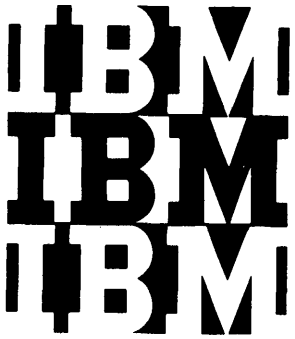


Form Y30-2002-2

Program Logic

IBM System/360 Operating System

Queued Telecommunications Access Method

Program Logic Manual

Program Number 360S-CQ-519

This Program Logic Manual describes the internal logic of the Queued Telecommunications Access Method (QTAM) under Option 2 and Option 4 of the IBM System/360 Operating System. This publication is intended for use by personnel involved in program maintenance and by system programmers who are altering the system design. Program logic information is not necessary for the use and operation of the program; therefore, distribution of this publication is limited to persons with program maintenance or modification responsibilities.

PREFACE

This Program Logic Manual is a guide to the internal structure of the Queued Telecommunications Access Method (QTAM). It is designed to be used with the program listing; program structure at the machine instruction level is not discussed.

Effective use of this manual requires a knowledge of the concepts presented in the following IBM System/360 publications:

IBM System/360 Principles of Operation, Form A22-6821

IBM System/360 Operating System: Queued Telecommunications Access Method, Message Control Program, Form C30-2005-2

IBM System/360 Operating System: Queued Telecommunications Access Method, Message Processing Program Services, Form C30-2003-3

In addition, the following publications may be used when information about other elements of the control program is required:

IBM System/360 Operating System: Assembler 32K, Form Y26-3598

IBM System/360 Operating System: Assembler 64K, Form Y26-3700

IBM System/360 Operating System: Basic Direct Access Method, Program Logic Manual, Form Y28-6617

IBM System/360 Operating System: I/O Supervisor, Program Logic Manual, Form Y28-6616

IBM System/360 Operating System: I/O Support (OPEN/CLOSE/EOV), Program Logic Manual, Form Y28-6609

IBM System/360 Operating System: Job Management, Program Logic Manual, Form Y28-6613

IBM System/360 Operating System: Linkage Editor, Program Logic Manual, Form Y28-6610

IBM System/360 Operating System: Sequential Access Method, Program Logic Manual, Form Y28-6604

IBM System/360 Operating System: Direct Access Device Space Management, Program Logic Manual, Form Y28-6607

IBM System/360 Operating System: Catalog Management, Program Logic Manual, Form Y28-6606

IBM System/360 Operating System: Fixed-Task Supervisor, Program Logic Manual, Form Y28-6612

This publication contains the following: discussions on the physical organization and logical organization as an introduction to QTAM, an outline of the QTAM operation as an overall logic flow, the function of BTAM within QTAM, a summary of the internal logic at the routine level, flowcharts of each routine, and appendixes. The routine names that appear as labels on the overall logic flowchart can be used to access the detailed flowchart for the specific routine. The labels on these detailed flowcharts relate to the labels on the listings for the routine.

Throughout this publication, option 2 of multiprogramming with a fixed number of tasks is assumed (MFT). QTAM also runs under option 4 of multiprogramming with a variable number of tasks (MVT). There are no major differences in these two options of the operating system for the logic of QTAM except that partitions are regions and priority of partitions must be assigned to jobs in MVT.

Third Edition, November 1968

This edition, Y30-2002-2, corresponds to OS Release 17. It is a major revision of, and renders obsolete, Form Y30-2002-1 and associated Technical Newsletters. Changes not documented in Technical Newsletters to the previous edition are indicated in the following manner: changes to the text are indicated by a vertical line to the left of the change; in the case of a page which contains all new information, a bullet (•) is placed next to the page number; similarly, changed or added illustrations are denoted by a bullet to the left of the caption.

Significant changes or additions to the specifications contained in this publication are continually being made. When using this publication in connection with the use of IBM equipment, check the latest SRL Newsletter for revisions or contact the local IBM branch office.

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 reader's comments. If the form has been removed, comments may be addressed to IBM Corporation, Programming Documentation, Dept. 844, P.O. Box 12275, Research Triangle Park, North Carolina, 27709.

This section describes the various parts of the total package called QTAM and explains what the parts are, where they come from, how they get into the system, and their relationship to the rest of the package. The function of these QTAM parts and the logic of their operation are discussed in detail in subsequent sections.

Figure 1 shows the steps taken to begin processing in the QTAM environment. The following discussion deals with these steps:

1. System generation.
2. Assembling and linkage editing a message control program.
3. Assembling and linkage editing a message processing program.
4. Initializing a message control program.
5. Initializing a message processing program.

SYSTEM GENERATION

QTAM NUCLEUS

When QTAM is called for during a system generation procedure (QTAM operand in DATAMGT system generation macro instruction), a number of routines collectively called the QTAM nucleus are included as a permanent part of the System/360 Operating System supervisor nucleus. These routines are then always present in the system, whether or not a telecommunications application is being run.

The QTAM nucleus is packaged as a single module named IECKQQ01. During system generation, it is linkage edited from SYS1.MODLIB into SYS1.NUCLEUS. It is loaded from there by the IPL program as one of the resident SVC routines. The QTAM nucleus consists of the following nine subroutines, each of which is discussed later in this manual:

1. Entry interface
2. QTAM wait
3. QTAM post
4. Qdispatch
5. Defer entry
6. Priority search
7. Queue insert
8. Exit select
9. Exit interface

QTAM MACRO DEFINITIONS

The operating system macro definition library (SYS1.MACLIB) includes the macro definitions used during the assembly of the message control program and message processing programs. Appendix D lists the QTAM macro instructions.

EXTERNAL ROUTINES

When performing a system generation to include QTAM, the user must define a special library area named SYS1.TELCMLIB. During the generation run, all routines that will later be linkage edited with message control and message processing object modules are copied from SYS1.MODLIB into SYS1.TELCMLIB. In this publication, these routines are defined as external routines. Appendix D lists the modules in SYS1.TELCMLIB and indicates the function performed by the routine or routines in each module.

SUPPORT MODULES

During the generation run, all modules that are loaded into main storage by the various Open executors and the QTAM Open and Close executors are copied from SYS1.MODLIB into the SYS1.SVCLIB. In this publication, these modules are defined as support modules. Appendix D lists the QTAM support modules in SYS1.SVCLIB.

ASSEMBLING AND LINKAGE EDITING A MESSAGE CONTROL PROGRAM

The user codes the QTAM macro instructions necessary to design a message control program. The output of this assembly includes: several tables and control blocks, a buffer area, linkages to QTAM external and support routines, and, except for these linkages and a few minor Line Procedure Specification (LPS) macro instruction expansions, very little other executable code. The message control object module may include some user-written routines, but these usually will not be extensive.

The assembled object module is then linkage edited to include the necessary external routines from SYS1.TELCMLIB. These external routines are the LPS routines used in processing header information, translating from one code to another, directing messages to the proper lines and queues, etc.

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

ASSEMBLING AND LINKAGE EDITING A MESSAGE PROCESSING PROGRAM

A message processing program normally needs only the OPEN, CLOSE, GET, and PUT macro instructions and some data set definition macro instructions. When this is the case, no external routines are required to be linked with the object module. An installation will also write one or more message processing programs that use the following macro instructions to examine and modify the status of the control program:

- CHNGP
- CKREQ
- CLOSEMC
- CHNGT
- COPYP
- COPYT
- COPYQ
- RELEASEM
- RETRIEVE
- STOPLN
- STARTLN

When any of these macros are used, the linkage editor will include the corresponding external routines in the load module. The load module is stored into a system library for execution.

INITIALIZING THE MESSAGE CONTROL PROGRAM

The QTAM message control program is normally executed in partition 0 as the highest priority task in the system. The initiator/terminator loads and transfers control to the message control program. The first QTAM macro instruction executed must open the DASD queue area. When the system Open routine detects the unique organization code for the QTAM DASD queue, it loads and transfers control to the first QTAM Open executor (module IGG01930). The Open routine performs several functions described in more detail in subsequent sections. For the purposes of this section, however, we need note only that the Open routine loads a large module called the QTAM Implementation module (IGG019NG) and Checkpoint/Restart module (IGG019NH) into partition 0, along with the Message Control Load module.

The Implementation module contains three distinct types of routines - distinct as far as their logical relationship to the rest of the system. The three types are:

1. Problem program routines - executed enabled to all interruptions as part of the message control program task. These routines receive control through branches from the external routines linkage edited with the message control program.

2. Supervisory routines - executed disabled to all interruptions as part of the QTAM nucleus "task." These routines receive control through branches from the QTAM nucleus.

3. I/O appendages - executed disabled to all interruptions, again logically as part of the QTAM nucleus "task." These appendages receive control from the I/O Interruption Handler in the input/output supervisor (IOS).

The logical relationship of the preceding routines is discussed more fully in the next section. When only physical organization is considered, this collection of routines represents no more than a convenient and efficient packaging technique. The Implementation module can in no way be thought of as a "program."

When the DCBs for the communications line groups are opened, four other QTAM Open executors are used (modules IGG0193N, IGG0193R, IGG0193T, and IGG0194A). These routines also perform several functions to be discussed later. For this discussion, however, note that only the WTTA Line Appendage (IGG019QB) is loaded by the first of these four executors when opening a WTTA line group; the BTAM Read/Write routine and BTAM modules containing model channel programs are loaded by the third of these four executors. These modules are also loaded into partition 0. The BTAM Read/Write routine is run in the problem program state as part of the user's message control task.

INITIALIZING A MESSAGE PROCESSING PROGRAM

It is possible to run a message control program with no message processing program. For example, a message switching application can be handled entirely within the message control program with a single message processing program loaded at the end of the day to initiate a system shutdown procedure. However, there is usually at least one, and possibly two or three, message processing programs being executed at

the same time as the message control program.

In this discussion, assume the normal case where a message processing program is to be loaded into partition 1 immediately after the message control program is loaded and initiated. A Start Initiator Function should be employed. This will load the message processing program into partition 1. When the message control task goes into the wait state, the message processing program opens the process queues, at which time the GET/PUT macro instruction support routines needed are also brought into partition 1. There are three Get modules and three Put modules. The modules selected depend on the unit of data processed by the program: segment, message, or record.

At any point during the initialization of this message processing program task, control may return to the message control program because of an I/O interruption from one of the communications lines or from a direct access storage device. More often, execution of the processing program task continues up to the pcint of a GET instruction before the message control task has a message to pass on. In this case, the processing task is placed in a wait state until the conditions for accomplishing the GET are satisfied. At any rate, the process of initialization is complete at this point with all of the parts of QTAM in place and running.

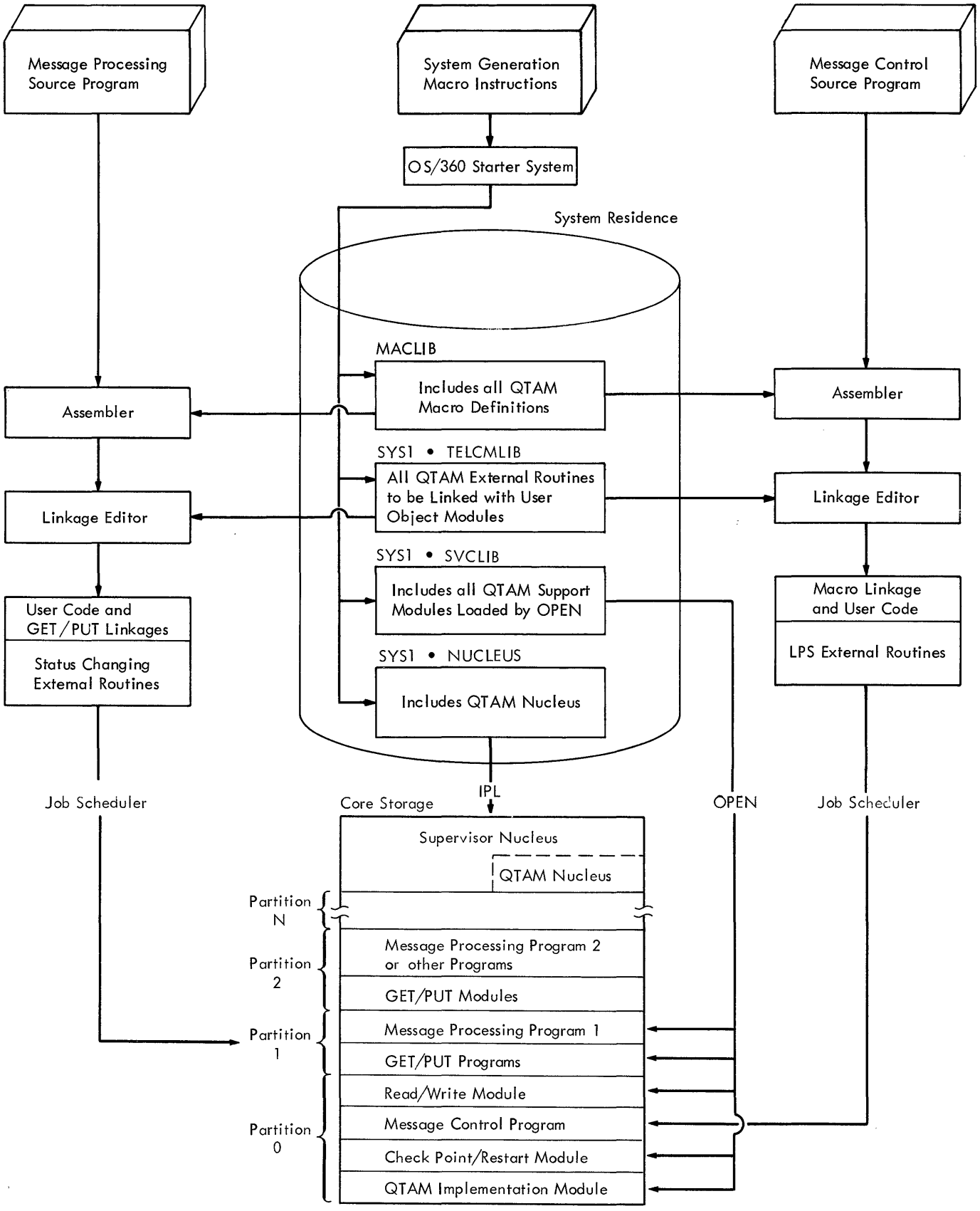


Figure 1. Physical Organization of QTAM

The previous section explained how the physical pieces of QTAM are positioned in main storage. This section discusses how these pieces are logically related and how they pass control back and forth.

In this section, the logical organization of QTAM is discussed within two different frameworks. First, QTAM is considered as a part of operating system task management and within the structure and categories of that control program. Then QTAM is considered as a separate logical entity outside of the framework of the operating system control program, and is viewed as a control program in its own right. The key to understanding the logical organization of QTAM lies in understanding the overlap of the two control program structures.

QTAM WITHIN THE OPERATING SYSTEM CONTROL PROGRAM STRUCTURE

The various pieces of QTAM discussed in the preceding section can be grouped into three logical categories:

1. Message control program
2. Message processing program(s)
3. QTAM supervisory routines

The message control program and message processing programs are both run under control of the operating system task management routines. When considered as a part of operating system task management, these programs are in no way different from any other processing program tasks. 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.

After distinguishing and separating the two processing program tasks, only the third category, the QTAM supervisory routines, remains. These routines are executed as type 2 SVC routines or as asynchronously scheduled I/O interruption-handling routines. Strictly speaking, they are executed as part of the processing program tasks. Practically speaking, however, it is more meaningful to think of these routines as a separate category outside of the task framework established by operating system task management. This section is primarily an explanation of the nature of this third category in relation to the other two categories. The discussion continues subsequently in the section QTAM Supervisory Routines, but first the message control program and message processing problem programs are more closely defined.

MESSAGE CONTROL PROBLEM PROGRAM

The message control problem program includes the following:

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

2. The external routines linkage edited with the assembly output.

   Note: If the DLIST macro instruction is used, a single supervisory routine, called the Distribution List routine in a module named IECKDLQT, is linkage edited into the message control load module. This routine is one of the supervisory routines, and is not part of the problem program.

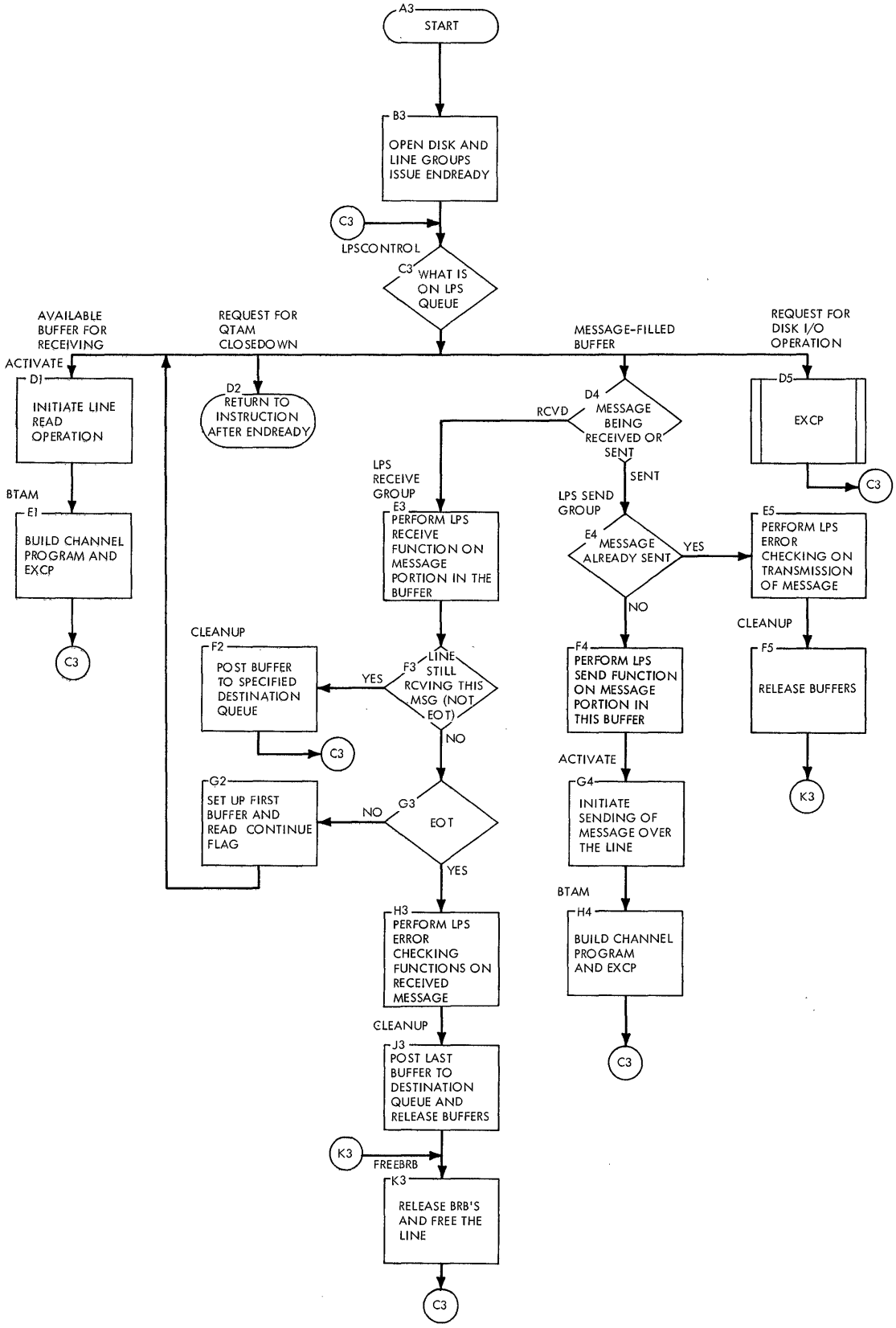
3. Five of the routines in the implementation module brought into partition 0 by the DASD OPEN:

   Note: Full descriptions of these routines may be found under the heading QTAM Implementation Module Routines. Flowchart "IDs" for each are identified below:

   - Activate (Chart DP)
   - LPS Control (Chart DO)
   - Buffer Cleanup and Recall (Charts DD, DE)
   - Free BRB (buffer request block) (Chart DF)
   - End Insert (Chart DG)

4. The BTAM Read/Write routine and BTAM Device I/O modules (modified for and incorporated into QTAM) brought in by the line group OPEN.

A simplified flowchart of the message control problem program is shown in Figure 2. The flowchart is included to show how four problem program routines in the Implementation module and the BTAM Read/Write routine are related to the rest of the message control program.



●Figure 2. Flowchart of Message Control Program

MESSAGE PROCESSING PROBLEM PROGRAM

A message processing problem program includes: the assembled user code, any external routines linkage edited with the code, and the Get and Put routines. The only difference between a QTAM message processing program and any other processing program is the requirement for and the implementation of interpartition communication. The various macro instructions that can be used in a message processing program are handled as follows:

1. COPYP, COPYT, and COPYQ. These macro instructions present no problem. The corresponding external routine simply reads the requested information from partition 0, using address pointers stored in the communications vector table (CVT) and in the terminal table.

2. All other macro instructions. The remaining macro instructions cause SVC interruptions to the QTAM supervisory routines. Any cross-partition communication is done by the supervisory routines, operating under the storage protection key of the supervisor.

The only unusual operation to be noticed when logical organization is considered is in the case of a PUT macro instruction. To avoid including a large amount of code in the supervisory routines for each of the three types of PUT (segment, record, or message), certain code that must be executed in the supervisor state is packaged within the Put modules. The SVC routine entered as a result of a PUT branches directly back to these routines in the problem program Put modules to execute them in the supervisor state.

QTAM SUPERVISORY ROUTINES

This discussion of the QTAM supervisory routines is still within the framework of the operating system control program. When the physical organization of these routines is considered, they consist of:

1. The routines within the supervisor nucleus.

2. The routines within the Implementation module (in partition 0) that are executed in the supervisor state. This includes all except the five routines previously identified as part of the message control problem program.

3. The Distribution List routine linkage edited with the message control program.

4. Part of the Put modules in the message processing problem partition(s).

When the interruption-handling facilities of the operating system are considered, the QTAM supervisory routines consist of:

1. Type 2 SVC routines entered by SVCs 65 and 67 from problem program partitions.

2. Asynchronously scheduled I/O interruption-handling routines entered from IOS.

Although the QTAM supervisory routines can be considered from either point of view, neither is very helpful in understanding the logical organization of QTAM. For example, a routine within an appendage, to which control is passed to process an I/O interruption, may also be executed as the result of an SVC interruption. The problem is that both points of view are taken from within the framework of the operating system control program environment and are seen within the categories of that system. The solution to the problem lies in understanding the implications of the statement: "QTAM is a Control Program."

QTAM is a control program that is within a second control program. Before discussing how the two control programs overlap, it is important to describe the QTAM control program within its own framework as a separate logical entity.

QTAM AS A SEPARATE CONTROL PROGRAM

The one essential function of a control program is the allocation of system resources. The system resources to be allocated by QTAM are:

1. CPU processing time
2. Main storage space
3. I/O paths

In order to perform this allocation function efficiently, it is necessary to break up the system resources into the smallest practical number of pieces. This is done as follows:

1. The work to be done is broken into many separate work units that are defined as QTAM subtasks of message processing and message control tasks. Small pieces of the time resource are then allocated to individual subtasks.

2. The main storage space to be allocated is broken into a large number of buffers. Thus, only that amount of storage absolutely required at a given

time need be tied up for a given function.

3. The I/O paths controlled by QTAM are the communications lines and the disk queue. Only that I/O path absolutely required at a given time need be tied up for a given function.

The term allocation is usually used only to refer to physical resources; scheduling refers to time resources. In a QTAM control program (as opposed to the operating system) the entire allocation function is performed by a single mechanism. This allows interdependence of scheduling and allocation.

The following sections describe the resource allocation mechanism of QTAM. The key to the mechanism is the ready queue, the structure through which a resource is allocated to a subtask. The actual mechanism of allocation is the Qwait and Qpost operations performed by the QTAM subtasks. Qwait, in effect, puts a request for a resource on the ready queue. Qpost passes an available resource to the ready queue. The QTAM nucleus performs a queue management function that includes dispatching the subtask that is at the top of the ready queue.

QUEUE MANAGEMENT

Elements, Queues, and Subtasks

The physical resources of the system are broken into elements (e.g., the buffer pool, a resource, is broken into individual buffers, the elements) with each element represented by an RECB (resource element control block), which can be thought of as an 8-byte identifying prefix.

If the RECB points to an available buffer queue, the buffer is free and not in use. The RECB is an identifier.

For every element in the system, there is at least one subtask that works with the element. These subtasks are represented by STCBs (subtask control blocks).

The elements, and the subtasks that operate on these elements, are associated with one another through the use of a third control block, the QCB (queue control block). Thus, a QCB will have a pointer to the chain of elements under its control and a pointer to the chain of subtasks waiting to operate on these elements.

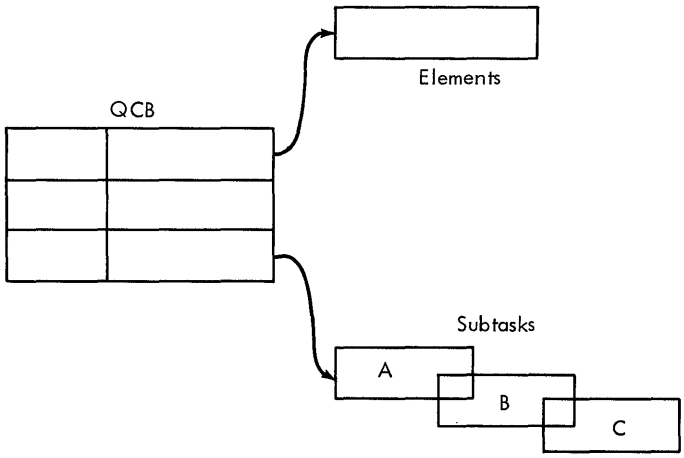


Figure 3. (Part 1 of 2)

When a subtask needs an element, it requests one from the QCB that handles that particular element by "Qwaiting" in the STCB chain of the QCB. If the element is available, the subtask that Qwaited is dispatched.

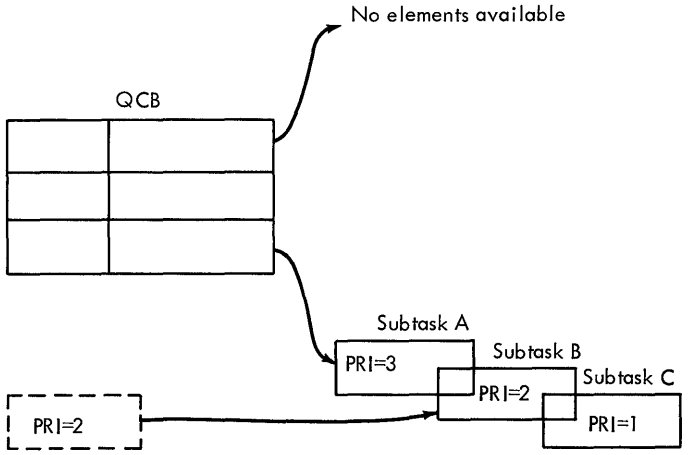


Figure 3. (Part 2 of 2)

When a subtask has finished using an element, it gives (Qpcsts) the element to the appropriate QCB (Figure 4). The QTAM nucleus gives this element to the first (highest priority) subtask in the STCB chain of the QCB. (Subtask A in Figure 5 would be dispatched). Note, however, that STCB A is not usually removed from the STCB chain unless it Qwaits on another QCB.) If another element is posted to this QCB, subtask B will be dispatched. The STCB chains end with a permanent STCB. (STCB C, in Figure 4, will remain the last STCB in the chain.) STCB C might point to a routine that does nothing more than chain elements into the QCB's element chain. Subtask C would have a lower priority than any other subtask that might use the element and would, therefore, be dispatched only if no other subtask needed the element.

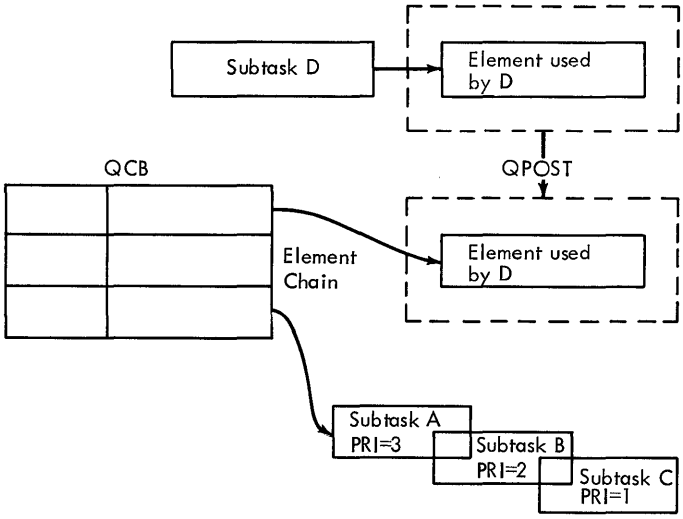


Figure 4.

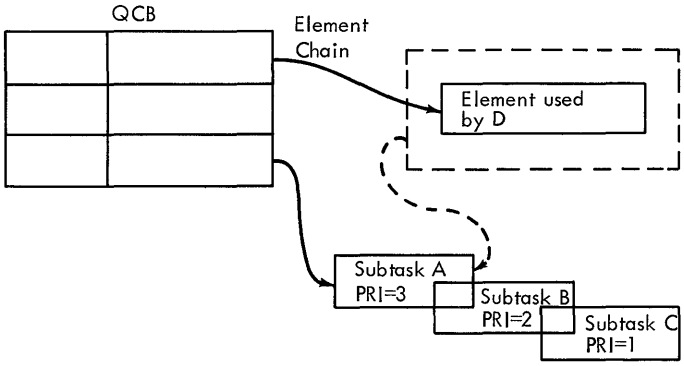


Figure 5.

The Ready Queue

The previous discussions pointed out that subtasks gain control depending on:

1. The availablity of elements;

2. The priority of the subtask.

Since QTAM is a control program, it is responsible for allocating CPU processing time to the various tasks under its control. The mechanism it uses is called the ready QCB, which can be thought of as a QCB whose element chain is "time" and whose subtask STCB chain is all the work to be done in the system. (Note that the ready QCB's subtask chain is called the ready queue). The work to be done is represented by the various QCBs and RECBs. These QCBs and RECBs, just like the STCBs within their own chains, appear on the ready queue in a priority order (Figure 6).

To complete the general picture (Figure 6), an RECB (resource element control block) appears on the ready queue. As was mentioned previously, when an element is Qposted to a QCB, the first subtask in the QCB's chain gets control (register 1 points to the RECB being Qposted).

In most cases, however, the Qpost is a two step operation. The element's RECB contains a pointer to the queue to which it is being posted and is placed on the ready queue in priority order (this is the first step). As time becomes available for processing, the ready queue is examined by a routine called Qdispatch in the QTAM nucleus. If the routine finds an RECB on the ready queue, the RECB is replaced with its QCB; then the first subtask in the QCB's chain is executed (this is the second step of the Qpost operation [Figure 7]).

The ready queue's operation can be understood through an illustrative example dealing with two simultaneous events:

First Event: A message starts coming across the line into an allocated buffer. Other buffers must be obtained to accommodate the message in case its length exceeds that of one buffer (high priority event).

Second Event: At the same time, a subtask that has written a buffer to a disk now frees the buffer by posting it to some QCB whose subtask will chain it into an element chain (low priority event).

In order to obtain a buffer, a BRB (buffer request block) is posted to a QCB whose subtask will eventually fulfill the request for the buffer. The empty buffer and a BRB will be placed on the ready queue "on their way" to their appropriate queues. It is much more vital to obtain a buffer for the incoming message than to chain the freed buffer, so QTAM assigns a higher priority to the BRB than to the buffer and chains them both into the ready queue in priority order. The BRB will, therefore, be handled first (i.e., the BRB will be replaced on the ready queue by the QCB to which it was posted; and the first subtask in the QCB's STCB chain will get control to obtain the needed buffer).

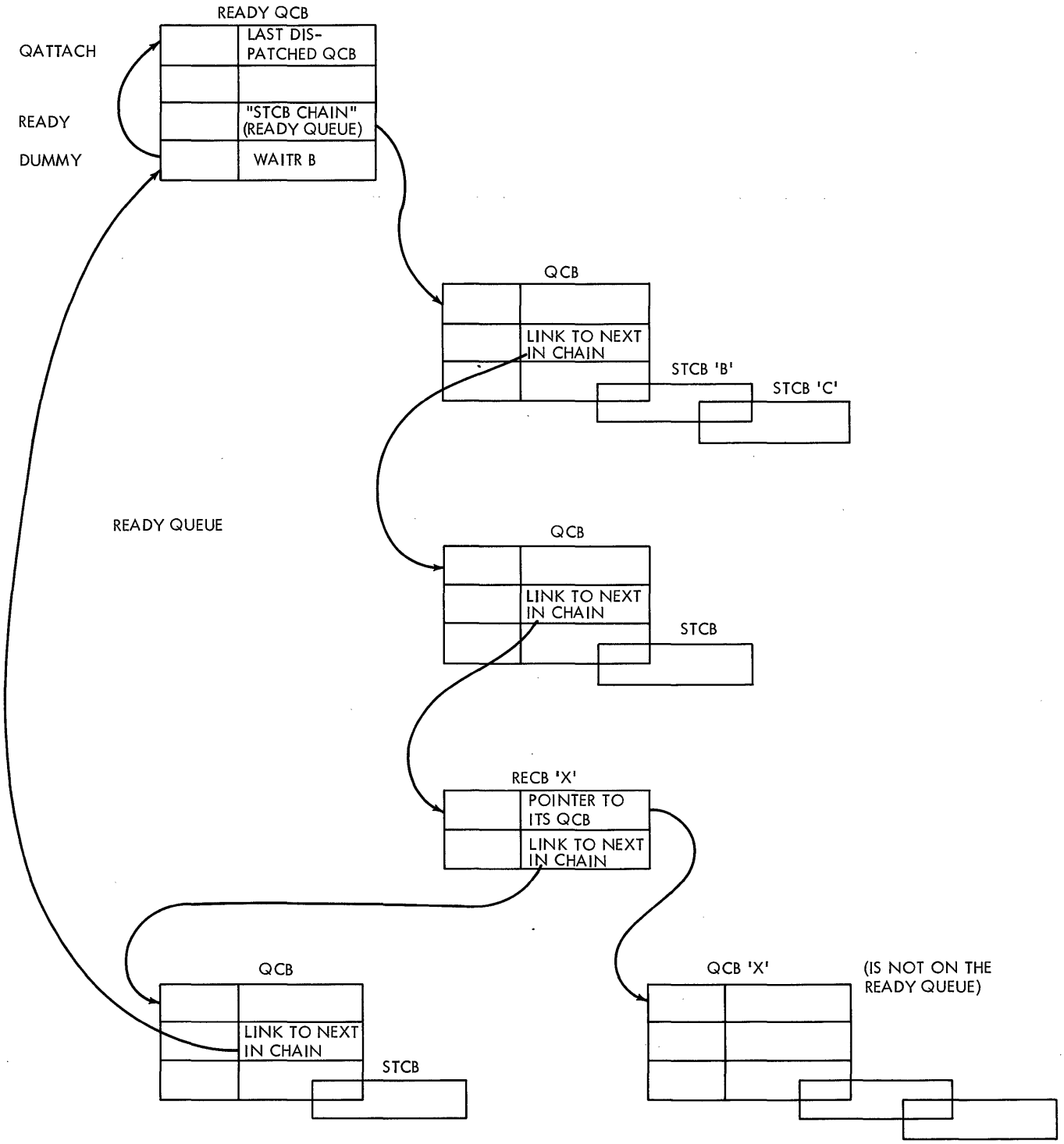
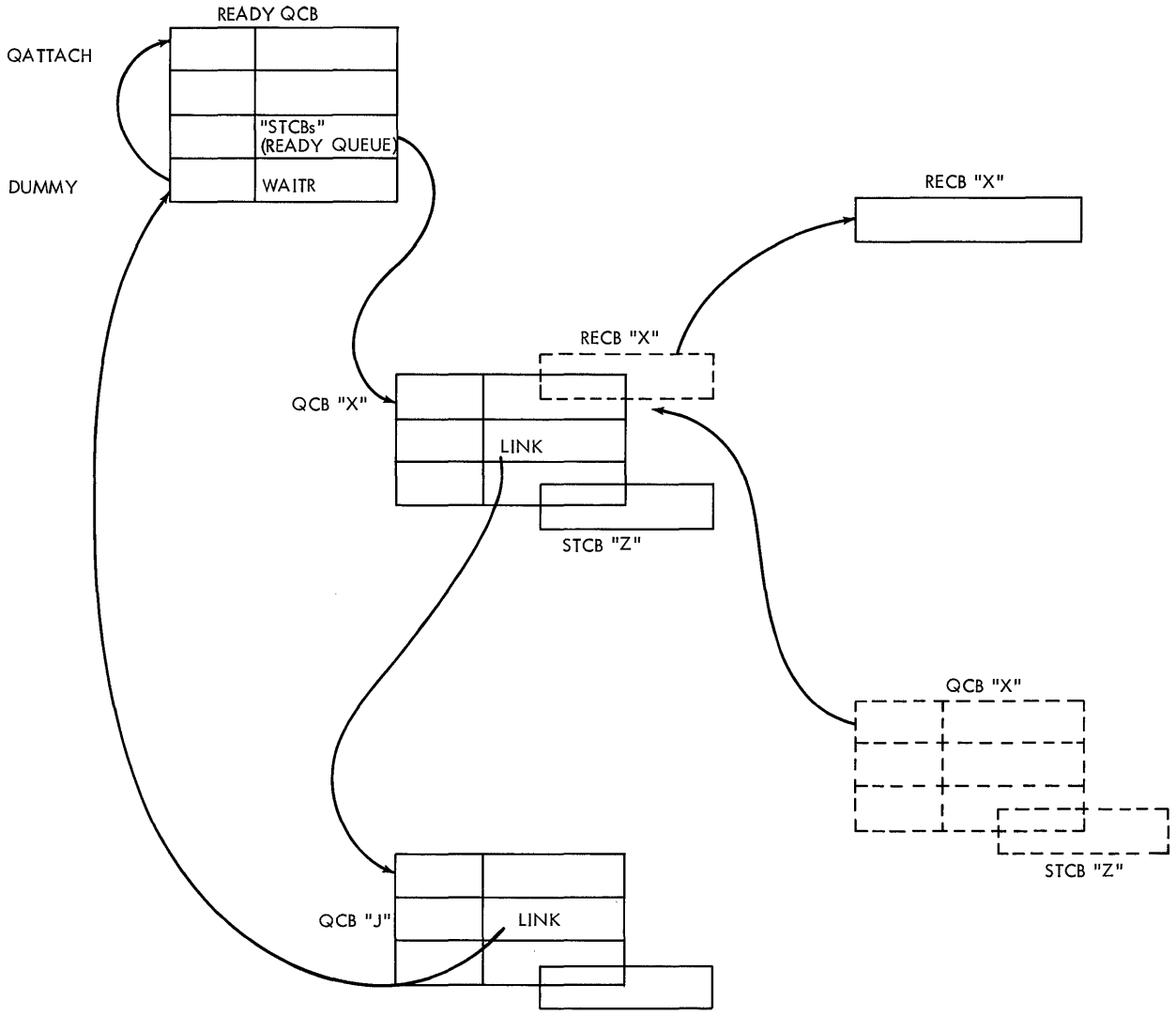


Figure 6. Element Chain



●Figure 7. Second Step of Qpost Operation

CONTROL BLOCKS

Resource Element Control Blocks

There are three main types of permanent resource element control blocks (RECBs):

1. Buffer RECBs
2. Communications line RECBs
3. Buffer request RECBs

Figure 8 shows the general form of a RECB.

Buffers are areas of main storage used to contain message data and/or control information. The first 8 bytes of each buffer comprise an RECB. As with all QTAM elements, the identity of a buffer at a particular time depends solely upon the queue its representative RECB is chained into at that time. The buffer itself is always physically identifiable as a fixed number of bytes of main storage. If the RECB representing the buffer is chained into a destination queue control block (QCB), the buffer is full; that is, it contains a message segment to be transmitted to a destination. When the same RECB is subsequently chained into the available buffer QCB, the element involved is an available buffer, even though there has been no change in the physical storage area.

Communications lines are represented to QTAM through the line control block (LCB). There is an LCB for each line. When a subtask has control of an LCB, it has control of the line. Therefore, the LCB itself is

treated as the resource element. The RECB is contained within the LCB.

In order to avoid preassigning buffers before they are actually needed, QTAM uses buffer request blocks (BRBs) to queue buffer requests. (This process is explained later in the section entitled Outline of QTAM Operation.) These BRBs are elements. The RECB is contained within the BRB. There are at least as many BRBs in the system as the number of buffers in the buffer pool. Thus, this pool of BRBs is itself a pool of resources to be allocated to the various subtasks that use them.

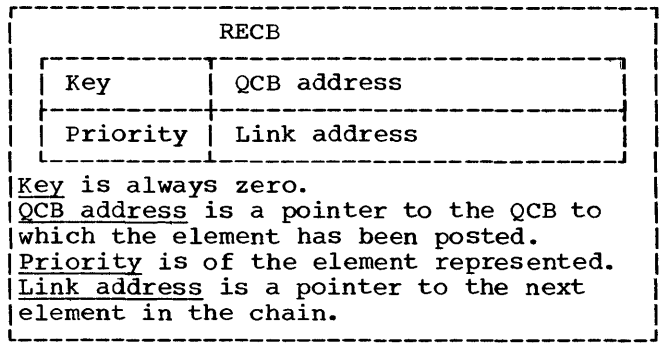


Figure 8. Resource Element Control Block

Subtask Control Blocks

There are two types of subtask control blocks (STCBs):

1. Truncated STCBs
2. Full STCBs

These are shown in Figure 9.

Truncated STCBs represent subtasks that are executed in supervisory state. These subtasks are performed by routines that are packaged within the Implementation module (and also by the Distribution List routine linked with the message control program). These routines are called implementation routines and the truncated STCB represents an implementation subtask.

Full STCBs represent subtasks that are executed in problem program state. These subtasks are performed by the message control program and message processing problem programs. At this point, we see the overlap of the operating system control program structure with the QTAM control program structure. A QTAM problem program subtask is created when an SVC 65 (Qwait) or 67 (Qpost) is issued within an operating system task. More specifically, the supervisor request block (SVRB) created by the second-level Interruption Handler is modified and used as a QTAM STCB. As a subtask, the problem program is placed under the subtask management of QTAM and must

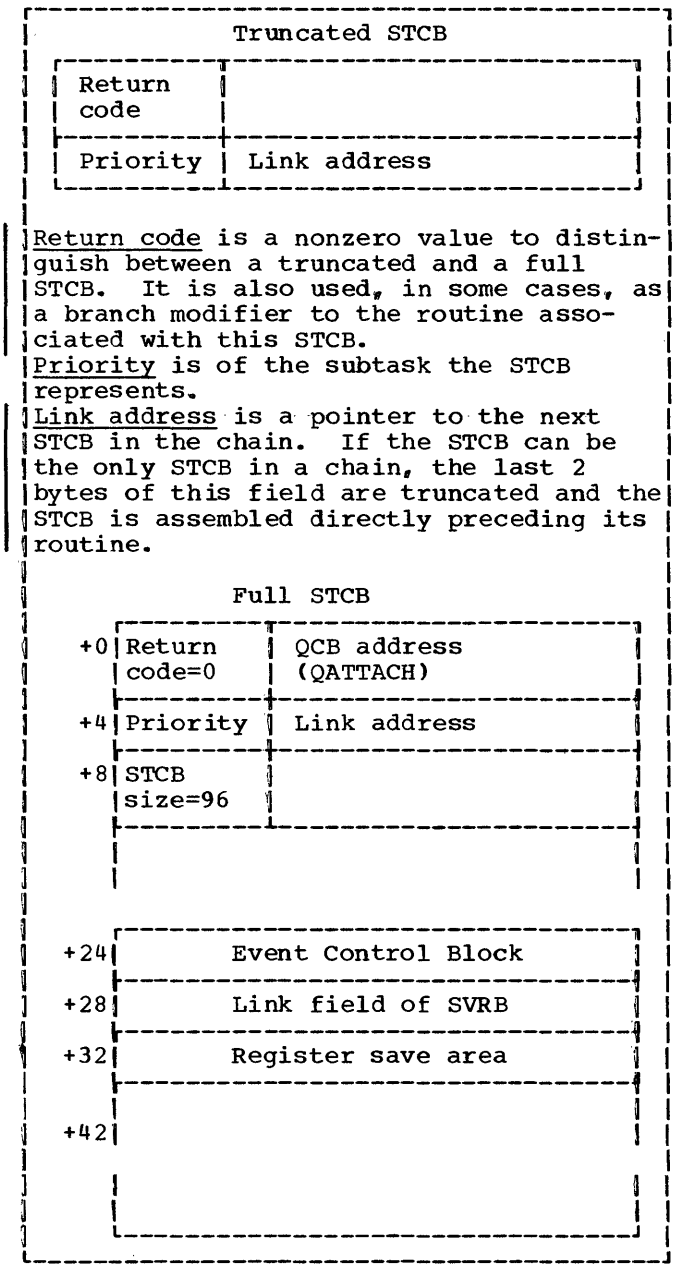


Figure 9. General Form of Full and Truncated STCB

contend for control in that multitask environment before it is released to contend with other operating system tasks in the system. The way in which this is implemented is discussed more fully in the following sections. Note at this point, however, that every problem program request that results in a QTAM SVC 65 or 67 causes a subtask to be created. These problem program subtasks are always lower in priority than any implementation subtask; thus they are never considered for dispatching until all of the internal implementation subtasks have done all of the work possible with the resources available.

There can never be more than one full STCB per problem program partition at a time.

Queue Control Blocks

The ready queue can be thought of as a queue of queues, each queue being associated with a queue control block (QCB). Figure 10 gives the general form of all QCBs that are on the ready queue and an example of a QCB that has replaced an RECB on the ready queue. The types of queues that may appear at any given time on the ready queue are discussed in the following paragraphs. A more complete and detailed list of queues is given in Appendix A.

Available Buffer Queue: This queue is used to keep track of unassigned buffers. The element chain is the chain of all buffers that are not assigned. As soon as a buffer is no longer needed, it is posted to this queue. The STCB chain for this QCB is limited to the STCB for the available buffer subtask, which is used whenever a buffer is made available.

LPS Queue: This queue is used to pass elements from the QTAM control program to the message control problem program. As shown in Figure 2, the element chain may point to:

1. An empty buffer, signifying that a Line Read operation is to be initiated.
2. A message-filled buffer to be passed through some portion of the LPS.
3. A request for a disk I/O operation to be started.
4. A request for a QTAM closedown.

The LPS queue controls the problem program of the message control task. The LPS Control routine in the message control program waits for the LPS queue. When an element is available, the LPS Control routine is given control. This routine examines the element to determine which of the four possibilities is the first item in its element chain. Figure 2 shows the action that is taken for each case. The STCB chain for this QCB is the STCB for the LPS Control routine.

Main Storage Process Queue: This queue is used to pass full buffers from the QTAM control program to a message processing program. The element chain is the chain of buffers containing the message unit that is passed to the message processing program. This is the QCB that a message processing program GET waits for.

Inactive BRB Queue: This queue is used to keep track of inactive buffer request blocks. The element chain is the chain of all BRBs that are not assigned. As soon as a BRB is no longer needed, it is posted to this queue. The STCB chain may contain the STCB for a receive-scheduling subtask and/or one or more send-scheduling subtasks.

Active BRB Queue: This queue is used to pass active buffer requests from the various subtasks that require buffers to the active buffer request subtask, which obtains the buffers. The element chain is the chain of active BRBs. The STCB chain is limited to the STCB for the active buffer request subtask.

Additional CCW Queue: This is a queue of insert blocks containing the CCWs used to transmit idle characters when certain line control characters are encountered in an outgoing message. When one of these line control characters is encountered by the send portion of the LPS, the problem program waits for this queue to obtain one of these insert blocks.

Disk Input/Output Queue: BRBs containing channel command words are posted to this queue when a Disk Read operation is required. Full buffers are posted to the same queue for writing messages on the disk. The STCB chain is limited to the STCB for the disk input/output subtask.

Communications Line Queue: There is one QCB for each communications line. The QCB is created from the LCB itself when the LCB is encountered on the ready queue. This occurs as follows:

1. When a send or receive operation is completed, the LCB is posted to the ready queue as an element.
2. When the LCB reaches the top of the ready queue, a field within it is initialized as a QCB.
3. The element chain is then completed by posting the LCB to itself.
4. A receive-scheduling subtask is then dispatched for the line unless there is already a send-scheduling subtask waiting for the line.

Return Buffer Queue: This queue is used by the GET macro instructions to return a buffer. After the data has been transferred to the work area, the buffer is returned to the available buffer queue via this QCB.

Time Queue: This queue is used to delay the polling of a line for a specified amount of time. The element chain for this queue is the LCB waiting for an interrupt from the Timer.

QCB

key	element chain pointer
priority	link address
	STCB chain pointer

Example of the Available Buffer QCB on the Ready Queue

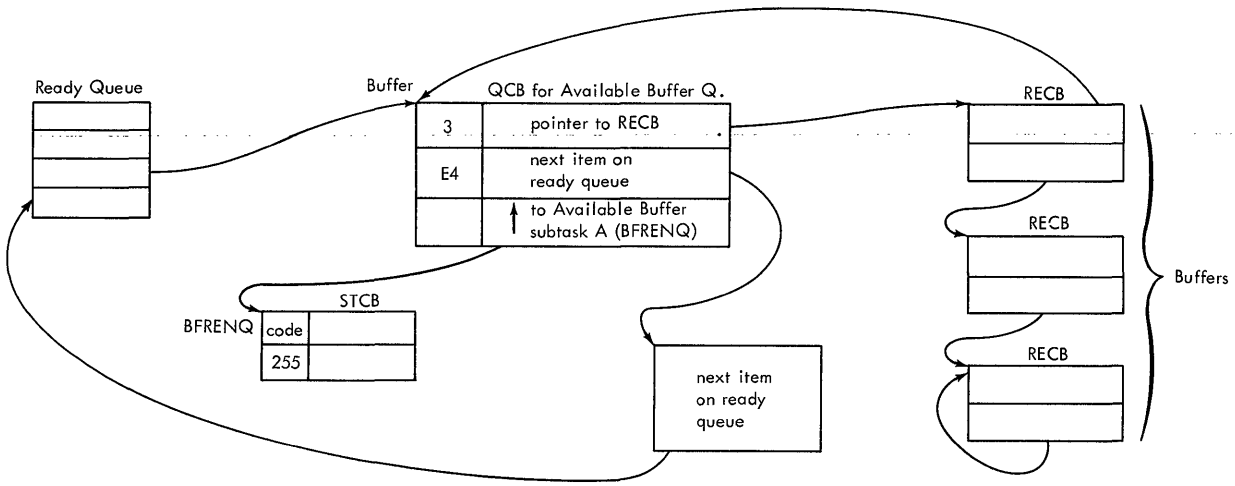


Figure 10. General Form of QCB and Example of QCB on the Ready Queue

Move Data Queue: This queue is used to move data while in supervisor mode. Data can be moved within a partition or across partitions.

Interim LPS Queue: This queue is used to delay processing of buffers until all requests have been processed. Elements of this queue are transferred to the LPS queue.

Note: Both the DASD destination QCB and the DASD process QCB never appear on the ready queue. They are assembled off the ready queue, but with a Key of 3 (see the following discussion of Keys and Appendix A for a description of the above QCBs).

QWAIT AND QPOST

A subtask requests a resource from a queue by issuing a Qwait on the associated QCB. A subtask passes a resource that it is finished with by Qposting the resource to the proper QCB.

Qwait from Problem Program: A problem program (either message control or message processing) requests an element from the QTAM system by issuing an SVC 65.

Note: All QTAM SVCs are macro generated. The programmer should never have to issue one directly. Because this is a type 2 SVC, the supervisor call second-level Interruption Handler (SVC SLIH) creates an SVRB and passes control to the Entry Interface routine in the QTAM nucleus.

The operating system SVRB is converted to a QTAM full STCB and is temporarily chained into the STCB chain of the last dispatched QCB (i.e., if the message control problem program [LPS] was issuing the Qwait, the LPS QCB would have been the last dispatched QCB). The address of the QCB for the element queue being waited for is passed in register 2. If the element is available, the full STCB is removed from its temporary chain, the element's address is placed in register 1, and control is returned to the problem program.

If an element is not available, the full STCB is added to the STCB chain of the QCB whose element chain is being waited for. An SVC 1 (WAIT) is issued to place the requesting task in the wait state. The operating system task management routines then dispatch some other task if there is one waiting. Otherwise, these routines place the entire system in the wait state.

When another subtask subsequently posts an element to the queue that the problem

program waited for, QTAM dispatches the problem program subtask by posting the event control block waited for as complete. The problem program is then dispatched in its proper task priority by operating system task management.

Qpost from Problem Program: A problem program (either message control or message processing) passes an element to the ready queue by issuing an SVC 67. As with the Qwait, the SVRB contains the address of the Qattach QCB and is converted to a QTAM full STCB. The Qpost STCB is then chained into the STCB chain of the last dispatched QCB. However, in the case of the Qpost, the last dispatched QCB will usually be the ready QCB itself. Thus, the full STCB will be chained directly on the ready queue (see the discussion of Keys for an example of Qpost). The address of the queue that the element is being posted to is passed in register 2, and the address of the RECB for the element being passed is in register 1. The RECB is placed on the ready queue. (Note that when the ready QCB is the last one dispatched, the RECB is placed on the ready queue above the full STCB. The RECB has a higher priority.) If a subtask is waiting for the element, it is dispatched in priority order. If no subtask is waiting for the element, the RECB is chained to the proper QCB. When the full STCB gets to the top of the ready queue, control is returned to the problem program by the OS supervisor routines.

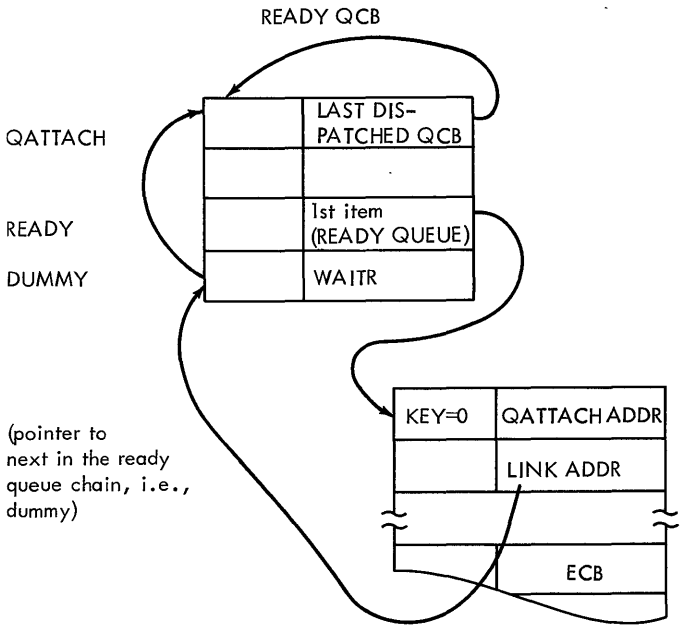
Qwait from Internal Implementation Subtask: When one of the implementation subtasks requires an element, the subtask checks the QCB for the element queue being waited for. If the element is available, the subtask removes it from the chain and relinks the element chain, if necessary.

If the element chain is empty, the subtask branches directly to the queue management routines in the QTAM nucleus. If the STCB for the requesting subtask is not already chained to the QCB for the requested element, it is placed on the chain. Control then passes to the Dispatch routine to activate the next subtask.

Qpost from Internal Implementation Subtask: When one of the implementation subtasks has an element to pass to the ready queue, it branches directly to the Qpost routine in the QTAM nucleus. The RECB, containing the address of the QCB to which it was posted, is placed on the ready queue. The STCB for the subtask that posted the element is left chained to the QCB that it was already on, and either the Qposting subtask or the subtask waiting for the element will be executed.

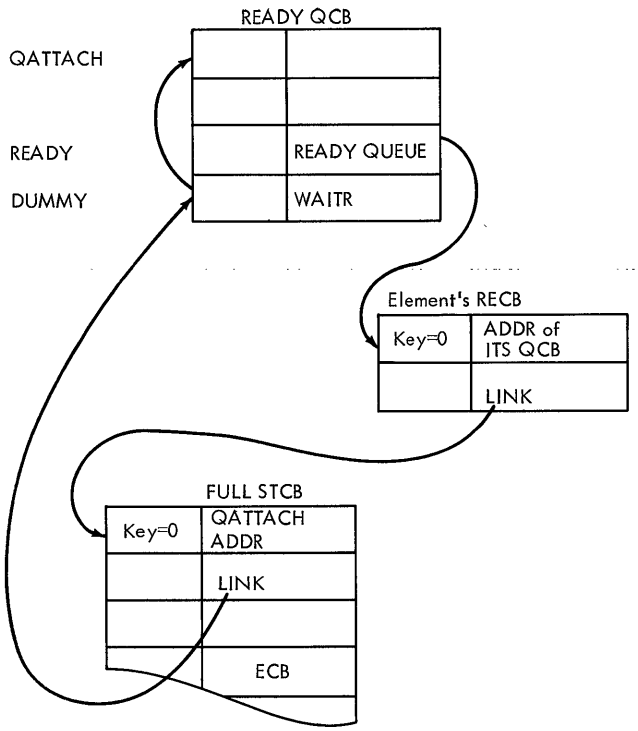
QPOST EXAMPLE

The new, full STCB is placed on the chain of the last dispatch QCB (the ready queue).

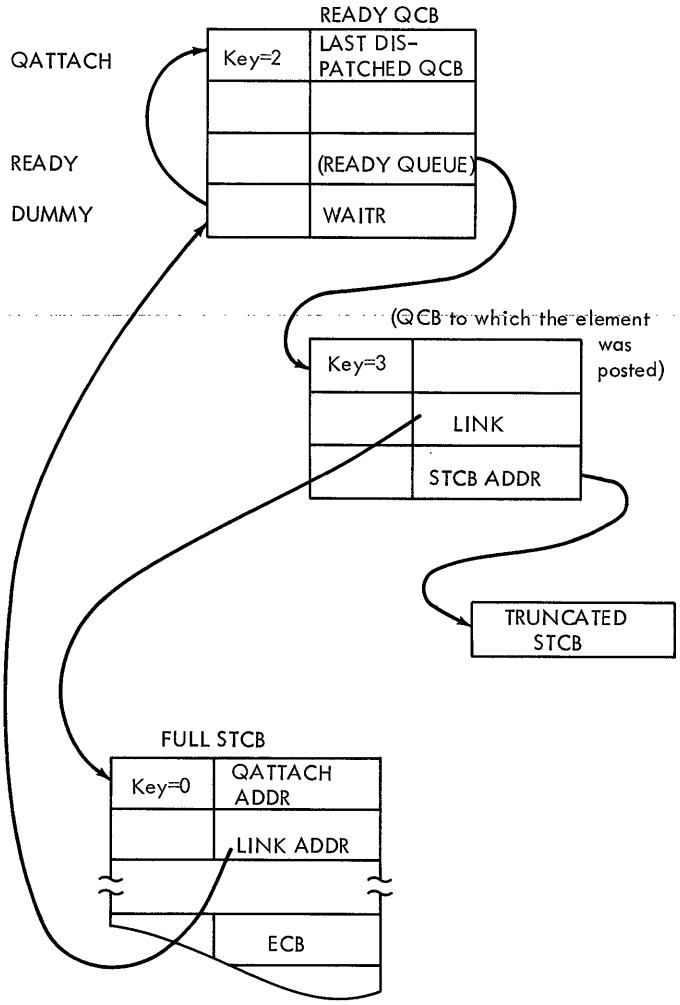


The Priority Search/Queue Insert routine places the posted element on the ready queue in priority order in front of the full STCB.

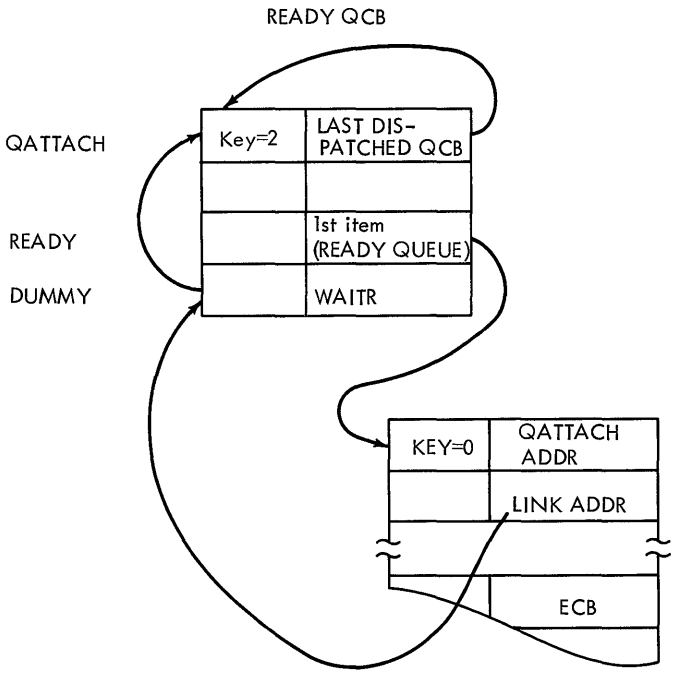
The Dispatch routine finds an RECB with a key of 0 and replaces it with its QCB (III).



The QCB has replaced the RECB. The QCB has a key of 3, and the truncated STCB in its chain is dispatched. When the subtask terminates, the Dispatch routine is entered. The key in the ready queue's QCB is set to 2. A QCB with a key of 3 is found at the top of the ready queue. The QCB is removed and its key is set to 1.



The QCB at the top of the ready queue has been removed. The ready QCB is dispatched by setting the key to 2. The ready queue is examined and an RECB (full STCB) with a key of 0 is found. The QCB pointed to (Qattach or the ready queue) looks as if it were on the ready queue (since it has a key of 2). The Exit Select routine sets the ready queue's key to 3; finds the full STCB in the chain (this chain is the ready queue); and posts it complete and exits.



QTAM NUCLEUS

The QTAM nucleus provides the overall queue management facilities. These facilities include:

1. Interfacing with the operating system to convert SVRBs to STCBs.
2. Placing problem programs in the wait state and then posting them as complete.
3. Chaining RECBs to the ready queue, and STCBs to QCBs in the proper priority sequence.
4. Dispatching the highest-priority subtask.

The nucleus is composed of several subroutines; each is discussed in the section QTAM Control Module. At this point, however, we can look at the queue management facility as a whole. Figure 11 shows a generalized flowchart of the nucleus. The Qdispatch routine examines the item at the head of the ready queue. The position of all items on the ready queue is determined by the relative priorities of elements as they are posted to the queue. Generally speaking, the priority of an element is determined by the type of subtask to which it is being passed. There are four priorities, indicated by a hexadecimal code in the RECB.

Highest-Code (EC): The only elements ever given a code of EC are a BRB or special dummy element. This is done in five instances:

1. To indicate that the buffer request for a disk operation has been unable to be assigned a buffer.
2. To indicate that a buffer request is made by the PCI interruption routine for the first PCI on a receive operation.
3. To execute a portion of the code of the Put routine in supervisor mode.
4. To recognize that a SIO is to be issued to the DASD.
5. To recognize that a QTAM closedown is in progress.

Second Highest-Code (E4): This priority is given to all elements being passed to implementation subtasks that are disabled to interruption, except those with a code of EC.

Third Highest-Code (E0): This priority is given to all elements being passed to the message control program.

Lowest-Code (DC): This lowest priority code is given to all elements being passed to message processing programs.

QDISPATCH ROUTINE

Qdispatch follows the address pointer in location READY to the item (either an RECB or QCB) at the top of the ready queue. To determine whether the item is an RECB or QCB, Qdispatch examines the key field in the first byte.

- Key=0: All RECBs have a key of zero. In some instances full STCBs appear directly on the ready queue instead of being chained to the QCB. Qdispatch will find a full STCB during initialization, when the ENDREADY macro instruction is executed and during a Qpost from a processing program (see sample Qpost above in Qpost Example). This full STCB appears to Qdispatch as an RECB pointing to a location labeled QATTACH at READY-8, the QCB of the ready queue. Therefore, the full STCB, whose address is at location READY (the top of the ready queue), appears at the head of an STCB chain in a QCB labeled QATTACH, and the full STCB is given control . If Qdispatch finds an RECB, one of the following events will result:

1. If the QCB pointed to by the RECB is not on the ready queue (key=1), the RECB is replaced by its QCB, and the first STCB in that QCB's chain is dispatched (see the queue management discussion).

2. If the QCB pointed to by the RECB has a key of 2, the RECB remains chained to the ready queue, and the first subtask in the QCB's STCB chain is dispatched.

3. If the QCB has a key of 3, the RECB is removed, and then the first subtask in the QCB's chain is dispatched. Note that the QCB does not, in the this case, replace the RECB on the ready queue.

- Key=1: Indicates a QCB that is not on the ready queue.

- Key=2: A key of 2 indicates a QCB with a subtask at the top of its STCB chain that is ready to be dispatched. A QCB with a key of 2, however, represents a special case. The STCB that is ready was previously entered when an element was made available to it. At some point in its processing, it exited (by Qposting or branching to either another Implementation module routine or to another part of the nucleus). Before it exited, however, it elected to be reentered whether or not another element was made available to it. In order to be reentered, this STCB had set its own QCB key to 2. Now, when an element is posted to this QCB, Qdispatch will discover that it is already on the ready queue with a key of 2. The STCB will, at this point, be reentered immediately. The element, however, will not be removed from the ready queue.

  In summary then, when Qdispatch finds an RECB pointing to a QCB with a key of 2, the first STCB in its chain will be gaining control for a second time (reentered), and that RECB will not be removed from the ready queue.

- Key=3: A key of 3 indicates a QCB with an associated subtask that has been dispatched, and the subtask has finished all the processing required with the element passed to it.

  Note: The dispatched STCB may or may not be the top STCB of the QCB's chain. The subtask might have, during the course of its operation, Qwaited on another QCB, in which case it would have been chained into the new QCB's STCB chain. Regardless of the location of the STCB, when Qdispatch finds a QCB with key=3, it removes the QCB from the ready queue and sets its key to 1.

The flowchart in Figure 11 further shows how control is passed to the dispatched subtask. If the subtask is represented by a truncated STCB, the Exit Select routine simply branches to the entry point of the subtask. If it is a problem program subtask (full STCB), the Exit Interface routine branches to the Supervisor Post routine to post this SVRB/STCB as complete, and then issues an SVC1 (WAIT) on the STCB that the QTAM control program is currently operating under. These SVRBs may or may not be the same. When they are not the same, we see the case where QTAM is placing one problem program task in the wait state and enabling another task that was previously placed in the wait state to again be dispatched by the operating system task supervisor.

There is one dummy element that is used to indicate the end of all element chains and is permanently the last item on the ready queue. This dummy element is preassembled in the ready queue's QCB (see Figure 6). Note that the physical blocks of main storage--the RECBs, QCBs, STCBs, and this dummy element--are never physically moved in main storage. Their pointers are merely changed to reflect their current relative positions (on or off the ready queue, in a chain, etc.). When this dummy element reaches the top of the ready queue, a final wait is issued to place the last QTAM problem program in the wait state until an asynchronous item is put on the ready queue.

Summary: The ready queue controls allocation of the resources. The contents of the ready queue tie an element with a subtask. Each resource element is represented by an RECB (Resource Element Control Block), which contains a pointer to an appropriate QCB. The QCB contains a pointer to an STCB associated with a routine that performs the desired function. To allow more than one item to request a subtask or wait for a resource, items are chained or queued to a QCB. Each subtask has an associated truncated STCB that contains a code that is used to gain access to the routine address. RECBs to be acted upon, QCBs with associated STCBs waiting for a resource, and full STCBs representing processing programs are chained to the ready queue. The second word of each item on the chain of the ready queue contains the address of the next item on that queue. The last item points to a dummy item. The position of all items on the ready queue is determined by priorities of the resource. These priorities, set by the subtask posting the resource, are determined by the type of function to be performed.

A subtask requests the resource (Qwaits) it requires for its execution from the appropriate QCB, performs its function, and passes (Qposts) the resource to another QCB for the next function to be performed. The Qposting and Qwaiting is done by the QTAM control program (IECKQQ01 in the nucleus). After chaining the item into its proper place, the QTAM nucleus examines the first item in the ready queue chain to determine which routine is to receive control. Three items can appear on the ready queue:

1. RECBs
2. Full STCBs
3. QCBs

The first byte of these control blocks contains a key, QKEY. A key of zero indicates an RECB or a full STCB. A QCB has a nonzero key whose value shows the status of the QCB. These keys are either preassembled in the QCB or set by IECKQQ01.

The three main types of elements represented by RECBs are: buffers, buffer request blocks (BRBs), and line control blocks (LCBs). By posting an element to a queue the QTAM nucleus (refer to Figure 11) causes:

1. The QCB address, passed in register 2, to be placed in the RECB whose address is passed in register 1.

2. The RECB to be inserted into the chain of the ready queue in priority order.

3. When the RECB reaches the top of the ready queue, the QCB, in the RECB, to replace the RECB on the ready queue if the QCB is not on the ready queue.

4. A subtask to be given control to perform the function. The truncated STCB in the STCB chain of the QCB provides the address of the routine for the subtask.

There are three ways of posting this element:

1. If a Qpost is issued via an SVC (only done in the problem program), an SVRB is created by the system, and the nucleus is entered at the Entry Interface subroutine. This subroutine transforms the SVRB into a full STCB that is used to return to the problem program. The RECB is chained as described above.

2. If posting is done in the implementation subtasks, registers 1 and 2 are set with the address of the RECB and QCB respectively, and the subtask branches directly to the post subroutine in the nucleus.

3. If the implementation subtask wishes to post several elements before another subtask gets control, the implementation subtask places the RECB containing a QCB address directly on the ready queue.

A full STCB is made from an SVRB created by the operating system as the result of an SVC. The STCB is chained to the last dispatched QCB. If this QCB is the ready queue, then the STCB is chained directly onto the ready queue. This STCB appears to the nucleus as an RECB whose QCB is on the ready queue with key=2. When the Exit Select subroutine discovers that it is a full STCB (by a zero code for the address of the routine), control is not given to a routine. The Exit Interface routine posts the ECB in the STCB as complete and issues a WAIT (SVC 1) for the entry STCB. Normally IOS, through the SVRB-STCB, returns control to the problem program. If this STCB that was serviced was not for the SVC that caused the entry, the message control task is in a wait state until there is an asynchronous interrupt.

The special form of the QCB (12 bytes) is the only type of QCB that appears on the ready queue. DASD process and destination QCBs (full QCBs) are not chained on the ready queue. A QCB can be placed on the ready queue by a Qpost or Qwait.

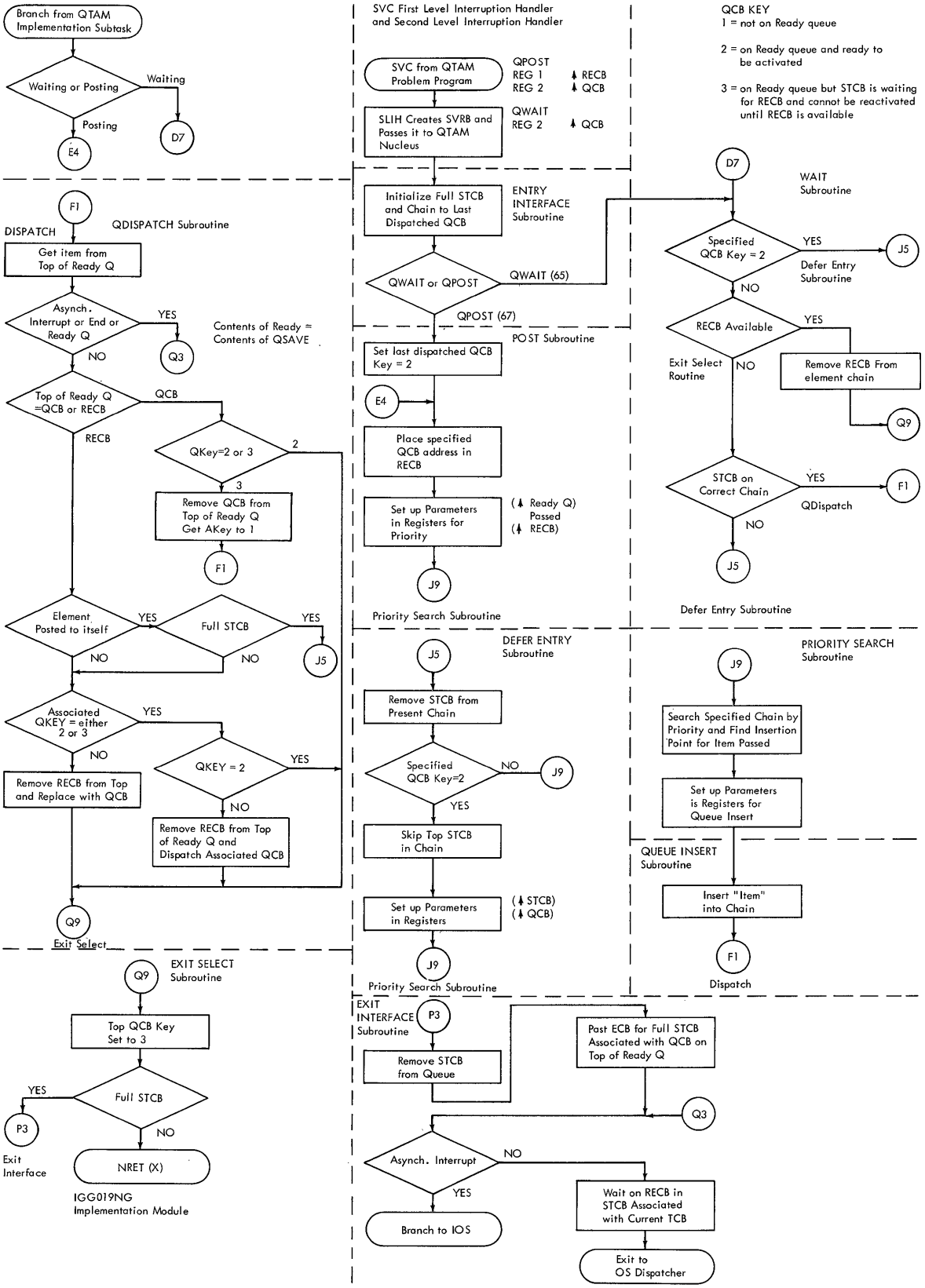
When an element has been posted to a queue and that queue is not on the ready queue, then the QCB is chained on the ready queue in place of the RECB. The key of the QCB is set to 3 to indicate that the QCB is on the ready queue but has been dispatched. When this QCB is encountered on the ready queue with a key of 3, it is removed and the key is set to 1 to indicate that it is not on the ready queue.

The above occurs for all QCBs with the exception of DASD destination and DASD process QCBs. As mentioned previously, these two QCBs never appear on the ready queue. They are preassembled off the ready queue with key=3. When an element is posted to one of these QCBs, it appears (to Qdispatch) that it is on the ready queue (since its key=3), and the first STCB in its chain will get control. Note, however, that these two QCBs do not replace the RECBs on the ready queue.

As the result of a Qwait, the full STCB is chained to the last dispatched queue. If there is an element available in the QCB being waited for (passed in register 2 to the Wait subroutine), the subtask is given control. The ECB in the full STCB is posted complete. Control returns to the problem program as the Qwait was satisfied.

If the QCB being waited on is ready to be activated (key=2), the Defer Entry subroutine causes the STCB to be chained to the correct QCB but deferred. If the QCB is not ready to be activated (key=1 or 3), the STCB is chained to the correct QCB (if the last dispatched queue is the one being waited for, the QCB is immediately dispatched), and the QCB is inserted in priority order on the ready queue. When an element is posted to this QCB, the STCB chained to the QCB is the full STCB previously chained. The QTAM control routine recognizes the full STCB and posts the event as complete. Return is made to the routine that issued the Qwait, for now this wait has been fulfilled.

Therefore, at a given moment, the ready queue consists of a chain of full STCBs, QCBs, and RECBs, arranged in priority sequence. The rate at which a subtask acquires resources is based on the availability of the resources and priority of subtasks. This allocating and dispatching of subtasks and resources is done by the single mechanism of the control program acting on the ready queue.



● Figure 11. QTAM Nucleus

OUTLINE OF QTAM OPERATION

The following description is intended to give a functional flow of messages through the QTAM operation.

Processing of a QTAM message control task is activated as a result of interrupts (SVC, program control, disk, line end, and line SIO) that occur during the sending and receiving of a message. These interrupts result in the processing of one or more asynchronously operating QTAM subtasks or appendages. These subtasks communicate with one another and the message control task by means of the Qpost and Qwait functions (see section on Qwait and Qpost). When a subtask has a resource element to be processed by another subtask, the element is posted to a queue representing that subtask. This is done in the problem program by a Qpost supervisory call; the implementation subtasks branch to post in IECKQQ01. Or an effective Qpost is issued by placing the resource element control block (RECB) in the buffer on the ready queue and the address of the QCB in the first word of the RECB. When a subtask is ready to receive an element, the Qwait function is used. The subtask sequence is managed by queuing to the ready queue as discussed earlier under Queue Management. The interference of one line with another is handled by the queuing provided within the Qpost/Qwait functions. This description shows the logical sequence of events for a message without regard for other subtasks and interrupts that may occur and that do not effect the processing of the message. Therefore, when an element is posted to a queue, the subtask associated with that queue is activated immediately. Also for the sake of continuity and simplicity, that function of the QTAM nucleus that is entered as the result of Qposting and Qwaiting is not included in this discussion. The description takes the example of a multisegment message ending in an EOB-EOT from a nonswitched terminal.

Figure 22 is a functional flowchart of the components of QTAM: message control task, opens and closes, message processing task, subtasks, and appendages. These components are separated by solid lines. Also on the flowchart, each subtask or module is separated by broken lines. The labels on the flowchart, Figure 22, are the names of the routines. The functional blocks for the routine follow the label. When more detailed information is needed for a particular functional block, refer to the detailed explanation of each routine. This detailed description also gives the sequence number of the logical flowchart for that routine. This detailed flowchart contains the labels that are in the listing of the routine. Note that some of the labels in Figure 22 are names of LPS delimiter macros for that group of the LPS. The function of the expansion of these macros is also represented with a functional block. For the QTAM nucleus subroutines, see Figure 11 in the Logical Organization of QTAM section.

This description is divided into five sections: Initialization, Receiving, Sending, Message Processing, and Closedown. The flow of QTAM operation can be traced by following the steps in the description of the flowchart, Figure 22.

INITIALIZATION

The initial function of QTAM is initiated by the OPEN macros in the problem program. Upon discovering QTAM, the system Open shifts control to the Open routines in the transient area. These routines obtain and initialize the control blocks (DEB, DCB, and LCB/IOB), load QTAM resident routines into partition 0, and prepare the lines for transmission.

Enter Message Control Task

1. Open disk (see Figure 12)
2. Open checkpoint data set
3. Open line groups (see Figure 13)

Enter QTAM Open Routines

- Open DASD

Message queues

1. Put the address of the terminal table in the CVT.
2. Build DEBs.
3. Load Implementation module and store the address in the terminal table.
4. Load the Checkpoint/Restart module and store the address in the Implementation module.

Load 2

1. Initialize the QCB whose address is in the terminal table with the address of the DASD destination STCB.
2. Execute subtask to put address of IECKQQ01 in the Implementation module.
3. Free main storage for secondary DSCBs.
4. Replace offset to polling list with

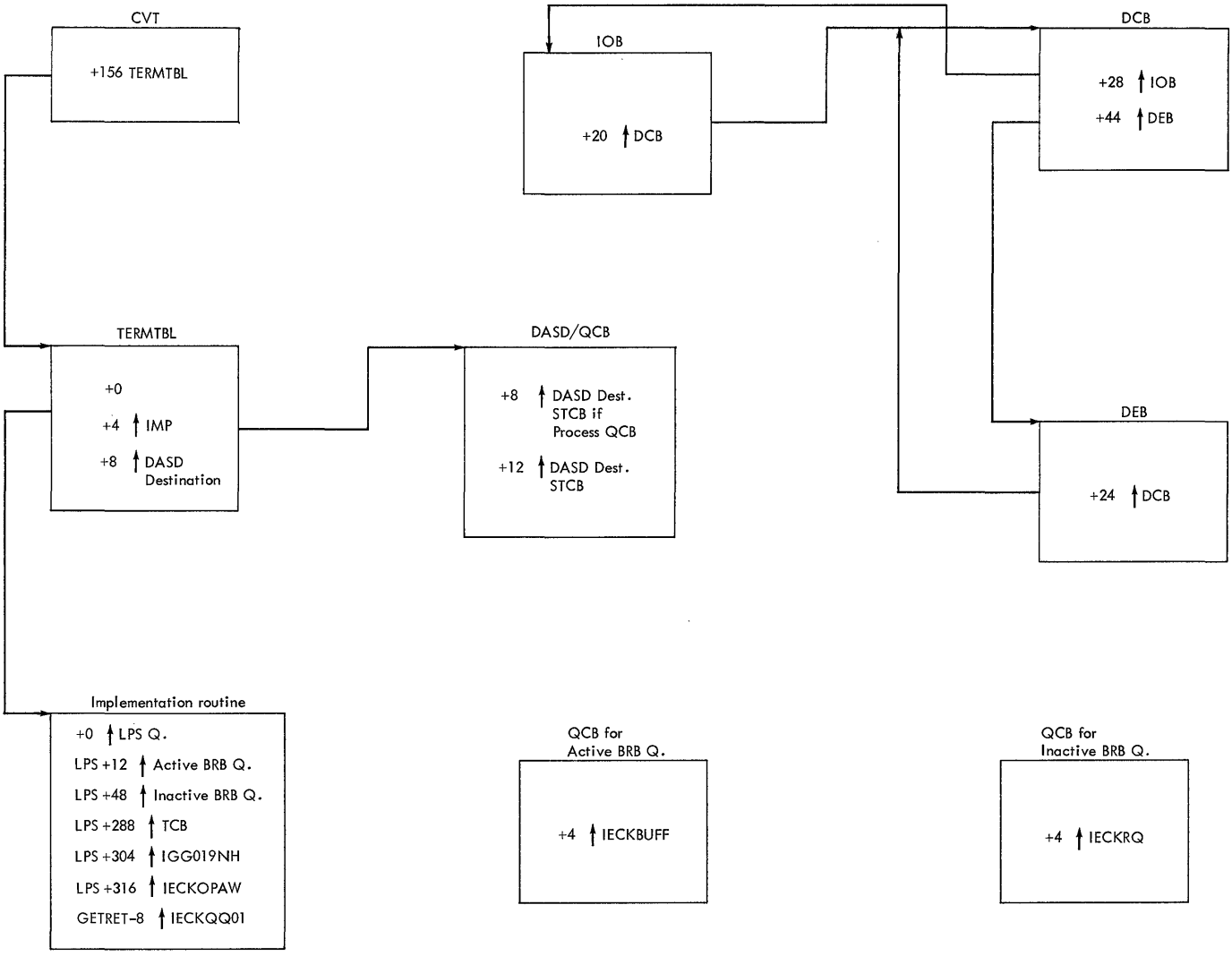


Figure 12. Blocks Initialized by Open Direct Access Device

the polling characters and index bytes.
5. Put buffers in the available buffer queue.
6. Put buffer request blocks (BRB) in inactive BRB queue.

- Open Checkpoint/Restart

1. Calculate size of checkpoint records.
2. For disposition NEW, write control record for first record of data set and two dummy checkpoint records.
3. For disposition OLD, the control record is read from the disk.
4. If not for a restart, the data set is formatted. If this is initialization for a restart, the checkpoint record is read (into the work area obtained by a GETMAIN). The data previously recorded is restored.

- Open Line Group Executors

Load 1

1. Build DEB.
2. If the line group is a WTTA line group, load the WTTA Line Appendage module and establish linkages with the Implementation module.

Load 2

1. Build LCBs and IOBs.
2. Build NOP, SAD, or Enable commands.

Load 3

1. Load BTAM Read/Write module and skeleton CCWs.
2. EXCP for each line.

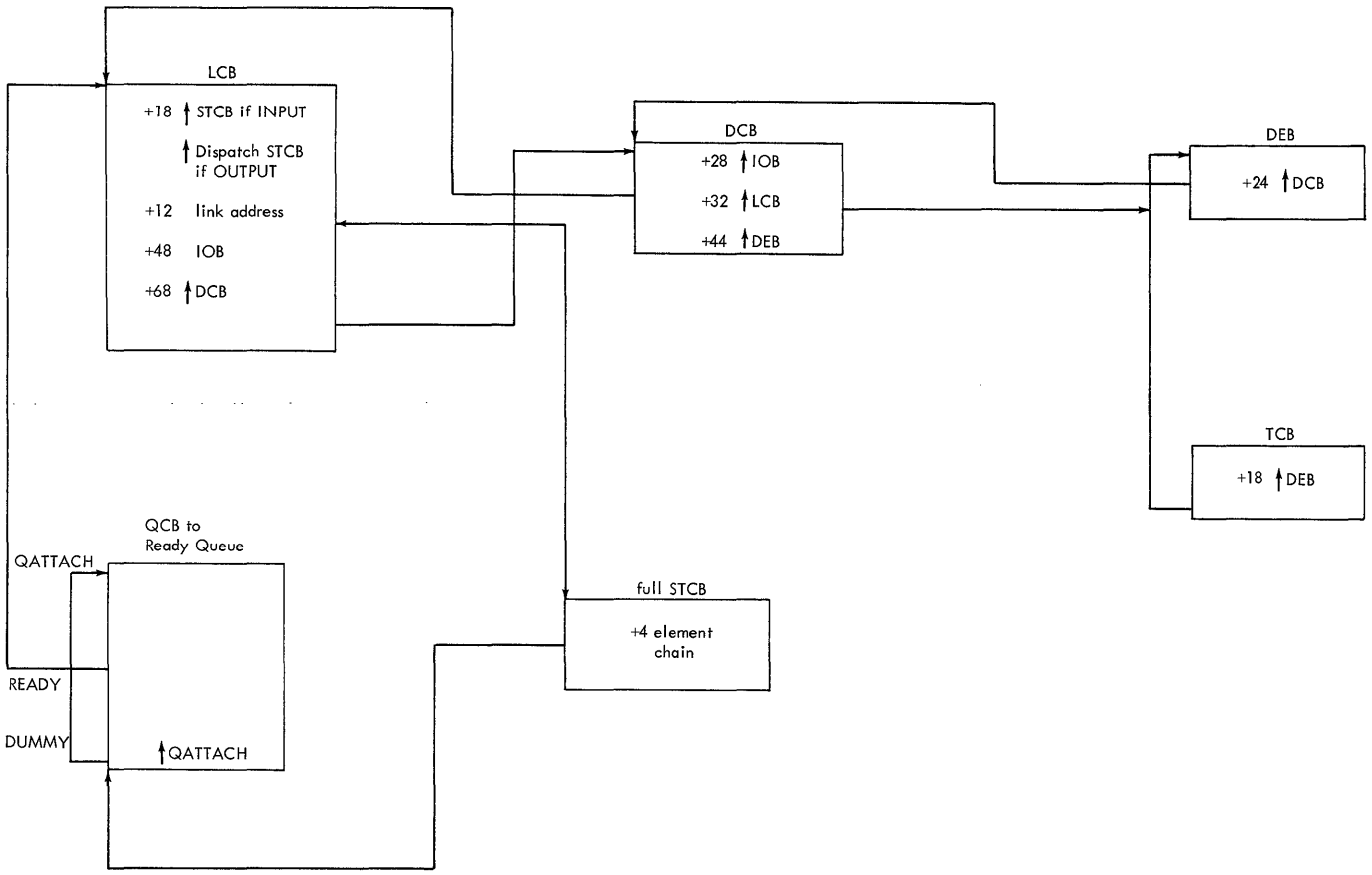


Figure 13. Control Block after Open Line Groups

The channel program for a SAD, Enable or NOP is executed to put the line in receive status. IOS gives control to the SIO Line Appendage routine, which requests that Error Recovery Procedures be given control. The Special Open and Checkpoint routine in the ERPs checks for SIO errors. If there was a normal Start I/O, return is made to IOS. The channel end/device end interrupt gives the Line End Appendage control. If IDLE has been specified, return is made to IOS. Otherwise, the LCB is posted to itself, i.e., the QCB and RECB are the same address.

The STCB for the Receive Scheduler is in the LCB if the line was opened for input. (If the line was opened for output, the STCB is for the Qdispatch subroutine.) The receive scheduler STCB contains the address of the Receive Scheduler routine, which gets control.

Enter Receive Scheduler subtask

- Receive Scheduler routine

1. Test for end of polling list.
2. If not end of polling list, set LCB-STATE for receiving (X'08). (Assume not end of polling list.)
3. Branch to BRB-Ring routine.

If it is the end of the polling list, the End of Poll Time Delay routine is entered. If a time interval is specified, the Send Scheduler is placed in the LCB that transmits messages during the interval (receive has priority over sending) or until all messages on the queue have been sent (receive and send have equal priority).

- BRB-Ring routine

1. Build ring of buffer request blocks (BRB) to be used for dynamic buffer allocation. (BRBs are obtained from the inactive BRB queue.)
2. Make BRBs unaddressable.
3. Post the first BRB to the active BRB queue with high priority.

The number of BRBs in the ring is equal to the value specified in the BUFRQ operand. The address of the first BRB in the ring is stored in the LCB so the Activate routine can gain access to it later. The first BRB is then posted with a priori-

ty of X'EC' to cause immediate servicing of the request for a buffer.

Enter Active BRB subtask

- Active BRB routine (High priority)

1. Test for available buffer (assume available). If the buffer is not available, the request is put into the active BRB chain of requests.
2. Branch to Buffer BRB routine.

- Buffer BRB routine

1. Assign empty buffer for receiving; i.e., the LCB address is placed in the prefix of the buffer.
2. Post empty buffer to LPS queue with priority of X'E0'.

Return is to Open Load 3 as the result of the EXCP.

Open Line Group Executors

Load 3

1. EXCP is issued for each line to cause each line to be made ready.

Load 4

1. Test for completion of I/O on each line. If I/O has not completed there is a 30-second delay.
2. Return is to the message control program.

For option 2 (MFT) and option 4 (MVT) a Start Initiator function should be employed. This will load the message processing program into another partition or region. (See the section on Initializing Message Processing Program.) The message processing program gains control when the message control task enters a WAIT state. Figure 14 illustrates the formation of the BRB ring and relation of the buffer to queues.

RECEIVING

Now there is an empty buffer for each line chained on the LPS queue and a ring of

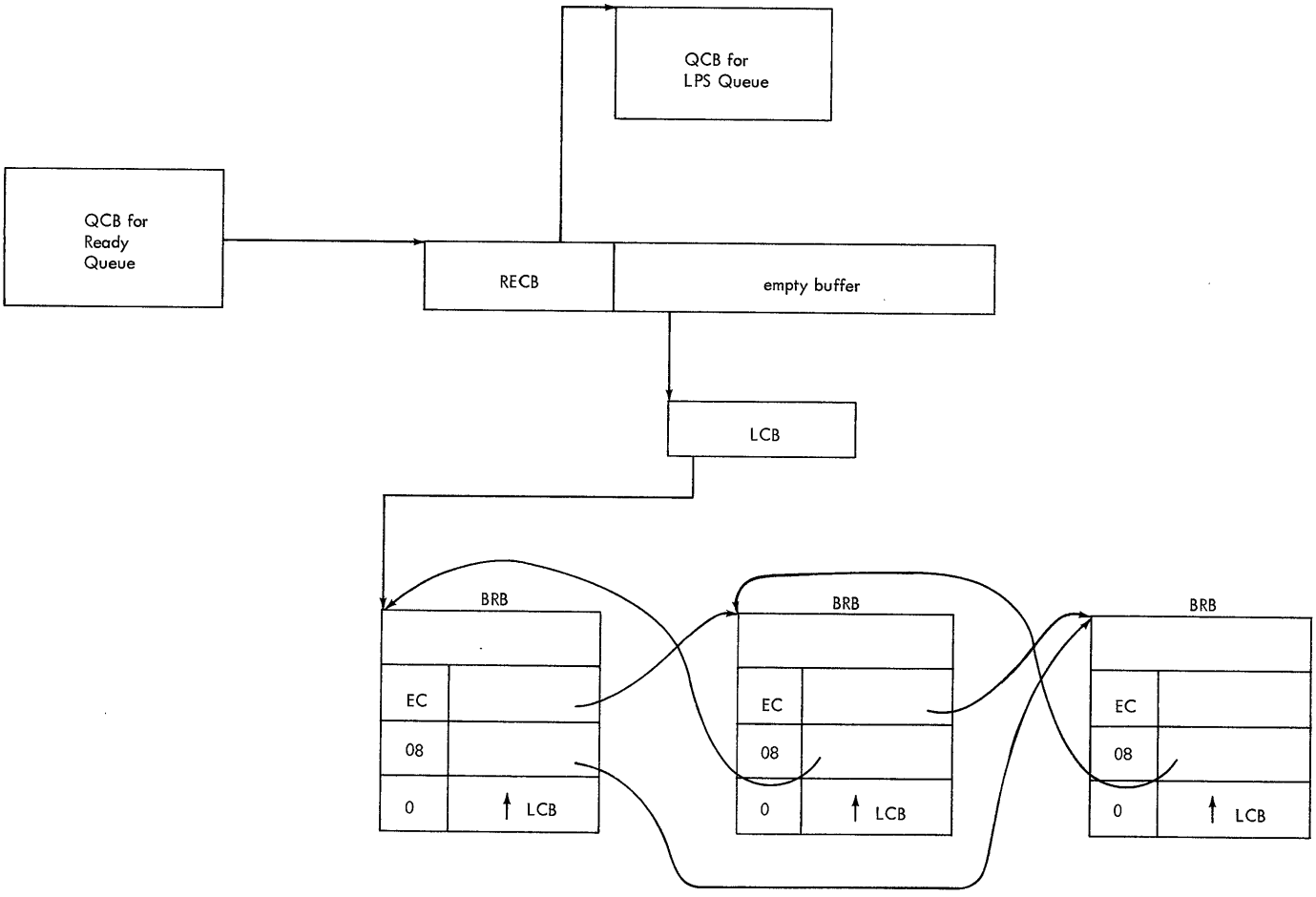


Figure 14. Buffer Ready to Receive Message from Line

BRBs for each line. The next function is to read the messages from the terminal into the buffers. To do this, the CCWs must be prepared for a particular terminal.

- ENDREADY macro instruction

1. Establish save registers.
2. Issue an SVC Qpost to enter the checkpoint subtask.

Enter Checkpoint Subtask

- Checkpoint routine

1. Set interval time via the Time Delay routine (if CPINTV is specified).
2. Release main storage obtained in the Open Checkpoint (if restart).
3. Return to ENDREADY via the full STCB of the SVC Qpost.

ENDREADY continued

1. Branch to LPS Control routine.

- LPS Control Routine

1. Set up registers for Activate routine.
2. Issue an SVC Qwait for buffer in LPS queue (empty buffer posted in Buffer BRB routine).
3. Since buffer is available, set MSTATUS to 5 and branch to the Activate routine.

- Activate routine

1. Prepare CCW for entire buffer in first BRB (buffer address, operation code, count).
2. Build DECB for BTAM Read/Write routine.
3. Branch to BTAM Read/Write routine.

- BTAM Read/Write routine

1. Prepare CCWs for terminal selection and reading first segment (address in DECB).
2. Issue EXCP supervisor call.

- IOS branches to SIO Appendage

- Line SIO Appendage routine (refer to Figure 11)

1. Move TIC command from BRB to end of BTAM-prepared CCWs.
2. Test for send request on line or end of polling list.
3. Get poll characters for next terminal that can be polled.
4. Change poll CCW to point to poll

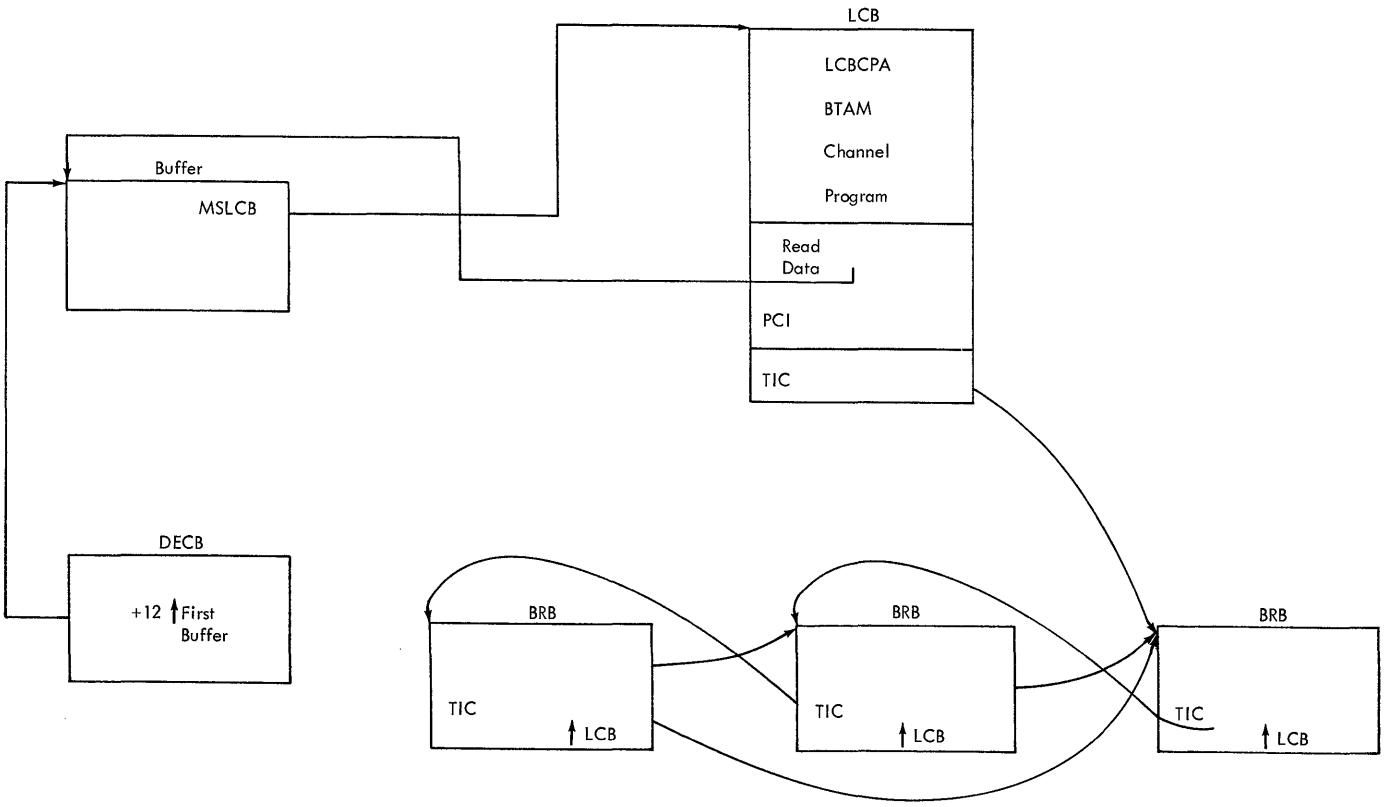


Figure 15. Channel Program Prepared for First Buffer

characters found in terminal table.
5. Set PCI flag in the BTAM Read CCW.

- IOS issues Start I/O

  Return is to the LPS Control routine.

- LPS Control routine

1. Issue a SVC Qwait for buffer in LPS queue.

After a Start I/O is executed for each line, the LPS Control routine will find no buffers on the LPS queue. The message control task will enter a wait state. Subsequent I/O interrupts activate subtasks that cause buffers to be posted to the LPS queue, allowing the message control task to continue.

PCI INTERRUPT (RECEIVING THE FIRST BUFFER)

The PCI Appendage is entered as a result of a PCI flag set in the BTAM CCW by the Line SIO Appendage. This PCI interrupt is to allow buffers to be assigned to the BRBs in the chain.

- PCI Appendage routine

1. Post (effective) all BRBs except the first to the active BRB queue with high priority to obtain a buffer.

Enter Active BRB subtask

- Active BRB routine

1. Obtain empty buffer from the available buffer queue (assume available).
2. Branch to Buffer BRB routine.

- Buffer BRB routine

1. Assign empty buffer for receiving.
2. Set MSTATUS to 5 to signify empty buffer.
3. Post empty buffer to the LPS queue with priority of X'E0'.

Enter Message Control task

- LPS Control routine

1. Set up register for Activate routine.
2. Branch to Activate routine (empty buffer).

- Activate routine

1. Prepare CCW for entire buffer in BRB.
2. Clear low-order bits from TIC command in previous BRB to make it addressable.
3. Branch to LPS Control routine.

- LPS Control routine

1. Issue an SVC Qwait for buffer in LPS queue.

PCI INTERRUPT (RECEIVING ALL BUFFERS EXCEPT FIRST)

PCI Appendage is entered as a result of a PCI flag in the QTAM CCW in the BRB in the ring. The PCI interrupt is needed to return the BRB to the active BRB queue so it can be reassigned. This interrupt also indicates that the preceding buffer is full and ready for the LPS macro instructions as shown in Figure 16.

- PCI Appendage routine

1. Post (effective) preceding BRB to active BRB queue with low priority.
2. Post (effective) all message-filled buffers to LPS queue (via interim LPS queue).

Enter Active BRB subtask

- Active BRB routine (low priority)

1. Chain BRB into active BRB element chain.

The interim LPS subtask is entered to post the buffer to the LPS queue. This subtask provides a means of delaying the processing of all buffers until all BRBs are processed. Since a PCI interrupt may be missed due to extended CPU disable time, a buffer may be out of order.

Enter Message Control task

- LPS Control routine

1. Set up registers for LPS.
2. Branch to LPSTART (message-filled buffer).

- RCVSEG portion to LPS

- RCVHDR portion of LPS (if header)

- ENDRCV macro instruction

1. Test for end of message, MSTATUS=X'42' (assume not end of message).
2. Branch to Cleanup routine.

- Cleanup routine

1. Post buffer to DASD process or destination queue specified by the ROUTE or DIRECT macro.

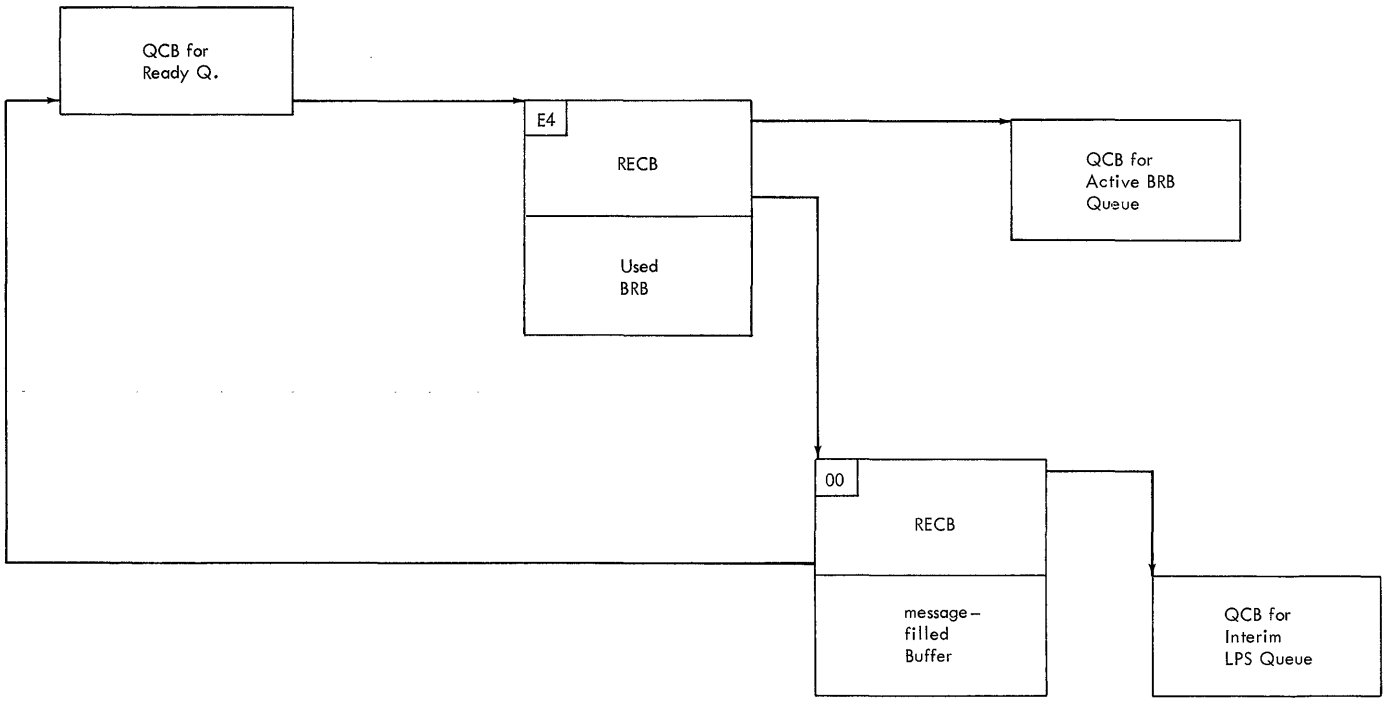


Figure 16. Effect of PCI Interrupt

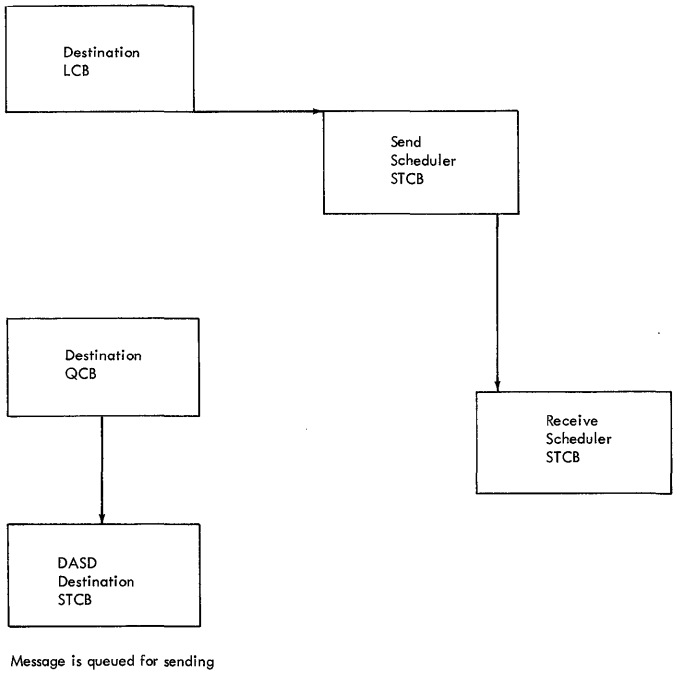
If posted to a process queue, the Get Scheduler routine is entered; if posted to a destination queue, the Send Scheduler routine is entered. After the Send Scheduler has been entered, the STCB in the DASD destination queue changes to point to the DASD Destination routine. Therefore, control would pass to the DASD Destination routine and exit to the Qdispatch subroutine to dispatch the next item on the ready queue. Both schedulers use the common code of the DASD destination routine. (If GET has been previously issued in message processing program, posting to the process queue is changed. This is covered later in the Message Processing section.)

- DASD Destination routine

1. Assign direct access location.
2. Reserve and record location of direct access space for next message and/or segment.
3. Post (effective) buffer to disk I/O queue.
4. Return to scheduler.

- Send Scheduler routine

1. Set 'line trying to send' bit in LCB (LCBINCAM = X'01') (Assume line is not free so the Send Scheduler will wait for the line to be free.)
2. Place Send Scheduler STCB in LCB's STCB chain.



Message is queued for sending

Enter Disk I/O subtask

- Disk I/O routine (write)

1. Convert relative record number to actual DASD address.
2. Execute EXCP supervisor call.

Return to Message Control task

- LPS Control routine

1. Issue a SVC Qwait for buffer in LPS queue. At this point there is no buffer on the LPS queue so the message control program enters a wait state.

TIMER INTERRUPT - CHECKPOINT INTERVAL

Enter Checkpoint/Restart routine

1. Issue a GETMAIN for main storage required for checkpoint record.
2. Transfer data to work area (information from terminal table, polling list, LCB, and QCB).
3. Chain element to disk I/O queue below any other request for a Disk Write. (If no elements are in the queue and EXCP is issued for the disk operation.)

Note: The first buffer has now been read from the line and processed by the LPS macros. The operations now in progress, filling the second buffer from the line and writing the first buffer to the disk, cause the following possible interrupts.

1. Channel end/device end from the disk indicating the Disk Write operation is complete. Control passes to the Disk End Appendage routine.
2. PCI indicating another full buffer has been received.
3. Channel end/device end from the line indicating an EOB was received from the terminal. Control passes to the Line End Appendage routine.
4. Channel end/device end/unit exception from the line indicating an EOT was received from the line. Control passes to the Line End Appendage.

In this example, it is assumed that the channel end/device end from the disk operation occurs first and the others follow in order given.

DISK INTERRUPT (RECEIVING)

The Disk End Appendage is entered as the result of a disk operation. This interrupt is used to free the message-filled buffer and to initiate for another disk or read operation.

- Disk End Appendage routine

1. Place the disk I/O QCB (effective Qwait) on the ready queue to initiate another disk operation if one is stacked. (Assume none.)
2. Post buffer to available buffer queue.

Enter Available Buffer subtask

- Available Buffer routine

1. Find and remove BRB (from PCI interrupt) from active BRB element chain.
2. Branch to Buffer BRB routine.

- Buffer BRB routine

1. Assign empty buffer for receiving.
2. Post empty buffer to LPS queue.

Now that a buffer is available, it can be assigned to a BRB and used to continue reading the message. Note that the basic structure of the channel program has been set, therefore all that is needed is to complete the CCW. Figure 17 shows the normal path of a buffer. Actually the buffer is chained to the ready queue; however, the diagram shows the logical association between the buffer and function to be performed.

Enter Message Control task

- LPS Control routine

1. Set up registers for Activate routine.
2. Branch to Activate routine (empty buffer).

- Activate routine

1. Prepare CCW for entire buffer in BRB.
2. Clear low-order bit from TIC command in previous BRB.

- Return is to LPS Control routine

1. Issue a SVC Qwait for buffer in LPS queue.

DISK INTERRUPT--CHECKPOINT WRITE

Enter Checkpoint/Restart Routine

1. If there are errors, a WTO macro is issued for a message. (Assume no errors.)
2. If the complete record has not been written, another disk operation is started.

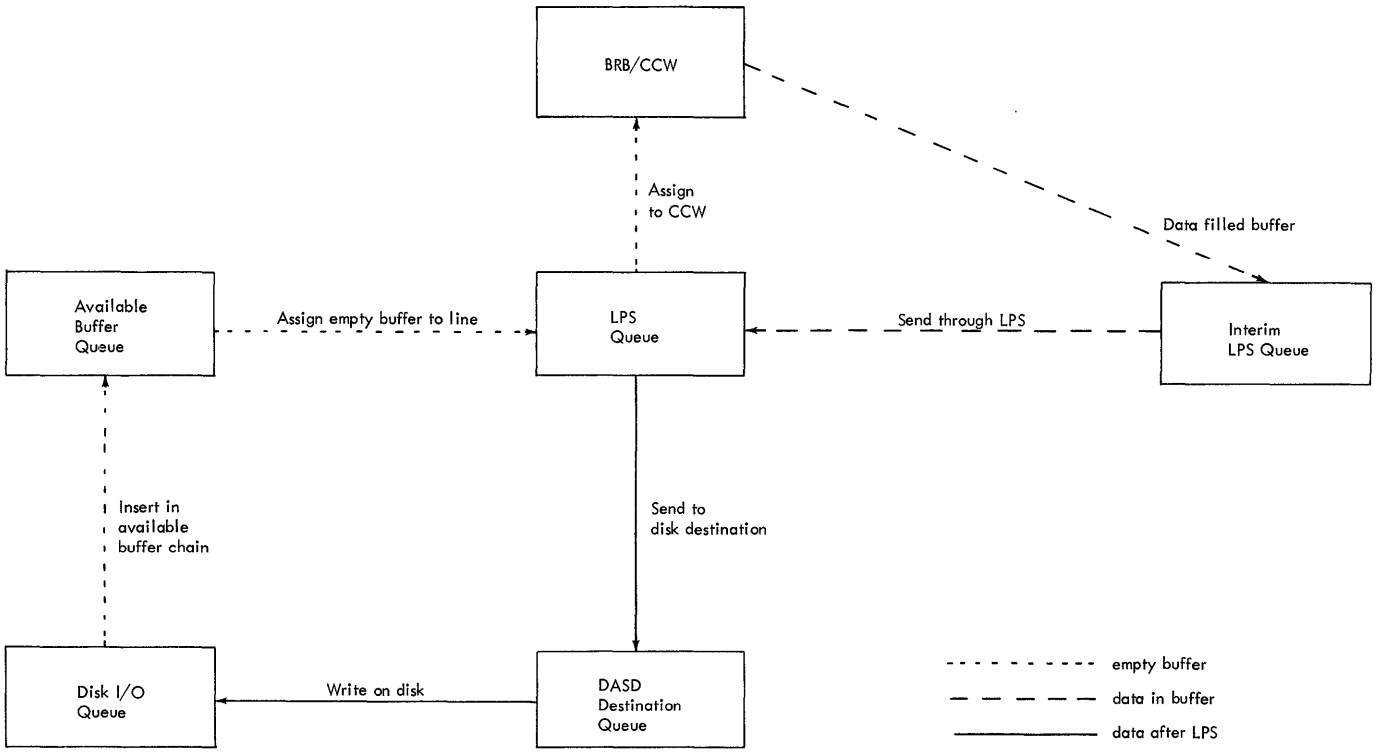


Figure 17. Path of a Buffer for Receiving

3. When complete record is written, the control record is written.

4. FREEMAIN is issued to free the checkpoint record.

5. Timer is reset.

6. Dispatch next item on queue.

LINE END INTERRUPT (RECEIVE AN EOB)

The Line End Appendage routine is entered as a result of an EOB indication. The CCW must be set up to read the rest of the buffer.

- Line End Appendage routine

1. Check for errors.
2. Post message-filled buffer to LPS queue.

Enter Message Control task

- LPS Control routine

1. Set up registers for LPS.
2. Branch to LPSTART.

- RCVSEG portion of LPS
- RCVHDR portion of LPS (if header)
- ENDRCV macro instruction

1. Test for end of message (MSTATUS=X'42').

- EOB or EOBLC macro instruction

1. Branch to EOB or EOBLC routine.

- EOB or EOBLC routine

1. Set up "first buffer" and "read continue" flags for Activate routine.
2. Branch to Activate routine.

- LPS macro instructions for error checking of received messages.
- Activate routine

1. Prepare CCW for entire buffer in BRB.
2. Prepare DECB for BTAM Read/Write routine.
3. Branch to BTAM Read/Write routine.

- BTAM Read/Write routine

1. Prepare CCWs to respond to EOB and read portion of buffer that follows EOB.
2. Execute EXCP supervisor call.

- IOS branches to Line SIO Appendage
- Line SIO Appendage routine

1. Move TIC command from BRB to end of BTAM-prepared CCWs.

- IOS issues Start I/O
- LPS Control routine

1. Issue an SVC Qwait for buffer in LPS queue.

LINE END INTERRUPT (RECEIVE WRU SIGNAL ON WTTA LINE)

The WTTA Line Appendage routine is entered as a result of a WRU indication. If EOM is different from WRU, the CCW must be set up to read the rest of the buffer.

- WTTA Line Appendage routine

1. Check for errors.
2. If this is the first buffer, the requested identification exchange is performed. On completion, restart the Read CCW. If this is not the first buffer, post it to the LPS queue, and set the "WRU" flag in the LCB.

Enter Message Control Task

- LPS Control routine
- RCVSEG portion of LPS
- ENDRCV macro instruction

1. Test for end of message (MSTATUS=X'42').
2. Branch to EOB routine.

- EOB routine

1. Set up "first buffer" and "read continue" flags for Activate routine.
2. Branch to Activate routine.

- Activate routine

1. Prepare CCW for entire buffer in BRB.
2. Prepare DECB for BTAM Read/Write routine.
3. Branch to BTAM Read/Write routine.

- BTAM Read/Write routine

1. Prepare CCW for ID Exchange and read portion of buffer including WRU.
2. Execute EXCP supervisor call.

- IOS branches to line SIO Appendage.
- Line SIO Appendage routine

1. Move TIC command from BRB to end of BTAM-prepared CCWs.

- IOS issues Start I/O.
- LPS Control routine

1. Issue an SVC Qwait for buffer in LPS queue.

LINE END INTERRUPT (RECEIVE EOT--RECEIVE EOT/EOM ON WTTA LINES)

The Line End Appendage is entered as a result of an EOT indication.

- Line End Appendage routine

1. Check for errors.
2. Post buffer to LPS queue.

Enter Message Control Task

- LPS Control routine

1. Set up registers for LPS.
2. Branch to LPSTART.

- RCVSEG portion of LPS
- RCVHDR portion of LPS (if header)
- ENDRCV portion of LPS
- EOB or EOBLC macro instruction

1. Branch to EOB or EOBLC routine.

- EOB or EOBLC routine

1. Test for EOT.
2. Return to LPS macro instruction.

- LPS macro instructions to perform error checking
- POSTRCV macro instruction

1. Branch to Cleanup routine.

- Cleanup routine

1. Issue a SVC Qpost to post buffer to DASD process or destination queue.

Note: Enter DASD Destination routine and disk I/O subtask as already explained under the PCI Interrupt section (receiving all buffers except first). Upon returning to the Cleanup routine the following functions have been performed:

1. Allocated disk location for text segment.

2. Placed necessary linkages in text prefix.

3. Initiated Disk Write operation for last buffer.

Return to Cleanup routine

- Cleanup routine (continued)

1. Issue an SVC Qpost to post any assigned but unused buffers to the available buffer queue.
2. Branch to Free BRB routine.

- Free BRB routine

1. Issue an SVC Qpost to post BRBs to inactive BRB queue. (If BRB is in the active BRB queue it is not posted. A flag is set so that when this buffer is available it is not assigned and the BRB is posted to the inactive BRB queue.)
2. Issue an SVC Qpost to post the LCB to itself to free the line.

The LCB contains the STCB for either the Receive or Send Scheduler depending upon the priority of sending and receiving. The following priorities may be specified for nonswitched lines.

1. Receive over send: Messages are sent only during the polling interval delay. If no polling delay is specified, no messages are sent.
2. Receive equal to send: For WTTA lines, messages are sent if an EOT signal has been received. For all other lines, messages are sent at the end of the polling list. All messages queued for that line are sent before polling is reinitiated.
3. Send over receive: Messages are sent at EOT time, at the end of polling list, and after a negative response to poll.

The STCB contains the address of the scheduler subtask in the link field. When the LCB is posted to itself and is subsequently dispatched, the STCB is activated so that the Send Scheduler routine (assume line is free to send) is entered.

SENDING

Sending is initiated when a line is free, and a full message has been received. The message must be read into buffers and then the header rewritten on the disk with the "message sent" flag set. (See Figure 18.) The buffers are then routed through the send LPS.

Enter Send Scheduler subtask

- Send Scheduler routine

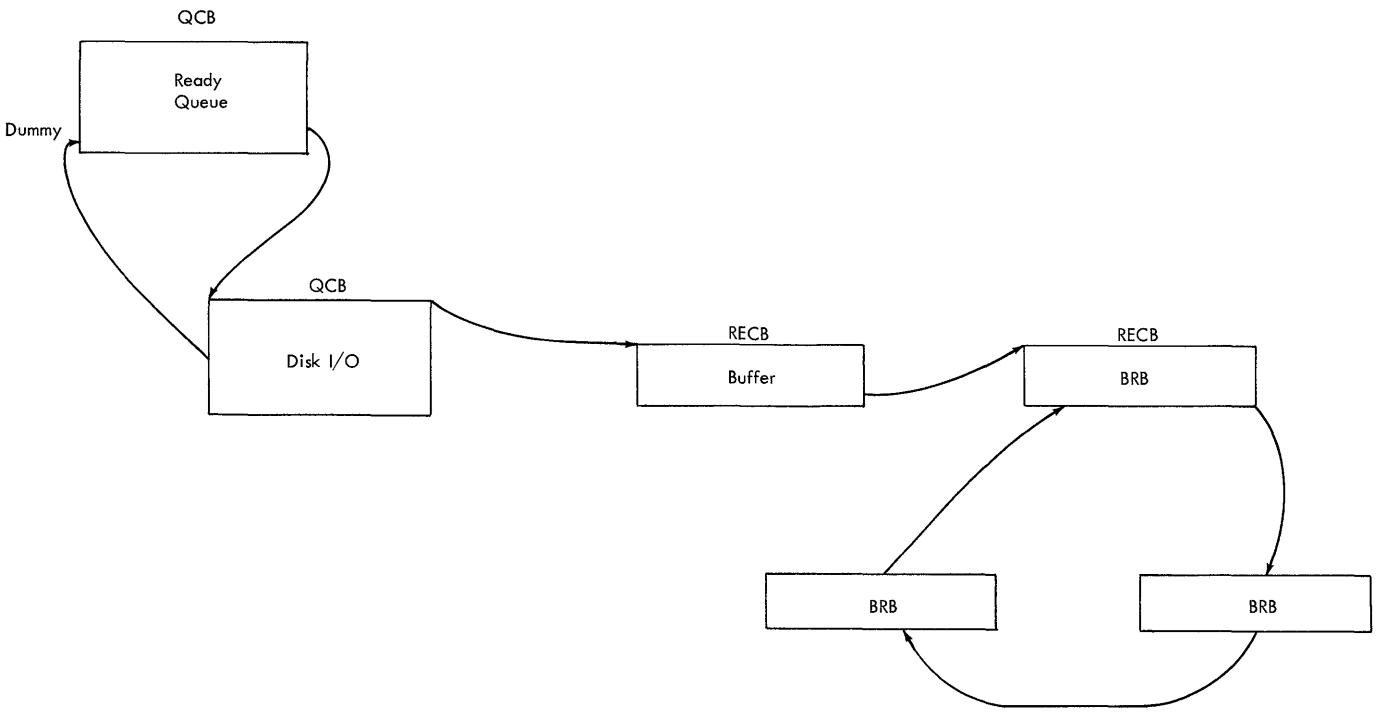


Figure 18. Ready Queue at Sending Time

1. Test for full message in queue.
2. Branch to BRB-Ring routine.

- BRB-Ring routine

1. Build ring of BRBs used for dynamic buffer allocation. (BRBs are obtained from the inactive BRB queue.)
2. Post first BRB to disk I/O queue.

Return to Message Control task

- LPS Control routine

1. Issue an SVC Qwait for buffer in LPS queue.

Enter Disk I/O subtask

- Disk I/O routine (read)

1. Assign buffer from available buffer queue for Disk Read. (If no buffer available, BRB is posted to active BRB queue.)
2. Put buffer on disk I/O queue before BRB (BRB is a request to read buffer.)
3. Convert relative record number to actual DASD address.
4. Execute EXCP supervisor call.

DISK INTERRUPT (SENDING - HEADER)

The Disk End Appendage is entered as a result of a disk operation. This interrupt is used to initiate the writing of the buffer back on this disk.

- Disk End Appendage routine (read)

1. Assign sequence number and set "message sent" flag in prefix.
2. Return to IOS to rewrite buffer on disk.

DISK INTERRUPT (SENDING - ALL BUFFERS)

The Disk End Appendage is entered as a result of a disk operation. Note that the buffer containing the header enters the Disk End Appendage twice (read, rewrite). Now that the header has been written back on the disk the message-filled buffer can be sent through the send LPS. This interrupt also provides the opportunity to initiate the reading of the next buffer from the disk.

- Disk End Appendage routine

1. Post (effective) buffer to LPS queue.
2. Set up next BRB to read next segment of message.
3. Turn off the "send" bit so that the buffer can go through send LPS.
4. Post next BRB to disk I/O queue if available buffer for read.

Enter Disk I/O subtask

- Disk I/O routine (read)

1. Assign buffer from available buffer queue for read. (If no buffer is available, BRB is posted to active BRB queue.)
2. Put buffer on disk I/O queue ahead of BRB (BRB is a request to read buffer).
3. Convert relative record number to actual DASD address.
4. Execute EXCP supervisor call. Since this routine was entered through an appendage, an EXCP may not be able to be executed. If the disk is idle, a SIO element (STARTIO) is posted to the LPS queue. The LPS Control routine would then issue the EXCP.

Enter Message Control task

- LPS Control routine

1. Set up registers for LPS.
2. Branch to LPSTART.

- SENDHDR macro instruction

1. Test for complete "message sent," "serviced" bit MSTATUS=X'10'. (Assume complete message not sent).
2. Branch to header portion of LPS.

- SENDHDR portion of LPS (if header)

- ENDSEND macro instruction

1. Branch to Activate routine.

- Activate routine

1. Prepare CCW for entire buffer in BRB.
2. Indicate "message sent" flag in prefix.
3. Prepare DECB for BTAM Read/Write routine (first buffer).

   For all buffers except first:

4. Clear low-order bits from TIC command in previous BRB.
5. Set "PCI" flag in CCW.
6. Branch back to LPS Control routine.

- BTAM Read/Write routine

1. Prepare CCWs for terminal selection and writing first segment.
2. Execute EXCP supervisor call.
3. IOS branches to Line SIO Appendage

- Line SIO Appendage routine

1. Move TIC command from BRB to end of BTAM-prepared CCWs.
2. IOS issues Start I/O

The sequence of Disk End Appendage, disk I/O subtask, and message control task is repeated for each buffer. For the last buffer the BRB is not posted to the disk I/O queue, but the disk I/O QCB is chained to the ready queue to request another operation.

Return to LPS Control routine

- LPS Control routine

1. Issue an SVC Qwait for buffer in LPS queue.

PCI INTERRUPT (SENDING)

The PCI Appendage is entered as a result of a "PCI" flag set in the CCW for every buffer except the first.

- PCI Appendage routine

1. Post (effective) preceding BRB to active BRB queue with low priority.
2. Post (effective) buffer to available buffer queue.

Enter Active BRB subtask

- Active BRB routine (low priority)

1. Chain BRB into active BRB element chain.

Enter Available Buffer subtask

- Available Buffer routine

1. Find and remove BRB (from PCI interrupt) from active BRB element chain.
2. Test if valid or idle BRB. When there is no more to read, the buffer is placed in the available buffer chain and the next item is dispatched.
3. Branch to Buffer BRB routine.

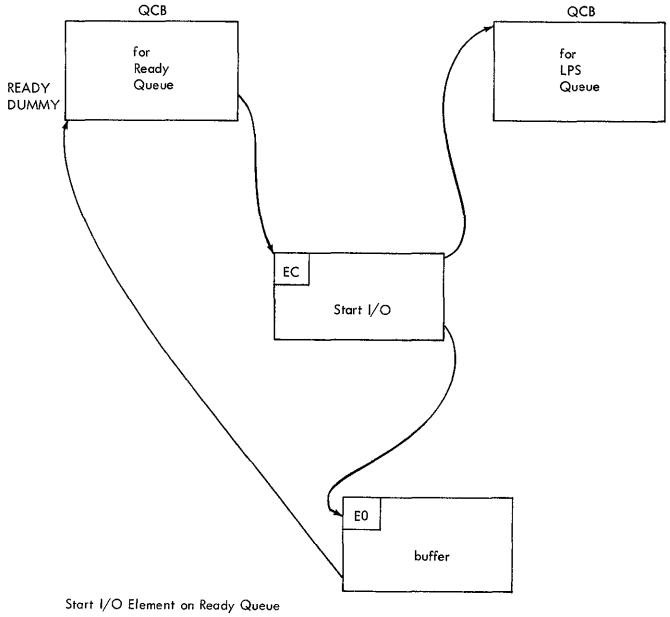
- Buffer BRB routine

1. Reserve buffer for Disk Read.
2. Post BRB to disk I/O queue.

Enter Disk I/O subtask

- Disk I/O routine (read)

1. Assign buffer from available buffer queue for Disk Read.
2. Put buffer on disk I/O queue ahead of BRB (BRB is a request to read buffer).
3. Convert relative record number to actual DASD address.
4. Post disk request element to LPS queue, if disk is idle (assume true for this case). Execute EXCP supervisor call, if disk is not idle. The Start I/O element is the CCWs created by Disk I/O routine for reading the next segment.



Start I/O Element on Ready Queue

Enter Message Control task

- LPS Control routine

1. Issue EXCP supervisor call for disk.

Disk End Appendage is same as explained under Sending - All Buffers.

LINE END INTERRUPT (SENDING - EOB)

The Line End Appendage is entered as a result of an EOB indication.

- Line End Appendage routine

1. Check for errors.
2. Return to IOS to read EOB.

LINE END INTERRUPT (SENDING - RESPONSE TO EOB)

The Line End Appendage is entered as a result of a response to an EOB.

- Line End Appendage routine

1. Check for errors.
2. Post buffer to LPS queue.

Enter Message Control task

- LPS Control routine

1. Set up registers for LPS.
2. Branch to LPS.

- SENDHDR macro instruction

1. Test for complete message sent.
2. Branch to the macro instructions following ENDSEND, as a complete message has been sent (EOB).

- EOB or EOBLC macro instruction

1. Branch to EOB or EOBLC routines.

- EOB or EOBLC routine

1. Set up "first buffer" and "write continue" flags for Activate routine.
2. Branch to Activate routine.

- Activate routine

1. Prepare CCW for entire buffer in BRB.
2. Prepare DECB for Read/Write routine.

- Read/Write routine

1. Prepare CCW to write portion of buffer that follows EOB.
2. Execute EXCP supervisor call.

- IOS branches to Line SIO Appendage

- Line SIO Appendage routine

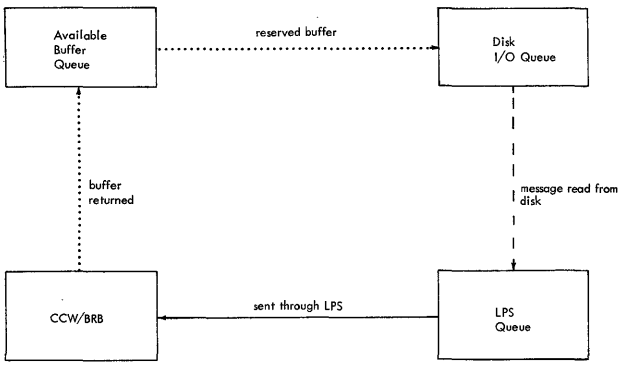
1. Move TIC command from BRB to end of prepared CCW.

- IOS issues Start I/O

Return to LPS Control routine

- LPS Control routine

1. Issue an SVC Qwait for buffer in LPS queue.



Path of Buffer for Sending

LINE END INTERRUPT (SEND EOB/EOT)

The Line End Appendage is entered as a result of an EOT indication. Now the buffer is ready for the send LPS. Also the EOT indicates that all BRBs and the line can be freed.

- Line End Appendage routine

1. Check for errors.
2. Post buffer to LPS queue.

Enter Message Control task

- LPS Control routine

1. Set up registers for LPS.
2. Branch to LPSTART.

- SENDHDR macro instruction

1. Test for complete message sent (MSTATUS=X'10').
2. Branch to the macro instructions following ENDSEND, as a complete message has been sent (EOB).

- EOB or EOBLC macro instruction

1. Branch to EOB or EOBLC routine.

- EOB or EOBLC routine

1. Test for EOT following EOB.
2. Return to LPS since line interrupt is for EOT.

- LPS macro instruction for error checking

- POSTSEND macro instruction

1. Branch to Cleanup routine.

- Cleanup routine

1. Issue an SVC Qpost to post the buffer to available buffer queue.
2. Issue an SVC Qpost to post BRBs to inactive BRB queue. (If BRB is in the active BRB queue, it is not posted. A flag is set so that when a buffer is available it is not assigned and the BRB is posted to the inactive BRB queue.)
3. Issue an SVC Qpost to post the LCB to itself.

Enter Send Scheduler subtask

- Send Scheduler routine

1. Test for full message in queue.
2. Since no messages are now in the queue, the Send Scheduler removes the STCB from the line and places it back in the destination line QCB's STCB chain.

The line (LCB) would now be free to execute the next STCB on its chain, which may be the Receive Scheduler or another Send Scheduler for another terminal on its line.

Enter Receive Scheduler (If send and receive have equal priority)

Cycle now complete.

MESSAGE PROCESSING

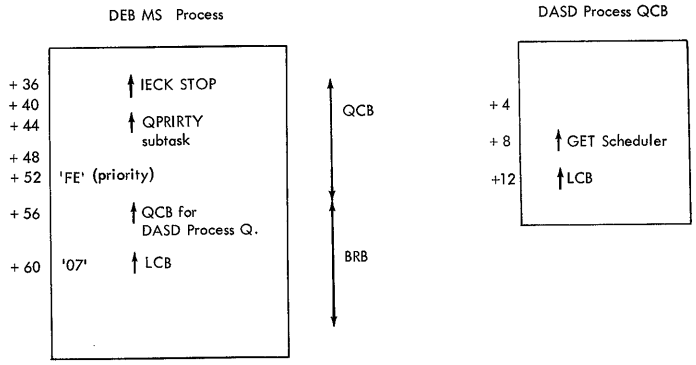
The procedure for routing buffers to a message processing program before a GET has been issued is similar to the description in the Receiving section. The only difference is that the messages are posted to the DASD process queue and the GET Scheduler is entered, which branches to the DASD destination routine. Prior to the first GET the incoming buffers accumulate on the DASD process queue.

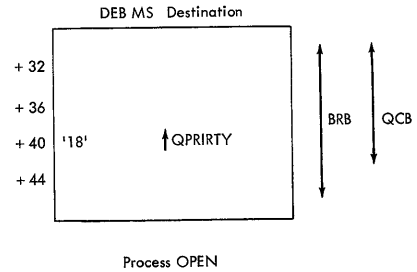
Enter Message Processing task

1. Open process queues.

• Open Process Queue routine

1 Build DEB (144 bytes).
2. Build chain of message processing DEBs.
3. Initialize BRB and QCB in DEB.
4. Load Get and/or Put modules.





Process OPEN

Return to Message Processing task

The first GET is to initialize the process of reading the buffers from the disk. No buffers could be queued to the MS process queue until this time because the message processing queue may not have been opened.

1. Issue GET.

• Get routine

1. Test for message in queue (if none, exit to EODAD).
2. Issue an SVC Qpost to post the preceding buffer to return buffer queue (first time dummy buffer in BRB of process DEB is used).

Enter Return Buffer subtask

• Return Buffer routine

1. Make BRB eligible for reading into MS process queue.
2. Branch to Get Scheduler routine.

• Get Scheduler routine (special entry)

1. Get address of DASD process queue.
2. Test to see if BRB is eligible for a read MSTIC=3. (assume it is eligible)
3. Set the relative record number of the header segment on the DASD process queue in BRB.
4. Indicate disk operation for buffer in BRB (MSTATUS=9).
5. Post BRB to disk I/O queue for read.

Enter Disk I/O subtask (read)

• Disk I/O routine

1. Test for buffer available (assume available)
2. Assign buffer from available buffer queue for Disk Read.
3. Put buffer on disk I/O queue ahead of BRB. (BRB is requested to read buffer.)
4. Convert relative record number to actual DASD address.
5. Execute EXCP supervisor call.

Return to Get routine

1. Issue an SVC Qwait for a buffer.

If the MS process queue had a message, this wait would be satisfied. However to illustrate a complete cycle, the disk end procedure follows. The disk operation replenishes the MS process queue depleted by a GET (if there is a buffer in the DASD process queue). Therefore the disk I/O operation overlaps with the processing in the user's processing program.

DISK INTERRUPT (FIRST BUFFER - HEADER)

The Disk End Appendage is entered as a result of a disk operation.

Disk End Appendage routine (read)

1. Indicate message sent and assign sequence number.
2. Return to IOS indicating that Start I/O is for a rewrite to write the message back on disk.

DISK INTERRUPT (REWRITE)

The Disk End Appendage is entered as a result of a disk operation.

- Disk End Appendage routine (BRB is still a request to read a buffer).

1. Remove BRB and buffer from disk I/O queue.
2. Put buffer in MS process queue.
3. Test for more space in MS process queue. (Assume more space.)
4. Set up for new Disk Read to fill MS process queue.
5. Post BRB to disk I/O queue to cause the reading of the next segment.

The wait is now satisfied for a buffer in the MS process queue.

Return is to the Get routine

- Get routine (continued)

1. Move buffer to work area.
2. Return is made to the message processing program.

For Get Message and Segment if the buffer is empty or it is not end of message, another buffer is requested.

Enter Message Processing task

1. Execute modifying and examining macro instructions.

After the first GET has been issued, then the MS process queue can continue to be filled. If a message is posted to the process queue after the first GET and there is space in the MS process queue, the buffer is put in the MS process queue without actually doing the Disk Read. (See Figure 19.) This procedure is initiated when the Cleanup routine posts a buffer to the DASD process queue as follows.

Enter the Get Scheduler subtask (activated by posting the buffer to the DASD process queue)

- DASD Destination routine

1. Assign direct access location.
2. Reserve and record location of direct access space for next message.
3. Post (effective) buffer to disk I/O queue for write.

- Get Scheduler routine

1. Test for EXPEDITE (assume not EXPEDITE).

   If EXPEDITE, the message is not put on the disk but is put directly into the MS process queue.

2. Test for space in MS process queue. (Assume space.)

3. Test for disk address in BRB. (Assume disk address is the same as for write in DASD destination routine. This BRB is in the active BRB queue as a result of the post to the disk I/O queue by Disk End Appendage.)

4. Post BRB to disk I/O queue for read. (Second element on disk I/O queue, BRB, is now a request to read first element.)

Enter Disk I/O subtask

- Disk I/O routine (write)

1. Convert relative record number to actual DASD address.
2. Execute EXCP supervisor call.

DISK INTERRUPT

The Disk End Appendage is entered as a result of a disk operation.

- Disk End Appendage routine (write)

1. Remove BRB and buffer from disk I/O queue.
2. Put buffer in MS process queue.
3. Test for more space in MS process queue. (Assume space.)
4. Set up for disk read.
5. Post BRB to disk I/O queue. (Continue to fill up MS process queue.)

Enter Message Processing task (when Message Control task enters a WAIT state)

1. Issue GET (not first time)

- Get routine

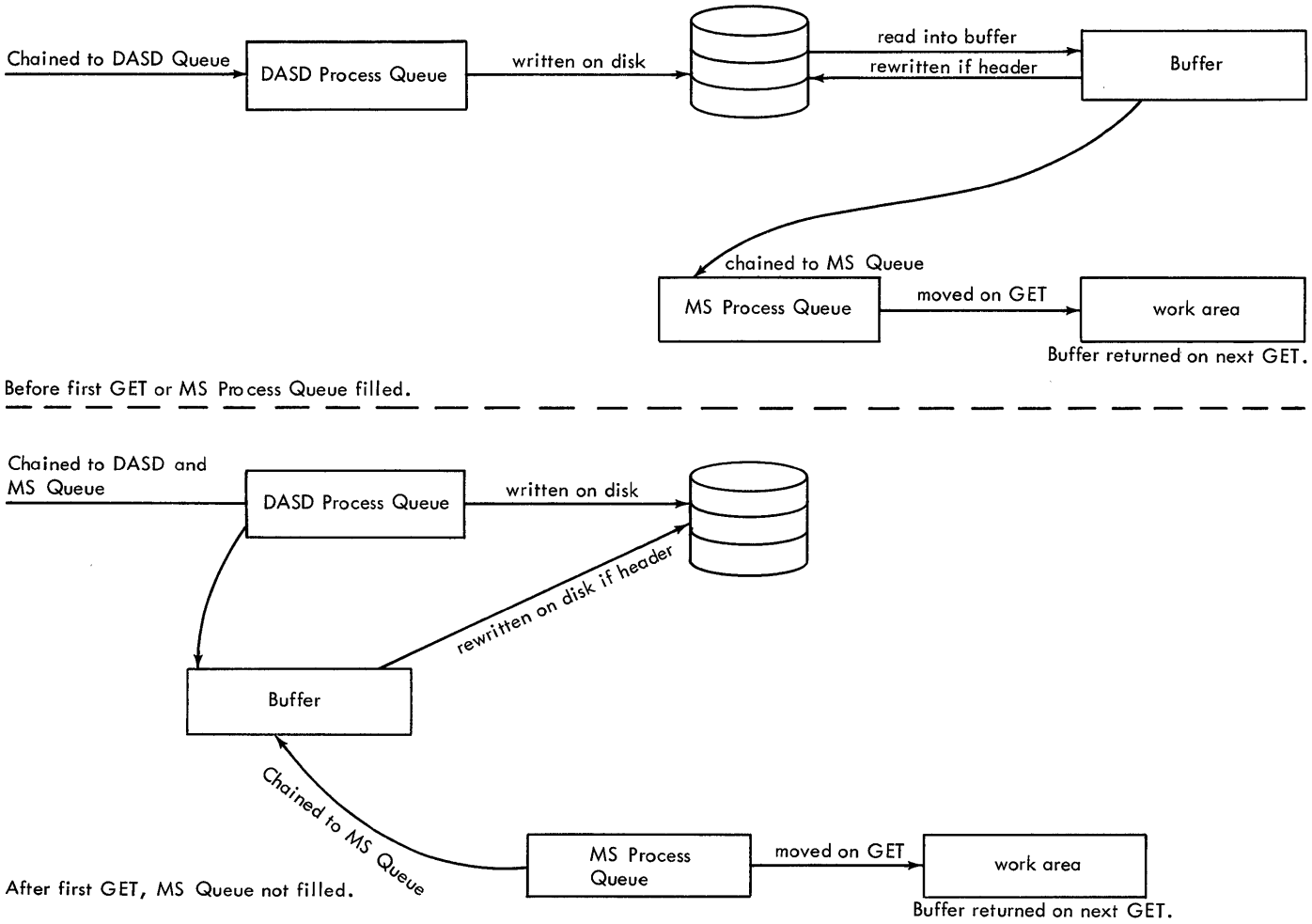


Figure 19. Queuing in Message Processing

1. Test for message in DASD process queue.
2. Post preceding buffer to return buffer queue.

Enter Return Buffer subtask

- Return Buffer routine (not first time)

1. Make BRB eligible for Disk Read.
2. Post (effective) buffer to available buffer queue.
3. Branch to Get Scheduler routine.

Note: Get Scheduler, Disk I/O, Disk End Appendage, and Message Processing are the same as in posting buffer to process queue after first GET.

Enter Message Processing task

1. Issue PUT macro instruction.

- Put routine

1. Set high priority in BRB in destination queue in DEB.
2. Issue an SVC Qpost to post BRB to active BRB queue.

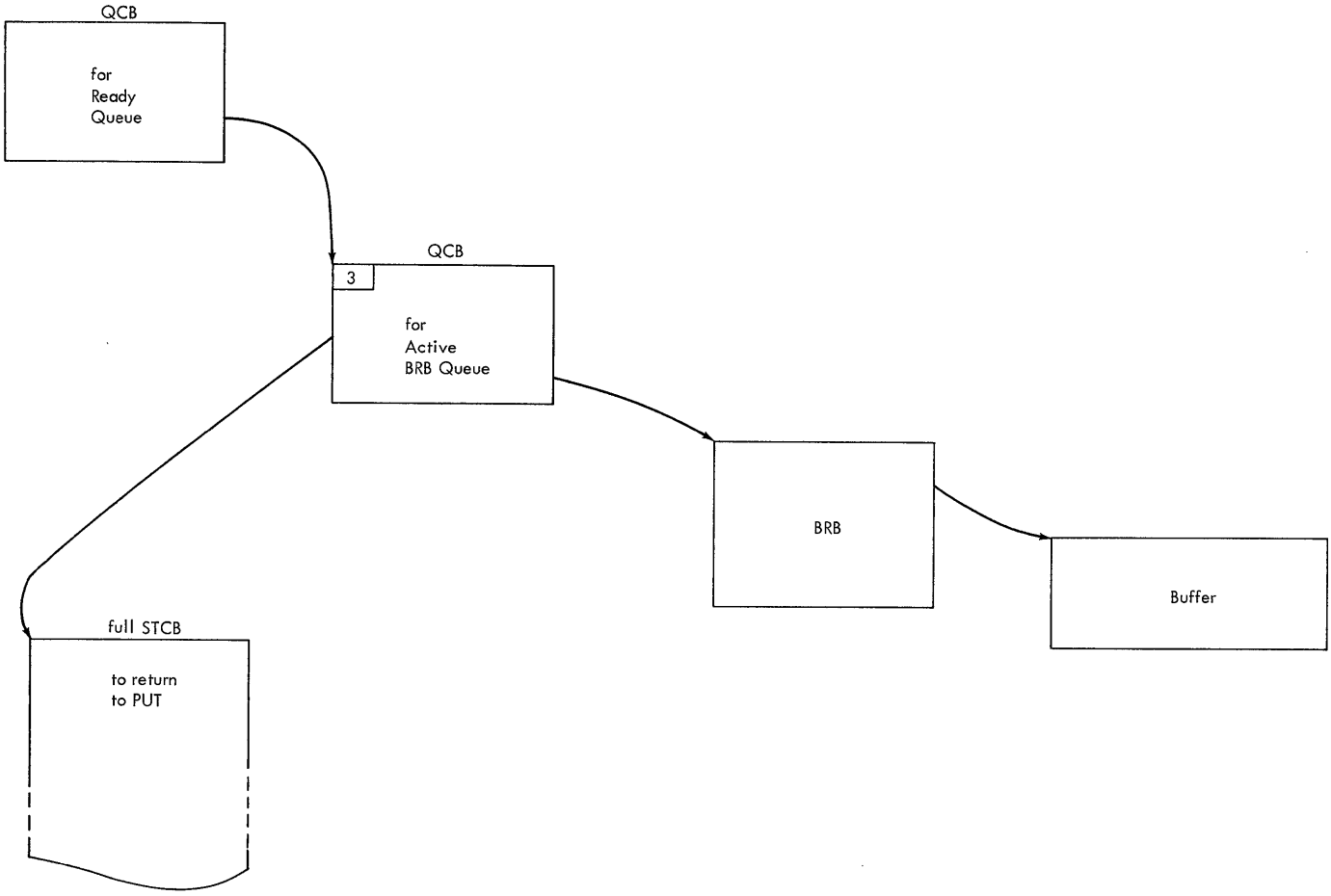


Figure 20. Ready Queue to Obtain Message

Enter Active BRB subtask

- Active BRB routine

1. Assign empty buffer from available buffer queue.
2. Branch to buffer BRB routine.

- Buffer BRB routine

1. Make BRB into QCB for MS destination queue.
2. Exit to Put routine

- Return to Put routine (special entry in supervisory mode)

1. Move message from work area to buffer.
2. Post buffer to MS destination queue.

The STCB for the MS destination queue is QPRIORTY, which inserts the buffer in the queue and dispatches the next item on the ready queue. In Figure 21 the MS destination queue will be removed and the full STCB will be dispatched to return to the Put routine.

- Return to Put routine

1. Issue an SVC Qwait for new filled buffer.
2. Issue an SVC Qpost to post the buffer to DASD destination QCB.

Note: The results of the post to the DASD destination queue are as explained in the section on Receiving. The message is now sent out to the terminal as explained in the section on Sending.

CLOSEDOWN

Enter Message Processing Task

1. Issue CLOSEMC macro instruction.

- Close routine

1. Turn off master receive switch by the move data subtask. This is to prevent further receive operations.

2. Issue a STOPLN macro for all active lines.

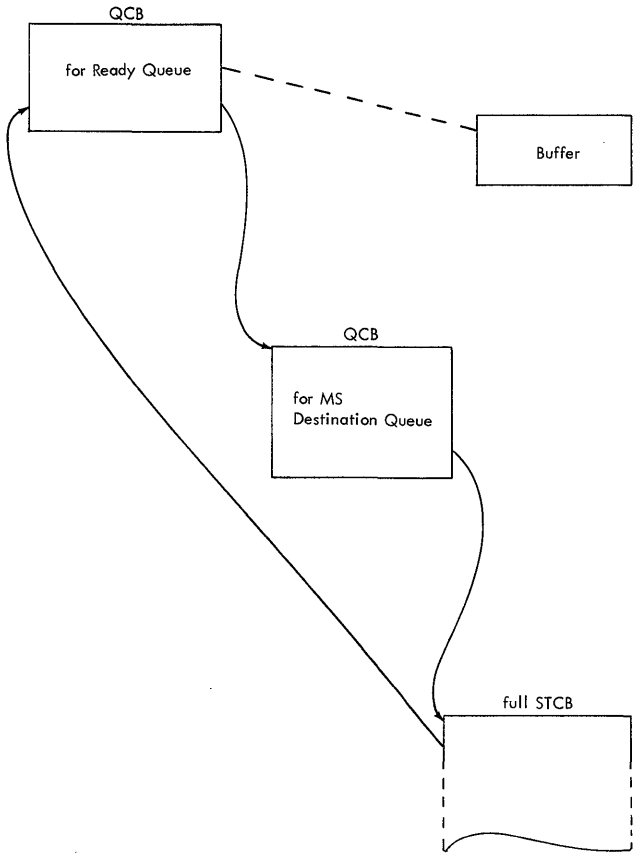


Figure 21. Ready Queue After Obtaining Message

- Stop Line routine

1. Issue a Halt I/O for all dial lines or 2740 terminals (basic or with checking) that are not in active transmission.

2. Issue an SVC Qwait for the LCB. This wait can be satisfied by
   a. End of poll list,
   b. Negative response with "send" flag for the LCB,
   c. Completion of current operation,
   d. Completion of interval delay, which will indicate that the line is free.

3. Return to Close routine.

- Close routine

1. Issue STARTLN macro instruction.

- Start Line routine

1. Set up SAD/Enable or NOP command.
2. Issue a SVC Qpost to post LCB to queue QCB to get in supervisor mode.

- Queue routine (in Line Change Routine)

1. EXCP
2. Dispatch next item on queue. This should be the full STCB to return to the Start Line routine, which returns to the Close routine.

This starts all lines for output only. The master receive switch keeps the input lines inactive. The Close routine returns to the message processing task.

Return to Message Processing task

1. Issue CLOSE process queue macro instruction.

- Enter Close Process Queue routine

1. Remove DEB for each DCB from DEB chain and TCB chain.
2. Test for general closedown. (Assume general closedown. If not, return.)
3. Issue a STOPLN macro instruction.

- Stop Line routine

1. Issue a Halt I/O for all dial lines or 2740 terminals (basic or with checking) that are not in active transmission.
2. Issue an SVC Qwait for the LCB.
3. Return to Close Process Queue routine. (All process queues have been closed.)

- Return to Close Process Queue routine

4. Post request for message control close to LPS queue.

Return to Message Control Task

- LPS Control routine

1. Test for request for closedown.
2. Branch to CLOSE macro instructions.

- Close line group routine

1. Free main storage for LCB.
2. Clear DCB pointers.
3. Purge request for I/O on each line.
4. Disable all dial lines.

- Close DASD routine

1. Clear terminal table from the communications vector table.
2. Post as complete the event control block to return to the message processing task.

The message processing task is now complete and the system does the deallocation to terminate the job.

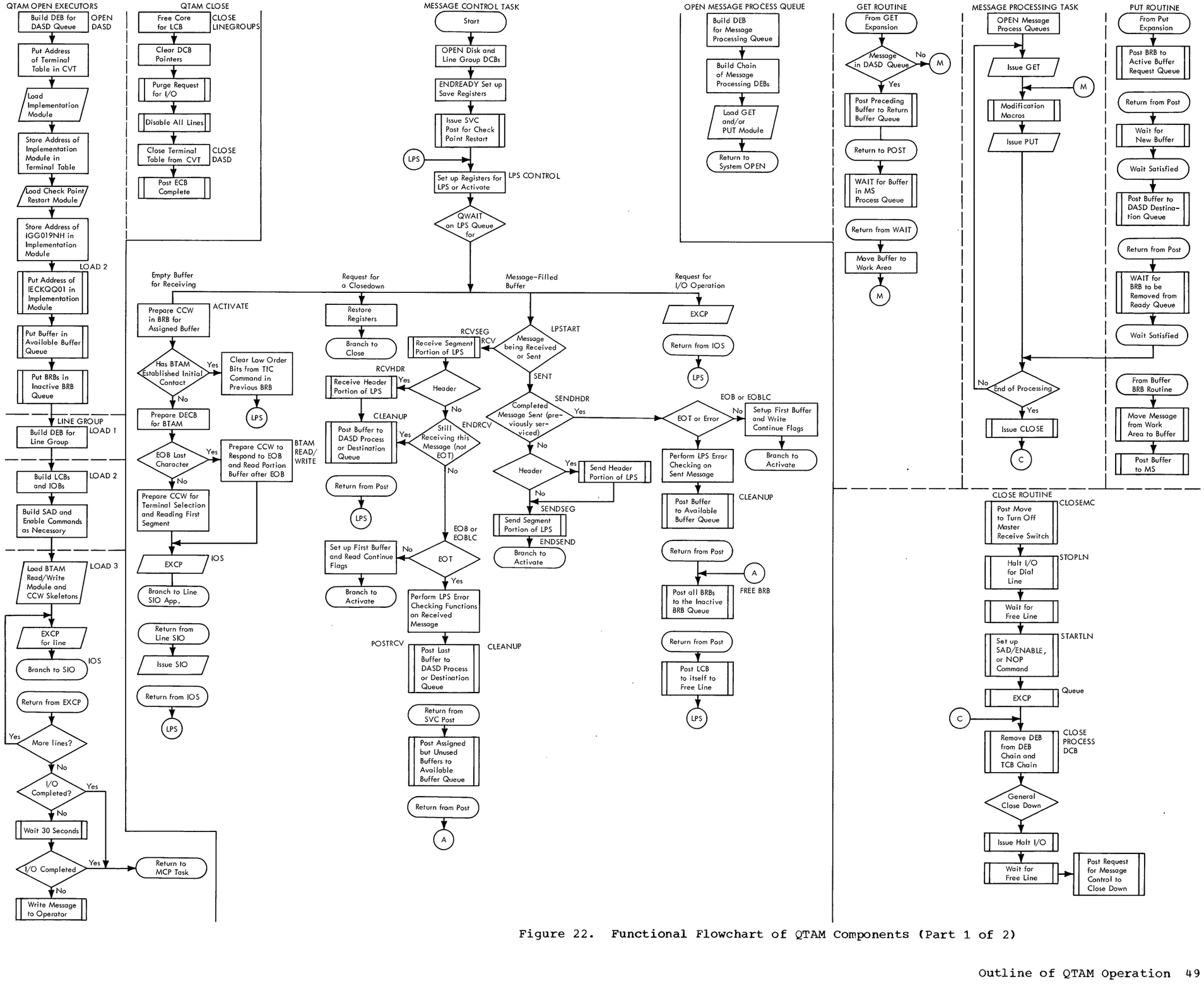


Figure 22. Functional Flowchart of QTAM Components (Part 1 of 2)

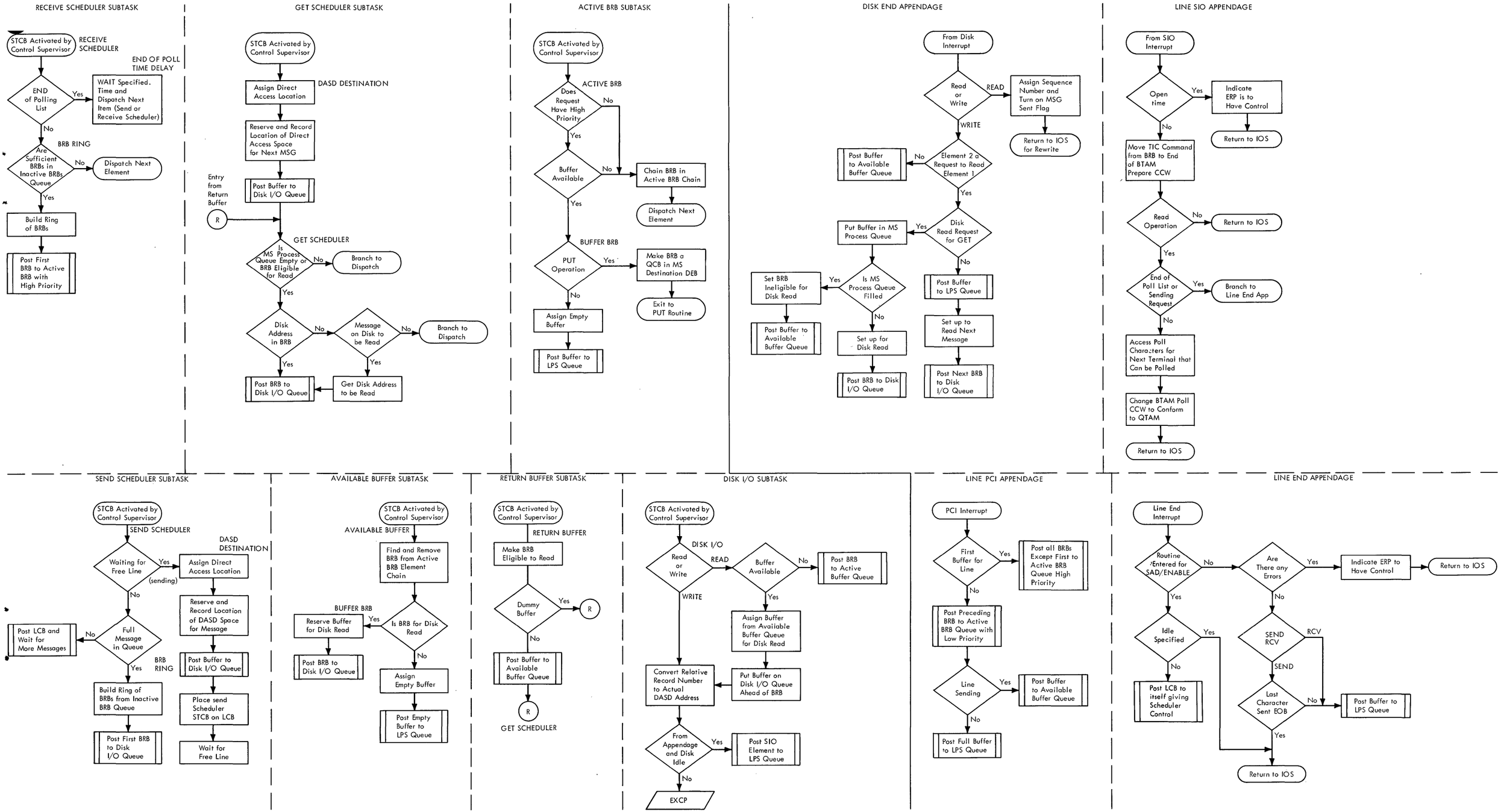


Figure 22. Functional Flowchart of QTAM Components (Part 2 of 2)

QTAM uses the services of BTAM to perform the read and write operations for the system. The BTAM module IGG019MA is the Read/Write routine that QTAM uses under the name IGG019NZ.

QTAM uses BTAM's Device I/O modules for every device type. However, QTAM does not use BTAM's appendages or other routines. QTAM appendages and routines are located in module IGG019NG.

QTAM uses BTAM's channel programs for the operations used by BTAM. The following channel programs are provided for each communications line type, where applicable:

- Read Initial
- Write Initial
- Read Continue
- Write Contiune
- Read Repeat
- Write Conversational
- Write at Line Address
- Write Erase
- Write Negative Acknowledgment

The BRB-Ring routine, LPS Control, Activate, Line SIO Appendage, and Line End Appendage routines modify BTAM's channel programs for QTAM use.

The data event control block required by BTAM is constructed in the Activate routine in the Implementation module, and is labeled LINEDECB (see Appendix B: System Control Blocks).

The following sections will explain BTAM Read/Write routine, BTAM control information for channel program generation, and BTAM channel programs.

BTAM READ/WRITE ROUTINE (IGG019NZ)

The BTAM Read/Write routine is entered by a branch and link from the Activate routine in IGG019NG and acts as an intermediary between the Activate routine and the input/output supervisor. Read/Write performs the following functions.

- Gets the IOB of the LCB from a pointer in the DCB specified by the Activate routine.
- Obtains the Device I/O module, computes, if necessary, the area address and length, and loops on the CCW count until all CCWs have been moved and completed in the channel program area of the LCB for the line.
- Issues an execute channel program SVC, passing control to the I/O supervisor with the address of the IOB as a parameter. The I/O supervisor checks access method and, upon discovering QTAM, gives control to QTAM's SIO Appendage routine.

The CCWs in the Device I/O modules are complete except for the area address and count fields. An index in the second and seventh bytes of the CCW determines which subroutine is used to complete either the area address or the count. If an offset to the normal address is required, this value already exists in the CCW. The subroutines for computing the area address are:

INDEX VALUE

00 TESTLNG - If there is no area address index byte then Read/Write go directly to compute the length.

04 DATAREA - The fourth byte of the CCW is added to the address of the area. The area address is the DECAREA field in the DCB for this Read or Write operation. This subroutine computes the area address for a Read or Write Data or Read Response CCW. If there is a Read Response CCW, then it will read into the first two bytes of the area, and the Read Data CCW that follows will read into the original area address plus two.

08 RESPAREA - The address of the response field in the DCB (DECRESPN) is loaded into the area field of the CCW to read the response to addressing or to text.

0C SPECCHAR - The address of the control characters are provided for the CCW. The control characters are defined as constants at the end of the Device I/O module. The count is not computed.

10 LIST - The number of dial characters is moved to the count field. The address of the dial digits is set in the CCW. This sets up the field to dial a terminal on a switch line.

14 PALIST - The offset in the CCW is picked up to load the polling or addressing pointer, if necessary. The count of characters is added to the

terminal address. The subroutine finds the polling or addressing entry and places the address in the CCW.

18 TWXIDENT - The number of dial characters plus one is added to the list address and then the number of ID characters is moved into the count field. The address of the ID characters is placed in the area address field of the CCW.

1C PA1050D - The address of the addressing characters in the 1050 Dial list is placed in the area address field.

20 DISABLE - The entry is checked to see if this is an Answer list. If it is an Answer list, then an Enable CCW is set up instead of a Dial in the channel program area.

24 AUTO POLL - This subroutine builds the additional CCWs in the channel program in the Device I/O module. The second and third CCWs (poll and TIC) are copied into the channel program as the fifth and sixth CCWs. A TIC CCW, defined in this subroutine, is moved into the channel program as the fourth CCW.

The data address, obtained from the IOBPOLPT field of the IOB, and the count are set in the first poll CCW. The data address, address of first polling character, and the count are set in the second poll CCW.

If there was a permanent error, the first poll CCW is adjusted to start polling at the next terminal. If at the end of the polling list, the first poll CCW is made the same as the second CCW.

If there is a message to be sent, then the STCB in the link field of the LCB requires a channel program that allows a message to be sent to the end of a polling list. For this case, the TIC after the first and second poll CCW are changed to NOP to prevent continuous polling.

If the line is in conversational mode, the "converse mode" flag is turned off. Also the first poll CCW is changed to NOP to cause an immediate interrupt.

28 WTTASNS - This subroutine builds the Sense CCW of the Read Initial channel program for WTTA. The TP Op Code address of the CCW is placed in the area address field of the CCW. The EOT flag in the LCB is reset.

2C WTTATIC - This subroutine builds the transfer address of the TIC CCW. If ID exchange is requested at the beginning of an outgoing message, this will be the address of the third CCW; if not, this will be the address of the sixth CCW.

30 WTTADID - A number of characters equal to the terminal ID minus one is moved into the count field of the CCW. The address of the area reserved by the TERM macro instruction is place in the area address field of the CCW.

34 WTTATID - A number of characters equal to the computer ID is moved into the count field of the CCW, and the address of the computer ID is placed in the area address field of the CCW. If WRU=YES has been coded in the DCB, the CC flag of the CCW is set on.

The subroutines for computing the count field, if not already computed, are:

00 TESTLAST - There is no length to compute; the number of CCWs is checked to see if they have finished building the channel program.

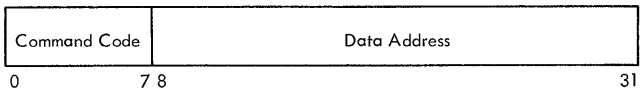
04 DATALNG - The length is picked up from the area length in the DECB for Read or Write Data CCWs.

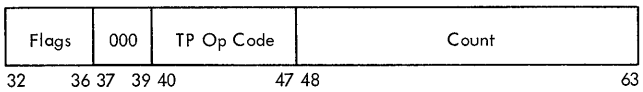
BTAM CONTROL INFORMATION FOR CHANNEL PROGRAM GENERATION

This section describes the form and content of the channel command word (CCW) of the channel program generated by the Read/Write routine.

Channel Command Word

The format of the BTAM channel command word is as follows:





The CCW used within BTAM is identical to that used throughout System/360 except for the addition of an operation code (TP Op Code) in the sixth byte (bits 40 through 47). This byte, which is unused in other environments, has no effect upon channel operations. Bit 0 in the byte is set to one in the last CCW created dynamically for

a channel program. Bit 1 is reserved for use with dynamic buffering. The use of bits 2 through 7 is described in the section on Channel Programs.

DEVICE I/O MODULE: A Device I/O module contains the control information for the generation of channel programs for a given device type. Every device type (e.g., IBM 1050, IBM 1030, 115A, 83B3, etc.) specified for a data set opened in a problem program is represented by a Device I/O module in main storage.

The Device I/O module has four parts as shown in Figure 23:

- A 16-byte table of offsets.
- The offset to the channel command words.
- A table of special characters.
- The channel command words for the channel programs.

The 16-byte table of offsets is at the beginning of the Device I/O module. Each byte contains the binary offset factor used to gain access to the model channel program for an I/O operation. Unused bytes, reserved for future use, contain an offset value of all ones (hexadecimal FF). If access is gained to a reserved byte, control is immediately returned to the calling routine with register 15 containing return a code of 8 to indicate that the requested operation is not valid for the device type involved. The sixteenth byte contains the offset factor for the table of special characters.

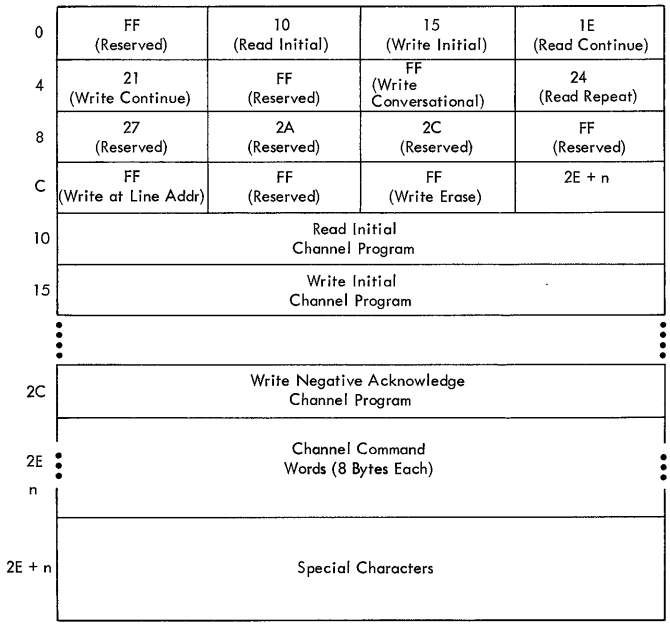


Figure 23. 1050 Nonswitched Device I/O Module

An operation type is associated with each byte in the table of offsets:

Byte	Operation Type
0	Reserved
1	Read Initial
2	Write Initial
3	Read Continue
4	Write Continue
5	Reserved
6	Write Conversational
7	Read Repeat
8	Reserved
9	Reserved
A	Write Negative Acknowledgment
B	Reserved
C	Write at Line Address
D	Reserved
E	Write Erase

Thus, byte 4 in the table of offsets contains the appropriate offset value for any device for which the Write Continue operation is valid. (Otherwise, byte 4 will contain a hexadecimal FF.)

Note: Although the position of the offset byte for an operation is fixed, the actual offset value contained in that byte is not fixed. The offset value is a function of the number of bytes occupied by preceding channel programs, which varies depending on the device involved.

All offset factors are calculated with respect to the first byte following the table of offsets.

Following the table of offsets in the Device I/O module are the offsets to the CCW for the channel program for the device; they are contiguous, beginning immediately after the sixteenth byte of the table.

Following the last offset to the CCW for the channel program in the Device I/O module is the table of special characters for the device (e.g., circle C, circle N, etc.). The field contains the actual hexadecimal representations of the character sequence.

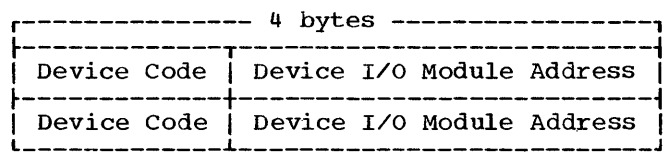
Device I/O modules are loaded into main storage by the QTAM Open routine. The names of the modules are:

Device	I/O Module
IBM 1050 (nonswitched)	IGG019NY
IBM 1060	IGG019NW
IBM 1030	IGG019NV
AT&T 83B3	IGG019NU
Western Union 115A	IGG019NT
IBM 1050 (switched)	IGG019NX*
TWX 33/35	IGG019NS*
IBM 2740 (basic)	IGG019NJ
IBM 2740 (with dial)	IGG019NK
IBM 2740 (with transmit control and checking)	IGG019NL
IBM 2740 (with dial and transmit control)	IGG019NM
IBM 2740 (with dial and checking)	IGG019NN
IBM 2740 (with station control and checking)	IGG019NO
IBM 2740 (with station control)	IGG019NP
IBM 2740 (with checking)	IGG019NQ
IBM 2260	IGG019NR
World Trade Telegraph Adapter	IGG019QA

*This module supports both Auto Call and Auto Answer facilities.

Device I/O Directory

The format of the Device I/O directory is as follows:



The Device I/O directory, contained within module IGG019NZ and initialized by the Open executor when Device I/O modules are loaded, contains the address of each Device I/O module in main storage. The directory allows up to 21 separate devices.

As each DCB is opened, the Device I/O module for the corresponding device is loaded into main storage, unless the required module is already present. The address of the module is placed in the first available directory word, and an index value, representing the position of the entry within the directory, is placed in field DCBDEVTP of the DCB. The index value for the first directory entry is 0; for the second, 1, etc.

Note: The value contained in DCBDEVTP is not a fixed code related to a physical device type.

Before the index value is placed into DCBDEVTP, the contents of that field are placed into the first byte of the directory entry. This data is a fixed code for each device type, specifying physical device type and optional features or mode of operation. This device code is used by the Open routine in determining whether a given Device I/O module is in main storage.

BTAM CHANNEL PROGRAMS

This section describes the BTAM channel programs that are generated by the Read/Write routine, and describes the action of the Read/Write routine during channel program operation. Channel programs are listed by operation types within communication line types.

Each description begins with a graphic representation of the model channel program, as follows:

1. Operation - Command code type with brief description of information being transferred.

2. Flags - Flags that are set in the generated CCW: chain command (CC), chain data (CD), suppress length indication (SLI), etc.

3. TP Op Code - Code carried in bits 2 through 7 in the generated CCW through channel program execution and retrieved by the Channel End Appendage on channel and device end. Bit 0 is on (in addition to the TP Op code) in the last CCW generated in the channel program. Currently defined TP Op codes are:

Code	Definition
01	Disable (only when first CCW of channel program)
	Dial
	Enable
	Prepare
	Write pad characters
02	Write circle D and three circle Cs prior to selection
	Write EOT sequence prior to selection
	Write circle D and 15 idle characters (basic 2740)
	Write response to text
03	Write polling or addressing characters or / space (2740)

Turnaround sequence (TWX)

CPU - ID sequence (TWX)

04 Write space (2740)

Write code (2260)

Write shift (83B3)

Write one (1030)

Write WRU (WTTA).

05 Read response to polling

06 Read response to addressing

07 Read ID response (TWX, WTTA)

Write CPU-ID sequence (WTTA).

08 Write EOA character following addressing (1050, 1030, 2740), or
STX (2260)

10 Write at line address (2260)

Break sequence (WTTA).

0A Read index (Auto Poll only)

11 Read text

Write text

20 Read response to text

88 Sense for 2740 (basic or with checking)

4. Count - Data count set in the generated CCW before execution. Length refers to the buffer length (number of bytes) specified by the Activate routine.

5. Address - Data address set in the generated CCW before execution. "Area" refers to the buffer address specified by the calling program. "Table" refers to the appropriate location in the table of special characters provided in the Device I/O module. "Respn" refers to the DECRESPN field in the DECB. "List" refers to the applicable polling or addressing list entry.

For a description of the subroutines that compute the address and count value for generated CCWs, refer to the section on Read/Write Subroutines.

CHANNEL PROGRAMS FOR AT&T 83B3 SELECTIVE CALLING STATION LINES

Read Initial Channel Program

Operation	Address	Flags	TP-Op Code	Count
1.Write deselect chars (Figs H Ltrs)	Table	CD	02	3
2.Write polling chars	List	CC,SLI	03	2
3.Read response	Area	CD	05	2
4.Read data	Area +2	SLI	11	Length -2

Initiated by the Read/Write routine, the Read Initial channel program places the line in control mode, polls the terminals, and reads the response to polling. If the response is positive, the response will be read into the first byte of the input area. The positive response is followed by the message. Since the Read Response command specifies a count of 2 (with no suppressed length), the positive response followed by the message will reduce the count to zero and data-chain to the Read will continue to read the data until the transmission is ended with an EOT. When a negative response is received on the Read Response, only one byte of data (the negative response) will be read into the message area and channel end/device end occurs (no unit exception). With the "wrong length" flag on and a nonzero data count, there is no data-chaining to the next Read command. Instead, QTAM's channel end detects the polling TP Op code and initializes for the next terminal to be polled by returning to IOS for execution of the CCW beginning with the one containing a 03 TP Op code.

Write Initial Channel Program

Operation	Address	Flags	TP-Op Code	Count
1.Write deselect chars (Figs H Ltrs)	Table	CD	02	3
2.Write addressing chars	List	CD	03	2
3.Write Shift chars (Ltrs)	Table	CC,SLI	04	1
4.Read response	Respn		06	1
5.Write data	Area	SLI	11	Length

The Write Initial channel program places the line in control mode (to allow selection of the terminals) by sending Figs H Ltrs and addresses the terminal by sending two addressing characters. The 83B3

requires a shift character after the addressing characters. The response is read.

CHANNEL PROGRAMS FOR WESTERN UNION PLAN 115A OUTSTATIONS

Read Initial Channel Program

Operation	Address	Flags	TP-Op Code	Count
1. Write deselect characters (Figs H Ltrs)	Table	CD	02	3
2. Write polling characters	List	CC, SLI	03	2
3. Read response	Area	CD	05	2
4. Read data	Area+2	SLI	11	Length-2

The Read Initial channel program initiated by the Read/Write routine places the line in control mode by sending the Figs H Ltrs sequence, polls the terminal with the two polling characters, and reads the response.

The Read Response command specified a data count of 2, with wrong length indication not suppressed, whereas the length of the response is one byte. When a positive response character and the first byte of the message are read under control of the Read Response CCW, it reduces the data count to zero and causes data-chaining to take place. The rest of the message is read under control of the address and count fields of the Read Data CCW. The execution of the Read continues in the channel until an interrupt occurs at the end of transmission. When, on a Read Response, a negative response (one byte) is received, a channel end/device end interrupt occurs. There is no unit exception indication. The data count of 2 for a one-byte polling response character signals wrong length, which suppresses data-chaining and allows BTAM to determine that a negative response was received.

The channel end routine detects the polling restart TP Op code and reinitializes for the next terminal to be polled. Control is returned to IOS for execution of the CCWs beginning with the one containing a 03 TP Op code.

Write Initial Channel Program

Operation	Address	Flags	TP-Op Code	Count
1. Write deselect characters (Figs H Ltrs)	Table	CD	02	3
2. Write addressing chars	List	CC, SLI	03	2
3. Read response	Respn		06	1
4. Write data	Area	SLI	11	Length

The Write Initial channel program, initiated by the Read/Write routine, places the line in control mode (which allows it to be selected), addresses the terminal, and reads the response to addressing.

CHANNEL PROGRAMS FOR IBM 1030 LINES

Read Initial Channel Program

Operation	Address	Flags	TP-Op Code	Count
1. Write deselect characters (3 circle Cs)	Table	CD	02	3
2. Write polling character	List	CC, SLI	03	1
3. Read response	Area	CD	05	2
4. Read data	Area+2	SLI	11	Length-2

The Read Initial channel program places the line in control mode by sending three circle Cs, polls a terminal with one polling character, and reads the response to polling. The Read Response command has a data count of 2 with no suppressed length. Thus, when the response (one byte) is read and it is a positive response, the response will be followed by data. This will reduce the count to zero and cause data-chaining to read the rest of the data until an EOB or EOT is received or the count is zero. If the negative response is received, channel end/device end interrupt occurs with unit exception. There was no data-chaining because of wrong length indication and QTAM reinitializes to poll the next terminal if one was specified in the list.

Read Continue Channel Program

Operation	Address	Flags	TP-OP Code	Count
1. Write positive response and 3 deselect characters	Table	CD	01	4
2. Write 3 circle Cs	Table		11	3

The Read Continue channel program sends a positive response to the previous message block, followed by three circle Cs to put the terminal in control mode. These are followed by three additional circle Cs.

Read Repeat Channel Program

Operation	Address	Flags	TP-OP Codes	Count
1. Write negative response and 3 deselect characters	Table	CD	02	4
2. Write 3 circle Cs	Table	SLI	01	3

The Read Repeat channel program sends a negative response followed by three circle Cs to put the terminal in control mode. These are followed by three additional circle Cs.

Write Initial Channel Program

Operation	Address	Flags	TP-Op Code	Count
1. (Write 3 circle Cs, circle S) Write deselect characters	Table	CD	02	4
2. Write addressing characters	List	CD	03	1
3. Write "1"	Table	CC, SLI	04	Table
4. Read addressing response	Respn	CC	06	1
5. Write circle D	Table	CD	08	
6. Write data	Area+1	CC, SLI	11	Length
7. Read response to LRC	Respn+1		20	1

The Write Initial channel program sends out a circle C and a circle S to deselect the 1030 terminals, transmits a single addressing character followed by a 1, and reads the addressing response into the first byte of the DECRESPN in the DECB. Because multiple addressing is not possible with 1030 lines, the Read Response CCW is command-chained to a Write Circle D CCW to send a circle D before the message. The Circle D CCW is data-chained to write the message. This is followed by a CCW with read the LRC response.

Write Continue Channel Program

Operation	Address	Flags	TP-Op Code	Count
1. Write data	Area	CC, SLI	11	Length
2. Read response to LRC	Respn+1		20	1

Initiated by the EOB or EOBLC routine after a successful Write Initial or Write Continue operation, the Write Continue channel program writes data and then command-chains to read the response to longitudinal redundancy checking. This response is read into the DECRESPN+1, which is the second byte of a two-byte response field in the DECB.

CHANNEL PROGRAMS FOR IBM 1050 LINES

Read Initial Channel Program

Operation	Address	Flags	TP-Op Code	Count
1. Write deselect characters (3 circle Cs)	Table	CD	02	3
2. Write polling characters	List	CC, SLI	03	2
3. Read response	Area	CD	05	2
4. Read data	Area+2	SLI	11	Length-2

The Read Initial channel program initiated by the Read/Write routine places the line in control mode, polls a terminal, and reads the response. (Control mode is that state of the system that allows a terminal to be selected.) The third command (Read Response character) specifies a data count of 2, with wrong length indication not suppressed, while the length of the response character is one byte. Under the existing configuration of BTAM, 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 end of transmission.

2. Negative response. This response causes channel end and device end with unit exception and wrong length record indicated. The QTAM Appendages detect the polling restart TP Op code, reinitialize for the next terminal to be polled, and return control to IOS for execution of the CCWs beginning with the one containing a 03 TP Op code.

Read Continue Channel Program

Operation	Address	Flags	TP-Op Code	Count
1. Write response (circle Y)	Table	CC, SLI	02	1
2. Read data	Area	SLI	11	Length

The Read Continue channel program is initiated by the EOB or EOBLC routine after a successful Read Initial or Read Continue operation; the program writes the response character and command-chains to Read Data.

Read Repeat Channel Program

Operation	Address	Flags	TP-Op Code	Count
1. Write negative response (circle N)	Table	CC, SLI	02	1
2. Read data	Area	SLI	11	Length

The Read Repeat channel program is initiated by the EOBLC routine after a data check occurs during execution of the Read Data command of a Read Initial or Read Continue operation. The program transmits a negative response, and then chains to the second CCW to read data into the main storage area originally specified.

Write Initial Channel Program

Operation	Address	Flags	TP-Op Code	Count
1. Write deselect characters (3 circle Cs)	Table	CD	02	3
2. Write addressing chars	List	CC, SLI	03	2
3. Read response	Respn		06	1
4. Write circle D	Table	CD	08	1
5. Write data	Area	CC, SLI	11	Length
6. Read response to LRC	Respn+1		20	1

The Write Initial channel program, initiated by the Read/Write routine, places the line in control mode, addresses a terminal, and reads the response. Following the Read Response, a circle D is written to the terminal and is followed by the data.

Write Continue Channel Program

Operation	Address	Flags	TP-Op Code	Count
1. Write Data	Area	CC, SLI	11	Length
2. Read response to LRC	Respn+1		20	1

The Write Continue channel program is initiated by the EOB or EOBLC routine after a successful Write Initial or Write Continue operation; the program writes data and command-chains to read the response to longitudinal redundancy checking. The response is read into DECRESPN+1, the second byte of the 2-byte response field in the DECB.

CHANNEL PROGRAMS FOR IBM 1050 DIAL (SWITCHED CONNECTION LINES)

Read Initial Channel Program

Operation	Address	Flags	TP-Op Code	Count
1. Disable	Zero	CC, SLI	01	1
2. Dial	List	CC, SLI	01	Dial List
Enable	Zero	SLI	01	1
3. Write pad characters	Table	CD	01	15
4. Write deselect characters (3 circle Cs)	Table	CD	02	3
5. Write polling characters	List	CC, SLI	03	2
6. Read response	Area	CD	05	2
7. Read data	Area+2	SLI	11	Length-2

The Read Initial channel program initiated by the Read/Write routine disables and then enables the line adapter so that a remote terminal may dial the CPU.

When a terminal dials the CPU, the enable is complete, and 15 pad characters are sent. These are followed by three circle Cs to place the terminal in control mode. The two polling characters are sent. The sixth command (Read Response character) specifies a data count of 2, with wrong length indication not suppressed, while the length of the response character is one byte. Under BTAM, 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 end of transmission.

2. Negative response. This response causes channel end and device end with unit exception and wrong length record indicated. The Channel End routine detects the Read Response to polling TP Op code, reinitializes for the next terminal to be polled, and returns control to IOS for execution of the CCWs beginning with the one containing a 03 TP Op code.

Read Continue Channel Program

Operation	Address	Flags	TP-Op Code	Count
1. Write response (circle Y)	Table	CC, SLI	02	1
2. Read data	Area	SLI	11	Length

The Read Continue channel program is initiated by the EOB or EOBLC routine after a successful Read Initial or Read Continue operation; the program writes the response character and command-chains to read data.

Read Repeat Channel Program

Operation	Address	Flags	TP-Op Code	Count
1. Write negative response (circle N)	Table	CC, SLI	02	1
2. Read data	Area	SLI	11	Length

The Read Repeat channel program is initiated by the EOBLC routine after a transmission error occurred during execution of the Read Data command of a previous Read operation. The program transmits a negative response, and then chains to the second CCW to read data into the main storage area originally specified.

Write Initial Channel Program

Operation	Address	Flags	TP-Op Code	Count
1. Disable	Zero	CC, SLI	01	1
2. Dial	List	CC, SLI	01	List
Enable	Zero	SLI	01	1
3. Write pad characters	Table	CD	01	15
4. Write deselect characters (3 circle Cs)	Table	CD	02	3
5. Write addressing chars	List	CC, SLI	03	2
6. Read response to address	Respn		06	1
7. Write end-of-addressing	Table	CD	08	1
8. Write data	Area	CC, SLI	11	Length
9. Read response to LRC	Respn+1		20	1

The Write Initial channel program initiated by the Read/Write routine disables and dials a terminal. When the remote terminal answers, the pad characters and three circle Cs are sent to place the terminal in control mode. The addressing characters are sent to address the component. This is followed by a circle D and then the data.

Write Continue Channel Program

Operation	Address	Flags	TP-Op Code	Count
1. Write data	Area	CC, SLI	11	Length
2. Read response to LRC	Respn+1	SLI	20	1

After the line connection has previously been established, the Write Continue channel program is initiated by the EOB or EOBLC routine; the program writes data and command-chains to read the response to longitudinal redundancy checking. The response is read into DECRESPN+1, the second byte of the two-byte response field in the DECB.

Write Conversational Channel Program

Operation	Address	Flags	TP-Op Code	Count
1. Write response and deselect character (circle D and circle C)	Table	CD	02	2
2. Write addressing chars	List	CC, SLI	03	2
3. Read response	Respn		06	2
4. Write circle D	Table	CD	08	1
5. Write data	Area	CC, SLI	11	Length
6. Read response	Respn+1		20	1

The channel program transmits a circle D and a circle C with a single CCW. For a discussion of the channel program see the Write Initial Channel Program.

Write Negative Acknowledgment Channel Program

Operation	Address	Flags	TP-Op Code	Count
1. Write circle C	Table	CC, SLI	01	1
2. Disable	Zero	SLI	01	1

The Write Negative Acknowledgment channel program sends a circle C to deselect the remote terminal and then issues a disable to disconnect the line.

CHANNEL PROGRAMS FOR IBM 1060 LINES

Read Initial Channel Program

Operation	Address	Flags	TP-Op Code	Count
1. Write deselect characters (3 circle Cs)	Table	CD	02	3
2. Write polling characters	List	CC, SLI	03	2
3. Read response	Area	CD	05	2
4. Read data	Area+2	SLI	11	Length-2

The Read Initial channel program places the line in control mode by sending three circle Cs, polls a terminal with one polling character, and reads the response to polling. The Read Response command has a data count of 2 with no suppressed length. Thus, when the response (one byte) is read and it is a positive response, the response will be followed by data. This will reduce the count to zero and cause data-chaining to read the rest of the data until an ECB or EOT is received or the count is zero. If the negative response is received, channel end/device end interrupt occurs with unit exception. There is no data-chaining because of wrong length indication and QTAM reinitializes to poll the next terminal if one was specified in the list.

Read Continue Channel Program

Operation	Address	Flags	TP-OP Code	Count
1. Write response and deselect characters (circle Y and 3 circle Cs)	Table	CD	02	4
2. Write 3 circle Cs	Table	CD	01	3

The Read Continue channel program sends a positive response to the previous message block, followed by three circle Cs to put the terminal in control mode. This is followed by three additional circle Cs.

Read Repeat Channel Program

Operation	Address	Flags	TP-OP Code	Count
1. Write deselect characters (3 circle Cs)	Table	SLI	01	3

The Read Repeat channel program sends a negative response and 3 circle Cs are sent to put the terminal in control mode.

Write Initial Channel Program

Operation	Address	Flags	TP-Op Code	Count
1. Write deselect characters (3 circle Cs)	Table	CD	02	3
2. Write addressing chars	List	CC, SLI	03	2
3. Read response	Respn		06	1
4. Write circle D	Table	CD	08	1
5. Write data	Area	CC, SLI	11	Length
6. Read response to LRC	Respn+1	SLI	20	1

The Write Initial channel program, initiated by the Read/Write routine, places the line in control mode, addresses a terminal, and reads the response. Following the Read Response, a circle D is written to the terminal and is followed by the data.

CHANNEL PROGRAMS FOR TTY MODELS 33 AND 35 TWX LINES

Read Initial Channel Program

Operation	Address	Flags	TP-Op Code	Count
1. Disable	Zero	CC, SLI	01	1
2. Enable	Zero	SLI	01	1
3. Write pad characters	Table	CD	01	15
4. Write identification	List	CC, SLI	03	List
5. Read data	Area	SLI	11	Length

The Read Initial channel program, initiated by the Read/Write routine, disables the line in case this was not done previously. The enable latch is set within the line adapter so that the remote terminal may dial the CPU. After the pad characters have been sent, the fourth command writes the identification assigned to the CPU in the polling list for the line. This is followed by the data transmitted by the terminal.

Write Initial Channel Program

Operation	Address	Flags	TP-Op Code	Count
1. Disable	Zero	CC, SLI	01	1
2. Dial	List	CC, SLI	01	List
3. Read identification	List		07	List
4. Write data	Area	SLI	11	Length

The Write Initial channel program, initiated by the Read/Write routine, disables and dials a terminal and, if the identification received was valid, writes the data to the terminal. If the identification was invalid, the channel program is terminated.

After the CPU has read the identification sent from the terminal, an interrupt occurs and the Channel End routine compares the identification supplied by the user in the TERM macro. If an unequal compare results, the addressing bit in the error halfword (bit 12) is set. If an equal identification is received, it is assumed the correct terminal has been contacted and the channel is restarted with the Write Data command.

Write Conversational Channel Program

Operation	Address	Flags	TP-Op Code	Count
1. Write data	Area	SLI	11	Length

The Write Conversational channel program is initiated by QTAM after a successful Write Initial operation.

Write Negative Acknowledgment Channel Program

Operation	Address	Flags	TP-Op Code	Count
1. Write circle C	Table	CC, SLI	01	1
2. Disable	Zero	SLI	01	1

The Write Negative Acknowledgment channel program sends a circle C to deselect the remote terminal and then issues a disable to disconnect the line.

CHANNEL PROGRAMS FOR IBM 2740 COMMUNICATIONS LINES

IBM 2740 BASIC CHANNEL PROGRAMS

Read Initial Channel Program

Operation	Address	Flags	TP-OP Code	Count
1. Write deselect characters (3 circle Cs)	Table	CD	02	3
2. Prepare	Zero	CC,SLI	01	1
3. Read data	Area	SLI	11	Length

The Read Initial channel program places the line in control mode by sending three circle Cs. The Prepare command is sent to condition the control unit to receive a message from a terminal. The Prepare command removes the circle D from the beginning of the message and the count is reduced to zero, causing command-chaining to the Read Data command, which reads the message.

Write Initial Channel Program

Operation	Address	Flags	TP-Op Code	Count
1. Write circle D and 15 idle characters	Table	CD	02	16
2. Write data	Area	SLI	11	Length

The Write Initial channel program sends a circle D and 15 idle characters, and data-chains to the Write Data command to send the message.

Read Repeat Channel Program

Operation	Address	Flags	TP-Op Code	Count
1. Write negative response (circle N)	Table	CC, SLI	02	1
2. Read data	Area	SLI	11	Length

The Read Repeat channel program is initiated by the EOBLC routine after a transmission error occurred during execution of the Read Data command of a Read Initial or Read Continue operation. The program transmits a negative response, and then chains to the second CCW to read data into the main storage area originally specified.

IBM 2740 WITH CHECKING

Read Initial Channel Program

Operation	Address	Flags	TP-OP Code	Count
1. Write deselect characters (3 circle Cs)	Table	CC,SLI	02	3
2. Prepare	Zero	CC,SLI	01	1
3. Read data	Area	SLI	11	Length

The Read Initial channel program places the line in control mode by sending three circle Cs. The Prepare command conditions the control unit to receive a message and then command-chains to the Read command when a character is received. The circle D sent by the transmitting terminal is deleted by the Prepare command.

Read Continue Channel Program

Operation	Address	Flags	TP-Op Code	Count
1. Write response (circle Y)	Table	CC, SLI	02	1
2. Read data	Area	SLI	11	Length

The Read Continue channel program is initiated by the EOB or EOBLC routine after a successful Read Initial operation; the program writes the response character and command-chains to Read Data.

Write Initial Channel Program

Operation	Address	Flags	TP-Op Code	Count
1. Write circle D and 15 idle characters	Table	CD	02	16
2. Write data	Area	CC, SLI	11	Length
3. Read response to VRC/LRS	Respn+1		20	1

The Write Initial channel program sends the circle D to put the terminal in control mode and 15 idle characters to allow terminal motors to get up to speed. This Write command data-chains to the Write Data command, which sends the message and command-chains to the Read Response command.

Write Continue Channel Program

Operation	Address	Flags	TP-Op Code	Count
1. Write data	Area	CC, SLI	11	Length
2. Read response to VRC/LRC	Respn+1		20	1

The Write Continue channel program will write the data and then command-chain to read the response into DECRESPN +1 (VRC/LRC response field in the DECB).

Write Conversational Channel Program

Operation	Address	Flags	TP-Op Code	Count
1. Write circle D	Table	CD	02	1
2. Write data	Area	CC, SLI	11	Length
3. Read response to VRC/LRC	Respn+1		20	1

The Write Conversational channel program first writes a circle D to put the terminal in receive mode, and then data-chains to the next write to send the data. When the count is zero, this command chains to read the VRC/LRC response into the response field of the DECB (DECRESPN+1).

IBM 2740 WITH DIAL

Read Initial Channel Program

Operation	Address	Flags	Tp-Op Code	Count
1. Disable	Zero	CC, SLI	01	1
2. Enable	Zero	CC, SLI	01	1
3. Prepare	Zero	CC, SLI	01	1
4. Read data	Area	SLI	11	Length

Initiated by the Read/Write routine, the Read Initial channel program disables and then enables the line to receive a call. When a call is received, the Enable command chains to the Write Deselect Characters to set the terminal in control mode. The Prepare command conditions the control unit to receive a message. When a character is received, the count goes to zero and the Prepare command chains to read the data. The Prepare command deletes the circle D, which is sent by the depression of the BID key at the transmitting terminal.

Write Initial Channel Program

Operation	Address	Flags	TP-Op Code	Count
1. Disable	Zero	CC, SLI	01	1
2. Dial	List+1	CC, SLI	01	List
3. Write pad characters	Table	CD	01	15
4. Write circle D	Table	CD	02	1
5. Write data	Area	SLI	11	Length

Initiated by the Read/Write routine, the Write Initial channel program disables the line and command-chains to the Dial command to dial the terminal specified in the terminal table. After dialing, the channel program sends 15 pad characters before data-chaining, when the count is zero, to a Write Circle D command, which is sent before the data.

Write Conversational Channel Program

Operation	Address	Flags	TP-Op Code	Count
1. Write circle D	Table	CD	02	1
2. Write data	Area	SLI	11	Length

The Write Conversational channel program sends a circle D, and then data-chains when the count is zero to a Write Data command to send the message.

Write Negative Acknowledgment Channel Program

Operation	Address	Flags	TP-Op Code	Count
1. Write 3 circle Cs	Table	CC, SLI	01	3
2. Disable	Zero	SLI	01	1

To disconnect, the channel program sends three circle Cs to put the terminal in control mode and command-chains to disable the line.

IBM 2740 WITH DIAL AND CHECKING

Read Initial Channel Program

Operation	Address	Flags	TP-OP Code	Count
1. Disable	Zero	CC,SLI	01	1
2. Enable	Zero	CC,SLI	01	1
3. Write deselect characters (3 circle Cs)	Table	CC,SLI	01	3
4. Prepare	Zero	CC,SLI	01	1
5. Read data	Area	SLI	11	Length

Initiated by the Read/Write routine, the Read Initial channel program disables and then enables the line to receive a call. When a call is received, the Enable command chains to the Write Deselect Characters command, which places the line in control mode and is chained to the Prepare command, which conditions the control unit to receive a message. When a character is received, the count goes to zero and the Prepare command chains to read the data. The Prepare command deletes the circle D, which is sent by the depression of the BID key at the transmitting terminal.

Read Continue Channel Program

Operation	Address	Flags	TP-Op Code	Count
1. Write response (circle Y)	Table	CC, SLI	02	1
2. Read data	Area	SLI	11	Length

The Read Continue channel program is initiated by the EOB or EOBLC routine after a successful Read Initial operation; the program writes the response character and command-chains to read data.

Read Repeat Channel Program

Operation	Address	Flags	TP-Op Code	Count
1. Write negative response (circle N)	Table	CC, SLI	02	1
2. Read data	Area	SLI	11	Length

The Read Repeat channel program is initiated by the EOBLC routine after a transmission error occurred during execution of the Read Data command of a previous Read operation. The program transmits a negative response, and then chains to the second CCW to read data into the main storage area originally specified.

Write Initial Channel Program

Operation	Address	Flags	TP-Op Code	Count
1. Disable	Zero	CC, SLI	01	1
2. Dial	List	CC, SLI	01	List
3. Write pad characters	Table	CD	01	15
4. Write circle D	Table	CD	02	1
5. Write data	Area	CC, SLI	11	Length
6. Read response to VRC/LRC	Respn+1		20	1

Initiated by the Read/Write routine, the Write Initial channel program disables the line before dialing the terminal specified in the terminal table. The 15 pad characters are sent to allow the terminal motors to reach the necessary speed before the message is sent to it. Before the data is sent, a circle D is sent to the terminal. After the message is sent the response to VRC/LRC is read into the response field in the DECB (DECRESPN+1).

Write Continue Channel Program

Operation	Address	Flags	TP-Op Code	Count
1. Write data	Area	CC, SLI	11	Length
2. Read response to LRC	Respn+1	SLI	20	1

After the line connection has previously been established, the Write Continue channel program is initiated by QTAM; the program writes data and command-chains to read the response to longitudinal redundancy checking. The response is read into DECRESPN +1, the second byte of the two-byte response field in the DECB.

Write Conversational Channel Program

Operation	Address	Flags	TP-Op Code	Count
1. Write circle D	Table	CD	02	1
2. Write data	Area	CC, SLI	11	Length
3. Read response to VRC/LRC	Respn +1		20	1

The Write Conversational channel program sends a circle D after the line has previously been established. The data is sent and the Write Data command chains to the Read Response CCW.

Write Negative Acknowledgment Channel Program

Operation	Address	Flags	TP-Op Code	Count
1. Write circle C	Table	CC, SLI	01	1
2. Disable	Zero	SLI	01	1

To disconnect, the channel program sends a circle C to put the terminal in control mode and command-chains to disable the line.

IBM 2740 WITH DIAL AND TRANSMIT CONTROL

Read Initial Channel Program

Operation	Address	Flags	TP-Op Code	Count
1. Disable	Zero	CC,SLI	01	1
2. Enable	Zero	SLI	01	1
3. Write pad characters	Table	CD	01	15
4. Write selection chars (/ space)	Table	CC,SLI	03	2
5. Read response	Area	CD	05	2
6. Read data	Area+2	SLI	11	Length-2

The Read Initial channel program initiated by the Read/Write routine disables and sets the enable latch within the line adapter so that the remote terminal may dial the CPU.

After writing 15 pad characters, 3 circle Cs are sent to place the line in control mode. They are followed by the selection characters (/ space) to select the fifth command (Read Response character). The Read Response CCW specifies a data count of 2, with wrong length indication not suppressed, while the length of the response character is one byte. Under BTAM, the effect of this technique is as follows:

1. Positive response. The response character, a circle D caused by the depression of the transmitting terminal's BID key, 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 the end of the transmission.

2. Negative response. Only one byte is received on this response, which causes channel end and device end with unit exception and wrong length record indicated. There is no polling of components or terminals on the 2740 with dial and transmit control; only the sending of the selection characters.

Write Initial Channel Program

Operation	Address	Flags	TP-OP Code	Count
1. Disable	Zero	CC,SLI	01	1
2. Dial	List	CC,SLI	01	List
3. Write pad characters	Table	CD	01	15
4. Write data	Area	SLI	11	Length

The Write Initial channel program initiated by the Read/Write routine disables and then dials a terminal. After writing the pad characters, the channel program sends the data.

Write Conversational Channel Program

Operation	Address	Flags	TP-Op Code	Count
1. Write data	Area	SLI	11	Length

The Write Conversational channel program sends a Write Data command to send the message.

Write Negative Acknowledgment Channel Program

Operation	Address	Flags	TP-Op Code	Count
1. Write circle C	Table	CC, SLI	01	1
2. Disable	Zero	SLI	01	1

To disconnect, the channel program sends a circle C to put terminal in control mode and command-chains to disable the line.

IBM 2740 WITH DIAL, TRANSMIT CONTROL, AND CHECKING

Read Initial Channel Program

Operation	Address	Flags	TP-Op Code	Count
1. Disable	Zero	CC,SLI	01	1
2. Enable	Zero	SLI	01	1
3. Write pad characters	Table	CD	01	15
4. Write selection chars	Table	CC,SLI	03	2
5. Read response	Area	CD	05	2
6. Read data	Area+2	SLI	11	Length-2

The Read Initial channel program initiated by the Read/Write routine disables and sets the enable latch within the line adapter so that the remote terminal may dial the CPU.

After writing 15 pad characters, 3 circle Cs are sent to place the line in control mode. They are followed by the selection characters (/ space) to select the fifth command (Read Response character). The Read Response CCW specifies a data count of 2, with wrong length indication not suppressed, while the length of the response character is one byte. Under BTAM, the effect of this technique is as follows:

1. Positive response. The response character, a circle D caused by the depression of the transmitting terminal's BID key, 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 the end of the transmission.

2. Negative response. Only one byte is received on this response, which causes channel end and device end with unit exception and wrong length record indicated. There is no polling of components or terminals on the 2740 with dial and transmit control; only the sending of the selection characters.

Read Continue Channel Program

Operation	Address	Flags	TP-Op Code	Count
1. Write response (circle Y)	Table	CC, SLI	02	1
2. Read data	Area	SLI	11	Length

After the line connection has previously been established, the Read Continue channel program is initiated by the problem program through the Read/Write routine; the program writes the response character and command-chains to read data.

Read Repeat Channel Program

Operation	Address	Flags	TP-Op Code	Count
1. Write negative response (circle N)	Table	CC, SLI	02	1
2. Read data	Area	SLI	11	Length

The Read Repeat channel program is initiated by the EOBLC routine after a data check occurred during execution of the Read Data command of a Read Initial or Read Continue operation. The program transmits a negative response, and then chains to the second CCW to read data into the main storage area originally specified.

Write Initial Channel Program

Operation	Address	Flags	TP-OP Code	Count
1. Disable	Zero	CC, SLI	01	1
2. Dial	List	CC, SLI	01	List
3. Write pad characters	Table	CD	01	15
4. Write circle D	Table	CD	08	1
5. Write data	Area	SLI	11	Length

The Write Initial channel program initiated by the Read/Write routine disables and then sets the enable latch within the line adapter so that the remote terminal may dial the CPU. After writing the pad characters, a circle D is sent before the message is sent.

Write Continue Channel Program

Operation	Address	Flags	TP-Op Code	Count
1. Write data	Area	CC, SLI	11	Length
2. Read response to VRC/LRC	Respn+1		20	1

The Write Continue channel program will write the data and then command chain to read the response into DECRESPN +1 (VRC/LRC) response field in the DECB.

Write Conversational Channel Program

Operation	Address	Flags	TP-Op Code	Count
1. Write circle D	Table	CD	02	1
2. Write data	Area	CC, SLI	11	Length
3. Read response to VRC/LRC	Respn+1		20	1

The Write Conversational channel program first writes a circle D to put the terminal in receive mode, and then data-chains to the next write to send the data. When the count is zero, this command chains to read the VRC/LRC response into the response field of the DECB (DECRESPN +1).

Write Negative Acknowledgment Channel Program

Operation	Address	Flags	TP-Op Code	Count
1. Write circle C	Table	CC, SLI	01	1
2. Disable	Zero	SLI	01	1

To disconnect, the channel program sends a circle C to put the terminal in control mode and command-chains to disable the line.

IBM 2740 WITH STATION CONTROL

Read Initial Channel Program

Operation	Address	Flags	TP-Op Code	Count
1. Write deselect characters (3 circle Cs)	Table	CD	02	3
2. Write polling character	List	CD	03	1
3. Write space character	Table	CC, SLI	04	1
4. Read response	Area	CD	05	2
5. Read data	Area+2	SLI	11	Length-2

Initiated by the Read/Write routine, the Read Initial channel program places the line in control mode, polls the terminals, with one character followed by a space character, and reads the response to polling. If the response is positive, the response will be read into the first byte of the input area. The positive response is followed by the message. Since the Read Response command specifies a count of 2 (with no suppress length), the positive response followed by the message will reduce the count to zero, and data-chaining will occur to continue reading the data until the transmission is ended with an EOT. When a negative response is received on the Read Response, only one byte of data (the negative response) will be read into the message area and channel end/device end occurs (no unit exception). With the "wrong length" flag on and a nonzero data count, there is no data-chaining to the next Read command. Instead, QTAM channel end detects the polling TP Op code and initializes for the next terminal to be polled by returning to IOS for execution with a pointer to the Write Polling Characters CCW.

Write Initial Channel Program

Operation	Address	Flags	TP-Op Code	Count
1. Write deselect characters (3 circle Cs and circle S)	Table	CD	02	4
2. Write addressing chars	List	CD	03	1
3. Write space characters	Table	CC,SLI	04	1
4. Read response	Respn	CC	06	1
5. Write data	Area		11	Length

The Write Initial channel program places the terminal in control mode and sends a circle S to denote that addressing will follow. The terminal is addressed with a one-character code followed by a space character. The response to addressing is read into the first byte of the response field in the DECB (DECRESPN). The Read Response CCW is command-chained to write the data.

IBM 2740 WITH STATION CONTROL AND CHECKING

Read Initial Channel Program

Operation	Address	Flags	TP-Op Code	Count
1. Write deselect characters (3 circle Cs)	Table	CD	02	3
2. Write polling character	List	CD	03	1
3. Write space character	Table	CC, SLI	04	1
4. Read response	Area	CD	05	2
5. Read data	Area+2	SLI	11	Length-2

The Read Initial channel program initiated by the Read/Write routine places the line in control mode, polls a terminal with one character followed by a space character, and reads the response. (Control mode is that state of the system that allows a terminal to be selected.) The third command (Read Response character) specifies a data count of 2, with wrong length indication not suppressed, while the length of the response character is one byte. Under the existing configuration of BTAM, 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 the end of transmission.

2. Negative response. This response causes channel end and device end with unit exception and wrong length record indicated. The Channel End routine detects the polling restart TP Op code, reinitializes for the next terminal to be polled, and returns control to IOS for execution of the CCWs beginning with the one containing a 03 TP Op code.

Read Continue Channel Program

Operation	Address	Flags	TP-Op Code	Count
1. Write response (circle Y)	Table	CC, SLI	02	1
2. Read data	Area	SLI	11	Length

The Read Continue channel program is initiated by the EOB or EOBLC routine after a successful Read Initial operation; the program writes the response character and command-chains to read data.

Read Repeat Channel Program

Operation	Address	Flags	TP-Op Code	Count
1. Write negative response (circle N)	Table	CC, SLI	02	1
2. Read data	Area	SLI	11	Length

The Read Repeat channel program is initiated by the EOBLC routine after a data check occurs during execution of the Read Data command of a Read Initial or Read Continue operation. The program transmits a negative response, and then chains to the second CCW to read data into the main storage area originally specified.

Write Initial Channel Program

Operation	Address	Flags	TP-Op Code	Count
1. Write deselect characters (3 circle Cs and circle S)	Table	CD	02	4
2. Write addressing chars	List	CD	03	1
3. Write space characters	Table	CC, SLI	04	1
4. Read response	Respn	CC	06	1
5. Write circle D	Table	CD	08	1
6. Write data	Area	CC, SLI	11	Length
7. Read response	Respn+1		20	1

The Write Initial channel program, initiated by the Read/Write routine, places the line in control mode and informs it that the addressing function will follow by circle S, addresses a terminal with a one-character code followed by a space character, and reads the response. The status of the chaining flags for the third command depends upon the status of the addressing list. For multiple component addressing, all specified components must be logically connected to the line before message transmission occurs. A negative response from any component terminates the channel program and suppresses transmission.

Write Continue Channel Program

Operation	Address	Flags	TP-Op Code	Count
1. Write data	Area	CC, SLI	11	Length
2. Read response to VRC/LRC	Respn+1		20	1

The Write Continue channel program is initiated by QTAM after a successful Write Initial operation; the program writes data and command-chains to read the response to longitudinal redundancy checking. The response is read into DECRESPN +1, the second byte of the two-byte response field in the DECB.

CHANNEL PROGRAMS FOR IBM 2848 - 2260 REMOTE LINES

Specific Poll of a Display Station: On positive response (STX), chains the Read Response to read the message. On negative response (EOT), an interruption occurs. QTAM detects the polling restart TP code, initializes the channel program to poll the next entry within the list, and returns control to the supervisor.

Request of a Printer Status: If the printer is ready and the buffer is empty, a reservation is set on the printer buffer that prevents transmission of messages from the display stations to the printer buffer. If a message is received indicating these conditions, the Read Response chains to the Read Data CCW. The next EOT resets the reservation condition.

A negative response is either NAK, which indicates the printer is not ready, or EOT, which indicates the printer is ready but the buffer is not empty. Both negative responses set the printer request condition, which causes the 2848 Display Control (DC), upon receipt of a general poll, to sense if the printer is in a ready condition, and if the buffer is empty.

Read Initial Channel Program

Operation	Address	Flags	TP-Op Code	Count
1.Write deselect characters (3 circle Cs)	Table	CD	02	3
2.Write polling characters	List	CD	03	2
3.Write READ MI code	Table	CC,SLI	04	1
4.Read response	Area	CD	05	2
5.Read data	Area +2	SLI	11	Length-2

Initiated by the Read/Write routine, the Read Initial channel program places the line in control mode and polls a terminal with a two-character code. For the 2260 devices, the polling characters specify a general poll of the DC, a specific poll of a display station, or a request of a printer status. After the polling characters are sent, the special READ MI code is sent to inform the 2848 that the CPU wants a message.

General Poll of a DC: The polling list must specify a general poll, with the second byte a hexadecimal FF. If the printer has a status pending as a result of a previous request (printer status or Write Initial), this message will be transmitted and the Read Response CCW will chain to the Read Data CCW.

If the printer is not ready, the display stations are scanned for a message. If a message is pending, it is sent. If there is no message waiting for transmission, a

negative response EOT is received. The channel program is interrupted; QTAM detects the polling restart TP code, updates the channel program, and returns control to the supervisor.

Read Continue Channel Program

Operation	Address	Flags	TP-Op Code	Count
1.Write ACK	Table	CC,SLI	02	1
2.Read data	Area	SLI	11	Length

The Read Continue channel program sends a positive response ACK and reads the message. If the previous operation was a specific poll of a display station, an EOT will be returned, which ends the operation. If the previous operation was a general poll, a message (if one is sending) will be received; otherwise an EOT is received.

Read Repeat Channel Program

Operation	Address	Flags	TP-Op Code	Count
1.Write NAK	Table	CC,SLI	02	1
2.Read data	Area	SLI	11	Length

The Read Repeat channel program sends a negative response (NAK) and reads the data.

Write Initial Channel Program

Operation	Address	Flags	TP-Op Code	Count
1.Write deselect characters (3 circle Cs)	Table	CD	02	3
2.Write addressing chars	List	CD	03	2
3.Write WRITE code	Table	CC,SLI	04	1
4.Read response	Respn	CC	06	1
5.Write STX	Table	CD	08	1
6.Write data	Area	CC,SLI	11	Length
7.Read response to Text	Respn +1		20	1

The Write Initial channel program is for either the printer or the display station. The channel program places the line in control mode, sends the addressing characters, and sends the WRITE code. If a printer is addressed, the Read Response CCW reads the addressing sequence response. If either an EOT or an NAK (negative responses) is received, there is an interrupt. The EOT indicates the printer is not ready, and the NAK indicates the printer is ready but the buffer is not empty. Either of these sets is a printer request.

If the response is positive (ACK), which indicates that the printer is ready and the buffer is empty, the Read Response CCW command-chains to send the STX (Start of text character) and then sends the data. If a transmission error occurs, the operation is stopped and the printer buffer is cleared.

If a display station is addressed, the Read Response CCW reads the addressing sequence response, which is normally positive (ACK) and chains to read the data. If a transmission error occurs, the EOBLC routine will retry transmission three times before setting the error bit in the error halfword.

Write Continue Channel Program

Operation	Address	Flags	TP-OP Code	Count
1. Write STX	Table	CC,SLI	08	1
2. Write data	Area	CC,SLI	11	Length
3. Read response to test	Respn + 1		20	1

The Write Continue channel program writes the STX character and the data. The Write Data command is chained to the Read Response command, which reads the response into the second byte of the DECB response field.

Write Erase Channel Program

Operation	Address	Flags	TP-Op Code	Count
1.Write deselect characters (3 circle Cs)	Table	CD	02	3
2.Write addressing chars	List	CD	03	2
3.Write ERASE Code	Table	CC,SLI	04	1
4.Read response	Respn	CC	06	1
5.Write STX	Table	CD	08	1
6.Write data	Area	CC,SLI	11	Length
7.Read response to text	Respn +1		20	1

The Write Erase channel program places the line in control mode, addresses a terminal with the two-character code, and sends the special code ERASE. This operation is to erase the CRT and any message on

the display screen starting in the upper left-hand corner. The response to addressing is read in the first byte of the response field in the DECB (DECRESPN). If a negative response is received, the channel program is terminated.

If a positive response is received, the Read Response is chained to the Write STX character followed by data. The response to text is read into the second byte of the DECB response field.

Write at Line Address Channel Program

Operation	Address	Flags	TP-Op Code	Count
1.Write deselect characters (3 circle Cs)	Table	CD	02	3
2.Write addressing chars	List	CD	03	2
3.Write WRITE LINE code	Table	CC,SLI	04	1
4.Read response	Respn	CC	06	1
5.Write STX	Table	CD	08	1
6.Write data	Area	CC,SLI	11	Length
7.Read response to text	Respn +1		20	1

The Write at Line Address channel program places the line in control mode, addresses a terminal with a two-character code, and sends the WRITE LINE code to indicate the operation to the 2848. The response to addressing is read. If it is positive, the Read Response chains to write the STX character and the data. If the response is negative, the channel program is terminated. The cursor is positioned on a specified line and the characters are displayed from that point. The response is read into the second byte of the response field in the DECB. If a transmission error occurs, the EOBLC routine will retry three times before setting the error indication in the error halfword.

CHANNEL PROGRAMS EMPLOYING THE AUTO POLL FEATURE

The QTAM Device I/O modules incorporated for each of the terminals supported by OS QTAM with Auto Poll are the following:

IGG019N3 IBM 1030

IGG019N1 IBM 1050 (nonswitched)

IGG019N2 IBM 1060

IGG019N9 IBM 2740 (with station control)

IGG019N8 IBM 2740 (with station control and checking)

The Device I/O modules are essentially the same for Auto Poll as for the other terminal types except for the Read Initial operation for the 2740, which deletes the write space character.

After the Read/Write module (IGG019NZ) has built the channel program, the Read Initial channel program, independent of terminal type, is in the following form:

Operation	Address	Flags	TP-op Code	Count
1. Write EOT sequence	Table	CC,SLI	02	3
2. Poll	List	CC,SLI	03	k(n)*
3. TIC	2nd Poll Command	SLI	09	1
4. TIC	Read Response Command			
5. Poll	First entry in List	CC,SLI	03	k(n)*
6. TIC	2nd Poll Command	SLI	09	1
7. Read Response	Area	CD	0A	2
8. Read Data	Area + 2	CC,SLI,PCI	11	length -2

*k = 2 for IBM 1030, 3 for other devices
n = total number of entries in the polling list.

Where, on a Read Initial command, the CCWs (1-8) have the following effect:

1. EOT sequence of three circle Cs in line code.
2. Polls the terminal with polling character.
3. On a negative response to polling at the end of the list, this TIC will be executed to start the second Poll command.
4. On a positive response to polling, this TIC command will be executed to start the Read Response command.
5. If either Poll command terminates with negative response at the end of the list, this Poll command will restart polling at the beginning of the list.
6. On a negative response to polling at the end of the list, this TIC command will be executed to restart the second Poll command.
7. On a positive response to polling, this command will read the list entry

index byte and the first byte of text into the message area and then chain to 8.

8. This command causes the remainder of the text to be read into the message area.

CHANNEL PROGRAMS FOR WORLD TRADE TELEGRAPH ADAPTER

The channel programs for terminal-to-CPU transmission (Read Initial and Read Continue) and for CPU-to-terminal transmission (Write Initial) are made up of two parts:

- The first part (identification exchange channel program) is a channel subprogram automatically associated with the second part. On request, it performs identification exchanges at any time during message transmission.
- The second part (Read or Write channel program) is set up to receive input messages or to send output messages.

Read Initial Channel Program

Operation	Address	Flags	TP Op Code	Count
1. Write CPU-ID sequence (Note 2)	List	CD SLI	07	n
2. Write (see Note 1)	WRU	CC SLI	04	1
3. Read terminal-ID (Note 3)	List	SLI	07	Length-1
4. Prepare	0	CC SLI	01	1
5. Sense	TP Op Area	CC SLI	FF	1
6. Read	Area	SLI	11	Length

The Read Initial channel program is started by the Read/Write routine at the fourth CCW.

The Prepare command prepares the control unit to receive a message and, when a character is received, command-chains to the Sense command and to the Read command. When the Sense command is executed, the TP Op code of the Sense overlaid by the adapter sense byte (which is never X'FF'), the contents of the Sense command TP Op code indicates when data is to be received.

Read Continue Channel Program

Operation	Address	Flags	TP Op Code	Count
1. Write CPU-ID sequence (Note 2)	List	CD SLI	07	n
2. Write (see Note 1)	WRU	CC SLI	04	1
3. Read terminal-ID (Note 3)	List	SLI	07	Length-1
4. Read	Area	SLI	11	Length

The Read Continue channel program is initiated by:

- Tha EOB routine when a WRU signal has been received. The channel program started at the first CCW performs an ID exchange, and then the fourth CCW reads the remaining data into the main storage area originally specified.
- The Activate routine when the last message received was ended by EOM; the channel program is started at the fourth CCW.

Write Initial Channel Program

Operation	Address	Flags	TP Op Code	Count
1. Write	Table	CD SLI	04	1+m (Note)
2. Write	Table	CD SLI	01	12
3. TIC				
4. Write CPU-ID sequence (Note 2)	List	CD SLI	07	n
5. Write (see Note 1)	WRU	CC SLI	04	1
6. Read terminal-ID (Note 3)	List	SLI	07	Length-1
7. Write	Area	SLI	11	Length

Note: m is the number of mark characters specified by the user.

The Write Initial channel program is started by the Read/Write routine, and the CCWs have the following effect:

1. Twelve letters shift characters are sent at the beginning of the output message.

2. The transfer address in the TIC CCW is that of the third or of the sixth CCW, depending on whether the WRU macro instruction is present in the Send Header subgroup of the LPS.

Notes

1. When the Automatic Answerback Unit feature is installed on the terminal, the CPU sends the WRU signal to the terminal, which then sends its identification sequence to the CPU.

2. The computer identification (CPU ID) defined in the POLL macro instruction associated with the line is sent to the terminal.

3. The terminal identification is read into the area reserved by the TERM macro instruction associated with the line.

This section summarizes the operation of each of the LPS routines from which the user selects those required for his particular message control functions. The routines selected form collectively the Line Procedure Specification (LPS) section of the message control program. Each LPS routine is contained within a module; each module contains a single routine.

The majority of the LPS routines correspond to LPS macro instructions, and are linkage edited into the Message Control Program Load module because of the inclusion of the macro instructions in the message control source program. They are entered upon execution through linkages generated in the macro expansions.

The remaining LPS routines are general routines; each of these is linkage edited into the Message Control Program Load module because of a linkage generated in any of several LPS macro instructions.

Each of the following LPS routine descriptions provides:

- Name of the routine.
- Name of the module that contains the routine.
- Function of the routine.
- Entry point and linkage information.
- Names of external routines used.

BREAKOFF ROUTINE (CHART BY)

Function: This routine causes a message to be terminated and an error bit to be set, if the incoming message exceeds maximum length, or if the characters in the buffers are identical (usually an indication of terminal or line malfunction). If the characters are identical, the routine skips the length comparison and sets up for an error. If the characters are not identical, the routine adds the previous count of characters in LCBERCCW+6 field of the LCB to the length of the current message, and restores the LCBERCCW+6.

The Breakoff routine obtains the specified maximum length of a message passed by register 14. If the specified length is greater than zero, the accumulated length is compared with the maximum length specified; otherwise, the length comparison is bypassed. If the accumulated length is greater than the maximum length, the routine sets up for an error by turning off the "receive" bit in the LCBSTATE field of the LCB: this keeps buffers from being assigned, which causes a program check. If the accumulated length is less than or equal to the maximum length, it tests for end of message. If it is not the end of message, the routine returns to the next LPS instruction; otherwise, it tests for program check. Breakoff characters are not written until end of message, and a program check indicates Breakoff characters are to be written. If there is no program check, return is made to the next LPS instruction. If there is a program check as a result of no buffer assignment, the Read Initial operation code is cleared, the "breakoff" bit in the error halfword is set, and LCBTRST field of the LCB is set to the EOB of the text segment. The address of the CCW with a BREAK command code is moved into LCBSTART field of the LCB. The channel program is executed to write the control characters necessary for the breakoff. The Breakoff routine branches to the LPS control to wait for the breakoff.

Module Name: IECKBRKF

Entry Point: Expansion of the BREAKOFF macro instruction generates a BALR to the routine at IECKBRKF, using register 15 as the branch register and register 14 as the return register. Register 14 also serves as a parameter register. The parameter list passed to the routine consists of the maximum length of a message.

External Routines Used: EXCP (SVC 0)

CANCEL MESSAGE ROUTINE (CHART CL)

Function: This routine causes the message to be cancelled when any of the error conditions specified by the error mask is indicated in the error halfword, or when the error mask is zero. If the error mask is not zero, and none of the error conditions specified by the error mask is indicated in the error halfword, return is made to the next LPS instruction. If the error mask is zero or the specified errors are detected, and if the destination code has not been specified in the error mask, or no destination code error is set in the error halfword, linkage is made to the Recall routine to obtain the header. When the header is available, the "cancel" bit is set on in the MSTATUS field of the header

prefix. When any of the error conditions specified by the error mask is indicated in the error halfword, or when the error mask is zero, the previous sequence number is stored in the TSEQUIN field of the terminal table unless it is equal to zero. Zeros are moved into the LCBMPLRT byte and LCBDLPTR of the LCB to cancel the multiple route option and distribution list, and the conversational mode bit in the LCB is cleared before returning to the next LPS instruction.

Module Name: IECKCNCL

Entry Point: Expansion of the CANCELM macro instruction generates a BALR to the routine at IECKCNCL, using register 15 as the branch address register and register 14 as the return register. Register 14 also serves as a parameter list register. The parameter list passed to the routine consists of the error mask in hexadecimal notation.

External Routines Used: Recall (IECKRC in module IGG019NG)

Check I/O has been deleted in QTAM.

DATE STAMP ROUTINE (CHART CH)

Function: This routine obtains the current date in packed decimal form (via a TIME macro), unpacks the date, and inserts it in the message header in the format byy.ddd, where b = blank, yy = year, and ddd = day of the year. Prior to inserting the date, the Date Stamp routine links to the Expand routine (which "expands" the header by shifting, seven places to the left, all message characters from the end of the prefix plus seven, up to and including the character pointed to by the scan pointer). The date is then inserted in the field created. The scan pointer points to the last character in the date.

Module Name: IECKDATE

Entry Point: Expansion of the DATESTMP macro instruction generates a BALR to the routine at IECKDATE, using register 15 as the branch address register and register 14 as the return register. Register 14 also serves as the parameter list register. The parameter list passed to the routine consists of a halfword containing, in binary form, the length (7) of the date field to be inserted in the message header.

External Routines Used: Expand (module IECKEXPD)

DISTRIBUTION LIST ROUTINE (CHART DB)

Function: This routine stores the destination key in the LCBDLPTR field of the LCB. The address of the terminal list is obtained from the terminal table. If there is an entry in the list, the address of the QCB for the destination is put in the LCBDESTQ field of the LCB. If there is no entry in the list or after the address of the QCB has been stored in the LCBDESTQ, the destination key is stored in the TTDKEY field of the prefix. The address of the QCB is moved into the RECB section of the buffer. If the Distribution List routine was previously entered, it branches to the Priority subroutine in IECKQQ01. If the Distribution List routine has not been entered previously, it branches to the Endinsrt (End Insert) routine. The Endinsrt routine places the address of a special entry point in the Distribution List routine, which the Cleanup routine will process in a chain according to the priority specified. Endinsrt replaces the second operand of the BAL instruction to the Endinsrt routine with the address of the Priority subroutine. Endinsrt returns via the return register minus four, which returns to the same BAL instruction. This time the BAL instruction branches to the Priority subroutine.

The code is entered from the Cleanup routine by a branch to a routine in the chain. If there is no entry in the list, the routine branches to the next routine in the chain. If there is an entry in the list, the address of the header is saved for the next destination. The destination key of the next entry is stored in the LCBDLPTR field of the LCB. If there is an entry in the terminal table, the address of the QCB is placed in the LCBDESTQ field of the LCB. The routine links to the Recall routine to obtain the header. Upon return, the destination is stored in the TTDKEY field of the prefix. When there are more destinations to be satisfied, return is made to the start of the Cleanup routine.

Module Name: IECKDLQT

Entry Point: The routine is entered from the module IECKQQ01.

External Routines Used:

- End Insert (Endinsrt in module IGG019NG)
- Priority (in module IECKQQ01)
- Recall (IECKRC in module IGG019NG)

END OF ADDRESS ROUTINE (CHART DC)

Function: The EOA macro expansion branches to the Message Type routine, which branches to the End of Address routine if the EOA character is not found by the Scan routine. The End of Address routine computes and saves the offset of the destination in the header from the start of the header. If this is the first time the EOA macro appears in the LPS, the routine branches and links (BAL) to the Endinsrt (End Insert) routine. The Endinsrt routine places the address of a special entry point in the End of Address routine, which the Cleanup routine will process in a chain according to the priority specified by a DC in the routine. Endinsrt replaces the second operand of the BAL instruction to Endinsrt with the address of the Skip Character Set routine. Endinsrt returns via the return register minus four, which returns to the same BAL instruction. This time the BAL instruction branches to the Skip routine.

If the EOA macro has been entered before, the routine branches immediately to the Skip routine because the code has already been inserted in the chain. The Skip Character Set routine advances the scan pointer past the specified EOA character and returns to the code generated by the macro. If the EOA character specified in the macro is found by the Scan routine through the use of Message Type routine, return is made to the code generated by the EOA macro, which tests to determine if the header is being copied. If the header is being copied, a branch is made to the ENDRCV macro expansion, which branches to the Cleanup routine. If no header is being copied, return is made to the next LPS instruction.

The End of Address routine is entered at the special entry point from the Cleanup routine by a branch to the next routine in the chain. The routine tests for multiple routing. If there is no multiple routing indicated in the LCBMPLRT field of the LCB, the routine returns to the Cleanup routine, which links to the next routine in the chain. If there is another destination, the routine links to the Recall routine to obtain the header from the disk. Multiple routing is set up by clearing the error indication in the LCBERRST field, the distribution list pointer in the LCBDLPTR field, and the multiple routine indicator in the LCBMPLRT field in the LCB. The scan pointer is reset to the offset of the next destination from the end of the prefix. The routine branches to the Route routine, which handles the next destination code and returns to the first instruction in the EOA macro expansion. The End of Address routine is repeated until all destination codes in the header have been handled.

Module Name: IECKEOAD

Entry Point: Expansion of the EOA macro instruction generates a BALR to the Message Type routine at IECKTYPE, using register 15 as the branch address register and register 14 as the return register. Register 14 also serves as a parameter list register. The parameter passed to the routine consists of the field size and the EOA character specified in the macro. The parameter register 1 contains the address of the End of Address routine.

External Routines Used:

- Skip Character Set (module IECKSKPS)
- End Insert (Endinsrt in module IGG019NG)
- Recall (IECKRC in module IGG019NG)
- Route (module IECKROUT)

END OF BLOCK ROUTINE (CHART CY)

Function: The function of this routine depends on whether it is entered from the EOB macro expansion or from the ENDRCV macro expansion.

1. When this routine is entered from the EOB macro expansion, and if the message has been cancelled, an error message has been sent, or the message has been rerouted, this macro is not executed and return is made to the next LPS instruction. Similarly, if the status byte of the CSW indicates either an end of transmission, a unit exception, or a residual count of zero in the CSW, return is made to the next LPS instruction. The end-of-block bit in the prefix is set because there was a positive indication that the message was correctly transmitted. In setting up for transmission of the next message, the scan pointer is adjusted to segment size and is stored in the header prefix. The LCBTRST halfword is updated by storing the segment size in this field.

   If the transmission was a Write operation, a check is made for 1030 or 1060. If a 1030 or 1060, return is made to the next LPS instruction. Otherwise, the operation code is set for Write Continue. If the transmission was a Read, the operation code for Read Continue is set in the LCBCECB field of the LCB. For both operations, the buffer is set to be

reused. The End of Block routine branches to the Activate routine. Return is made from the Activate routine.

2. When this routine is entered from the ENDRCV macro expansion, it tests the WRU flag in the LCB. If this flag is not set, return is made to the next LPS instruction. If the WRU flag is set, this indicates that the last character received is WRU. In this case, an identification exchange must be performed. The EOB bit in the buffer prefix is set, and the buffer is set to be reused. The Read Continue indicator is set in the LCBECB field of the LCB. The End of Block routine exits to the physical I/O routine, which generates and initiates execution of the appropriate channel program. This channel program performs an identification exchange and reads the rest of the input message, provided EOM is different from WRU.

Module Name: IECKEOBK

Entry Point: Expansion of the EOB macro instruction or of the ENDRCV macro instruction (if this is an LPS for a WTTA line) generates a BALR to the routine at IECKEOBK, using register 15 as the branch address register and register 14 as the return register.

External Routines Used: Activate (IECKACT in module IGG019NG)

END OF BLOCK AND LINE CORRECTION ROUTINE (CHART CZ)

Function: For the following conditions the retry counter and error flag in the LCB are cleared and return is made to the next LPS instruction.

1. A message has been cancelled.
2. An error message has been sent.
3. The message has been rerouted.
4. Transmission has failed three times and an EOT has been received.
5. Transmission has failed and there is a time-out or intervention required.
6. An EOT was received or other errors occurred.

If there was a transmission error, but not one of the above, the "transmission error" and "time-out" bits are set to zero in the error halfword unless the permanent error flag was set in the LCBERCCW field of the LCB. The routine branches and links to the Recall routine to obtain the header. If the line is sending, Write Continue is set in the LCBCECB field of the LCB. If the device is an IBM 1030 or IBM 1060, the Write Initial operation code is set in the LCBCECB field of the LCB. Return is made to the start of the LPS for another try at transmission.

If the line is not sending, action is taken to retry receiving the message. If an EOB is present in the header segment, the entire message is cancelled and a new buffer is set up. If no EOB is present in any buffer position other than the last position, the cancel bit of the MSTATUS field is set. If the sequence number is not zero, the last sequence number in the terminal table is stored in the TSEQUIN field of the terminal table. Linkage is made to the Recall routine to obtain the header. The scan pointer and the LCBTRST field of the LCB are updated. An end of address (EOA) character is set up, and the header is filled with idle characters in the space reserved for time, date, or sequence number. (The EOA and idles are not set for an IBM 2260 device.) The distribution list and multiple route indicators are cleared, and error flags are reset to zero. For all receiving messages, the end of message is set in the MSTATUS field of the prefix, and the transmission operand for the retry is set in the LCBCECB field of the LCB. The buffer size is stored in the MSEGSZE field of the prefix to indicate the message size. The LCBCLCCW field of the LCB is set to reuse the buffer. A branch is made to the Activate routine.

If there were no transmission errors or an EOT had not been received, tests are made on the CSW. If the status byte of the CSW indicates either an end of transmission or a residual count of zero, the retry counter and error flag in the LCB are cleared and return is made to the next LPS instruction. The end of block bit in the prefix is set, because there was a positive indication that the message was correctly transmitted. In setting up for transmission of the next message, the scan pointer is adjusted to segment size and is stored in the header prefix. The current segment is set as the last correctly transmitted message in the LCBRCADD field of the LCB. The LCBTRST halfword is updated by storing the segment size in this field. If the transmission was a Write operation, a Write Continue is set in the LCBCECB field of the LCB. If an IBM 1030 or IBM 1060 was the transmitting terminal, there is no further execution of this routine; the retry counter and error flag are cleared before returning to the next LPS instruction. For a Read operation, the Read Continue operation code is set in the LCBCECB field of the LCB. For both operations, the buffer is set to be reused. The routine branches to the Activate routine. Return is made from the Activate routine.

Module Name: IECKEOBC

Entry Point: Expansion of the EOBLC macro instruction generates a BALR to the routine at IECKEOBC, using register 15 as the branch address register and register 14 as the return register.

External Routines Used:

- Activate (IECKACT in module IGG019NG)
- Recall (IECKRC in module IGG019NG)

ERROR MESSAGE ROUTINE (CHART CQ)

Function: This routine causes a user-written error message to be sent to a designated terminal when any of the error conditions specified in the error mask is indicated in the error halfword, or when the error mask is zero. If the error mask is not zero and none of the error conditions specified by the error mask is indicated in the error halfword, return is made to the next LPS instruction. If there has been a sequence number error, the last valid sequence-in number is obtained from the terminal table. The error text is scanned for a dollar sign ($). If a $ is found, the sequence-in number is inserted, in decimal form, in the error text. Upon encountering a second $, the specified sequence number, obtained from the header prefix, is inserted in decimal form in the error text.

When an error condition is encountered, linkage is made to the Recall routine (in module IGG019NG) to obtain the header. A test is made for the option of including the header of the message in the error message. If the header is not to be included, the scan pointer is reset to the beginning of the header of the message in error. The specified error message then overlays the header. If the header is included, the pointer remains positioned at the end of the header. The buffer is loaded with the error text. If the error message exceeds the space in the buffer, the text is truncated. The size of the message is stored in MSEGSZE field, and single segment is indicated in MSTATUS field of the prefix. Linkage is made to the Lookup routine, which looks up the destination code in the terminal table and places the relative address in the TTDKEY field of the header prefix for the error message to be sent. Return to the next LPS instruction is made by the Lookup routine.

Module Name: IECKERMG

Entry Point: Expansion of the ERRMSG macro instruction generates a BALR to the routine at IECKERMG, using register 15 as the branch address register and register 14 as the return register. Register 14 also serves as a parameter list register. The parameter list passed to the routine consists of the error mask in hexadecimal notation. Register 0 contains the length of the error message (0 if an address is specified). The address of the destination terminal is contained in register 2; the address of the error message is in register 1.

External Routines Used:

- Recall (IECKRC in module IGG019NG)
- Lookup (module IECKLKUP)

EXPAND ROUTINE (CHART CU)

Function: If the scan pointer is pointing to a blank character, the pointer is shifted back one position. The number of characters to be shifted is computed by subtracting the end of the prefix and the number of spaces to be expanded from the value in the scan pointer. If the result is negative, return is made to the next LPS instruction because there is no space for the shift. If there is sufficient space, the characters are moved to the left the number of spaces indicated. After the characters of the header have been shifted, a blank is inserted as a left delimiter at the start of the field. The scan pointer for the next destination is shifted to the left the length of the new field. If an EOA has not been reached, the scan pointer is stored in LCBMPLRT of the prefix. Return is made to the calling routine.

Module Name: IECKEXPD

Entry Point: The routine is entered via a BALR from SEQOUT, TIMESTMP, or DATESTMP; register 15 is the branch address register and register 3 is the return register. The address of the parameter list is passed to the routine in register 14. The parameter list contains the number of spaces the header is to be expanded.

External Routines Used: None

INTERCEPT ROUTINE (CHART CT)

Function: This routine causes suppression of all message transmission to a terminal when any of the error conditions specified by the error mask is indicated in the error halfword, or when the error mask is specified as zero. If the error mask is not zero, and none of the error conditions specified by the error mask is indicated in the error halfword, return is made to the next LPS instruction. The routine makes linkage to the Recall routine to recall the header. The "serviced" bit is turned off and the "priority" bit is turned on in the prefix so that a new sequence number is not

assigned. The "send" bit of the TSTATUS byte of the terminal table for that entry is turned off to indicate that messages on the queue for the destination were withheld from transmission. If the "intercept" bit in the TSTATUS byte is on, indicating that a previous message is in the INTERCPT field, and if the header address is greater than the address in the INTERCPT field, return is made to the next LPS instruction. If the header address is less than the address in the INTERCPT field, and if the "intercept" bit in the TSTATUS byte is off, the "intercept" bit is set to one, to indicate that a message on the queue was not transmitted, and the header address is put into the INTERCPT field in the user area of the terminal table entry. The offset to the INTERCPT field in the terminal table is obtained and saved in LPSTART for the Release Intercepted Message routine. Return is made to the next LPS instruction.

Module Name: IECKITCP

Entry Point: Expansion of the INTERCPT macro instruction generates a BALR to the routine at IECKITCP, using register 15 as the branch address register and register 14 as the return register. Register 14 also serves as a parameter list register. The parameter list passed to the routine consists of the address of the error mask in hexadecimal notation for the communication line. The parameter register 1 contains the address of the INTERCPT field in the terminal table.

External Routines Used: Recall (IECKRC in module IGG019NG)

LOOKUP ROUTINE (CHART CO)

Function: This routine obtains, in succession, the destination name contained in each terminal table entry and compares it with the destination name provided in a work area. Each time a no-compare results, the process is repeated with the destination name from the next terminal table entry. When a match results, the routine obtains, from the terminal table entry, the address of the queue control block for the destination queue, and places this QCB address in the LCBDESTQ field of the LCB.

If the terminal name in the work area does not match any destination name in the terminal table, the routine turns on the invalid destination bit (bit 0) in the error halfword, and places the address of the QCB for the dead-letter queue in the LCBDESTQ field of the LCB.

Module Name: IECKLKUP

Entry Point: The Lookup routine is entered, either:

1. At IECKDRCT, via a BALR in the DIRECT macro expansion. (Register 15 is the branch address register and register 14 is the return register).

2. At IECKLKUP, via unconditional branches from the Routing, Error Message, and Reroute routines.

External Routines Used: None

MESSAGE MODE ROUTINE (CHART CW)

Function: This routine is entered when a specific character is specified in the second operand of the macro. Linkage is made to the Scan routine to obtain the next nonblank character in the header. If the character provided by the Scan routine is identical to the one specified in the MODE macro, the Message Mode routine branches to the routine designated in the first operand. If the characters do not match, the scan pointer is restored, and return is made to the next LPS instruction.

Module Name: IECKMODE

Entry Point: Expansion of the MODE macro instruction generates a BALR to the routine at IECKMODE, using register 15 as the branch address register and register 14 as the return register. The parameter list passed to the routine consists of the character that is compared with the first nonblank character in the header. Register 1 is a parameter register that contains the address of the routine specified by the first operand of the macro.

External Routines Used: Scan (module IECKSCAN)

CONVERSATIONAL MODE ROUTINE (CHART CX)

Function: The "converse" bit is set in the LCBSTATE field of the LCB. If this is the first appearance of the macro in the LPS, the Conversational Mode routine branches and links (BAL) to the End Insert routine. If the Conversational Mode routine has been previously entered, return is made to the next LPS instruction. The End Insert routine places the address of a special entry point in the code of the Conversational Mode routine, which the Cleanup routine will process for the conversational mode in a chain according to the priority specified by a DC in the routine. End Insert replaces the second operand of the BAL instruction to End Insert with the address of the next LPS instruction, and returns via the return register minus four, which

returns to the same BAL instruction. This time the BAL instruction branches to the next LPS instruction.

This section of the routine is entered from the Cleanup routine by a branch to the next routine in the chain. If the conversational mode has not been indicated, an error message is to be processed, or a polling or addressing error has occurred, return is made to the next routine in the chain. The conversational mode must be used for processing; therefore, if the queue is not a processing queue, return is made to the next routine in the chain.

If the line is receiving, the routine branches to the LPS Control routine at STARTUP to wait for the incoming message to be processed. If the line is sending, the line is turned around to receive by setting the highest priority, the "converse" and "receive" bits in the LCBSTATE field of the LCB, and the Read operation code in the LCBCECB field of the LCB. If the end of the polling list has been reached, the start of the polling list is stored in the LCB poll pointer. The LCBCLCCW field of the LCB is reset with the BRB address for initialization of the Activate routine. The routine issues a post to insert the BRB into the ready queue, and branches to the LPS Control routine at STARTUP to wait for the message to be received. The LPS Control routine returns to the code in the Conversational Mode routine. The "service" bit is set in the prefix of the message and the "converse" bit is cleared in LCBSTATE field of the LCB. If the source terminal was not identified, then return is to the start of the Cleanup routine. The chain, pointed to by the QCB for a source terminal, is searched for a DEB to see if the process queue contains a reply. If the chain has been searched completely, return is to the start of the Cleanup routine. If there is a reply, the LCBSTATE field is set to send, and the Write operation code is set in the LCBCECB field of the LCB. The routine posts the original message to the empty buffer queue. The LCB is restored with the disk address of the reply. The relative address of the EOB is set in the LCBTRST field of the LCB. The routine posts to the I/O queue, and branches to the LPS Control routine at STARTUP to wait for the message to be received.

Module Name: IECKCVRS

Entry Point: If there is no specified character in the second operand of the macro, the expansion of the MODE macro instruction generates a BALR to the routine at IECKCVRS, using register 1 as the branch address register and register 14 as the return register. If there is a character specified in the second operand, the address of the Conversational Mode routine is placed in the parameter register 1, and the routine is entered by a branch from the Message Mode routine.

External Routines Used:

- End Insert (Endinsrt in module IGG019NG)
- LPS Control (STARTUP in module IGG019NG)
- Qpost (IGC067 in module IECKQQ01)

INITIATE MODE ROUTINE (CHART CW)

Function: The routine sets the "initiate" bit in the LCBSTATE field of the LCB. Return is made to the next LPS instruction.

Module Name: IECKNATE

Entry Point: If there is no specified character in the second operand of the macro, the expansion of the MODE macro instruction generates a BALR to the routine at IECKNATE, using register 1 as the branch address register and register 14 as the return register. If there is a character specified in the second operand, the address of the Initiate routine is placed in parameter register 1, and the routine is entered by a branch from the Message Mode routine.

External Routines Used: None

PRIORITY MODE ROUTINE (CHART CW)

Function: Linkage is made to the Scan routine, which obtains and provides the address of the first nonblank character in the header. This character is moved into the LCBPTEMP field of the LCB to be the priority of the message. Return is made to the next LPS instruction.

Module Name: IECKPRTY

Entry Point: If there is no specified character in the second operand of the macro, the expansion of the MODE macro instruction generates a BALR to the routine at IECKPRTY, using register 1 as the branch address register and register 14 as the return register. If there is a character specified in the second operand, the address of the Priority subroutine is placed in parameter register 1, and the routine is entered by a branch from the Message Mode routine.

External Routine Used: Scan (module IECKSCAN)

MESSAGE TYPE ROUTINE (CHART CA)

Function: This routine saves the scan pointer, and then links to the Scan routine, which obtains and provides (for the Message Type routine) the message header character pointed to. The Message Type routine compares the character provided with the character specified in the MSGTYPE macro statement. If the characters are identical, the routine branches to the next executable LPS instruction. If they are not identical, the routine restores the scan pointer and branches to the next Message Type routine (if this is the last Message Type routine, it branches to the next delimiter routine). Because the scan pointer is restored when the two characters are not the same, a series of Message Type routines may be executed, each examining the same message type character in the header.

Module Name: IECKTYPE

Entry Point: Expansion of the MSGTYPE macro instruction generates a BALR to the routine at IECKTYPE, using register 15 as the branch address register and register 14 as the return register. Register 14 also serves as a parameter list register. The parameter list passed to the routine consists of:

- A halfword containing the field size (one byte).
- A character constant containing the character to be compared with the scanned character.
- A character constant containing a blank.

External Routines Used: Scan (module IECKSCAN)

OPERATOR AWARENESS (CHART EO)

Function: This routine is used to format messages to be sent to the operator control terminal. If an I/O error message is to be sent, indicated by a hexadecimal 'FF' in LCBCPA +28, the line number, status, sense, TP Op Code, index byte, and terminal ID are obtained from the LCB and put into the message in printable form in the following format:

```
I/O  ERR,aaa,bb,cccc,ddee,ffgghhhh
```

where: aaa = Line number
bb = Operation code
cccc = Status halfword
ddee = Sense information
ff = TP Op code
gg = Index byte
hhhh = Terminal identifier

If a threshold message is to be sent, indicated by a nonzero value in the next to the last byte in the LCB, the line number (in next to last byte) and counters are obtained from the LCB and placed in the message in printable form in the following format:

```
THRESHOLD aaa TRANS=bbb DC=ccc IR=ddd TU=eee
```

Where: aaa = Line number
bbb = Transmission counter
ccc = Data check counter
ddd = Intervention required counter
eee = Time-out counter

After the message has been formatted the threshold counters are cleared.

The routine branches and links to the Buffer Recall routine in the Implementation module to obtain the header. Idle characters are inserted if specified. The message is moved into the buffer. The QCB address is obtained from the operator control entry in the terminal table.

The routine exits by branching to the Buffer Cleanup routine to post the message and return all buffers.

Module Name: IECKOPAW

Entry Point: Expansion of the POSTSEND or POSTRCV macro instruction generates an unconditional branch to the Implementation module (IGG019NG). This location in the module is a branch instruction to an address constant of the Operator Awareness routine at IECKOPAW. The routine is passed the LCB address, which contains information for the messages, in register 4 and the address of the buffer, into which the message is to be placed, in register 6.

External Routines Used: Buffer Recall and Cleanup (IECKRC and IECKPR in IGG019NG)

OPERATOR CONTROL ROUTINE (CHART EE)

Function: After saving the base register, scan pointer register, and return register, the routine branches to the Scan routine to get the first field of the current message header. A test is made for a header error. If one has occurred, control is returned to the next LPS instruction after the scan pointer register and return register are restored.

The control message indicator characters are moved into a work area and compared against this first field of the header. If they are not the same, then that message is not a control message; scan pointer and return registers are restored and control returns to the next LPS instruction.

If this is a control message, a test is made to see if it is a single-segment message. If it is a multisegment message, or if it is cancelled, the source key is moved to the destination key field of the terminal table entry, the QCB for that terminal is moved into the destination queue field of the LCB, registers are restored, and control is returned to the macro, which branches to ENDRCV, thus initiating the resending of that message to the source terminal.

If this is a single-segment message and it was not cancelled, the next field of the header is obtained by the Scan routine. This field, which should be the message type, is compared to find the type of operation desired. When the operation type is found, control passes to the routine to handle that process. If the operation specified is not valid, the message is retransmitted to the source terminal as described above.

SUB1 (CHART EF): Common subroutine to get the next field in the header (termname), find the offset of that terminal from the beginning of the terminal table, get the size of the remainder of the buffer, and the address of the terminal table, IECKSCAN and IECKDRCT are used to scan the header and look up the next field in the terminal table. If the terminal table entry is found, control returns to the calling routine. If not found, control is passed to the routine that will initialize for resending the message to the source terminal.

UNPAK (CHART EF): Common subroutine used to unpack the terminal table entry or the counters and convert the data to EBCDIC. The data is unpacked 8 bytes at a time into the buffer until there are less than eight bytes left to be converted. When this occurs, the next 8 bytes are unpacked into a work area. The data in the work area is moved into the buffer for the remaining number of characters. A check is made to see if the EOB and EOT characters can fit in the buffer. If they cannot, the EOB and EOT are moved into the last two bytes of the buffer. Otherwise, they are moved into the two bytes following the converted data. The actual size of the data to be transmitted is stored in the MSEGSZE field of the message prefix.

RCOPYT (CHART EH): Control is passed here when a COPYT control message is received. The routine uses the SUB1 subroutine to get the offset of the termname and the number of bytes remaining in the buffer. The size of the terminal table entry is obtained and doubled to get the number of bytes after conversion. The size of the unpacked entry is compared with the number of bytes remaining in the buffer. The lesser figure is passed to the UNPAK subroutine to unpack the entry into the buffer and translate it. The message is then sent to the source.

RCOPYC (CHART EG): Control is passed to the SUB1 subroutine to get the termname, offset to that terminal table entry, and the number of bytes remaining in the buffer. The QCB address is obtained from the entry and a check is made to see if this is a terminal entry. If a list or process entry, then the message is returned to the source terminal. If it is a terminal entry, the relative line number and DCB address are gotten from the QCB. The LCB size and the address of the start of the LCBs are obtained from the DCB. The LCB address for this line is calculated, and the address of the counters in that LCB is calculated. The copy QCB is posted to itself. When the QCB comes to the top of the ready queue, the Copyclr routine is executed.

The Copyclr routine restores the registers to their values before the Qpost. The threshold counters are added to the cumulative counters. The number of bytes remaining in the buffer is compared with the number of bytes needed to receive the translated counter data. The smaller figure is used as the number of bytes to be unpacked and translated by the UNPAK subroutine. When control returns from the UNPAK subroutine, the threshold counters are cleared to zero and control passes to the Qdispatch subroutine.

RSWITCH (CHART EK): Control comes here when a SWITCH control message is received. A test is made to see if the ALTERM parameter was specified. If it was not, the message is returned to the source terminal. If ALTERM was specified, the offsets to the primary control terminal and the alternate are reversed in the macro-generated parameter list. The "serviced" bit in the prefix is set, and control is returned to the macro, which then branches to the ENDRCV macro expansion.

RCHNGT (CHART EI): Control comes here when a CHNGT control message is received. SUB1 is used to get the offset to the terminal table entry and the number of bytes left in the buffer. The blank delimiters following termname in the header are skipped over. The data to be placed in the terminal table is translated and this data is scanned for a delimited blank, EOB, EOT, or invalid character. If a delimiter is not found or if an invalid character is found, a test is made to see if there is enough room to insert the EOB and EOT in the buffer. If there is room, it is put in after the data. If not, it is put in after termname. In

either case, the message is returned to the source terminal.

If the data is valid, and the delimiter is found, the number of bytes to be moved into the terminal table entry is checked for zero. If zero, the serviced bit is turned on in the header prefix and control is returned to the macro, which branches to ENDRCV. If greater than zero, a test is made for an odd number of bytes to be moved. If odd, then the same procedure is followed as for an invalid character. If even, a test is made to see if the data in the buffer, when converted, will fit in the terminal table entry. If not, the same procedure is followed as for an invalid character.

If the data will fit, then registers are set up and Qpost is issued, posting the change QCB to itself. When the QCB comes to the top of the ready queue, the terminal table is changed. The routine doing this, Change, is disabled to interrupts.

Change routine gets the QCB address from the terminal table entry to be changed. If this is a list or process entry, the pointer to the area in the terminal table to be changed is bumped past the size and QCB address fields. The data to be inserted is packed and moved into the terminal table. Exit is to the Qdispatch subroutine. If this is a terminal entry, the relative line number and DCB address are taken from the QCB. The LCB size and start of the LCB is obtained from the DCB. With this information, the address of this LCB is calculated. A test is made to see if this line is active. If not, the pack and move is initiated (INACTIVE). If the line is active, the pointer to the area in the terminal table entry is bumped past the sequence numbers. If there is no data to move, control passes to the Qdispatch subroutine. If there is data to be moved, the pack and move operation is initiated.

RINTRCPT (CHART EJ): Control is passed here when an INTERCPT control message is received. A test is made to be sure INTRCPT was specified in the macro. If it was not, the message is returned to the source terminal. If INTRCPT was specified, SUB1 is used to get the offset of the terminal specified in termname, and the number of bytes in the remainder of the buffer. A test is made for a terminal entry. If not a terminal entry, the message is returned to the source terminal. If it is a terminal entry, the "send" bit in the TSTATUS field of the entry is set off, the "serviced" bit in the header prefix is turned on and control is passed to the macro, which branches to ENDRCV.

RRELEASM (CHART EJ): Control is passed here when a RELEASM control message is received. SUB1 is used to get the offset to the terminal table entry corresponding to the termname in the message header, and the number of bytes remaining in the buffer. A test is made to see if that terminal is in intercept mode. If not, the "serviced" bit in the header prefix is set on and control passes to the ENDRCV macro. If the terminal is in intercept mode, the address of the intercept field in the terminal table entry is obtained and the QCB address is acquired. The relative record number of the first message intercepted is compared with the relative record number of the highest-priority message received to see if any priority messages were intercepted. If priority messages were intercepted, the relative record number of the first message intercepted overlays the relative record number of the highest-priority message intercepted. In either case, the first message intercepted will be the first released.

The "intercept" bit and the bits in TSTATUS are set to 0 and the send bit is turned on. The "serviced" bit in MSTATUS is turned on and control is passed to ENDRCV.

RSTARTLN (CHART EK): Control is passed here when a STARTLN control message is received. SUB1 is used to get the offset of the terminal indicated by termname, and the number of bytes remaining in the buffer. The QCB for that entry is obtained and a test is made to make sure this is a terminal entry. (If it is a process or list entry, the message is returned to the source terminal.) The RLN and DCB address are acquired and the scan pointer is adjusted for the next field. The next characters are compared for an "All" entry. If "All" is specified, control passes to the Line Change routine (IECKLNCH) at the "start all" entry point. If "All" is not specified, then control passes to the Line Change routine at the "start one line" entry point. When control returns, a test is made of the error flags for errors. If there are no errors, then the "serviced" bit in the MSTATUS field of the header prefix is set on so the buffer will be returned to the available buffer queue, and control passes to ENDRCV. If there was an error, the message is returned to the source terminal.

RSTOPLN (CHARTS EL AND EM): Control is passed here when a STOPLN control message is received or from the RINTREL routine when an INTREL control message is received. SUB1 is used to get the offset of the table entry for the terminal specified in termname, and the number of bytes remaining in the buffer. SUB2 is used to check for a

terminal entry, and if it is one, to get the relative line number and the DCB address. The size of the buffer is checked to be sure it is at least the minimum size. If less than the minimum, the message is returned to the source terminal. The line count is set to 1 and the line number is obtained from the QCB. The DCB address (acquired in SUB2) is obtained and the address of the source terminal is calculated. If the source terminal is in the line group to be stopped and either

1. "All" is specified, or
2. The source terminal is on the line to be stopped, then the message is returned to the source. Any of these conditions would cause the control terminal to become permanently inoperative.

If the source terminal is not on a line to be stopped, a test is made to be sure the DCB for that line group has been opened. If it has not, the message will be returned to the source.

The address of the DEB is obtained from the DCB and the number of extents is acquired from the DEB. A test is made to see if the relative line number of the line to be stopped is too high. If so, the message is returned to the source terminal.

If a STOPLN is being handled, the next field in the message is compared for the characters "All". If "All" is specified, the line number is set to one and the line index is set to the number of lines in that group.

If this routine is executed as a result of receiving a STOPLN operator control message, and "All" was specified, the line count index is set to the number of lines in the line group, and the relative line pointer is set to one so that the first line in the line group will be the first one stopped. If this routine is executed as a result of receiving an INTREL control message, or if "All" was not specified in a STOPLN control message, the line count index is one and the relative line pointer has the relative line number of the line to be stopped. These values will be used in a BCT loop to stop the desired number of lines.

For either STOPLN or INTREL functions, the registers are saved and the QCB for the Stop routine is posted to itself, causing it to be placed on the ready queue with highest-priority. An SVRB is built by the SVC-handling routine with the address of the instruction following the Qpost as the point to receive control. This SVRB is placed on the ready queue following the QCB for the line to be stopped (i.e., the QCB is at the head of the ready queue and the STCB is next.)

When the next item on the ready queue is dispatched, the STOP1 routine receives control. STOP1 will execute disabled to interrupts as a result of Qposting.

STOP1 restores the registers and places the address of the Operator Awareness routine (IECKOPAW) in the buffer recall/cleanup address in IGG019NG to insure that IECKOPAW is executed at buffer recall/cleanup time.

The size of an LCB is obtained from the DCB for that line group and is stored in the header for that message. The relative line number is also stored in the header. The relative line number is multiplied by the size of an LCB. This product is added to the DCBLCBAD (address of first LCB minus the size of an LCB) in order to point to the LCB for that line. A test is made to see if this line is active. If it is not, control is passed to the loop control code to determine if another line is to be stopped.

If this line is active, a test is made to see if there is an STCB for an operator control subtask pending for this LCB. If there is, the STCB is tested for a STOPLN function. If the STOPLN function STCB is the one pending, then it is ignored.

If this line was already stopped, or if a STOPLN subtask was pending for this LCB, the relative line index is incremented by one, and a BCT is executed on the line counter. If this is the last line to be stopped, then one is subtracted from the relative line number to get the relative line number of the last line stopped. A test is made to see if this routine was activated as a result of an INTREL control message. If it was not, a test is made to see if this is the first pass. If it is, the "serviced" bit is set in the buffer and the next element on the ready queue is dispatched. If it is not the first pass, the buffer is posted to the available buffer queue.

If there is no operator control STCB on the queue for that LCB, or if the operator control STCB is not a STOPLN STCB, access is gained to UCB for that line. If this is a dial line or 2740 (basic or with checking), and not in active transmission, a Halt I/O command is issued to clear the enable. If the CSW was stored, the Halt I/O command is repeated until the CSW is not stored.

If this is a WTTA line, and not an inactive transmission, a Halt I/O command is issued to clear the Prepare command. If

the CSW was stored, the Halt I/O command is repeated until the CSW is not stored.

If this is not a dial line nor a WTTA line, or if the dial line or the WTTA line is active, or if the CSW was not stored after the Halt I/O, then a first pass switch is tested. If this is the first time through for this message, the address of the STOP2 routine is placed in the LCBDESTQ field of the LCB so this buffer will be posted to STOP2 when the LCB for the operator control terminal goes through the Cleanup routine. Control is passed to the Qdispatch subroutine in IECKQQ01 to activate the next item on the ready queue.

The next item on the ready queue is the SVRB created as a result of Qposting the QCB for STOP to itself. The SVRB activates the instruction following the Qpost, which is a branch back to the macro. The macro branches immediately to the ENDRCV macro, which performs buffer cleanup. The buffer is posted to the destination queue, which in this case is the queue for the STOP2 routine, and STOP2 receives control.

When STOP2 receives control, the address of the LCB for this line is retrieved from the QCB for STOP2. An STCB to stop the line is dynamically built in the buffer. The message itself is of no use, so the buffer is used as a convenient place. The STCB is placed at the head of the STCB chain for the LCB. If RSTOPLN was entered from RINTREL, the INTREL switch is turned off, and the INTREL switch in the LCB is set, the next element on the ready queue is dispatched.

Subtask1 is activated when the LCB for the line to be stopped is posted to itself, thus indicating that that line was stopped. The STOPLN STCB is removed from the LCB's STCB chain, and the LCBSTATE is set to zero to deactivate the LCB. The address of the DCB is obtained from the LCB, and the address of the DEB is obtained from the DCB. If entry was from RINTREL, control is passed to Subtask2.

The following procedure will be followed only if this was a STOPLN control message. The address of the next STCB in the LCB chain is obtained and a test is made to see if it is a full STCB. If it is not, the LCB is removed from the ready queue. In either case, if there are more lines to be stopped, the procedure is looped through again to stop the rest.

If there are not more lines to be stopped, the buffer is returned to the available buffer queue and the next item on the ready queue is dispatched.

If the INTREL switch is set, control passes to the Subtask2 routine to put the buffer on the time queue.

RINTREL (CHART EN): Control is passed to RINTREL when an INTREL operator control message is received. A switch is set to indicate to the RSTOPLN that an INTREL control message is currently being handled.

RSTOPLN will handle the stopping of the desired line. After the LCB address and the relative line number have been retrieved from the buffer by Subtask1, a test is made for an INTREL function. If the message was an INTREL message, control is passed to Subtask2.

Subtask2 removes the LCB for the line to be stopped from the ready queue. The INTREL switch is turned off. The address of the STCB2 routine is stored into the STCB in the buffer, as is the LCB address. If this routine is entered by way of an I/O interrupt, an SVC QCB is set up and put on the ready queue. When it comes to the top of the queue, an SVC interrupt will occur, and control will be passed to HAVESVC. When Subtask2 is sure it was entered via an SVC interrupt, the buffer is made to look like an LCB and placed on the time queue for two minutes.

When the two-minute interval has elapsed, the buffer comes to the head of the time queue, is posted to the queue for the STCB2 routine, and STCB2 receives control. The address of the available buffer queue is acquired and the LCB is posted to itself. This action posts the buffer to the available buffer queue and puts the LCB back on the ready queue.

When the operator awareness routine detects an irrecoverable error on a line in INTREL mode, it acquires a buffer and posts that buffer to the queue for the STOP4 routine. The relative line number and the LCB address of the line are stored in the buffer. If the line is already inactive, the buffer is placed on the time queue. If the line is active, control is passed to a point in RSTOPLN to stop the line.

The TIMEQ subroutine is entered from RSTOPLN when an INTREL message is being processed and a STOPLN is pending on that line. TIMEQ turns off the INTREL switch and sets the INTREL switch in the LCB. The line is always active at this point. A switch is set in the buffer to denote INTREL. The next item on the ready queue is dispatched.

Module Name: IECKOCTL

Entry Point: Expansion of the OPCTL macro instruction generates a BALR to the Operator Control routine at IECKOCTL, using register 15 as the branch register, and register 14 as the return address register. Before branching, the macro-generated code checks the message to assure that it is from either the control terminal or the alternate. If the message is a control message, control will eventually return to the macro-generated code. The first instruction executed upon return is a BALR to the ENDRCV macro.

Register 1 is the parameter register. The parameter list will vary from 20-50 bytes. If all parameters are specified, the list will look like this:

4 bytes		Address of Operator Awareness routine
1 byte		Length of control message name identifier
1 byte	status byte	X'00' neither INTRCPT nor ALTERM specified X'01' ALTERM specified X'02' INTRCPT specified X'03' Both ALTERM and INTRCPT specified
2 bytes	Offset to control terminal	
2 bytes	Offset to alternate terminal	(included only if ALTERM is specified)
2 bytes	Offset to INTRCPT field	(included only if INTRCPT is specified)
Variable	Control message name identifier	

External Routines Used:

SCAN (IECKSCAN)

Stop Line-Start Line (IECKLNCH)

Look-up (IECKDRCT)

PAUSE ROUTINE (CHART C0)

Function: The address of the first byte to scan for a special device control character is computed for a header or text segment. If there has been a previous pause, the number of remaining insert blocks is obtained from COUNT. If there have been no previous insertions, COUNT (a defined constant in the Pause routine) is the number of available insert blocks. The buffer is scanned for the special character specified in the PAUSE macro. If the specified character is not found, return is made to the next LPS instruction.

When the special character is found, the COUNT is decremented by one. If COUNT reaches 0, indicating that the limit of the insert blocks has been reached, return is to the next LPS instruction. The routine issues a wait for an insert block, obtained from the queue defined by the queue control block (INSERTQ) assembled in the routine. The address of the queue for the insert block is placed in the last word of the insert block. To fill in the insert block, the address of the next BRB is stored, and the command and the address of the next character after the previous special character are placed in the insert block. The next block in the chain is updated by placing the address of the next character after the special character in the data address and by adjusting the count. The count of characters up to the special character is placed in the second word of the insert block. The address of the next block of the previous BRB is updated to point to the insert block. The flags, counts, and TIC command are placed in the insert block to complete this block. The routine loops back to scan for other special characters in the buffer.

Module Name: IECKPAUS

Entry Point: Expansion of the PAUSE macro instruction generates a BALR to the routine at IECKPAUS, using register 15 as the branch address register. The routine returns via register 3. Register 14 is used as a parameter list register. The parameter list passed to the routine consists of:

- The number of idle characters.
- The special characters.
- The bit configuration for the idle characters.

External Routines Used: Qwait (IGC065 in module IECKQQ01)

POLLING LIMIT ROUTINE (CHART CR)

Function: This routine limits the number of messages to be accepted from a non-switched terminal during one polling pass. If the polling pointer is not equal to the terminal entry for the current message or is at the end of the polling list, return is made to the next LPS instruction. Otherwise, the current poll count is compared to the limit specified. If the count, incremented by one, is less than the limit, return is made to the next LPS instruction. If the count, incremented by one, exceeds or is equal to the limit, the pointer is set to the next terminal before returning to the next LPS instruction. If

this is an autopolled line, the length of the next entry is obtained from the start of the polling list. If the entry is the end of the polling list, the pointer is set to the first entry.

Module Name: IECKPLMT

Entry Point: Expansion of the POLLIMIT macro instruction generates a BALR to the routine at IECKPLMT, using register 15 as the branch address register and register 14 as the return register. The parameter register 1 contains the maximum number of messages.

External Routines Used: None

REROUTE ROUTINE (CHART CS)

Function: This routine causes a message to be sent to an alternate destination, in addition to its normal routing, when any of the error conditions specified by the error mask is indicated in the error halfword, or when the error mask is zero. If the error mask is not zero, and none of the error conditions specified by the mask is indicated in the error halfword, return is made to the next LPS instruction. Linkage is made to the Recall routine (in module IGG019NG), which obtains the header. Upon return, the Reroute routine branches to the Lookup routine, which looks up the destination code in the terminal table and places the relative address in the TTDKEY field of the incoming header prefix. Return to the next LPS instruction is made by the Lookup routine.

Module Name: IECKRRTE

Entry Point: Expansion of the REROUTE macro instruction generates a BALR to the routine at IECKRRTE, using register 15 as the branch address register and register 14 as the return register. Register 14 also serves as a parameter list register. The parameter list passed to the routine consists of the address of the error mask in hexadecimal notation. The parameter register 2 contains the address of the alternate destination.

External Routines Used:

- Recall (IECKRC in module IGG019NG)
- Lookup (module IECKLKUP)

ROUTE ROUTINE (CHART CN)

Function: This routine links to the Scan routine to obtain the destination code in the incoming message header, and then branches to the Lookup routine, which looks up the destination code in the terminal table and places the relative address in the TTDKEY field of the incoming header prefix. Return to the next LPS instruction is made by the Lookup routine, rather than the Route routine.

Module Name: IECKROUT

Entry Point: Expansion of the ROUTE macro instruction generates a BALR to the routine at IECKROUT, using register 15 as the branch address register and register 14 as the return register. Register 14 also serves as a parameter list register. The parameter list passed to the routine consists of a halfword containing, in binary form, either:

1. The maximum size of each destination code in incoming message headers, or
2. All ones, indicating that the destination code fields are of variable length (the end of the field is indicated by a blank).

External Routines Used:

- Scan (module IECKSCAN)
- Lookup (module IECKLKUP)

SCAN ROUTINE (CHART CF)

Function: This routine obtains one or more nonblank characters from a fixed length or variable length header field and places them in a work area, the address of which is provided in a parameter register to the calling routine. The Scan routine moves the scan pointer one position at a time, and places the header characters pointed to into the work area. This operation is repeated until either the end of the field is reached or the work area is filled (work area size is eight bytes).

If the field to be scanned is of fixed length, its size is provided to the routine in a parameter list; the routine places into the work area the number of characters specified. During scanning, any blank characters encountered are passed over. They are not placed in the work area and they are not included in the count of characters maintained by the routine.

If the field to be scanned is of variable length, an indicator (2X'FF') is passed to the routine in a parameter list. If end of segment is reached before the specified scan length has been satisfied, bit five ("incomplete message header") is set in the error halfword. The Scan rou-

tine places all header characters up to the first blank in the work area.

Module Name: IECKSCAN

Entry Point: The routine is entered via BALR from the IECKPRTY,IECKSEQIN, IECKSKPS, IECKMODE,IECKROUT,IECKSRCE, and IECKTYPE modules; register 15 is the branch address register and register 3 is the return register. The address of a single-item parameter list is passed to the routine in register 14. The parameter list contains the field length or variable field length indicator.

External Routines Used: None

SEQUENCE IN ROUTINE (CHART CV)

Function: This routine links to the Scan routine to obtain the sequence number from the header. All characters in sequence are converted to binary notation and put into the MSNUMIN field of the header prefix. If the number is not in sequence according to the TSEQUIN field in the terminal table entry, the "sequence error" bit is set accordingly in the LCB. If the sequence number is too low, the routine sets the "too low" bit in the LCBERRST field of the LCB, and return is made to the next LPS instruction. If the sequence is too high, the "too high" bit is set in the LCBERRST field of the LCB and return is made to the next LPS instruction. If the number is in the correct sequence, the expected sequence number is stored in the LCBBRKCT field of the LCB. The sequence number from the header is also incremented by one and restored in TSEQUIN field of the terminal table for the next message before returning to the next LPS instruction.

Module Name: IECKSEQN

Entry Point: Expansion of the SEQIN macro instruction generates a BALR to the routine at IECKSEQN, using register 15 as the branch address register and register 14 as the return register. Register 14 also serves as a parameter list register. The parameter list passed to the routine consists of the address of the number of character positions for the input message sequence number. If this operand is omitted, a hexadecimal 'FF' indicates a variable length field.

External Routines Used: Scan (module IECKSCAN)

SEQUENCE OUT ROUTINE (CHART CM)

Function: If the SEQOUT macro appears in the Receive header portion of the LPS, the destination QCB is checked for being a process QCB. If this is not a process QCB, return is made to the next LPS instruction. If this is a Process QCB, linkage is made to the expand routine, which expands the header to create a new field. The value of the scan pointer is stored in the MSNUMOUT field of the header. Return is made to the next LPS instruction.

If this macro is in the Send header portion of the LPS, linkage is also made to the Expand routine, which "expands" the header by creating a new field whose high-order byte is the location pointed to by the scan pointer. The binary sequence number is obtained from the header prefix and converted to decimal form. The sequence number is unpacked and inserted into the new header field. Return is made to the next LPS instruction.

Module Name: IECKSEQT

Entry Point: Expansion of the SEQOUT macro instruction generates a BALR to the routine at IECKSEQT, using register 15 as the branch address register and register 14 as the return register. Register 14 also serves as a parameter list register. The parameter list passed to the routine consists of the address of the number of character positions for the output sequence number.

External Routine Used: Expand (module IECKEXPD)

SKIP (CHARACTER COUNT) ROUTINE (CHART CJ)

Function: This routine advances the scan pointer from its current position past a specified number of nonblank header characters. The pointer then points to the last nonblank character needed to complete the count. Moving the scan pointer causes all characters bypassed to be ignored during header processing. If the scan pointer reaches the end of the segment prior to exhausting the specified count, bit five, "incomplete message header," is set in the error halfword.

External Routines Used: None

Module Name: IECKSKPC

Entry Point: Expansion of the SKIP macro instruction specifying a number of characters to be skipped generates a BALR to the routine at IECKSKPC, using register 15 as the branch address register and register 14 as the return register. Register 14 serves

also as a parameter list register. The parameter list passed to the routine consists of a halfword containing the number of characters to be skipped.

SKIP (CHARACTER SET) ROUTINE (CHART CJ)

Function: This routine advances the scan pointer from its current position past all header characters up to and including a specified character sequence. The scan pointer then points to the last character in the sequence. Moving the scan pointer causes all characters bypassed to be ignored during header processing. If the Scan routine returns an error indication in LCBERRST, the Skip routine clears the multiple routing indicator in the LCB and points the scan pointer to the end of the buffer. Return is made to the next LPS instruction.

Module Name: IECKSKPS

Entry Point: Expansion of the SKIP macro instruction specifying a particular sequence of characters to be skipped generates a BALR to the routine at IECKSKPS, using register 15 as the branch address register and register 14 as the return register. Register 14 also serves as a parameter list register. The parameter list passed to the routine consists of:

1. A halfword containing the length of the character sequence to be found;

2. A character constant containing the characters to be found.

External Routines Used: Scan (module IECKSCAN)

SOURCE ROUTINE (CHART CI)

Function: This routine determines the validity of the source code field of an incoming message header. The routine links to the Scan routine to obtain the source code from the header.

If the message orginated from a non-switched terminal, the contents of the source code field are compared with the name of the originating terminal (as the name appears in the terminal table entry). If the characters match, return is made to the next LPS instruction because the source code is considered valid.

If tne message originated from a switched terminal or an autopolled line, the contents of the source code field are compared with each terminal entry name in the terminal table until a match is found. If a match is found, the addresses of the DCB, obtained from the LCB, and the destination QCB of the terminal table are compared. If the source is in the same line group then it is considered valid. Therefore, for dial lines only, the priority of the Send Scheduler is set to 2 to prevent a dial. This priority had been initialized to 1 by the expansion of the TERM macro. The routine places the relative address of the source entry in the TTSKEY field of the header prefix and in the LCBTTIND field of the LCB. Return is made to the next LPS instruction.

If no match is found or the switched line was not in the line group specified, the code is considered invalid. If the source code is invalid, the routine sets bit 6 ("invalid source code") of the error halfword for the line to 1. Control returns to the next LPS instruction.

Module Name: IECKSRCE

Entry Point: Expansion of the SOURCE macro instruction generates a BALR to the routine at IECKSRCE, using register 15 as the branch address register and register 14 as the return register. Register 14 also serves as a parameter list register. The parameter list passed to the routine consists of a halfword containing a source code field-length indicator.

External Routines Used: Scan (module IECKSCAN)

TIME STAMP ROUTINE (CHART CK)

Function: This routine obtains the current time in packed decimal format (via a TIME macro), unpacks the time, and inserts a specified portion of the time information in the message header. Prior to inserting the time, the routine links to the Expand routine, which "expands" the header by shifting to the left all message characters from the end of the prefix plus the count up to and including the character pointed to by the scan pointer. The number of character positions by which the header characters are shifted is equal to the length of the time information to be inserted. The time is then inserted in the field created. The maximum field size is nine characters in the format bhh.mm.ss, where b = blank, hh = hours, mm = minutes, and ss = seconds. The scan pointer points to the last character in the time. Smaller field sizes have a similar format, truncated from the right.

Module Name: IECKTIME

Entry Point: Expansion of the TIMESTMP macro instruction generates a BALR to the routine at IECKTIME, using register 15 as

the branch address register and register 14 as the return register. Register 14 also serves as a parameter list register. The parameter list passed to the routine consists of a halfword containing, in binary notation, the length of the time information field to be inserted in the message header.

External Routines Used: Expand (module IECKEXPD)

TRANSLATE ROUTINE (CHART CP)

Function: This routine translates message segments from one code to another. The number of characters to be translated is computed by subtracting the address of the first byte to be translated from the address of the end of the segment. If the number is not negative, the message is translated using the table specified in the macro. Return is made to the next LPS instruction.

Module Name: IECKTRNS

Entry Point: Expansion of the TRANS macro instruction generates a BALR to the routine at IECKTRNS, using register 15 as the branch address register and register 14 as the return register. The parameter register 1 contains the address of the translation code table named in the operand of the macro.

External Routines Used: None

The Open and Close routines are in the transient area. The expansion of these macro instructions is a system expansion. The system Open and Close routines branch to the QTAM Open and Close routines via an XCTL from the where-to-go-table.

CLOSE COMMUNICATIONS LINE GROUP ROUTINE (CHART EB)

Module Name: IGG0203N

Function: The Close Communications Line Group routine is entered from the system Close routine. This routine obtains the address of the DCB being closed. From this DCB and the associated DEB, the routine calculates the size and starting address of the LCB. A FREEMAIN is issued for the main storage occupied by the LCB. The LCB and the IOB pointers and status bytes are cleared in the DCB. Completion of this executor is indicated in the where-to-go table. If the routine is to be used again for another DCB, a branch is taken to the beginning of the routine. The normal exit of this routine is an XCTL to the next non-zero entry in the where-to-go table.

CLOSE DIRECT ACCESS MESSAGE QUEUE ROUTINE (CHART EC)

Module Name: IGG0203O

Function: The Close Direct Access Message Queue routine is entered from the system Close routine. If the CLOSE is for a checkpoint data set, a test is made for normal completion. If an abnormal completion is set in the TCB address, the next entry in the where-to-go-table is obtained. If normal completion, the TTR is converted to a disk address. A CCW is set up in the Checkpoint routine to write the control record to indicate that there was normal completion. A EXCP is issued to write the record. If, upon completion, there was a permanent I/O error, the routine is ended abnormally with an error code of 0A4. Otherwise, the next entry in the where-to-go table is obtained.

If the Close routine is entered for a direct access message queue data set, the routine clears the terminal table entry in the communications vector table. The address of the first available buffer is moved into the link field of the first available request. The IOB address is reset in the DCB.

If there are any items in the ready queue, each element is tested for a QCB. If it is a QCB and items are present in the chain of STCBs, a test is made for a full STCB. If the item on the ready queue or in the chain is a full STCB, the ECB address is obtained. The routine then links to the OS Post routine to post the ECB as complete and to remove the STCB. When there are no more items on the ready queue, the entry in the where-to-go table is cleared. If the routine appears again in the where-to-go table, it is ended with a dump because there is only one DCB for the direct access device. The normal exit of this routine is to the next nonzero entry in the where-to-go table.

CLOSE PROCESS QUEUE (INPUT AND OUTPUT) ROUTINE (CHART EA)

Module Name: IGG0203P

Function: The Close Process Queue routine is entered from the operating system Close routine. The DEB for the DCB being closed is searched for in the chain of processing program DEBs. If the DEB for the DCB is not found, a branch is taken to the end of the routine to set up for a new entry. If the DEB is found, it is removed from the message queue DEB chain and from the TCB chain of DEBs.

If the DCB is for input, the unprocessed header segment is placed into the disk queue and the unprocessed text segments are placed into the unavailable buffer queue by a Qpost. When the last dummy element is reached, a branch is taken to the common part of the program. If the DCB is for output and there is a remaining buffer, this buffer is returned to the available buffer queue by Qpost. If an LCB is found in the chain, the dummy LCB in the DEB is removed from the destination queue.

For both input and output DCBs, if there are no more DCBs to close, a test is made for general closedown. If a general closedown exists, each DEB in the chain is obtained. If the associated DCB is for communications, the routine prepares to stop each line represented in the extents. If the line is a dial line or an active transmitting line, the Halt I/O is skipped; otherwise, the line is stopped with a Halt I/O. The routine waits if the line is active. At the end of the chain of DEBs, the message control partition is requested to issue a CLOSE.

If there are more DCBs to close or it is not a general closedown, the current entry is cleared in the where-to-go table. If this routine is to be used again for another DCB, a branch is taken to the beginning of the routine. The normal exit of the routine is an XCTL to the next nonzero entry in the where-to-go table.

LINE GROUP OPEN EXECUTOR - LOAD 1 ROUTINE (CHART F1)

Module Name: IGG0193N

Function: This routine is entered from the system Open routine. The routine obtains the length of the DEB by adding the minimum size of a DEB, and four bytes for the extents of each device. The routine then performs a GETMAIN to obtain storage for the DEB. The DEB is cleared to zeros and then initialized. The pointer to the STCB of the dial out-call queue, located at DEB -28, and the STCB chain pointer are initialized to point to DEB -20. The priority field is set to X'FF' to indicate a dummy last element. The address of the basic DEB is stored in the TCB and the DCBDEBAD field of the DCB. The DCB address is stored in the DEBDCBAD field of the DEB. If the number of buffers is less than or equal to two, the DCBBUFRQ field of the DCB is set equal to two. The UCB information is moved from the TIOT to the DEB. If the direct access device has not been opened, an ABEND is issued with the completion code of 0A6.

Analysis of the device type set in the UCB is done on the telecommunication devices found in the UCBs. Error codes are set for incorrect specifications, and the program is ended with a dump. Tests are performed on the model type (1060, 2740, 1050, and 1030) for each terminal adapter. Error codes are set if adapter type or terminal is found to be in error, and the program is ended with a dump. Tests are made for the optional features of Auto Call, Auto Poll, terminal-to-terminal, or Auto Answer for the IBM 1060, 2740, 1050, and 1030. If the correct optional bits are not set, an error code is set and the program is ended with a dump. The device code used for the vector directory is set.

The WTTA Line Appendage module IGG019QB is loaded into main storage, and its address is set in the DEB appendage table. Linkages are established with the QTAM Line Appendage routine. The "WRU" bits of the LPS (LPSTART macro expansion) are moved into the DCB.

If this routine is required again, the routine branches to the beginning of this routine. Upon completion, control is passed via an XCTL to the Line Group Open Executor - Load 2 Routine.

LINE GROUP OPEN EXECUTOR - LOAD 2 ROUTINE (CHART F2)

Module Name: IGG0193R

Function: This routine is entered by an XCTL from the Line Group Executor-Load 1. Tests are continued for model types 83B3, 115A, TWX, and 2260 for each terminal adapter. Error codes are set if adapter type or terminal is found to be in error, and the program is ended with a dump. Tests are made for optional features of Auto Call, Auto Poll, terminal-to-terminal, or Auto Answer for 83B3, 115A, TWX, and 2260.

To the minimum size of an LCB (88 bytes) is added the size of the channel program for a particular device. A GETMAIN is done for an LCB for each line in the group. The address of the LCB minus the size of an LCB is stored in the DCBLCBAD field of the DCB, and the size of the LCB is stored in the DCBEIOBX field of the DCB. The IOB address is stored in the DCBIOBAD field of the DCB, and the device type is inserted in the DCBDEVTP field of the DCB. The IOB is initialized.

For an IBM 2702 adapter, a channel command word is built in the channel program area with the correct SAD command. Except for an IBM 2701 with type III adapter, if there is no Auto Call or Auto Answer feature, a CCW with an Enable command code is set in the channel program area. If there is no Auto Call or Auto Answer and a restart is in progress, a READ Skip command, for a TTY device, or a Break command, for other devices, is set in the CCW.

If the line is being opened for input, the receive scheduler STCB is initialized. For input and output, the return code, priority, and line start indication are set for posting the LCB to start the line.

If a restart is to be done, a search is done on the saved data. If the LCB was saved, the saved data is restored; otherwise, the line is treated as usual. If the line was checkpointed, it is only restarted if it was active. The dummy ECB address is stored in the IOB and the address of the next LCB is obtained. This process continues for all lines.

Upon completion, the routine tests for another DCB to be opened. If there is another DCB, a branch is made to the start of the Load 2 routine. The routine exits to Line Group Executor - Load 3 via an XCTL.

LINE GROUP OPEN EXECUTOR - LOAD 3 ROUTINE (CHART F3)

Module Name: IGG0193T

Function: This routine is entered by an XCTL from the Line Group Executor - Load 2 routine. The identification, relative track number, and record address of the BTAM Read/Write module is set for the Load subroutine. The Load subroutine loads the BTAM Read/Write module for use by QTAM. The offset to the channel identification table, record address, and relative track number are obtained to load in the Device I/O module. The model channel program for the device specified in the DCBDEVTP field of the DCB is loaded into the channel program area.

If a restart is in progress, a search of the terminal table is made for a destination QCB. If the DCB specified in the QCB is not the current DCB, the search is continued for another destination QCB. The address of the LCB is obtained for the DCB being opened.

If the "send" bit is not on, this is a receive-only terminal, therefore setting up the Send Scheduler is skipped.

If the "send" bit is on for this terminal, the send scheduler STCB is set up in the QCB. If the Send Scheduler is already in the QCB, the search is continued on the terminal table. If the address of the next segment is equal to the next message, one less segment is put in the link field of the QCB. If the line is trying to send, the address of the header is set in the QFAC field of the QCB.

If there is an incoming message, the header is read from the disk and the "cancel" bit is set to cancel the message. The header is rewritten on the disk. This is done for each DCB to be opened.

Each line is started by issuing an EXCP on the channel program. After all lines have been started, the next DCB is gotten. The routine exits to Line Group Executor-Load 4 via an XCTL.

OPEN LINE GROUP EXECUTOR LOAD 4 ROUTINE

Module Name: IGG0194A

Function: This routine is entered by an XCTL from the Line Group Executor - Load 3 routine. The time of day is obtained by the TIME macro instruction and saved in the routine. A test is made on each line to determine if I/O has completed. If I/O has not completed on a line, the time is obtained until there has been a 30-second delay from the time of entry to the routine. If the line still has not completed I/O after 30 seconds, an error message is put to the console.

IEC806I ENDING STATUS NOT RECEIVED FROM LINE XXX - LINE UNAVAILABLE

If I/O has not completed on any of the remaining lines, a message is also sent for each line.

If the line had completed I/O during the 30-second interval, the test continues on the remaining lines.

If I/O has completed on all lines, or after all messages have been written, an XCTL is taken to the next nonzero entry in the where-to-go table. If this routine is required again, a branch is taken to the beginning of the routine.

For WTTA lines, the LCB fields are completed as follows:

1. The LCBTTIND field is updated with the offset of the associated TERM entry of the terminal table. If this entry does not exist, the program is ended with a dump.

2. The LCBPOLPT field is updated with the address of the associated POLL macro instruction.

OPEN DIRECT ACCESS MESSAGE QUEUE ROUTINE (CHART F4)

Module Name: IGG0193O

Function: This routine is entered from the system Open routine. The size of the DEB is calculated by adding the basic size, appendage size, and the size of the extents. This routine issues a GETMAIN for the DEB and initializes it. The terminal table address is obtained from the DCB and stored in the communications vector table. The Implementation module is loaded into main storage. For each device type and each extent, the routine determines the number of bytes required for each record (other than the first) on a track for this device. The number of records that will fit on each track is determined.

If this is an OPEN for a checkpoint device, a test is made to ensure that the direct access data set has been opened. If it has not been opened, the program is ended abnormally. The module for checkpoint/restart (IGG019NH) is loaded by using the load subroutine in the Open module. The address of IGG019NH is stored in the Implementation module (IGG019NG) and the terminal table address is stored in

IGG019NH. The track length and overhead value are saved in IGG019NH. The number of tracks for the extent in the DEB is calculated and stored in the DEB. The Open Checkpoint Records Data Set routine address is set in the where-to-go table.

If this is not an OPEN for a checkpoint data set, the Direct Access-Load 2 address is set in the WTG table.

If this routine is needed for another DCB, a branch is taken to the start of the routine. Upon completion, control is passed to the next nonzero entry in the where-to-go table via an XCTL.

OPEN DIRECT ACCESS-LOAD 2 (CHART F5)

Module Name: IGG0193U

Function: Entered from the system OPEN, this routine initializes the Implementation module with the TCB address, priority, and master receive switch. The message processing DEB chain is zeroed. For each terminal table entry that is not a distribution list, the routine stores the address of the DASD destination STCB in the link field and priority in the send scheduler of the QCB. If the QCB is for a process queue, the DASD destination STCB address is set in the chain pointer of the STCB. If there are no more entries in the terminal table, the relative record number available for the next segment is determined and stored in the Implementation module for placement in the QCB. The address of the first buffer is stored in the available buffer QCB, the IOB address in the DCB, and the DCB address in the IOB.

A subroutine and STCB in this routine are moved into the OPEN work area. An STCB is set up for this subroutine. This STCB is posted to cause the subroutine to be activated. This subroutine obtains the address of IECKQQ01 (obtained from the base register set up as a result of the Qpost) and stores the address of the Implementation module.

If the DSCBs in the chain have not been freed, a FREEMAIN is issued to free main storage for these DSCBs.

If operator control has been specified in the terminal table, an address constant in the Implementation module is changed from the address of Buffer Recall/Cleanup routine to the address of Operator Awareness routine. For all the entries in the terminal table, the polling list address is obtained and tested for Auto Poll (bit 7 of the fourth byte is one). If this is an Auto Poll polling list, the polling characters and index bytes replace the offset value in the polling list.

If the where-to-go table indicates that this routine is required again, the program is abnormally ended because there is only one DCB for the direct access device for message queues. Normal completion of the routine is an XCTL to the next nonzero entry in the where-to-go table of the system OPEN.

OPEN CHECKPOINT RECORDS DATA SET ROUTINE (CHARTS F6 AND F7)

Module Name: IGG0193V

Function: This routine is entered by an XCTL from the QTAM Open Direct Access Message Queue routine when a DD card specifying the checkpoint records data set is entered. The name of this DD statement must be TPCHKPNT.

The first function performed is that of calculating the size of the checkpoint records. A scan of the terminal table provides access to all control information to be counted. Included in the size are: each terminal table entry, each polling list, 11 bytes for each destination QCB, 14 bytes for each process QCB, and 11 bytes for each LCB. After the checkpoint record size has been computed, the total size and offsets to each type of data are saved in the Checkpoint routine for later use by that routine.

A test is then made to determine if the disposition of the checkpoint records data set is old or new. If it is new, a four-byte control record is written in the first record of the data set. The data set is then further formatted by writing two complete dummy checkpoint records. Exit is then made via an XCTL.

If the data set disposition is old, the four-byte control record is read from the disk to determine if a restart procedure should be initiated. If the first byte of the control record is zero, the checkpoint records data set was properly closed and no restart is necessary. The procedure described for formatting the data set is performed, and exit is made by an XCTL.

If the first byte of the control record is nonzero, the checkpoint records data set was not properly closed. This indicates a system failure, and a restart procedure must be performed. A GETMAIN macro is issued to obtain a work area into which is read the current checkpoint record. The control record contains an indicator as to which checkpoint record is the most recent.

The checkpoint record is then read from the disk into the work area. A scan through the terminal table is again used to locate the control blocks and tables that must be restored with the data contained in the checkpoint record. At this point, the terminal table entries, polling lists, and QCBs are restored with the data previously recorded in the checkpoint record. Restoring of the LCB data is deferred until the line groups are opened because the storage required for the LCBs is not obtained until that time. A code of X'F2' is set in the TERMTBL field of the Checkpoint module to indicate that a restart procedure has been initiated. Exit is then made via an XCTL.

All of the disk I/O operations initiated in this routine are accomplished via an EXCP/WAIT sequence. If an error occurs on any disk operation, the job is terminated via an ABEND macro. If a checkpoint record cannot be contained on a single track, as many EXCP/WAIT sequences as are needed are issued to read or write the entire logical record. Linkage is generated to the Convert routine to convert a TTR to an actual DASD address.

The checkpoint interval or number of CKREQ macros specified in the TERMTBL macro is stored in the Checkpoint module prior to giving up control via an XCTL.

OPEN MESSAGE PROCESSING PROGRAM ROUTINE (INPUT AND OUTPUT) (CHART C4)

Module Name: IGG0193P

Function: This routine is entered from the system Open routine. If no TRMAD field has been specified in the DCB, the program is abnormally ended with a dump. An error code is set and the program is ended with a dump when any of the following conditions exist:

- The direct access device has not been opened.
- A MS process queue DCB is being opened and the SOWA field has not been specified.
- The DDNAME is not found in the terminal table.

A GETMAIN for the DEB is issued for 144 bytes, and part of the DEB is initialized. The DEB is chained to the chain of processing program DEBs. If an MS process queue DCB is being opened, the routine sets up the QCB and BRB in main storage obtained for the DEB. If an MS destination queue DCB is being opened, the routine sets the priority and link address of the BRB in the DEB. The address of the Get or Put module is obtained according to the mode (message, record, or segment), and the Get or Put module is loaded. The basic part of the DEB is initialized. If the routine is required again, a branch is taken to the beginning of the routine; otherwise, an XCTL is executed to the next nonzero entry in the where-to-go-table found in the system Open routine.

The Get and Put routines are in the partition that contains the message processing program. The expansion of these macro instructions is a system expansion, which branches to the QTAM routine. The remainder of the macros are used to examine and to modify the status of the control program. The expansion of these macro instructions link to a corresponding routine for the macro.

GET MESSAGE ROUTINE (CHART C6)

Module Name: IGG019NB

Function: The Get Message routine is entered by the system expansion of a GET macro, which obtains the address of the Get module from the DCB specified in the GET macro. If this is the first entry for a process queue, the routine posts a dummy buffer (in DEB) to the return buffer queue. If this is not the first entry, or upon returning from the POST, the routine checks for the inclusion of EODAD by the user. If it has been specified and the disk is not in the process of reading, a test is made for a dummy last element. If there are no more messages in the MS process queue, the routine branches and links to the user's exit address.

If EODAD was not specified, or the disk was in the process of reading, or there are messages in the MS process queue, the routine obtains the address of the work area. If this is not the first entry, the routine posts the previously used buffer to the return buffer queue. Upon returning from the post, or if this is the first entry, the routine issues a Qwait to obtain the next buffer.

If it is a header segment, the address of the source area is obtained. If the prefix indicates a process queue, the area specified by the TRMAD operand is cleared. The terminal ID is moved into the user's area.

If a header or text segment (but not the last segment) and the buffer is empty, the routine branches back and obtains another buffer. If the buffer is empty and it is the end of message, a zero is set in DCBSEGAD, data count is set in the work area, and return is made to the next instruction in the processing program. If the buffer is smaller than or equal to the work area, a test is made for end of message. If it is not end of message, a branch is taken to request another buffer; otherwise, a zero is set in DEBSEGAD, data count is set in the work area, and return is made to the next instruction in the processing program. If the work area is filled, data remains in the buffer, and no SYNAD is specified, a zero is set in DCBSEGAD, data count is set in the work area, and return is made to the next instruction in the processing program. If the SYNAD is specified, the routine branches and links to the user's synchronous exit routine before returning.

GET RECORD ROUTINE (CHART C7)

Module Name: IGG019NC

Function: The Get Record routine is entered by the system expansion of a GET macro, which obtains the address of the Get module from the DCB specified in the GET macro. If this is the first entry for a process queue, the routine posts a dummy buffer (in DEB) to the return buffer queue. If this is not the first entry, or upon returning from the POST, the routine checks for the inclusion of EODAD by the user. If it has been specified and the disk is not in the process of reading, a test is made for a dummy last element. If there are no more messages in the MS process queue, the routine branches and links to the user's exit address.

If EODAD was not specified, or the disk was in the process of reading, or there are messages in the MS process queue, the routine obtains the address of the work area. If this is not the first entry, the routine posts the previously used buffer to the return buffer queue. Upon returning from the post or if this is the first entry, the routine issues a Qwait to obtain the next buffer.

If it is a header segment, the address of the source area is obtained. If the prefix indicates a process queue, the area specified by the TRMAD operand is cleared. The terminal ID is moved into the user's area.

If the segment is not the last and the buffer is empty, the routine branches back to obtain another buffer. If the buffer is empty and it is the end of a message, the data count is set in the work area, and return is made to the next instruction in the processing program.

Each character is moved into the work area, if space is available. If there is no space available and SYNAD is specified, the count is stored in the work area. The program branches and links to the user's synchronous exit routine before returning to the next instruction in the processing program. If SYNAD is not specified, the count is stored in the work area, and the routine returns to the next instruction in the processing program. Each character in the buffer is checked for new line (NL), end of block (EOB), or start of text (STX for 2260). If the character is an NL, EOB, or STX, all consecutive NLs, EOBs, or STXs are moved into the work area, the count is stored in the work area, and the routine returns to the next instruction in the processing program. If the buffer is empty and it is not an end of message, the routine branches to request another buffer. If the buffer is empty and it is an end of message, the data count is stored in the work area, and the routine returns to the next instruction in the processing program.

GET SEGMENT ROUTINE (CHART C5)

Module Name: IGG019NA

Function: The Get Segment routine is entered by the system expansion of a GET macro, which obtains the address of the Get module from the DCB specified in the GET macro. If this is the first entry for a process queue, the routine posts a dummy buffer (in DEB) to the return buffer queue. If this is not the first entry or upon returning from the POST, the routine checks for the inclusion of EODAD by the user. If it has been specified and the disk is not in the process of reading, a test is made for a dummy last element. If there are no more messages in the MS process queue, the routine branches and links to the user's exit address.

If EODAD was not specified, or the disk was in the process of reading, or there are messages in the MS process queue, the routine obtains the address of the work area. If this is not the first entry, the routine posts the previously used buffer to the return buffer queue. Upon returning from the post, or if this is the first entry, the routine issues a Qwait to obtain the next buffer.

If it is a header segment, the address of the source area is obtained. If the prefix indicates a process queue, the area specified by the TRMAD operand is cleared. The terminal ID is moved into the user's area.

If the buffer is empty, the data count is stored in the work area, and return is made to the next instruction in the processing program. If the buffer is not empty, the segment is moved to the work area. If all the data in the buffer has been moved, the data count is stored in the work area and return is made to the next instruction in the processing program.

If the work area is filled and data is left in the buffer, a check is made for SYNAD specification. If SYNAD is not specified, the routine returns to the next instruction in the processing program; if specified, it branches to the user's synchronous exit routine before returning.

PUT MESSAGE ROUTINE (CHART DA)

Module Name: IGG019NE

Function: The Put Message routine is composed of two parts. The first section is entered by a branch and link from the system macro expansion, which obtained the address from the DCB specified. This section sets priority for the BRB and sets a new entry switch.

The following error codes are set in register 15 when an error is detected.

1. Bit 26 is set to 1 for an invalid terminal name.

2. Bit 27 is set to 1 for wrong length specified.

If any error flags are set, return is made to the next instruction in the processing program. The routine then issues a Qpost to request a new buffer from the active BRB queue, waits for the buffer, and posts the buffer to the destination queue. If there is any more data in the work area, the routine branches back to request a new buffer; otherwise, it returns to the next instruction in the processing program.

The second section of the routine is entered by a branch from the Buffer BRB routine. If this is the first PUT for this buffer, a new entry is set and the buffer size is loaded. For every request, a 0 is set in the source key (TTSKEY) in the prefix to show a process queue. If this is a header segment, the sequence number is zeroed and the scan pointer in the prefix is set to the number of idle characters in the header. If the user has specified priority, that priority is taken from the work area and placed in the DEB; otherwise, a blank is set for the priority. The offset to the terminal entry is stored in the TTDKEY field, and the EOM header bit is set in MSTATUS field of the prefix.

For a text segment, the EOM bit is set in the MSTATUS field of the prefix. For a header or text segment, the length of the buffer is compared with the work area. If the work area is larger than the buffer, the EOM bit in MSTATUS is turned off and the buffer length is used to move the data from the work area into the buffer. The work area length is used for the data move if the work area is smaller than the buffer. If there is more data to move, the work area pointer is updated before branching to the Interim LPS routine to post the buffer to the MS destination queue.

PUT RECORD ROUTINE (CHART C9)

Module Name: IGG019NF

Function: The Put Record routine is composed of two parts. The first section is entered by a branch and link from the system macro expansion, which obtained the address from the DCB specified in the PUT macro. This section sets priority for the BRB and sets a new entry switch.

The following error codes are set in register 15 when an error is detected.

1. Bit 26 is set to 1 for an invalid terminal name.

2. Bit 27 is set to 1 for a wrong length specified.

3. Bit 25 is set to 1 for invalid sequence; that is, if this is the last segment and the next segment is not a header, or if this is not the last segment and the next segment is a header (other than the first time).

If any error flags are set, return is made to the next instruction in the processing program.

The routine then issues a Qpost to request a new buffer from the active BRB queue, and waits for this buffer. If a buffer address has been saved, use this buffer and save the address of the new buffer. The Put Record routine issues a Qwait to wait for the BRB to be removed from the ready queue. If the buffer is full, the routine posts the buffer to the DASD destination queue. If there is more data to be moved, this routine branches to request a new buffer. If no more data is to be moved, or the buffer is not full, it switches buffer addresses with the saved buffer. If there is an extra buffer, the unused buffer is posted to the free buffer queue before return is made to the next instruction in the message processing program.

The second section of the routine is entered by a branch from the Buffer BRB routine. If this is the first PUT request for this buffer, a new entry is set, the buffer size is loaded, and the buffer save area is zeroed. The present buffer is saved for every PUT request if no buffer is left over from a previous PUT. The LCB address, source key, and text indicator are placed in the prefix.

If this is the start of the work area and a header segment, the sequence number is zeroed and the scan pointer in the prefix is set to the number of idle characters in the header. If the user has specified priority, that priority from the work area is placed in the DEB; otherwise, a blank is set for the priority. The offset to the terminal entry is stored in TTSKEY field of the prefix.

For a text or header segment and not the start of a work area, the length of the buffer and the work area are compared if the buffer has been filled. If the buffer is larger than the work area, the length of the work area is used to move the data; otherwise, the buffer length is used. The data is moved from the work area to the buffer and the data count is set in the prefix. If data is left in the work area, the address of the next character in the work area is obtained. If no data is left in the work area and it is the end of message, the EOM is set before the address of the next character is obtained.

If a buffer is left over from the previous PUT, the extra buffer address is stored. If this is not a text segment, the EOM is set in MSTATUS. The switch is set to indicate that no buffer is left over. If a buffer has been saved, the saved buffer and present buffer are exchanged. The routine exits to the Interim LPS routine in the Implementation module to post the buffer to the MS destination queue.

PUT SEGMENT ROUTINE (CHART C8)

Module Name: IGG019ND

The Put Segment routine is composed of two parts. The first section is entered by a branch and link from the system macro expansion, which obtains the address from the DCB specified in the PUT macro.

The following error codes are set in register 15 when an error is detected.

1. Bit 26 is set to 1 for an invalid terminal name.

2. Bit 27 is set to 1 for a wrong length specified.

3. Bit 25 is set to 1 for invalid sequence; that is, if this is the last segment and the next segment is not a header, or if this is not the last segment and the next segment is a header (other than the first time).

If any error flags are set, return is made to the next instruction in the processing program.

Otherwise, the Put Segment routine sets priority for the BRB, and issues a Qpost to request a new buffer from the active BRB queue, waits for the buffer, and then posts the buffer to the DASD destination queue. The routine waits for the BRB to be removed from the ready queue, and returns to the next instruction in the processing program.

The second part of the routine is entered by a branch from the Buffer BRB routine. This section of the routine moves the LCB address in the MSLCB field, moves the message type in the work area to the MSTATUS field, and zeros (shows process queue) into the TTSKEY field of the prefix. If it is a header, the message sequence number is set to zero, and the scan pointer in the prefix is set to the number of idle characters in the header. The header is moved in the buffer. If the user has specified priority, that priority is placed from the work area into DEB; otherwise, the priority is set to a blank. The offset to the terminal entry is stored into TTDKEY field (destination key) of the prefix. For a text segment, the text is moved into the buffer. For a header of text segment, the segment size is stored in the MSEGSZE field of the prefix. The routine exits to the Interim LPS routine in the Implementation module to post the buffer to the MS destination queue.

CHANGE POLLING LIST ROUTINE (CHART CD)

Function: This routine sets up the DCB base with the DCB address given in the macro. If the terminal name is specified, the macro expansion has branched to IECKDCBL to find the address of the DCB. If the DCB specified has not been opened, an error code of hex '01' is set in register 15, and return is made to the next instruction in the processing program. If the relative line number specified is too high, an error code of hex '08' is set in register 15, and return is made to the next instruction in the processing program. If character number is specified in the third operand of the macro, the numerics are moved into the STATUS field of the polling list by the Cross Partition Move routine.

If the third operand is an address, the length of the new polling list is compared to the present one. If they are not equal, an error code of hex '10' is placed in register 15, and return is made to the next instruction in the processing program. Otherwise, the routine obtains the address of QMOVE and posts the QCB to itself to execute the move. The new polling list in the area specified is moved to the address of the polling list area. A normal completion code of hex '00' is placed in register 15 before return is made to the next instruction in the processing program.

Module Name: IECKCHPL

Entry Point: Expansion of the CHNGP macro instruction generates a BALR to the routine at IECKCHPL, using register 15 as the branch address register and register 14 as the return register. Parameter register 0 passes to the routine the address of the DCB with the relative line number in the high-order byte. Parameter register 1 contains the address of the area that contains the new polling list of the character number, either 0 or 1, which results in deactivation or activation of the polling list, respectively.

External Routines Used:

- Qpost (IGC067 in module IECKQQ01)
- Cross Partition Move (QMOVER +6 in module IGG019NG)

CHANGE TERMINAL TABLE ROUTINE (CHART CB)

Function: The terminal name specified in the macro is compared with each TERMID field in the terminal table. If the terminal entry specified is not found in the terminal table, an error code of hex '20' is set in register 15, and return is made to the next instruction in the processing program. When the entry is found, the length of the entry is compared to that of the present entry. If the size specified in the work area is not equal to the size in the terminal table, an error code of hex '10' is set in register 15, and return is made to the next instruction in the processing program.

If the QCB from the terminal table is not a destination queue or a STOPLN has been issued, the move data QCB is posted to itself to execute the move. The new entry from the work area is moved into the specified entry of the terminal table. If the QCB is a destination queue and a STOPLN has not been issued, the change is made in two moves, leaving the new sequence number. Because the sequence numbers may have been incremented after the entry was copied into

the work area, the old sequence numbers are not changed. Return is made to the next instruction in the processing program.

Module Name: IECKCHGT

Entry Point: Expansion of the CHNGT macro instruction generates a BALR to the routine at IECKCHGT, using register 15 as a branch address register. Parameter register 0 contains the address of the work area. Parameter register 1 passes to the routine the address of the name of the entry in the terminal table.

External Routines Used:

- Qpost (IGC067 in module IECKQQ01)
- Cross Partition Move (QMOVER +6 in module IGG019NG)

CHECKPOINT REQUEST ROUTINE (CHART C3)

Function: This routine initiates a request for a checkpoint record to be written on the checkpoint records data set.

If any of the following error conditions are detected, the request is ignored and return is made to the calling program with an error code in register 15, right-adjusted.

1. QTAM message queues data set not opened (error code = X'01').
2. A checkpoint interval was specified in the CKITV operand of the TERMTBL macro (error code = X'02').
3. Checkpoint records data set not opened (error code = X'04').

If no errors are detected, a checkpoint is requested by Qposting the passed ECB to the checkpoint request queue. An SVC WAIT is then issued to wait for the checkpoint to be taken. Return is made to the calling program.

Note: The checkpoint record is not written until the number of message processing partitions specified in CKPART operand of the TERMTBL macro have initiated checkpoint requests.

Module Name: IECKCKRQ

Entry Point: Expansion of the CKREQ macro instruction generates a BALR to the routine at IECKCKRQ, using register 15 as the branch address register and register 14 as the return register. Upon entry, parameter register 1 contains the address of an event control block (ECB) representing the checkpoint request from this partition.

External Routines Used:

- Qpost (IGC067 in IECKQQ01)
- WAIT (SVC 1)

CLOSE MESSAGE CONTROL ROUTINE (CHART ED)

Function: This routine is entered for a complete closedown. To turn off the master receive switch, the routine sets up for the Cross Partition Move routine by placing the address of the master receive switch and the mask in registers 4 and 5, respectively. The routine posts the move data QCB to itself; the Move Data subtask moves the mask to turn off the master receive switch.

The TCB for message control is used to obtain the DEB chain. For each DEB on the chain, the associated DCB is refered to. If the DCB is not for a communications line, the next DEB is obtained. If the DEB is for a line, each LCB is obtained. If the line for that LCB is active, the routine issues a STOPLN macro instruction to stop the line. The STARTLN macro is then issued to put out all messages. Only the lines for output will be started because the master receive switch has been turned off. If the line is not active, the next LCB is obtained. When the end of the DEB chain is reached, the routine returns to the next message processing program instruction. The net effect is that all input lines to the system are stopped, while line output operations continue as normal.

Module Name: IECKCLOS

Entry Point: Expansion of the CLOSEMC macro instruction generates a BALR to the routine at IECKCLOS, using register 15 as the branch address register and register 14 as the return register.

External Routines Used:

- Qpost (IGC067 in module IECKQQ01)
- Line Change (in module IECKLNCH)
- Cross Partition Move (QMOVER + 6 in module IGG019NG)

COPY TERMINAL TABLE ROUTINE (CHART CG)

Function: After saving the registers, the routine obtains the address of the terminal table from the communications vector table. The terminal table is searched for the name of the entry specified in the macro. If no entry of the specified name is found or the entry size is zero, an error code of a hex '20' is placed in register 15. When the name is found, the table is moved to the work area specified in the macro, and a

normal completion code of hex '00' is placed in register 15. After restoring registers, the routine returns to the next processing program instruction.

Module Name: IECKCPYT

Entry Point: Expansion of the COPYT macro instruction generates a BALR to the routine at IECKCPYT, using register 15 as a branch address register and register 14 as a return register. Parameter register 0 contains the address of the work area specified in the macro. Register 1 contains the address of a location containing the terminal name.

External Routines Used: None

COPY POLLING LIST ROUTINE (CHART CC)

Function: After saving the registers, the routine obtains the size of the polling list, using the relative line number and the address of the DCB specified in the macro. If the terminal name is specified, the macro expansion has branched to IECKDCBL to find the address of the DCB. If the DCB has not been opened, an error code of hex '01' is set in register 15, and return is made to the next instruction in the processing program. If the relative line number is too high, an error code of hex '08' is set in register 15, and return is made to the next instruction in the processing program. The polling list is moved into the work area specified in the macro. Registers are restored before return is made to the next instruction in the processing program.

Module Name: IECKCPPL

Entry Point: Expansion of the COPYP macro instruction generates a BALR to the routine at IECKCPPL, using register 15 as the branch address register and register 14 as the return register. Parameter register 0 passes, to the routine, the address of the DCB specified in the macro plus four times the relative line number. Register 1 contains the address of the work area that contains the new polling list.

External Routines Used: None

COPY QUEUE CONTROL BLOCK ROUTINE (CHART CE)

Function: After saving the registers, the routine searches the terminal table for the name of the terminal specified in the macro. Upon obtaining the address of the QCB from the terminal table, the queue of 32 bytes is moved into the area specified by the macro. If the terminal entry is not found, an error code of a hex '20' is placed in register 15. If there was no error, the routine returns a hex '00' in register 15. The routine restores registers and returns to the next LPS instruction.

Module Name: IECKCPYQ

Entry Point: Expansion of the COPYQ macro instruction generates a BALR to the routine at IECKCPYQ, using register 15 as a branch address register and register 14 as a return register. Parameter register 0 contains the address of the work area specified in the macro. Parameter register 1 passes, to the routine, the address of a location that contains the terminal name.

External Routines Used: None

LOCATE DCB ROUTINE (CHART BW)

Function: The routine obtains the maximum size of the terminal name. If the specified terminal name is not in the terminal table, an error code of hex '20' is set in register 15 and return is made to the macro expansion. If the terminal name is found, the QCB address is obtained from the terminal entry. If this is a list or process entry, an error code of hex '20' is set in register 15 and return is made to the macro expansion. If it is not a list or process entry, the DCB address is obtained from the QCB and placed in register 0. The relative line number is inserted in the high-order byte. The normal exit code of hex '00' is set in register 15 and return is made to the macro expansion.

Module Name: IECKDCBL

Entry Point: This routine is entered via a BALR from the macro expansion of STOPLN, STARTLN, COPYP, or CHNGP. The address of the terminal name is passed in parameter register 1.

External Routines Used: None

RELEASE INTERCEPTED MESSAGE ROUTINE (CHART BZ)

Function: Each TERMID field of the terminal table is compared with the specified terminal name, until there is an equal compare. If the name is not found in the terminal table, the routine returns to the next instruction in the processing program with an error code of hex '20' in register 15. If the entry is found, the TSTATUS field is tested. If the "intercept" bit is not on, return is made to the next instruction in the processing program with a code of hex '04' in register 15. If the intercept bit is on, indicating that messages

may have been intercepted, the address of the INTERCPT field is obtained from the LPS routine.

If the message header address in the INTERCPT field is greater than the message address in queue, a priority message has been intercepted. The "intercept" bit is reset to zero, the "send" bit is set on, and return is made to the next instruction in the processing program with the code of hex '00' in register 15 for normal completion. If the header address is less than the address in queue, the header address of the intercepted message is inserted (using Cross Partition Move routine) as the first message to be released. This is done by posting the move data QCB to itself. The "intercept" bit is reset to zero, the "send" bit is set on, and return is made to the next instruction in the processing program with a code of hex '00' in register 15 for normal completion.

Module Name: IECKRELM

Entry Point: Expansion of the RELEASEM macro instruction generates a BALR to the routine at IECKRELM, using register 15 as the branch address register and register 14 as the return register. Register 1 is the parameter register, which passes the address of the terminal name to the routine.

External Routines Used:

- Cross Partition Move (QMOVER+6 in module IGG019NG)
- Qpost (IGC067 in module IECKQQ01)

RETRIEVE - DASD ROUTINE (CHART C1)

Function: This routine causes a message segment to be retrieved by direct access address from the DASD destination or process queues, and to be placed into the work area. The routine saves registers and sets up addressability. If an invalid disk address is received, an error code of hex '02' is set in register 15, and return is made to the next instruction in the processing program.

A combination BRB and QCB is built in the user's work area. The routine stores the address of the STCB, queue-insert-by-priority, into the QTRAN field. It stores the direct access address in the BRB as the relative record address of the next segment to be read. The routine then sets the MSTATUS field equal to 9 in the BRB and sets priority in the QPRI field to a hexadecimal 'E4'. The routine posts the BRB/QCB to the disk queue, and waits for disk completion. Another Qwait is issued to ensure that the BRB/QCB is off the ready queue so that the work area can be used. The message is moved into the work area specified by the user. The buffer is returned by posting the buffer to the available buffer queue. The registers are restored and return is made to the next instruction in the processing program.

Module Name: IECKRETD

Entry Point: Expansion of the RETRIEVE macro instruction generates a BALR to the routine at IECKRETD, using register 15 as the branch register and register 14 as the return register. Registers 0 and 1 are used as parameter registers. Register 0 contains the work area specified by the user, and register 1 contains the relative record address of the message segment to be retrieved.

External Routines Used:

- Qwait (IGC065 in module IECKQQ01)
- Qpost (IGC067 in module IECKQQ01)

RETRIEVE BY SEQUENCE NUMBER ROUTINE (CHART C2)

Function: This routine causes a message segment to be retrieved and placed in a work area specified by the user. After saving registers and setting up addressability, the terminal table address is obtained from the CVT. The terminal table is searched for the destination named in the operand of the macro instruction. If the name of the destination is not found, an error code of a hexadecimal '20' is placed in register 15, and return is made to the next instruction in the processing program. If the name is found, the offset of the entry from the start of the terminal table is saved, and the destination queue address is obtained from the terminal table entry.

The routine branches to the Retrieve DASD routine passing in register 1 the direct access address of the message (a negative address indicates sequence in and a positive address indicates sequence out) and in register 0 the address of the work area.

The Retrieve DASD routine retrieves the next segment of the message and places it in the work area. If the direct access address is zero, an error code of hexadecimal '40' is placed in register 15, and return is made to the next instruction in the processing program. If the sequence number specified by the macro instruction is greater than the sequence of the retrieved message or it was a priority mes-

sage, the next message is obtained by the Retrieve DASD routine. If the sequence number of the retrieved message is larger than the one specified, an error code indicating invalid sequence of hexadecimal '40' is set in register 15. Return is made to the next instruction in the processing program. If the correct message is retrieved and registers are restored, return is made to the next instruction in the processing program with a hexadecimal '00' set in register 15.

Module Name: IECKRETS

Entry Point: Expansion of the RETRIEVE macro instruction generates a BALR to the routine at IECKRETS, using register 15 as the branch address register and register 14 as the return register. Register 0 is used as a parameter register. It contains the address of the work area into which the message segment is to be placed.

External Routines Used: Retrieve DASD (in module IECKRETD)

START LINE - STOP LINE ROUTINE (CHART BX)

Function: This routine sets up the DCB base with the DCB address given in the macro. If the terminal name is specified, the macro expansion has branched to IECKDCBL to find the address of the DCB. If the DCB has not been opened, an error flag of hex'01' is set in register 15, and return is made to the next instruction in the processing program. If the relative line number is greater than the number of lines, an error code of hex '08' is set for invalid relative line number, and return is made to the next instruction in the processing program.

For each line to be stopped or started, the associated LCB is obtained. If the routine was entered for a Start Line and is not active, the operation codes for the SAD and Enable commands, needed for starting a line, are stored in the QCB/STCB. Unless there is a type III adapter, 2260, or switched connection, an Enable operation code is set. Otherwise, NOP is set in the QCB. The LCB is posted to the queue QCB to enter the subtask in the routine.

If the routine is for a stop line, the UCB address for the line is obtained. If the line is a dial line or a WTTA line and is not in active transmission, a Halt I/O is issued to disable the line. If it is in active transmission, a Qwait is issued to wait for the line to become inactive. For an autopolled line, the line is stopped by causing the TIC after the second Poll CCW to be replaced with a NOP. The move data QCB is posted to itself to cause the NOP to replace the TIC across partitions. If there are no more lines to change, the normal exit code of hex '00' is set in register 15, and return is made to the next instruction in the processing program.

The Queue routine associated with the queue subtask is in this module. This routine takes the Op code set in the Start line-Stop Line routine and places it in the channel program area. A flag is set in the LCBCPA + 32 field of the LCB so that Line SIO Appendage will give control to ERP at completion. An EXCP is issued for the line. Upon return, the routine exits to Qdispatch subroutine in IECKQQ01.

Module Name: IECKLNCH

Entry Point: Expansion of the STOPLN or STARTLN macro instruction generates a BALR to the routine IECKLNCH, using register 15 as the branch address register and register 14 as the return register. Parameter register 0 passes to the routine the relative line number in the high-order byte and the DCB address in the three low-order bytes.

External Routines Used:

- Qpost (IGC067 in module IECKQQ01)
- Qwait (IGC065 in module IECKQQ01)

QTAM CONTROL MODULE SUBROUTINES

The QTAM control module (module IECKQQ01), consisting of nine subroutines, is included in the supervisor nucleus as a resident routine at system generation.

ENTRY INTERFACE SUBROUTINE

This subroutine performs initialization for the QTAM control program. It is entered from the first-level Interrupt Handler (FLIH) of the supervisor whenever a QTAM supervisor call (Qwait or Qpost) is issued.

Associated with each entry to the Entry Interface subroutine is a supervisor-created supervisor request block (SVRB); the SVRB is converted to a subtask control block (STCB). One otherwise unused word in the SVRB is zeroed and is later used as an event control block (ECB) for controlling the dispatching of its associated subtask.

The "new" STCB is placed at the head of the STCB chain of the QCB for the last dispatched queue (i.e., the queue from which QTAM last activated a subtask). The subroutine then exits to the Qwait or the Qpost subroutine, depending on which SVC was issued.

QTAM POST (QPOST) SUBROUTINE

The Qpost subroutine places the address of the QCB named by the calling routine into the QCB address field of the specified resource element control block (this is the means by which an element becomes associated with a QCB). The subroutine then branches to the Priority Search subroutine to cause the element to be placed on the ready queue in priority order.

QTAM WAIT (QWAIT) SUBROUTINE

When the Qwait subroutine is entered, the STCB representing reentry to the calling routine (when the wait condition is satisfied) has already been chained into a QCB by the Entry Interface subroutine. The Qwait subroutine determines what further disposition should be made, based on current conditions, to schedule the subtask for activation. Four sets of conditions determine the disposition:

1. If the QCB into which the STCB has been chained (i.e., the user-specified or "new" QCB) has a key of 2 (the highest-priority subtask is "not waiting"), the Qwait subroutine makes no further disposition, but branches immediately to the Defer Entry subroutine at UNAVAIL. Reasons for this branch are explained in the discussion of that subroutine.

2. If the new QCB has an element available on its element control block chain, the STCB remains linked into the QCB (the "old" QCB) that had been selected by the Entry Interface subroutine. The Qwait subroutine then branches to the Exit Select subroutine at RETURNX. This causes the address of the element found on the element chain of the new QCB to be placed in the calling routine's parameter register, which is itself stored in the save area of the STCB. The net effect is that at the time the subtask is activated, it appears in the STCB chain of the Qattach QCB, or the STCB chain of the dispatched (old) QCB; the element chain from which it is drawing elements, however, is that of the QCB specified by the calling routine (i.e., the new QCB). This action insures immediate satisfaction of the wait condition when the requested element is available.

3. If the new QCB has no elements available, but the last dispatched queue (old QCB) and the queue specified by the calling routine (new QCB) are the same, the STCB is already chained into the correct QCB and that QCB is already waiting on the ready queue. The Qwait subroutine branches to the Qdispatch subroutine.

4. If the new QCB has no elements available and is not the QCB for the last dispatched queue, the STCB must be linked into the STCB chain of the new QCB; therefore, the Qwait subroutine branches to the Defer Entry subroutine at UNAVAIL.

The Wait subroutine is also entered at a special entry point, UNAVAIL-6, by the BRB Ring and Send Scheduler routines in the Implementation module. The purpose is to determine if the last dispatched QCB and a QCB specified by the calling routine are the same.

DEFER ENTRY SUBROUTINE

This subroutine causes entry to a subtask to be deferred. When a control subroutine encounters an STCB for a subtask that cannot be activated, a branch to the Defer Entry subroutine is taken; this subroutine causes the STCB to be removed from the position at which it was encountered and linked into the appropriate STCB chain. After retrieving the pointer to the STCB from the location where it was encountered and restoring that location to its former state, the Defer Entry subroutine branches to the Priority Search subroutine; this causes the STCB to be placed, by priority order, into the STCB chain of the QCB specified by the calling routine.

An exception arises if the key of the QCB specified by the calling routine is 2. This condition indicates that the highest-priority subtask on the QCB's STCB chain is a ready subtask (not waiting for elements) and is ready to receive control. The STCB being processed, however, is not ready; if it is of higher priority than the ready subtask, it cannot be placed at the head of the STCB chain without preempting the "ready" status that applies to the current top STCB, and that STCB should be honored first for maximum efficiency. Therefore, the Defer Entry subroutine enters the Priority Search subroutine by a path that ensures that the new STCB is enqueued by priority order below the current top STCB.

PRIORITY SEARCH SUBROUTINE

This is a generalized subroutine that determines the position within a chain that an item should assume in order to be in correct priority sequence; items in the chain are arranged in descending order of priorities from the top of the chain. This subroutine acts on all chains including the ready queue.

The subroutine examines each item on the chain until it finds either an item with lower priority than that of the search argument, or the last item on the chain (signalled by priority 255). When either condition is met, the subroutine exits to the Queue Insert subroutine.

QUEUE INSERT SUBROUTINE

This is a generalized subroutine that links items into a chain; it is applied to all chains including that of the ready queue. When this subroutine is entered, a register contains a pointer to the link address portion of the item at the point in the chain at which the new item is to be inserted; a second register holds the address of the item to be inserted. (The point of insertion is the head of the chain except when this subroutine is entered from the Priority Search subroutine, which selects the insertion point according to the item's priority.) The subroutine places the old link address in the new item, replaces the old link address with the new item's address, and exits to the Qdispatch subroutine.

QDISPATCH SUBROUTINE

This subroutine performs the primary internal management function within QTAM, except for those cases in which another subroutine is able to determine the next subtask to be activated (e.g., when the Qwait subroutine finds that elements are already available to a subtask requesting elements and that the Qdispatch subroutine can be bypassed). The Qdispatch subroutine maintains continuity by receiving control from a completed subtask and by selecting another subtask that is to receive control.

The Qdispatch subroutine examines the item at the head of the ready queue and takes one of four courses of action, depending on the type of item encountered. Items that can appear on the ready queue are:

1. Queue control blocks for which the highest-priority subtask is not waiting for elements (QCB key is 2).
2. Queue control blocks waiting for elements (QCB key is 3).
3. Resource element control blocks containing the address of the QCB to which the element has been posted (RECB key is 0).
4. Full subtask control blocks for which the key value is also zero. The first word of a full STCB contains the address of the Qattach QCB.

The effect of the appearance of each type of item at the head of the ready queue is described in the following paragraphs.

Queue Control Block - Not Waiting (Key Is 2): When the item at the head of the ready queue is a "not waiting" QCB, control is given to the first (highest-priority) subtask represented in the QCB's STCB chain, and the QCB's key is set to 3.

Queue Control Block - Waiting (Key Is 3): A "waiting" QCB at the head of the ready queue is removed from the ready queue (i.e., replaced by the item linked to it), and its key is set to 1. A QCB waiting for elements cannot contend for control; how-

ever, it is automatically returned to the ready queue when an element becomes available.

When a subtask requiring more than one element (e.g., a series of buffers) to accomplish its function receives control, the associated QCB continues to appear as "waiting" (key is 3) until all required elements have been received. Before a waiting QCB is removed, it is determined whether the QTAM subtask that had control last was associated with that QCB. If it was, that subtask is again given control. This cycle continues until the subtask fulfills all of its requirements or until the subtask exhausts the queue's element chain.

Resource Element Control Block: Each resource element control block (RECB) that has been posted to the ready queue contains the address of the QCB for the queue to which the element has been posted. When an element reaches the top of the ready queue, it is immediately replaced by the QCB to which it points. However, the QCB pointer in the RECB is retained. That QCB is then treated as though it, rather than an element associated with it, had been encountered; its highest-priority subtask is activated, and its key is set to 3.

This convention has several significant aspects:

1. It is the means by which a removed "waiting" QCB is returned to the ready queue.

2. It illustrates the case where the active QCB (i.e., the QCB with which the active subtask control block is associated) is not necessarily at the head of the ready queue.

3. It explains the fact that an RECB need not be physically chained into a QCB to become associated with that QCB. Specifically, it ensures that an element is immediately acted upon, except in the case where the queue involved already has at least one other "real" element and is already contending for computing time.

Full Subtask Control Block: This is the only form of STCB that appears on the ready queue. Its appearance at the head of the ready queue has exactly the same effect as the appearance of a "not waiting" (key=2) QCB with this STCB at the head of its STCB chain; the subtask is activated and the key of the QCB with which it is associated (Qattach) is set to 3.

The mechanism by which this is accomplished is as follows:

1. Location READY contains a pointer to the STCB; this situation is the physical counterpart of the STCB's being at the head of the ready queue.

2. The STCB itself has the appearance (to the QTAM control routines) of an element. Its QCB address is QATTACH.

3. QATTACH is a storage location equivalent to READY minus 8 bytes; it also appears to be the first word of a "not waiting" QCB.

4. Since the STCB is apparently a resource element control block associated with a "not waiting" QCB, the first STCB in that QCB's chain should be selected for activation. The address of the first STCB is to be found in the third fullword of the QCB.

5. The third fullword of the QATTACH, which appears as a QCB, is the location READY. Therefore, the full subtask whose address is at READY is selected for control.

EXIT SELECT SUBROUTINE

This subroutine activates subtasks represented by truncated STCBs or falls through to the Exit Interface subroutine if the STCB is a full STCB.

The first byte of a truncated STCB is a branch modifier of the form (entrypt-NRET), where entrypt is the address of the desired entry point. NRET is the location from which the branch offset is applied. When the Exit Select subroutine encounters a nonzero return code, it computes the branch address and branches to the computed entry point.

EXIT INTERFACE SUBROUTINE

This subroutine is entered to process full STCBs. First the subroutine determines whether or not the subtask is being scheduled for activation because it was represented in the STCB chain of a waiting QCB for which an element has been encountered. If this condition exists, the address of the element is placed in the parameter register in the save area of the full STCB.

The subroutine links to the operating system Post routine, which posts completion in the event control block of the STCB (SVRB) being dispatched. The subroutine

then exits in one of two ways, depending on how the Qdispatch subroutine was entered:

1. If entry to Qdispatch resulted from an asynchronous interrupt, the subroutine branches back to the I/O supervisor.
2. If entry to Qdispatch resulted from an SVC, the subroutine issues a wait on the resource element control block of the entry SVRB. After the wait is satisfied, the subroutine returns to the routine that issued the SVC.

The QTAM Implementation module (module IGG019NG), consisting of 21 routines, is loaded into main storage by the Open Executor used to open the direct access queues data set.

RECEIVE SCHEDULER ROUTINE (CHART DH)

If the line is a WTTA line, this routine tests the EOT flag. If this flag is set, exit is made to the Defer Entry subroutine at UNAVAIL in order to enter the line's Send Scheduler subtask (if any). If the EOT flag is not set, exit is made to the BRB-Ring routine at RQCONST to initialize for receiving.

If the line is not a WTTA line, this routine examines the current polling list entry for a line. If polling is to be performed on the line and the current entry is valid (i.e., is not the dummy entry signalling the end of the polling list), the routine branches to the BRB-Ring routine (RQCONSTR) to initialize for receiving. Before the branch is executed, the routine sets the 'line receiving' code (LCBSTATE=8) in the LCB to indicate the kind of operation anticipated.

If the current polling list entry is the dummy last entry, the routine clears the line receiving code and branches to the Defer Entry subroutine at UNAVAIL after resetting the current entry pointer to the top of the polling list. The purpose of this branch is to permit the line's Send Scheduling subtask to become eligible for activation if its STCB is also in the chain. If the Send Scheduling subtask is not in the chain, the Receive Scheduling subtask will again be activated to start polling at the top of tne polling list.

Before the branch to the Defer Entry subroutine is taken, a possible endless loop is avoided by a test to determine that the polling list contains at least one entry in addition to the dummy last entry. If the list contains no true entry and the next STCB is a full STCB, the LCB is removed from the ready queue and its address is stored in the save area of the STCB. This is done in case of a closedown. The routine branches to the Exit Interface routine to post the ECB completed. If the list contains no true entry and the next STCB is not a full STCB, the Receive Scheduling subtask is left at the head of the STCB chain but is skipped over. Control passes to the second subtask in the chain.

BRB-RING ROUTINE (CHART DI)

This routine constructs the BRB ring used to send or to receive a message, and begins initialization of a CCW in each BRB. The BRBs are drawn from the pool generated on expansion of the BUFFER macro instruction; the routine attempts to form a ring containing the number of BRBs specified in the BUFRQ parameter of the DCB.

As an extension of the Receive Scheduler routine, the BRB-Ring routine checks for messages on the dial out-call queue. If the line is a dial line and the terminal has transmitted all messages, the dial out-call queue for the line is obtained in the DEB. The STCB chain is searched for a message whose relative line number is equal to or less than the relative line number of the free line. If the terminal for the STCB is connected to the free line, the Send Scheduler is removed from the dial out-call queue and inserted into the STCB chain for the line by IECKQQ01.

If the terminal is not free, the search is continued for a message with a relative line number that is less than the relative line number of the current line. At the end of the chain, the dial digits are obtained and the line is set to allow a dial up. The STCB is removed from the dial out-call queue chain and inserted into the STCB chain for the line.

If no messages are found, the line was not a dial line, or the terminal could not be disconnected, the LCB is set to receive and the BRB ring is constructed.

When the routine is entered, a register is adjusted so that the line control block appears to the system to be an STCB. This anticipates the situation in which not enough BRBs are available to complete the ring; in this case, the LCB is placed on the STCB chain of the active buffer request queue (through a branch to UNAVAIL-6, the instruction preceding the Defer Entry subroutine). When a BRB is posted to that queue, the BRB-Ring routine makes another attempt to complete the ring. When sufficient BRBs are available, the resulting BRB ring consists of a series of BRBs, each containing: (1) in the third fullword, the transfer-in-channel operation code and the address of the preceding BRB/CCW in the ring, and (2) in the fourth fullword, a pointer to the LCB for which the ring was constructed.

Since each BRB/CCW contains a transfer-in-channel to the previously built BRB/CCW, the TIC address in the first BRB/CCW is initially meaningless. The last step in completing the ring (if enough BRBs were available) is, therefore, to reset the first BRB/CCW to transfer-in-channel to the last one. If the order of construction of a four-member BRB/CCW chain was A-B-C-D, the order of execution will be A-D-C-B.

When the BRB/CCW ring is completed, the LCB is removed from the location where it was encountered as an apparent STCB (that is, from the head of an STCB chain or the ready queue). Depending upon whether a Send or a Receive operation is being prepared for, further initialization is performed:

1. The element control block portion of the first BRB/CCW is given a priority value:

   RECEIVE - 12
   SEND - 0

2. Into the LCB is inserted an operation-type code for subsequent use by BTAM:

   RECEIVE - 1 (Read Initial)
   SEND - 2 (Write Initial)

3. A register is initialized for the QCB of the queue to which the first BRB/CCW is to be effectively (but not literally) posted:

   RECEIVE - Active buffer request queue
   SEND - Disk input/output queue (for Send operations, additional initialization consists of setting an MSTATUS code of 9 and of inserting the relative record number for this first segment of the message.)

ACTIVE BUFFER REQUEST ROUTINE (CHART DL)

This routine is entered on activation of the active buffer request subtask. The element passed to the routine is an active BRB; the routine determines whether a buffer to satisfy the request is available and should be assigned, or whether the active BRB should be enqueued for later servicing.

If the active BRB represents the beginning of a BRB ring to be used for a receive operation, the routine removes a buffer from the element chain of the available buffer QCB and exits to the Buffer BRB routine. Parameters passed to that routine are: the address of the active BRB, the address of the removed buffer, and the address of the available buffer QCB.

If the active BRB is not the first of a ring for a Receive operation, or if it is the first but no buffer is available, the routine branches to the Priority Search subroutine to cause the active BRB to be enqueued on the element chain of the active buffer request queue.

AVAILABLE BUFFER ROUTINE (CHART DM)

This routine is entered on activation of the available buffer subtask, or from the Buffer BRB routine. The routine responds to the availability of a buffer by attempting to locate an available BRB. If no BRB is available, the buffer is chained into the element chain of the available buffer queue through a branch to the Queue Insert subroutine. If a BRB is available, this routine branches to the Buffer BRB routine.

BUFFER BRB ROUTINE (CHART DN)

This routine is entered from either the Active Buffer Request routine or the Available Buffer routine. Its function is to examine a buffer request block and to make the appropriate disposition of the buffer depending upon the status of the BRB.

1. If the BRB is associated with a Read from DASD operation, the routine effectively (but not through an SVC) posts the BRB to the disk I/O queue and the buffer to the available buffer queue.

2. If the BRB is associated with a Read from line operation, the routine assigns the buffer to the line and exits to the Interim LPS routine to cause the buffer to be placed on the LPS queue.

3. If the BRB is associated with a PUT operation, this routine branches to the Put routine, which places the data into the buffer.

DISK I/O ROUTINE (CHART D2)

This routine is entered on activation of the disk input/output subtask. The routine chains message-filled buffers (for disk writes) and BRBs (for disk reads) to the element chain of the disk input/output queue, and issues the SIO command (through an SVC 0) to write on or read from disk. Before issuing the Start I/O command, the routine converts the relative record number used by QTAM into a relative track address, and then branches (through a BALR) to a module of the basic partitioned access method to convert the relative track address to an actual DASD address.

DISK END APPENDAGE (CHARTS D0 AND D1)

This routine is an I/O appendage entered from the I/O supervisor following a DASD Read or Write operation. In a Receive operation, the routine routes the empty buffer required for the next segment to be received to the available buffer queue. In a Send operation, the routine routes the message-filled buffer to the LPS queue, initializes the next BRB in the ring to read the next segment of the message, and if a buffer has been assigned, routes it to the disk input/output queue.

LPS CONTROL ROUTINE (CHART DO)

When entered, the LPS Control routine issues an immediate SVC to Qwait on the LPS queue for:

1. An available buffer into which a message segment is to be read.
2. A buffer containing a text or header segment (that is, a message-filled buffer) that has been read or is to be written.
3. The last segment of a message after it has been written.
4. A request to start a disk I/O operation.
5. A request for a closedown.

When an available "first" buffer is encountered, the routine exits to the Activate routine to cause receipt of the message to be initiated. When a full buffer or the last buffer is encountered, the routine branches to the beginning of the line group routine defined by the user through LPS macro instructions. If the LCB is for checkpoint, an SVC Qpost is issued to post the LCB. Upon encountering a request for a closedown, the routine returns to the problem program at the instruction following ENDREADY.

ACTIVATE ROUTINE (CHART DP)

This routine initializes for a communications line Read or Write operation and branches to the BTAM Read/Write routine. An exception arises if a Send operation is scheduled for a terminal not eligible for receiving; an error status code is set in the LCB, and the routine exits to the user's LPS routine.

If the line is a WTTA line with the receiving code in the LCB, and if the EOT flag is set, the Read Initial operation code is set in the LCB (01 in LCBCECB); if the EOT flag is not set, the Read Continue operation code is set (03 in LCBCECB).

Before entering BTAM, the routine initializes the LCB to route the received message segment to a queue of messages having erroneous destination information. This routing information will be overlaid if valid destination information appears later.

LINE SIO APPENDAGE ROUTINE (CHART DQ)

This routine is entered from the supervisor EXCP handler after an EXCP (SVC 0) has been issued by BTAM, but before an SIO command has been issued. The routine modifies the BTAM generated channel program to meet QTAM requirements.

If this routine was entered at Open time, flags are set to indicate ERP is in control. Return is made to IOS, which will give control to the Open and Checkpoint routines.

When this routine is entered, BTAM has generated a channel program consisting of several channel commands including a Write Data or Read Data CCW. QTAM has created a ring of BRB/CCWs, each containing the PCI flag and a transfer-in-channel command to the following BRB/CCW. The routine links these two channel programs together by altering the flags in the BTAM Read or Write Data CCW and by adding a transfer-in-channel command to the second QTAM BRB/CCW (see Figure 24).

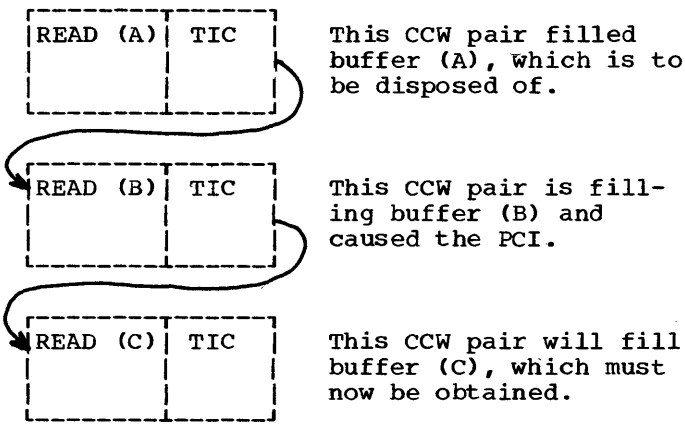
When the channel program is executed, the first buffer is transmitted under the BTAM Read or Write Data CCW. If QTAM has already scheduled a second buffer (usually this is the case), a transfer-in-channel to the QTAM CCW takes place, and the second buffer begins to fill. The PCI flag in the QTAM CCW causes the Line PCI Appendage to be entered as filling of the second buffer begins. (If a second buffer is not available, QTAM sets the PCI flag in the BTAM CCW.)

If the channel program involved is for an initial Read, the Line SIO Appendage also replaces the polling character pointer in the BTAM Write Polling Characters CCW with a pointer compatible with the QTAM polling technique. If the CCW is for an autopolled line, a header indicator is set and the polling pointer is not adjusted. If polling has been suppressed for the terminal, QTAM replaces the Write Polling Characters CCW with a No-poll CCW.

When the required adjustments have been made, QTAM branches back to the EXCP Handler to cause the SIO command to be issued.

LINE PCI APPENDAGE ROUTINE (CHART DR)

This routine is entered when a program controlled interrupt occurs during the execution of a QTAM channel command for the line. The function of the routine is to dispose of the buffer filled or emptied by the channel command preceding that which caused the PCI, and to place a request for the buffer that is to be emptied or filled by the CCW when it is again encountered in the ring. For an autopolled terminal, the routine adjusts the poll pointer to enable the terminal that has messages to send, to be repeatedly polled until it has no more messages or the limit is reached. Graphically:



For receive operations, buffer (A) is routed to the interim LPS queue; for send operations, the buffer is routed to the available buffer queue. In either case, the request for buffer (C) is routed to the active buffer request queue.

LINE END APPENDAGE ROUTINE (CHARTS DS AND DT)

This routine is an I/O appendage entered on channel end during line I/O operations or by the WTTA Line End Appendage when a channel end has occurred on a WTTA line. Normally, the routine routes a message-filled buffer to the LPS queue or exits to the supervisor to restart the channel program. When entered because of a negative response to polling, the routine resets the polling list pointer to the next entry in the polling list before initiating restart.

If the routine was entered due to a SAD or Enable command, return is to IOS if IDLE was specified; otherwise the LCB is posted to start the line. If there is terminal test activity the buffer is posted to the LPS queue.

If this routine was entered from the WTTA Line Appendage and if no program check has occurred, return is made to the WTTA appendage.

If the buffer is a header on an autopolled line, the indication is cleared. If the CCW is for a Read Text, the routine links to the Line PCI Appendage to adjust the polling pointer. Otherwise, a test is made for possible errors.

If there were no errors, a test is made for an autopolled line. If there is a message to send and it is either send priority or end of polling list, the message is sent by posting to the interim LPS queue.

If the status, or unit exception (not for a Read Response to polling CCW or enable) is not a normal indication, return is made to IOS to call in Error Recovery Procedures. If the completion code is not normal and the SIO condition code is 3, return is made to the LPS Control routine to free the buffer.

This routine may also be entered as a result of a program check occurring because a buffer was not provided on time, or because a CCW with a zero data count was accessed. The two low-order bits of the TIC command in each BRB/CCW are used to indicate BRB status. When a buffer has been allocated, these bits are set to zero. Because of timing considerations, a PCI flag in the CCW preceding a CCW containing a TIC may not interrupt the channel program before the transfer-in-channel command is executed. If this happens before the required buffer has been allocated and the BRB status code has been cleared, the requirement that the TIC address be on a doubleword boundary is violated by the non-zero low-order bits and a program check occurs. Four possibilities arise:

1. The check occurred on the TIC following the CCW for the last segment of an outgoing message. This is a normal situation and is ignored. (The missing buffer is for the next segment and there is no next segment.) If this is not the case, the start channel program pointer (LCBSTART) is reset to the CCW to which the TIC command was to have transferred control; this anticipates correction of the condition.

2. It is possible that through asynchronous operations a buffer was allocated and the TIC address was made valid in the period between the generation of

program check and its detection by the program. If this is true, the channel program is simply restarted.

3. The process of allocating a buffer may already have been initiated; if so, the routine exits to the supervisor to allow time for the process to complete.

4. If the process of allocating a buffer has not already been initiated, the routine branches to the Line PCI Appendage (at NOTINO) to release the buffer filled by the CCW immediately preceding the TIC that caused the program check.

WTTA LINE APPENDAGE ROUTINE (CHARTS R1, R2, R3, AND R4)

This routine is entered from the supervisor:

1. When a program controlled interrupt (PCI) occurs during the execution of a QTAM channel command for the line; or

2. On channel end during I/O operations.

Furthermore, this routine can be reentered from the QTAM Line End Appendage routine.

The WTTA Line Appendage is composed of the following two routines:

- The WTTA Line PCI routine
- The WTTA Line End routine

WTTA LINE PCI ROUTINE

The WTTA Line PCI routine is entered when a program controlled interrupt (PCI) occurs during execution of a QTAM channel command for the line.

If the interrupted channel command is a Write CCW or a Read CCW with a residual count in the CSW that is different from the initial count, control is returned to the QTAM Line PCI Appendage routine.

If the interrupted channel command is a Read CCW with identical initial and residual counts, the action taken depends on the type of Read CCW, as follows:

1. If the interrupted channel command is the first Read CCW, the PCI is ignored and control is returned to the supervisor.

2. If the interrupted channel command is a Read CCW in a BRB, the Line PCI routine tests the last character contained in the last filled buffer, as follows:

   a. If this character is EOM, EOT, or WRU, the residual count of the CSW is set to zero and the address of the CCW corresponding to the last filled buffer is inserted in the CSW.
   b. If this character is other than EOM, EOT, or WRU, the CSW remains unchanged.

Then the routine exits to the QTAM Line PCI Appendage routine.

WTTA LINE END ROUTINE

The WTTA Line End routine is entered when an I/O operation ends with a channel end condition, or is reentered from the QTAM Line End Appendage routine. If an I/O operation ends with channel end (C.E.) and unit check (U.C.), the result of the Sense operation is analyzed to check whether an abnormal condition occurred and, if so, the ERP routine is entered.

The operations executed by the WTTA Line End routine depend on whether this routine is entered on completion of a Halt I/O operation, of a Read channel program, of a Write channel program, or of an exchange of identifications, as follows:

1. On completion of a Halt I/O operation: If data is being received at the same time as the Halt I/O operation is executed, the interrupted Read Initial channel program is restarted. If no data is being received, a Write channel program is started to send a "letters shift" character followed by "n" padding characters (where "n" is the number specified in the DCB macro instruction). On completion of this Write channel program, control returns to the Interim LPS routine.

2. On completion of a Read channel program: The last character received in the corresponding buffer is analyzed, as follows:

   a. If this character is EOT, the EOT flag is set, and the buffer is posted to the Interim LPS routine.
   b. If this character is EOM, the buffer is posted to the Interim LPS routine.
   c. If this character is WRU, the action taken depends on whether or not the buffer is the first one. If the WRU character is in the first buffer, the Read CCW is updated to read the rest of the buffer, and the first part (identi-

fication exchange) of the Read channel program is started. If the WRU character is in another buffer, the "WRU" flag is set in the LCB, and the buffer is posted to the Interim LPS routine.

3. On completion of a Write channel program: The operations to be executed depend on how the I/O operation has ended:

   a. If the I/O operation has ended with a normal end condition, the buffer is posted to the LPS queue, provided no exchange of identifications is requested at the end of the output message. If this exchange is requested, the first part (identification exchange) of the Write channel program is started.
   b. If the I/O operation has ended with an abnormal end condition (contention), the contention counter is incremented, and a Write Break CCW is started (provided the threshold value has not been reached). On completion of this CCW, the interrupted Write CCW is restarted.

4. On completion of an exchange of identifications: The result of the exchange is analyzed to determine whether or not the exchange has been successfully performed and to take the appropriate action, as follows:

   a. If the exchange is unsuccessful, this condition is set in the line error halfword, and the buffer is posted to the Interim LPS queue (for receiving operations) or to the LPS queue (for sending operations).
   b. If the exchange is successful, the action taken depends on when the exchange has been performed.

      At the beginning of an output message: The Write channel program is restarted.

      At the end of an output message: The last buffer is posted to the LPS queue.

      When receiving an input message: If EOM=WRU, the last buffer is posted to the Interim LPS queue. If EOM is not WRU, the Read channel program is restarted to read the rest of the input message.

BUFFER CLEANUP AND RECALL ROUTINE (CHARTS DD AND DE)

This routine is entered through a branch instruction generated on expansion of a macro instruction in the problem program. The routine performs a cleanup function when entered at IECKPR through the calling sequence generated by a POSTSEND or POSTRCVE macro instruction. The recall function entry IECKRC is performed when entry is through the calling sequence associated with a CANCEL, EOBLC, ERRMSG, or REROUTE macro instruction. The difference between the two entry paths is that in the second case the recall flag is set on in the LCB (LCBSTATE = 64).

For either a cleanup or a recall operation, the routine releases all buffers assigned to the line. Buffers are released to the appropriate queue through an SVC 67. (The first buffer to be released may already contain a message segment; if so, it is posted to its destination queue.) The first buffer (if it does not already contain a message segment) and all subsequent buffers not scheduled to be filled are posted to the available buffer queue. Additional CCWs encountered in the BRB ring from which buffers are being released are posted to the additional CCW queue.

Buffers that have been assigned to the line and have also been scheduled for a read from direct access storage are treated differently. When such a buffer is encountered, the routine branches to the LPS Control routine. At that time, the "cleanup" flag or the "recall" flag (but not both) is on in the LCB for the line, indicating the type of operation in progress.

When the LPS Control routine is entered, it waits for a message-filled buffer and proceeds as usual unless the buffer is assigned to a line for which the "recall" or "cleanup" flag is on. When a buffer with either flag (but not both) on is found, the LPS Control routine branches back into the Buffer Cleanup and Recall routine, where the buffer is then released to the available buffer queue.

To recall a message segment, the routine provides the buffer request blocks required to read message segments from direct access storage, obtains the segment being recalled, and exits to the calling routine. When the cleanup operation is complete, exit is made to the Free BRB routine. This routine frees all BRBs in the BRB ring and posts each to the inactive BRB queue. The routine then posts the line to itself, which is the standard technique for returning a line to the free condition, and exits to the LPS Control routine.

DASD DESTINATION ROUTINE (CHART DX)

This routine is entered on activation of the DASD destination subtask, or by a branch-and-link from the Send Scheduler routine. The latter entry occurs when the Send Scheduling subtask is activated because of the availability of a message-filled buffer.

For buffers containing text segments, the routine routes a full buffer to the disk I/O queue and increments the message count (unless a CANCEL operation is in progress). The LCB for the source line (the line on which the segment now in the buffer was received) is removed from the source chain in which it previously appeared and linked into the source chain for the destination queue. The next segment relative record number is calculated and stored, and the routine either:

1. Returns to the Send Scheduler routine, or
2. Exits to the Qdispatch subroutine.

GET SCHEDULER ROUTINE (CHART DV)

This routine is entered when a buffer has been returned or when a disk read from a process queue has been completed. The routine makes three tests to determine whether the processing program is ready to accept another segment. If (1) there is no message segment in the DASD process queue, or (2) there are too many buffers in the process queue for the processing program to handle, or (3) a segment is currently being read from the DASD process queue, no further disk reading can be initiated, and this routine exits to the Qdispatch routine. If none of the three conditions exists, the routine initiates a disk read from the DASD process queue.

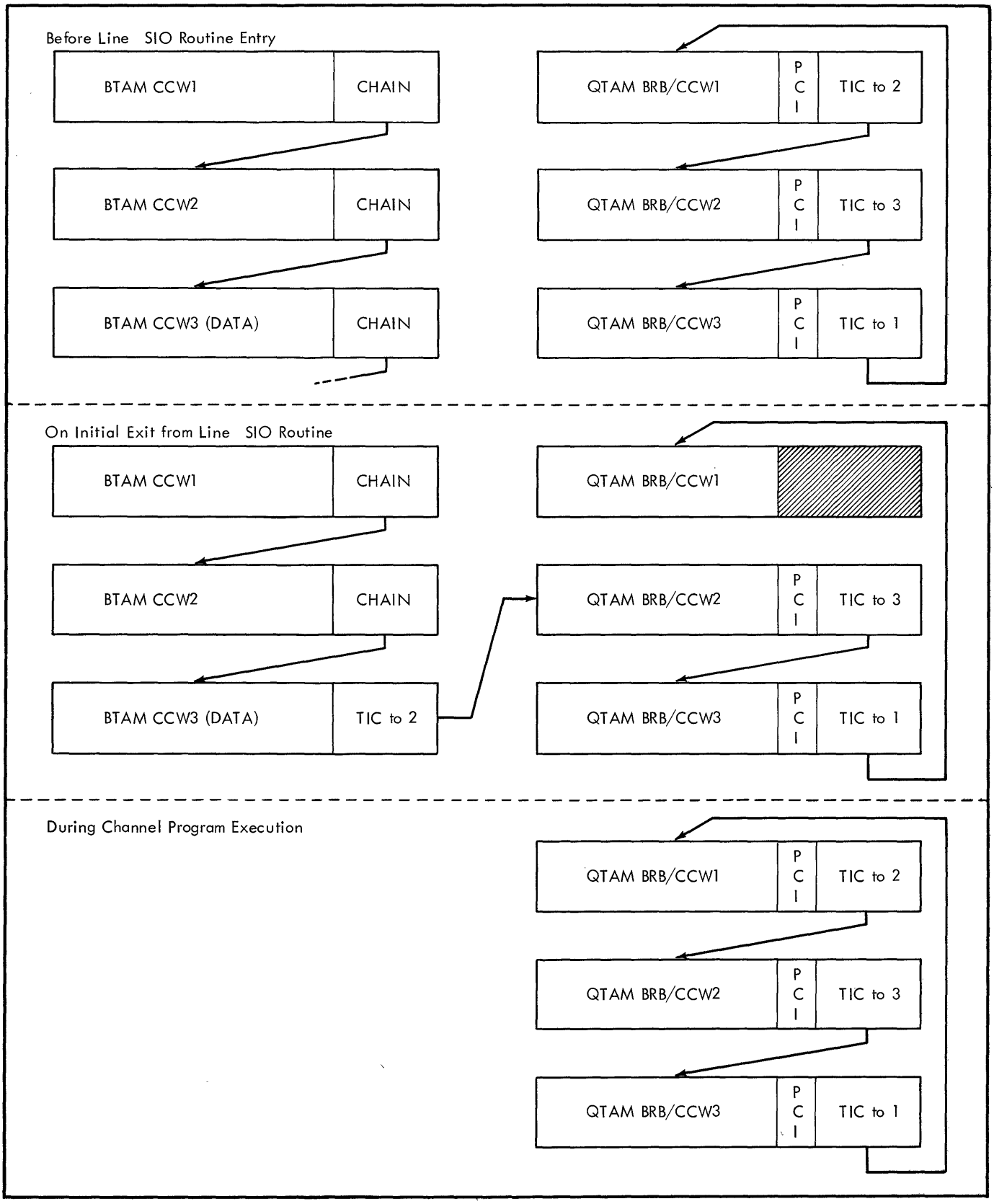


Figure 24. Interaction Between BTAM and QTAM Channel Programs

RETURN BUFFER ROUTINE (CHART DW)

This routine returns a buffer from the MS process queue and exits to the Get Scheduler routine to allow resumption of disk reading from the DASD process queue. If the buffer is not the dummy buffer for the first GET, the routine effectively posts the buffer to the available buffer queue if the buffer is not the last segment. If it is the last segment and if the "cleanup," "recall," and "converse" bits are set, the buffer is posted to the LPS queue.

END OF POLL TIME DELAY ROUTINE (CHART DJ)

This routine delays polling for a specified amount of time. If entered from an I/O interrupt, the routine goes to the Defer Entry routine, because the SVC cannot be given when an interrupt has occurred.

The routine issues the TIME macro instruction to obtain the time of day. The interval of intentional delay specified by the user is added to the time of day and stored in the LCB. The LCB is inserted into the time queue. If the interval of time has not yet elapsed, the routine issues an STIMER macro instruction to time the intentional delay. The exiting routine sets the condition code and obtains the address of the time queue QCB. The routine branches to the Line PCI Appendage to put the time queue on the ready queue.

When the time queue is dispatched from the ready queue and an LCB is in the time queue, the TIME macro is used to obtain the current time of day. If the time has elapsed, the LCB is removed from the time queue. If the LCB is for a checkpoint, a branch is taken to the Post subroutine to post the LCB. If the line is active, the LCB is placed into the top of the ready queue to activate the line before going to the Priority Search subroutine.

INTERIM LPS ROUTINE (CHART DU)

Before the buffers are processed, the INTERM queue is put on the ready queue behind the LPS queue. This is a special queue to delay the LPS until all buffer requests are processed.

SEND SCHEDULER ROUTINE (CHART DK)

This routine is entered when an LCB is on the top of the ready queue or when a message is to be written on a disk. If a message is to be written, the routine links to the DASD Destination routine at entry point SCREEN to cause a post to the disk I/O QCB. If the DCB has not been opened or the DEB is not open for output, the routine branches to the Dispatch subroutine (IECKQQ01).

If the DCB is open for output and the line is a WTTA line, the routine tests the line for availability. If the line is not available, a branch is made to the defer entry subroutine. If the line is available, the HIO flag is set in the LCB, and a Halt I/O operation is issued to clear the Prepare command.

For dial lines, if the relative line number of the STCB is greater than the relative line number of the current line, the STCB is placed on the dial out-call queue. If the line is connected to the destination terminal, the STCB is chained to the LCB and immediately dispatched. If the line is not connected to the correct terminal, the line group is searched by relative line number for a line that is free and the terminal available for dial. (This test is made on the priority of the Send Scheduler and is set to 1 by the TERM macro for dial lines that are not connected.) If no line is found, the STCB is chained to the dial out-call queue.

For all lines, the LCB is set to indicate that the line is trying to send. If the line is free the destination LCB with the send scheduler STCB is placed on the ready queue and dispatched. If the line is not free and is not an autopolled line, a branch is taken to the Defer Entry subroutine. If an autopolled line with receive status (LCBSTATE is X'08'), the TIC command code is changed to a NOP before branching to the Defer Entry subroutine.

If the routine was entered because the LCB was on top of the ready queue, the routine tests for an incoming priority message. If the line is sending or is in initiate mode, and if there are no complete nonpriority messages, or if there are priority messages coming in, the LCB is removed from the source chain. If the line is neither in an initiate mode nor sending, the status of the LCB is cleared. If a partial message is present in the queue (an invalid condition) the routine branches to the Wait subroutine. After setting the status code, the routine exits. If the routine was entered via the Get Scheduler routine, return is to that routine. If a line is sending, the routine branches to the BRB Ring routine at RQCONST.

FREE BRB ROUTINE (CHART DF)

This routine returns the BRBs to the inactive BRB queue. If a buffer request is pending (BRB is in the buffer request

queue), the BRB is not freed. If the BRBs are not in the active buffer request queue, the routine posts all BRBs to the inactive buffer request queue. The remaining BRBs will be freed by the Buffer BRB routine. When all the BRBs that have no buffer request pending are freed, the line is freed by posting the LCB to itself.

END INSERT ROUTINE (CHART DG)

This routine is entered by a branch and link from the End of Address, Conversational Mode, or Distribution List routines. The End Insert routine enters the address of a special entry point in these routines in a chain to be executed according to the priority specified by the Buffer Cleanup routine.

The End Insert routine compares the priority specified in the calling routine with the priority that has been set in the End Insert routine. If the priority is less than that of the highest-priority routine, the priority of the calling routine is compared with the next routine in the chain until the priority is higher.

When the priority of the calling routine is higher than the one in the chain, the address and priority of the calling routine are inserted in the constant of the higher-priority routine in the chain. The pointer to the calling routine is adjusted to the BAL instruction. The operand of this instruction in the calling routine is overlaid with the constant following the BAL instruction. This constant contains a register that has been set up by the calling routine. To complete the chain, the constant is overlaid with the address and priority of the lower-priority routine. This routine branches back to the calling routine at the BAL instruction.

CROSS PARTITION MOVE ROUTINE (CHART DY)

This routine is entered on activation of the Move Data subtask, and it is used to move data while in the supervisor mode. The routine allows data to be moved between partitions or within the same partition. Control is passed to the Dispatch subroutine. The routine is passed the address of the data to be moved in register 5, and the location into which it is to be placed in register 4.

This section summarizes the following operations of the services that QTAM provides to aid the user in error recovery:

- Checkpoint/Restart
- Error Recovery Procedures
- On-Line Terminal Tests

Note: Since Operator Control has an associated LPS macro instruction, the summary of this facility appears in the section Message Control Program (LPS) Routines.

CHECKPOINT/RESTART

Checkpoint/Restart is provided as an optional facility for the QTAM message control program at user-specified intervals (every 15 seconds to 15 minutes, or when a specified number of message processing partitions have issued CKREQ macros). By using the QTAM Checkpoint/Restart facility for the message control program and other QTAM facilities such as sequence numbers, an effective restart can be accomplished in a message processing program.

The Checkpoint routine (module IGG019NH) stores tables and other control information necessary for a subsequent restart after a system failure. Two such records are kept (flip/flop) with a pointer to the current record. For example, the initial checkpoint record is placed in area 1; after the user specified interval, the second record is placed in area 2; the third, after the interval, is placed in area 1, etc. The pointer is updated each time and also stored on the disk in a data set control record.

Restart of the QTAM job after a system failure is accomplished by initial program loading (IPL) the system again, and loading the QTAM message control program in the same location as it was when the failure occurred. QTAM automatically reinitializes the tables and pointers from the latest checkpoint record on the disk.

The Open Checkpoint Records Data Set routine checks the pointer to the latest checkpoint record to determine if the data set was properly closed, never opened, or left open due to a system failure. If the data set was never opened or properly closed, no restart procedure is performed. If the data set is left open due to a system failure, restart is performed in addition to normal open procedures for the data set.

Restart involves getting main storage for reading the checkpoint record. The checkpoint information is then moved to the proper areas overlaying the initial values. The checkpoint information includes the terminal table, polling lists, the disk pointers from the QCBs for destination and process queues, and the address of the next record to be written on the disk. An indicator is set for the line group DCB open routines to clear the lines in addition to normal open initialization.

The ENDREADY macro instruction initiates the initial time interval request for the first checkpoint if Checkpoint/Restart has been specified and the time interval method is used.

When the Checkpoint routine gains control after the initial time interval has elapsed or when the specified number of CKREQ macros have been issued, storage is reserved (GETMAIN) and the necessary data moved into this area. This record is written on the disk in the area specified by the pointer. The pointer is then updated and written on the disk. The storage is then freed (FREEMAIN).

Close for the Checkpoint/Restart data set sets the pointer on the disk to indicate that it has been properly closed to enable a subsequent OPEN of this data set to distinguish between a normal close and a system failure.

CHECKPOINT ROUTINE (CHARTS FA AND FB)

Module Name: IGG019NH

Function: This routine causes checkpoint records to be written on the Checkpoint Records data set at specified intervals or when CKREQ macros have been issued from a specified number of message processing partitions.

This routine is entered at QUEUEST +10

1. At ENDREADY time (A restart procedure may or may not be in process),
2. When a timer interruption occurs or the required number of CKREQ macros have been issued,
3. When the checkpoint element reaches the top of the disk I/O queue, or

4. When a disk Write operation has been completed.

The action taken for each type of entry is discussed in the following paragraphs.

ENDREADY Time: The expansion of the ENDREADY macro issues an SVC Qpost to cause the checkpoint subtask to be entered when the Checkpoint/Restart facility has been specified. The purpose of this entry is to set the timer for the first checkpoint interval and/or to release main storage obtained during a Restart operation.

If a Restart is in process, the storage obtained to read the checkpoint record by the Open Checkpoint Data Set routine (module IGG0193V) is released by issuing a FREEMAIN macro. A test is made to determine if the CKREQ or interval method of checkpointing is being used. If the CKREQ method has been specified (via the CKPART operand of the TERMTBL macro), no further action is required; therefore, exit is made to the Qdispatch subroutine for a return to the ENDREADY expansion.

If the interval method has been specified (via the CKINTV operand), exit is made to the Time Delay routine to set the timer for the first checkpoint interval. The checkpoint interval is passed in register 6, and the address of the checkpoint element (apparent LCB) is passed in register 4.

Timer Interruption or Required Number of CKREQ Macros Have Been Issued: The checkpoint subtask is entered for the purpose of collecting the data required for a checkpoint record and for preparing to write the record on the Checkpoint Records data set.

A GETMAIN macro is issued to obtain the main storage required to contain the checkpoint record. The following data is then located and transferred to the checkpoint work area: each terminal table entry; each polling list (except for the size byte); required data from each LCB (LCBCHDR, LCBNASEG, LCBTTIND, and LCBSTATE fields and unit address from the UCB); the data required from each destination QCB (QSIZE, QNASEG, QBACK, and QFAC fields) and each process QCB (same as for destination QCB plus the disk address of the current message); and the disk pointers in the error queue.

The element chain of the disk I/O queue is then examined. If other elements appear on the disk I/O queue, the checkpoint element is chained in below them to schedule the disk Write operation for the checkpoint record. Exit is then made to the Qdispatch subroutine to wait for the checkpoint element to reach the top of the disk I/O queue element chain. When this occurs, this routine will be reentered for writing of the checkpoint record.

If no element is on the disk I/O queue element chain, an EXCP is issued to start the disk Write operation. Linkage is made to the Convert routine to convert the TTR to an actual DASD address prior to issuance of the EXCP. After the I/O has been started, exit is made to the Qdispatch subroutine for dispatching the next ready item.

Checkpoint Element Reaches Top of Disk I/O Queue: The TTR is converted to an actual DASD address, and an EXCP is issued to start the disk Write operation.

Disk Write Operation Completed: When a write to the checkpoint records data set is completed, the disk interrupt is processed by an appendage within this routine, and control eventually returns to this routine at QUEUEST +10. This disk completion is recognized, and error checking is performed. If an error occurred on the disk Write, a WTO macro is issued to print an error message on the system console. The address of the Checkpoint routine is cleared in module IGG019NG to prevent any further attempts to write on the Checkpoint Records data set.

If no error is detected, a test is made to determine if the entire checkpoint record was written. If not, the new write address and count of remaining data to be written are computed, and another Disk Write operation is started. When writing of the checkpoint record has been completed, the current record indicator is set in the four-byte control record, and the counter of CKREQ macros required is reset to its initial value. An EXCP is then issued to start writing the control record.

When writing of the control record is completed, several cleanup and reinitialization procedures must be performed. The main storage obtained to build the checkpoint record is released via a FREEMAIN. If the time interval method is being used, exit is made to the Time Delay routine to set the timer for the next checkpoint interval. If the CKREQ method is being used, the ECBs for the waiting message processing partitions are removed from the wait queue and posted complete. Exit is then made to the Qdispatch subroutine.

When a CKREQ macro is issued in a message processing program, the Checkpoint Request routine (module IECKCKRQ) issues an SVC Qpost that causes this routine to be entered at CKSTCB +6. Upon entry, register 1 contains the address of an ECB associated

with the partition from which the CKREQ was issued. This ECB is chained into a wait queue. The CKREQ counter is decremented by one and tested to determine if the specified number of CKREQ macros have been issued. If not, exit is made to the Qdispatch subroutine. If CKREQ macros have been issued from the specified number of message processing partitions, an exit is made to the Post subroutine in IECKQQ01 to post the checkpoint element to itself. This causes this routine to be reentered at QUEUEST +10 so a checkpoint may be taken.

ERROR RECOVERY PROCEDURE

The Error Recovery Procedure (ERP) routines are designed to diagnose and recover, if possible, from all 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 End of Block and Line Correction routine.
- Statistical recording of all control unit errors.
- Error messages to the operator console for all permanent errors.
- Line error recording for all data checks, nontext time-outs, and intervention required errors.

ERP, which consists of 19 modules, operates in the nucleus error transient area within the supervisor protection key. IOS gives control to the QTAM/BTAM Control module (IGE0004A) on any error of a TP device. If the Line End Appendage routine finds any error in the status or sense, return is made to IOS indicating that control is to be given to ERP. Ten routines, module names ending in E, are called by IGE0004A according to the error found by the Control module. The remaining eight routines are linked by other ERP routines for error recording and other functions.

The ERP routines and module names are:

IGE0004E	Time-out and Data Check for Auto Poll
IGE0104E	Data Check
IGE0204E	Time-out
IGE0304E	Intervention Required
IGE0404E	Lost Data
IGE0504E	Error Post
IGE0604E	Bus-out and Overrun
IGE0704E	Link
IGE0804E	Status Check
IGE0904E	Command Reject, Equipment Check, SNO Error, SIO CC1
IGE0004F	Read Skip, Break Return
IGE0104F	Diagnostic Write/Read
IGE0204F	Line Error Recording
IGE0304F	Operator Control and LER Addition
IGE0404F	Special Open and Checkpoint Restart
IGE0504F	Not Operational SIO
IGE0604F	Bus-out and Overrun for Auto Poll
IGE0704F	Overrun

Linkage between the modules is done 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, 44 (CVT address) is placed in register 14. The possible linkages between modules are shown in Figure 25.

In this section there is a description of each module for the QTAM ERP. The descriptions explain the action taken under different commands and types of transfer.

Generally, if there has been no text transfer, the channel program is retried. If there is an error after two retries, the error is considered permanent. In the case of a permanent error, if on a nonswitched connection, a message is written to the operator. For a switched line, the sense bytes, CSW, and failing CCW are saved in the channel program area LCBCPA +32 through 40 for the message. A CCW for the Disable to hang up the phone is created as the first CCW in the channel program. A disable return (X'40') is set in LCBERRCT + 1. An EXCP is issued to execute the disable. Upon return, exit is made to IOS.

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

When there has been an error on a Read Response to autopolling, the polling list address and entry size are obtained. The

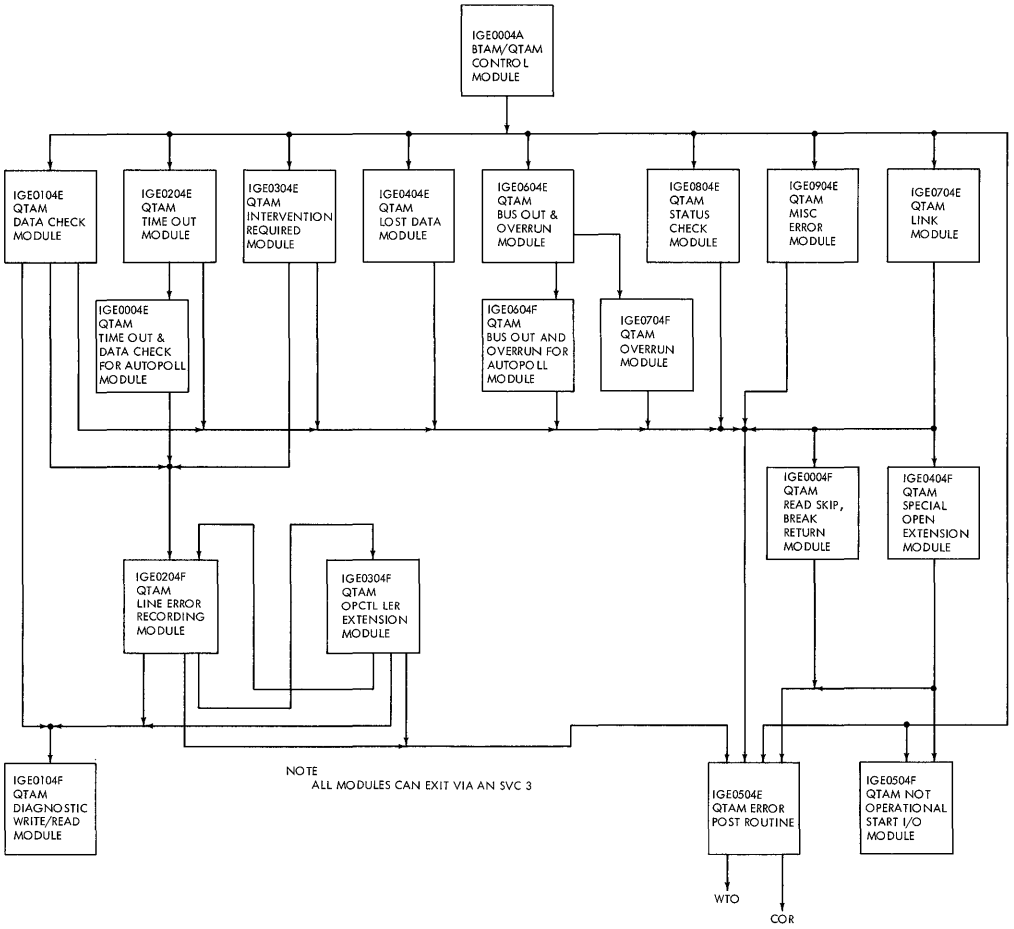


Figure 25. Linkage of ERP Modules

polling 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.

The following summarizes the switches that ERP sets in the LCB:

LCBERRCT Retry counter

LCBERRCT +1
- X'00' Normal return
- X'01' No message required
- X'02' Exit to Error Post routine
- X'04' Exit to Diagnostic Write/Read routine
- X'08' Read skip return
- X'40' Disable return
- X'0C' Special open for Checkpoint/Restart

LCBINCAM +1
- X'03' Time-out update for Line Error Recording
- X'01' Data Check update for Line Error Recording
- X'02' Intervention Required update for Line Error Recording

TIME-OUT AND DATA CHECK FOR AUTO POLL ROUTINE (CHART AF)

Module Name: IGE0004E

Function: After adjusting to the failing CCW, the routine tests the CCW:

- For a Read Response to autopolling, the polling list address and entry size are obtained. The polling list is searched to obtain the new count and data address for the poll CCW. The channel program is retried with the first CCW. Upon return, linkage is made to the Line Error Recording module.

- For a Poll CCW, the polling list address and entry size are obtained. The new count and data address are placed in the poll CCW. The channel program is retried with the first CCW. Upon return, linkage is made to the LER module.

If the retry has failed two times, the time-out error is set in the error halfword, and linkage is made in order to post with message.

DATA CHECK ROUTINE (CHART AB)

Module Name: IGE0104E

Function: After initializing, the routine indicates a data check update for the Line Error Recording module in LCBINCAM +1 (X'01')

If the failing CCW is a Read,

- If there has been a text transfer and no permanent error, linkage is made to the Error Post routine with an indication for no message.
- If a read TWX ID response, the channel program is executed to disable and redial for the retry.
- If a Read Response to autopolling, linkage is made to the Time-Out and Data Check for Auto Poll module.
- If a switched connection, the routine sets up for a retry after the disable-dial or disable-enable sequence. Upon returning from the EXCP, linkage is made to the Line Error Recording routine.

If the failing CCW is a Write,

- For a type I adapter,
  (a) If text transfer, linkage is made to the Error Post routine with no message indication.
  (b) If not text transfer, the channel program is restarted the same as a read. Return is made to the Line Error Recording routine.
- For a WTTA adapter, this is a contention situation, and the error recovery procedure has been performed by the WTTA Line End Appendage routine as long as the threshold value has not been reached. When the Data Check routine is entered, linkage is made to the Line Error Recording routine with a permanent error indication.
- For other adapters the routine indicates no linkage to the LER (Line Error Recording) routine and the "should not occur" bit in the error halfword of the LCB. A permanent error condition exists.

If the failing CCW is a poll, linkage is made to the Time-Out and Data Check for Auto Poll module.

If retry has failed two times, then it is considered a permanent error.

- On a 2701 control unit,
  (a) If the LER routine is required, an indication (X'04') for Diagnostic Write/Read is set in LCBERRCT + 1, and linkage is made to the LER routine.
  (b) If the LER routine is not required, linkage is made to the Diagnostic Write/Read routine.
- Otherwise a normal retry is executed.

If the failing CCW is a Break for a WTTA adapter,

- On a 2701 control unit, an indication (X'04') for Diagnostic Write/Read is set in LCBERRCT + 1, and linkage is made to the LER routine.
- Otherwise, linkage is made to the LER routine.

TIME-OUT ROUTINE (CHART AC)

Module Name: IGE0204E

Function: After initialization, the routine tests the failing CCW.

If the failing CCW is a Read,

- For a text transfer, linkage is made to the Error Post routine with a no message indication.
- For a Read Response to autopolling, linkage is made to the Time-Out and Data Check for Auto Poll module.
- For a Read Response to polling,
  (a) on a TWX terminal, return is made to IOS via an EXCP.
  (b) otherwise, the channel program is retried with the first CCW (third CCW for a Write Initial on a switched line). For switched lines, a Disable is set in the first CCW. Upon return, linkage

is made to the LER module (return to IOS if a text time-out).

If the failing CCW is a Dial, Enable, or Disable, the channel program is retried with the first CCW. Upon return, linkage is made to the LER module. If the failing CCW is a prepare, the channel program is retried beginning with the prepare CCW.

If the failing CCW is a poll, linkage is made to the Time-Out and Data Check for Auto Poll module.

If retried two times without success, a permanent error condition exists.

INTERVENTION REQUIRED ROUTINE (CHARTS AD AND AE)

Module Name: IGE0304E

Function: After the retry counter in the LCBERRCT field of the LCB is updated (if nontext transfer), the CCW is examined.

If the failing CCW is a Read or Write,

- If a Read Response to autopolling, the polling list address and entry size are obtained. The polling list is searched, and the new count and program is restarted with the first CCW.
- If a text transfer, the error halfword is updated and posted without a message (X'03') is indicated in LCBERRCT + 1. The "time-out" bit is set in the error halfword. Linkage is made to the LER routine.
- If this is a switched connection, a Disable is performed to hang up the transmitter.

If the failing CCW is a Prepare command or a Dial, the channel program is restarted with the first CCW. Upon return, normal retry is indicated, and return is made to IOS or the LER module, if required.

If the failing CCW is a poll CCW, the polling list address and entry size are obtained, the count and data address are set in the poll CCW. The channel program is restarted with the first CCW.

If the retry has failed two times, the routine considers it a permanent error.

LOST DATA ROUTINE (CHART AG)

Module Name: IGE0404E

Function: After initialization, the CCW is examined.

If a dial, the unit failure is recorded in the Statistical Data Recorder (SDR) and the channel program is retried using the first CCW.

If a Read command,

- For TWX ID response, the channel program is retried using the first CCW.
- For Read Response to autopolling, the new count and new data address are stored in the poll CCW before the channel program is retried with the first CCW.
- For a text transfer,

  (a) if the residual count is not zero, there is a permanent error condition.

  (b) if the residual count is zero, a Read Skip return indication (X'08') is set in LCBERRCT +1. The Read Skip CCW is set and executed. Upon return from the EXCP, the routine returns to IOS.

- For a switch initial program, the third CCW is used to restart. The control unit failure is recorded in the SDR.

If the retry has failed two times, the routine proceeds with a permanent error.

ERROR POST ROUTINE (CHARTS AH AND AI)

Module Name: IGE0504E

Function: After initialization, a branch is taken according to the indication set in LCBERRCT + 1 by the other ERP modules.

If from a normal post, a permanent error is indicated in the IOB. If no message is required, an EXCP is issued to return to IOS. IOS detects the permanent error condition.

If a disable return, the Redial/Enable sequence is indicated in LCBINCAM. For any error, the "hardware error" bit is set in the error halfword. The sense bytes, CSW and CCW are restored for use in message. A permanent error condition is set in IOBFLAG1. If a message is not required, an EXCP is issued to return to IOS.

If a message is required for either entry,

- If no operator control, linkage is made to the WTO (Write to Operator) module supplied by the system.

- If an error occurred at the operator control terminal, linkage is made to the WTO module supplied by the operating system.

- If operator control and outboard recording (OBR) bits are indicated, the OBR bit is turned off and linkage is made to the OBR module.

- If no outboard recording, a message is prepared.

If a message is to be prepared for the console,

- For a switched connection, include the dial digits in the message. The device type, adapter type, and terminal ID (if required) are put into the message. For an autopolled line, the index byte for the polling list is placed in the message.

- For operator control, return is made to IOS through an EXCP and RETURN.

- For no operator control, linkage is made to the Write to Operator module.

BUS-OUT AND OVERRUN ROUTINE (CHART AJ)

Module Name: IGE0604E

Function: After initializing, the routine determines if entered for bus-out check or overrun.

If the failing CCW is a poll CCW or a Read Response to autopolling, linkage is made to the Bus-out and Overrun for Auto Poll module.

Bus-out check:

If failing CCW is a Write,

- For a response expected (next CCW a Read) or an IBM Type III adapter, a Read Skip Return is indicated for the ERP Control module. The retry counter in the LCB is updated. The Read Skip CCW is set up in a save area after the CCWs. EXCP is issued to execute the Read Skip. Upon return, return is made to IOS.

- For a text transfer and a type I or II adapter, linkage is made to the Error Post routine with no message indicated.

- If not a text transfer,

  (a) If TWX, redial is set, and if a switched connection, the disable and redial sequence is bypassed. The retry counter in the LCB is updated and control unit failure is recorded in the SDR table. The channel program is restarted, and upon return, exit is made to IOS.

  (b) If dial, a disable-redial sequence is set. The retry counter in the LCB is updated, and control failure is recorded in the SDR table. The channel program is restarted. Upon return, exit is made to IOS.

Overrun check: Linkage is made to the Overrun module.

If an error occurs after two retries, a permanent error condition exists.

LINK ROUTINE (CHARTS AK AND AL)

Module Name: IGE0704E

Function: This routine is entered as a return from special functions performed by ERP.

If entry to routine was for the diagnostic Write/Read,

- For a Disable, control unit failure is indicated and the channel program restarted with the first CCW. Return is made to IOS.

- For a diagnostic Read that failed, if a teletype adapter, a check is made for unit exception in addition to channel end/device end. If an error is detected, control unit failure is indicated in the error halfword. If an Enable is not required, linkage is made to the Error Post routine with indication for a message. If Enable is required, the channel program is executed at the enable CCW. Return is made to IOS.

- For a diagnostic Write failure, control unit failure is indicated in the LCB. If Enable is required, the channel program is restarted at the Enable CCW. Return is made to IOS. If Enable is not required, the CSW, sense byte, and CCW is restored for the message, and linkage is made to the Error Post routine with a message indicated.

- For an Enable, if not channel end/device end, control unit failure is indicated in the LCB. The sense byte, CSW, and failing CCW are restored for message and linkage is made to the Error Post routine with message indicated.

If entry was for a Read Skip, post with message is indicated in the LCBERRCT +1

field of the LCB. Linkage is made to the Read Skip Return routine.

If entry to the routine was for a Write Break:

- If channel end, device end, and unit check are indicated, the sense byte is tested. If any indication other than bus-out, linkage is made to the Error Post routine.
- If a text transfer and if channel end, device end, or a bus-out indication, linkage is made to the Error Post routine with no message indicated.
- If no text transfer and initial type channel program, the channel program is restarted with the first CCW. Return is made to IOS. If not an initial channel program, linkage is made to the Error Post routine.

If entry to the routine is made for the special OPEN, linkage is made to the Special Open and Checkpoint/Restart module.

STATUS CHECK ROUTINE (CHART AM)

Module Name: IGE0804E

Function: The routine branches to the operating system supplied Interpreter to determine the type of status check.

For chaining, program or protection check,

- If a nonswitched connection or failing CCW is a Disable, the routine indicates an outboard recording, and linkage is made to the Error Post routine.
- If switched connection, the routine saves the CCW sense byte, CSW, and indicates a Disable Return. The channel program is restarted.

For an unit exception, the retry counter in the LCB is updated and,

- If teletype I adapter, the CCW for a break is set up.
- If 2701 control unit or permanent error, the Read Skip CCW is set up. The sense byte, and CSW are saved, and a Read Skip return is indicated. After execution of the Read Skip, return is made to IOS.
- Otherwise, a retry is done on the Write CCW. Upon return from the EXCP, return is made to IOS.

After two retries, a permanent error condition exists.

COMMAND REJECT, EQUIPMENT CHECK, SIOCC1, SNO ERROR ROUTINE (CHART AN)

Module Name: IGE0904E

Function: After initialization, action is taken according to the error.

If initial selection error (SIO condition code equal to 1), control unit failure is recorded and the retry counter is updated in the LCB. The channel program is restarted. Return is made to IOS.

If command reject error, the retry counter in the LCB is updated and the channel program is restarted at the command in error. Return is made to IOS.

If equipment check or "should not occur" (SNO) error, the outboard recording is indicated. The proper error indicator is set in the error halfword. For a nonswitched connection, linkage is made to the Error Post routine. For a switched connection, the routine indicates a Disable Return and saves the sense byte, and CSW. After the EXCP of the Disable, return is made to IOS.

After two retries, a permanent error condition exists.

READ SKIP RETURN ROUTINE (CHART AO)

Module Name: IGE0004F

Function: After initialization, action is taken according to errors found.

For the following indications the "should not occur" bit is set in the error halfword, and linkage is made to the Error Post routine: unit check, unit exception, command reject, bus-out check, equipment check, overrun, or residual count equal to zero.

If the second CCW is for a switched line, the Read Skip sense byte is checked.

- For intervention required or time-out, a Disable CCW and Disable Return is set. The channel program is restarted at the Disable CCW.
- Otherwise, if no text transmitted, the channel program is restarted at the third CCW. Return from the error EXCP is to IOS.

- For a text error, linkage is made to the Error Post routine with no message indication set.

If the second CCW is not for a switched line,

- For a text transfer, linkage is made to the Error Post routine with no message indicated. Prior to linking, if the Read Skip ended with a time-out or intervention required, the routine indicates that reselection is necessary.
- For no text transfer, the channel program is restarted at the first CCW. Return from the error EXCP is to IOS.

DIAGNOSTIC WRITE/READ ROUTINE (CHART AP)

Module Name: IGE0104F

Function: After initialization, the diagnostic Write/Read indication is set in the LCBERRCT +1 field of the LCB for returning. The CCW in LCBERCCW is set up with a Disable. Enable is set at completion unless a switched connection. If a TWX or 2260 and type III adapter, the Disable CCW for the 2260 is skipped and the chained Enable is removed. The address and command code for the diagnostic Read and Write are set in the channel program area. The test data for the particular device is also moved to the channel program area. The EXCP is issued, and upon return, exit is made to IOS.

LINE ERROR RECORDING ROUTINE (CHART AQ)

Module Name: IGE0204F

Function: A test is made on the LERFLG1 field of the LCB. If operator control is to put out a threshold message, linkage is made to the Operator Control LER module.

If a normal update to the counters, one is added to the proper error counter in the LCB. If the transmission threshold value specified has not been reached, the routine compares the updated threshold counter. If that threshold has not been reached, an exit is made to the module indicated in LCBERRCT +1, i.e., diagnostic Write/Read, Error Post routine, or IOS. If the threshold value has been reached (not transmission) message output is indicated in the LERFLG1 byte of the LCB. All threshold values are added to their respective accumulative counters.

If no message is to be printed, the counters are cleared and the exit is to the module indicated in LCBERRCT +1, i.e., Error Post routine, Diagnostic Write/Read routine, or IOS. If no operator control is specified, the threshold counters are converted to decimal and inserted into the message. A Write to Operator macro is issued to write the message.

OPERATOR CONTROL LER ADDITION ROUTINE (CHART AR)

Module Name: IGE0304F

Function: This module is linked by the Line Error Recording module. If this module was entered to update the temporary counters, because a message is to be written with existing counters, one is added to the temporary counters and exit is made to the module indicated in the LCBERRCT + 1 field of the LCB, i.e., Diagnostic Write/Read routine, Error Post routine, or IOS.

If the temporary counters are no longer needed, they are added to the corresponding threshold counters, and the temporary counters are cleared. Return is made to the Line Error Recording module, which proceeds as a normal update.

OPEN AND CHECKPOINT RESTART ROUTINE (CHART AS)

Module Name: IGE0404F

Function: This module is entered from the ERP control module after the SIO has been issued.

- If the condition code is 3, linkage is made to the Not Operational SIO module.
- If the condition code is a 0 or 1, the TP Op code is examined. For the condition code of 0 the failing CCW is used; for condition code of 1 the first CCW is considered in error.

  (a) For an Enable, NOP, or SAD command, a channel end or device end indication is valid, so the line can be started; otherwise, there is an error.

  (b) For a Write Break, if the CSW indicates channel end/device end alone, the break was successful so the line is started. If a unit check is indicated, the sense byte is examined. For a data check, the channel program is restarted with the first CCW to retry the Write Break unless retried two times. If retried two times, an error exists. For all other conditions an error exists.

(c) For a Read Skip, a test is made in the status and sense bytes. If no errors exist, the line is started.

- For an error, if it is Open time, linkage is made to the Error Post module to post complete with error. Otherwise, the line number, operation code, status, and sense bytes are placed in the message. A Write to Operator macro is issued to write the message. Upon return, the CCWs are restored and the line is started bypassing the check for OPEN.

- To start the line the CSW is initialized for the retry. An EXCP is issued to retry the channel program. If it is Open time, an indicator is cleared for Line SIO. Error corrected is indicated to IOS in LCBFLAG1. An ERREXCP (SVC15) is issued to return to IOS.

NOT OPERATIONAL START I/O ROUTINE (CHART AT)

Module Name: IGE0504F

Function: After initialization, the routine issues a Write to Operator macro, which writes the message, IEC8041 ---- CONTROL UNIT NOT OPERATIONAL. Upon return, a Write to Operator with Reply is issued to write the following message: IEC804A REPLY CONT OR POST.

If the reply is a Cont,

- If this is OPEN time, an EXCP is issued to return to IOS to retry the channel program.

- If this is not OPEN time, the SAD and Enable commands are needed. If a 2702, the SAD command is used and then stored in the channel program. A CCW is set up for an Enable except for the type III adapter. A Read Skip CCW is placed in the next CCW except for a type I adapter, which uses a Write Break CCW. The channel program is executed and upon return, exits to IOS.

If the reply is a Post,

- If this is OPEN time, Idle Open is indicated in the CCW in the LCB and normal completion is set in the IOB. Return is to IOS via an EXCP.

- If it is not OPEN time, the routine sets the "cleanup" flag in LCBSTATE and a special flag for Line End and Free BRB to ignore the line. The IOB is set to indicate a permanent error to IOS. Return is to IOS via an EXCP.

BUS-OUT AND OVERRUN FOR AUTO POLL ROUTINE (CHART AU)

Module Name: IGE0604F

Function: After initialization, the routine tests for bus-out or overrun checks.

For bus-out check,

- If the failing CCW is a poll operation, the address of the polling list and the length of the entries are obtained. The new count and data address are stored in the poll CCW. The retry counter is updated, and the control unit failure is recorded. The channel program is retried with the first CCW.

- If the failing CCW is a Read Response to polling, the address of the polling list and the length of the entries are obtained. A search is made for the correct data address and count for the Poll CCW. The retry counter is updated and control unit failure is recorded. The channel program is retried with the first CCW.

For an overrun check, the "should not occur" bit is set in the error halfword and a permanent error condition exists.

After two retries, a permanent error condition exists. For a bus-out the control unit failure is set in the error halfword. An indication is set for outboard recording. Linkage is made to the Error Post routine.

OVERRUN ROUTINE (CHART AV)

Module Name: IGE070F

Function: This module is linked to as the result of an overrun indication found by the IGE0604E module. After initialization, the failing CCW is examined.

For a Read CCW,

- If text transfer, linkage is made to the Error Post routine with no message indicated.

- If an initial channel program and a response to a TWX ID, the channel program is restarted after the Dial/Enable sequence.

- Otherwise, the control unit failure is recorded and the channel program is retried at the first CCW. Upon return from the error EXCP, return is made to IOS.

For a NO-OP CCW,

- If a Read Initial channel program, linkage is made to the Error Post routine with no message indicated.
- Otherwise, the "should not occur" bit is set in the error halfword and a permanent error condition exists.

If the retry has failed two times, a permanent error condition exists.

ON-LINE TERMINAL TEST

The Resident Terminal Test routine is the only routine of the on-line terminal test that remains in storage at all times. This module is located by a "V" type address constant in the LPSTART macro expansion.

The Header Analysis routine is brought into the SVC transient area and executed by a SVC 77 from the Resident Terminal Test routine.

The Header Analysis routine brings the needed terminal test routines into the SVC transient area.

These routines perform the function of examining the test request message and performing the desired test.

The test request message is sent from the terminal to initiate the test. The format of this message is:

```
99999 format-integer test-integer type-
      integer [addr-char(s)] [unit-char(s)]
      [text-char(s)] end-char
```

where:

99999 is the test activation code.

format is zero or one.

test specifies kind of test (1 through 9).

type specifies type of terminal test is for one (1 through 6).

addr address of the terminal.

- Format 0 means exact address.
- Format 1 means a symbolic address.

unit specifies particular unit of the terminal.

text is the text of the message.

end specifies the end of the Test Request Message.

RESIDENT TERMINAL TEST ROUTINE (CHARTS QL AND QS)

Module Name: IECKONLT

Function: This routine recognizes terminal test activity, calls terminal test transient routines, sends test messages, cleanup, stops and restarts line operation.

The LPSTART macro generates a linkage to the module that checks the incoming messages for the test activation code. If the code is not present, normal operation of LPS continues.

If the test activation code is present, the buffers associated with the line operation are posted to a test QCB. The subtask activated (Terminal Test Buffer Routing subtask) will set test identification flags in the buffer prefix containing the test request and post it to the LPS queue. Subsequent buffers will be posted to the available buffer queue. (terminal tests will utilize only the buffer containing the header segment.)

Upon the next execution of the LPSTART macro, the buffer with the "test request" and "test identification" flags is processed by the routine. The "test identification" flags are recognized at entry to the module and the terminal test transient routines are called. These routines validate the test request and set up the appropriate test.

The buffer is then posted to another test queue control block. The subtask activated stops the line to be utilized by the terminal tests by placing a test subtask control block in the STCB chain of the appropriate LCB.

After the line operation has been stopped, further identification flags are set in the buffer prefix and again it is posted to the LPS queue. Upon the following execution of the LPSTART macro, these flags are recognized at entry into the module and the test message is sent to the terminal.

Upon completion of the test message transmission, the Line End Appendage posts the buffer to the LPS queue. All areas and buffers utilized by the terminal tests will then be freed and QTAM line operation will be restarted on the subject line.

If a test message is to be returned to the requesting terminal on a dial line, the transient routines are called immediately upon recognition of the test activation code. The test message is then sent to the terminal without utilizing the Stop Line subtask. Buffers are released to the available buffer queue by the buffer routing subtask.

The Terminal Test Buffer Routing subtask and Terminal Test Stop Line subtask are a part of the Resident Terminal Test routine.

TERMINAL TEST HEADER ANALYSIS ROUTINE (CHART QA)

Module Name: IGC0007G

Function: This routine performs preliminary validation of the test request, translates the input message as necessary, and sets up terminal addressing characters.

The input message is located and any translation necessary is performed. Translations that may be needed are symbolic addresses of terminals and translation between ASCII and BCD.

The proper terminal addressing characters are placed in the buffer prefix along with the addresses of the LCB and UCB.

Control is then passed to the proper terminal test routine to complete the activation of the on-line terminal test.

TERMINAL TEST ROUTINES (CHARTS Q3, Q4, Q5, Q6, AND Q8)

Module Names: IGC0107G, IGC0207G, IGC0407G, IGC0307G, IGC0507G, IGC0607G

Function: These routines cause the generation of channel programs according to the terminal used.

All of the following attributes are independent of the terminal type. Additional functions are present in specific Terminal Test modules (example: the IBM 1050 module, IGC0207G, considers the dial capability).

The header of the test request message is inspected. If any part of the header is found to be invalid (example: test-integer of zero), the no-test switch is set and control is returned to the Resident Terminal Test module. If the header is valid, processing of the test request message continues.

If the format-integer is zero, the addr-char is exact and is moved directly into the buffer. With a format-integer of 1, the addr-char must be interpreted and the proper characters placed in the buffer.

A GETMAIN is issued to provide area for building channel programs and output data. If no main storage is available, the no-test switch is set and control is returned to the Resident Terminal Test module.

All general CCWs are built before determining the type of test to be performed. After the test type has been determined, the CCWs necessary for the data are constructed and any message to be sent to the terminal is prepared. Control is then returned to the Resident Terminal Test module.

Chart AB. Data Check Routine

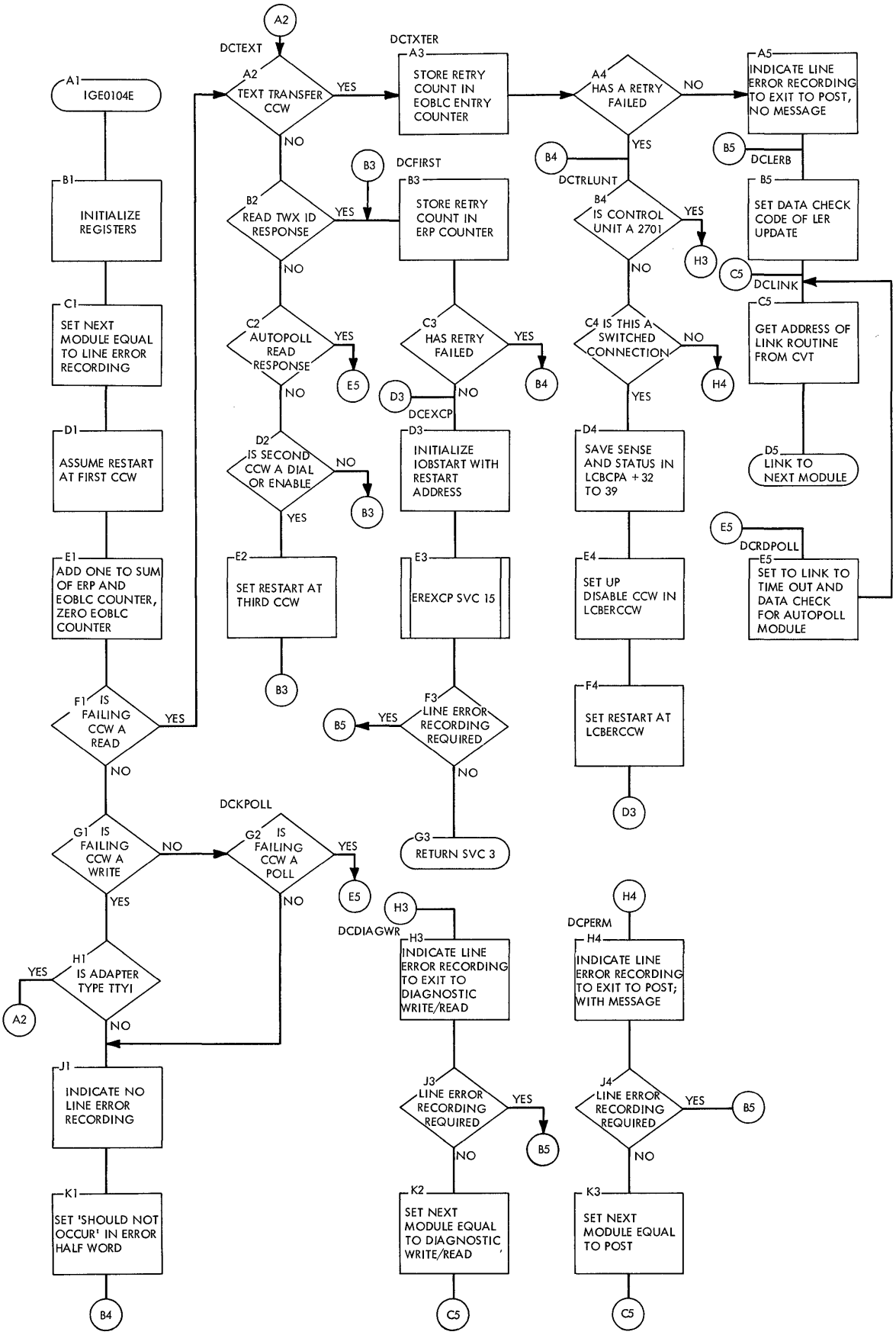


Chart AC. Time Out Routine

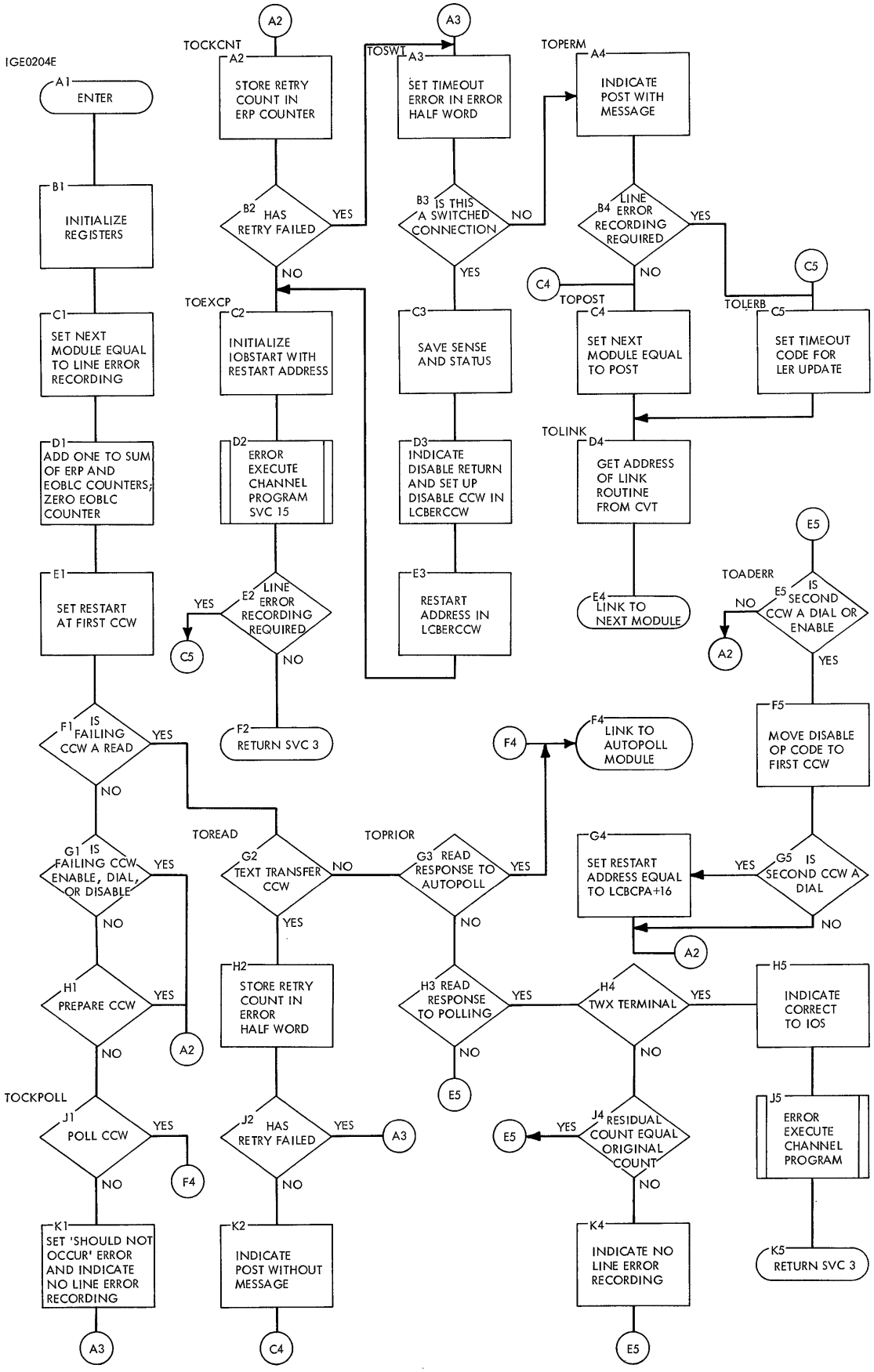


Chart AD. Intervention Required Routine

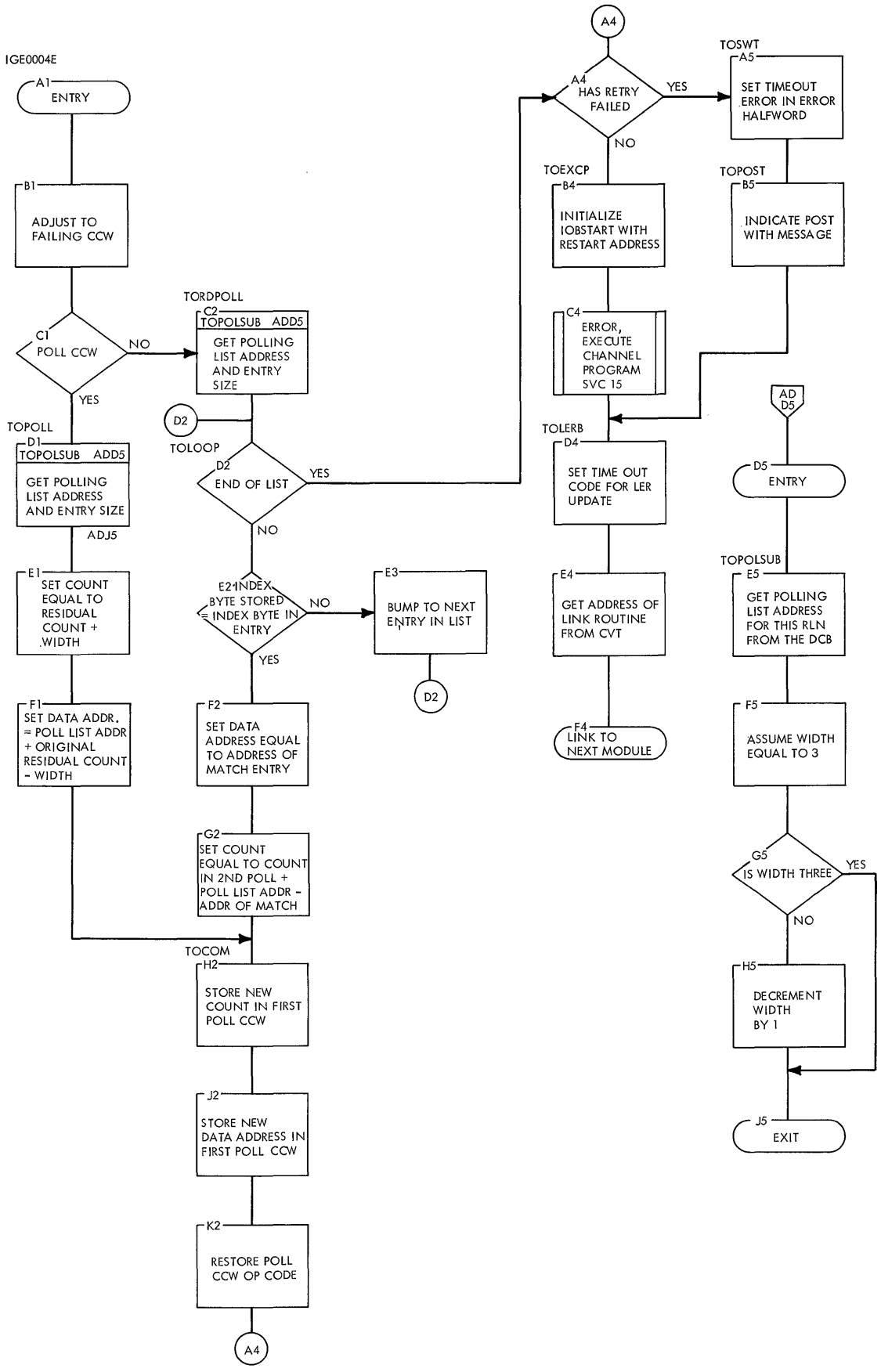


Chart AE. Intervention Required Routine (Continued)

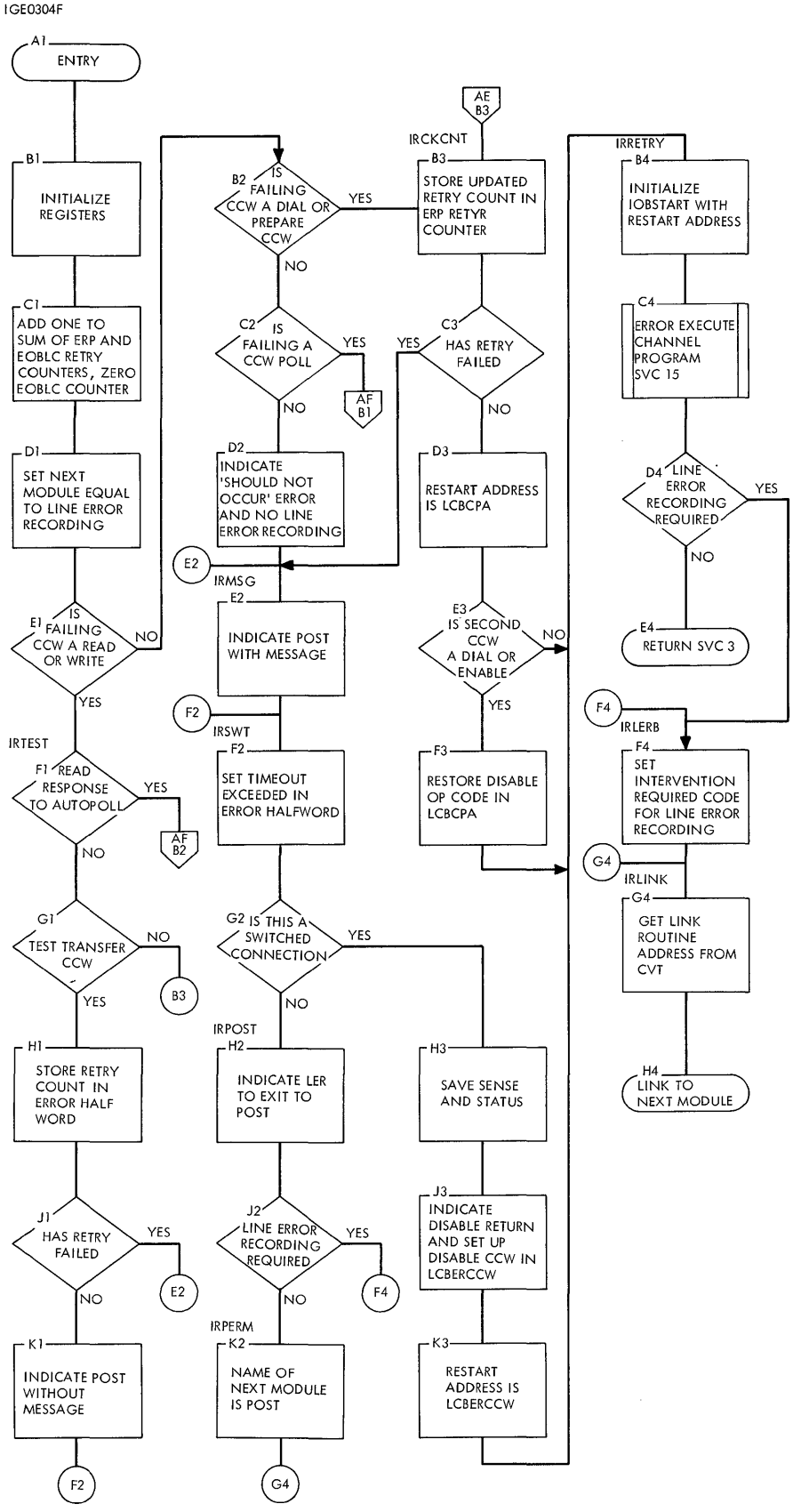


Chart AF. Time Out and Data Check for Auto Poll Routine

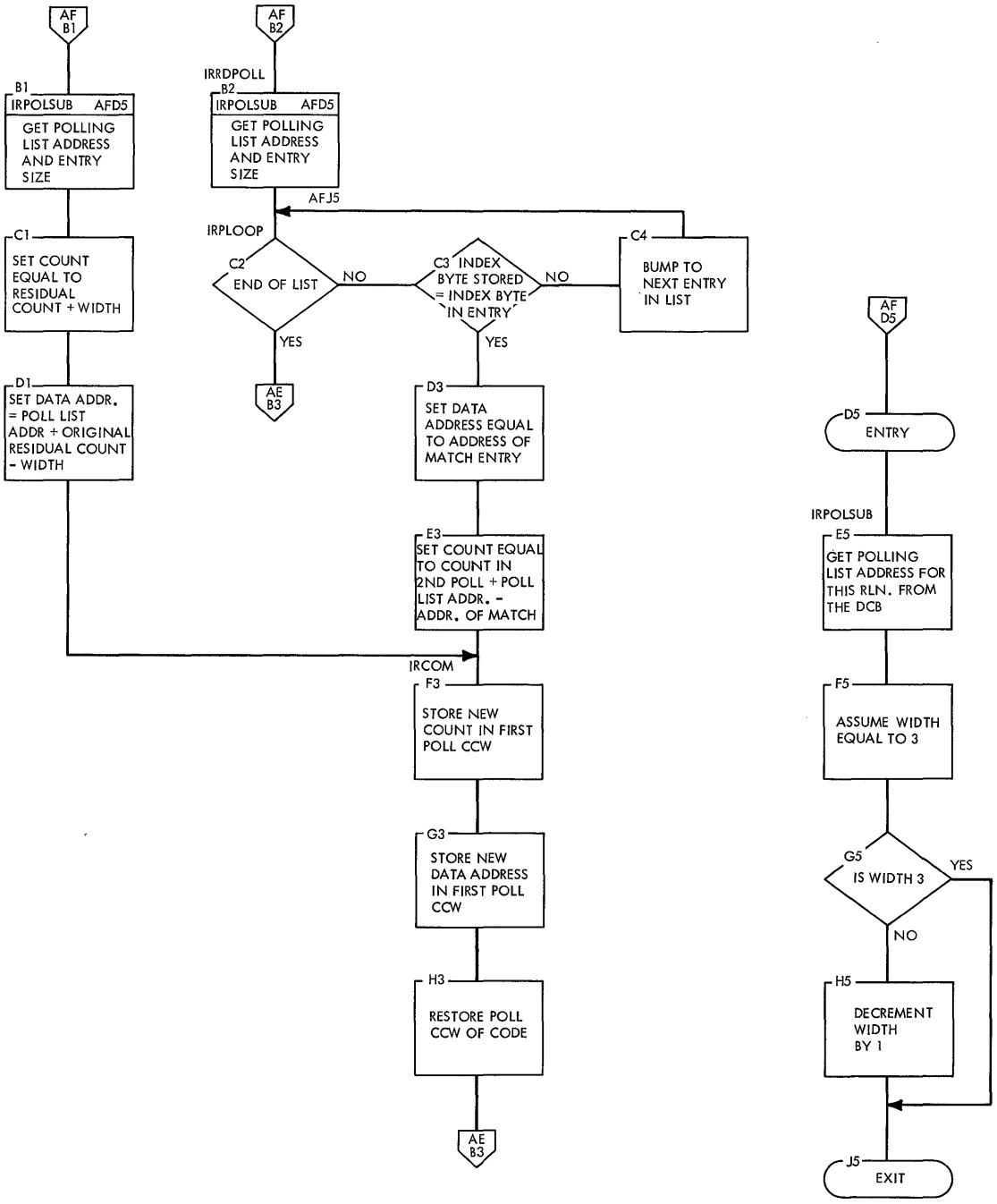


Chart AG. Lost Data Routine

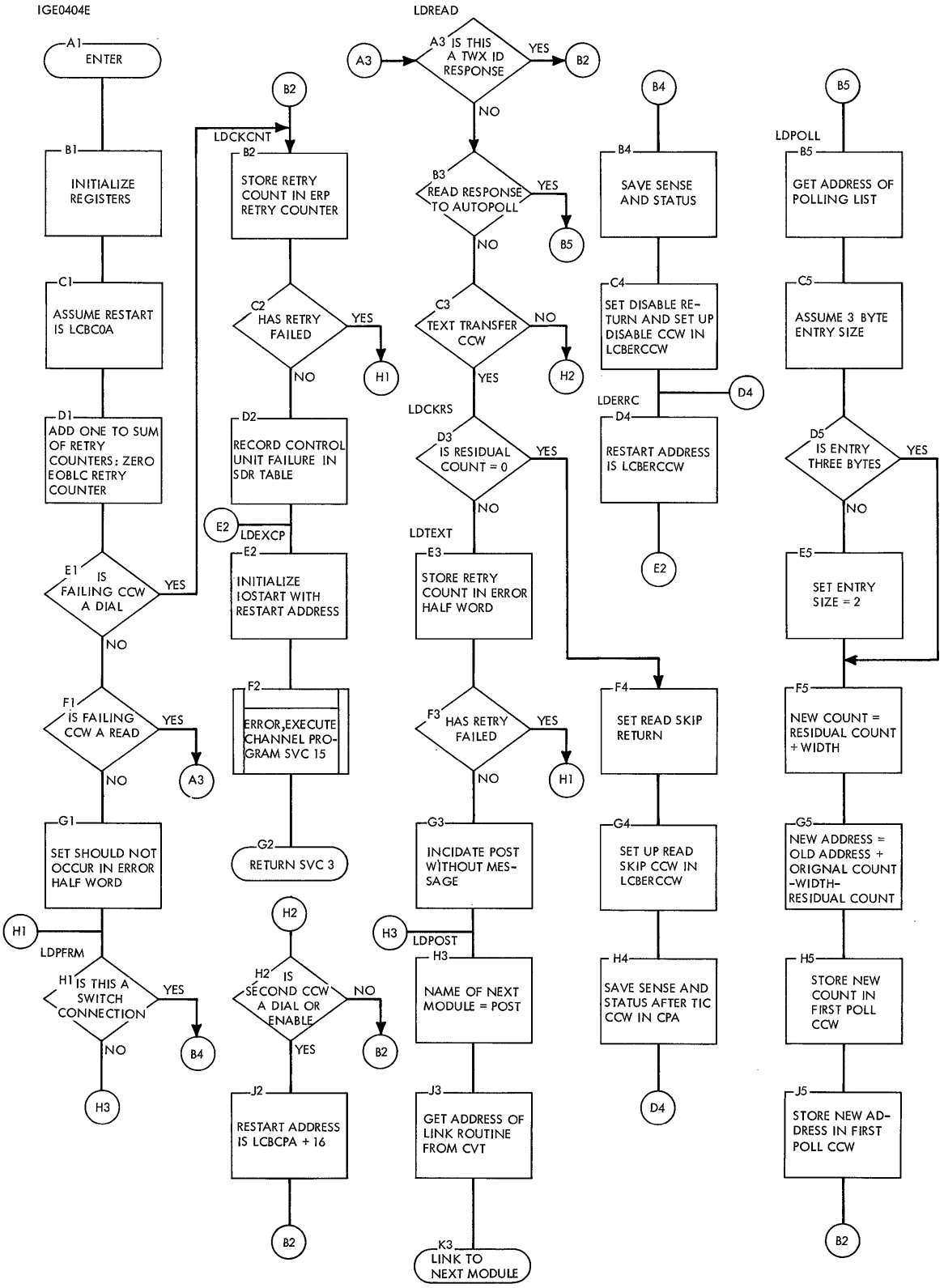


Chart AH. Error Post Routine

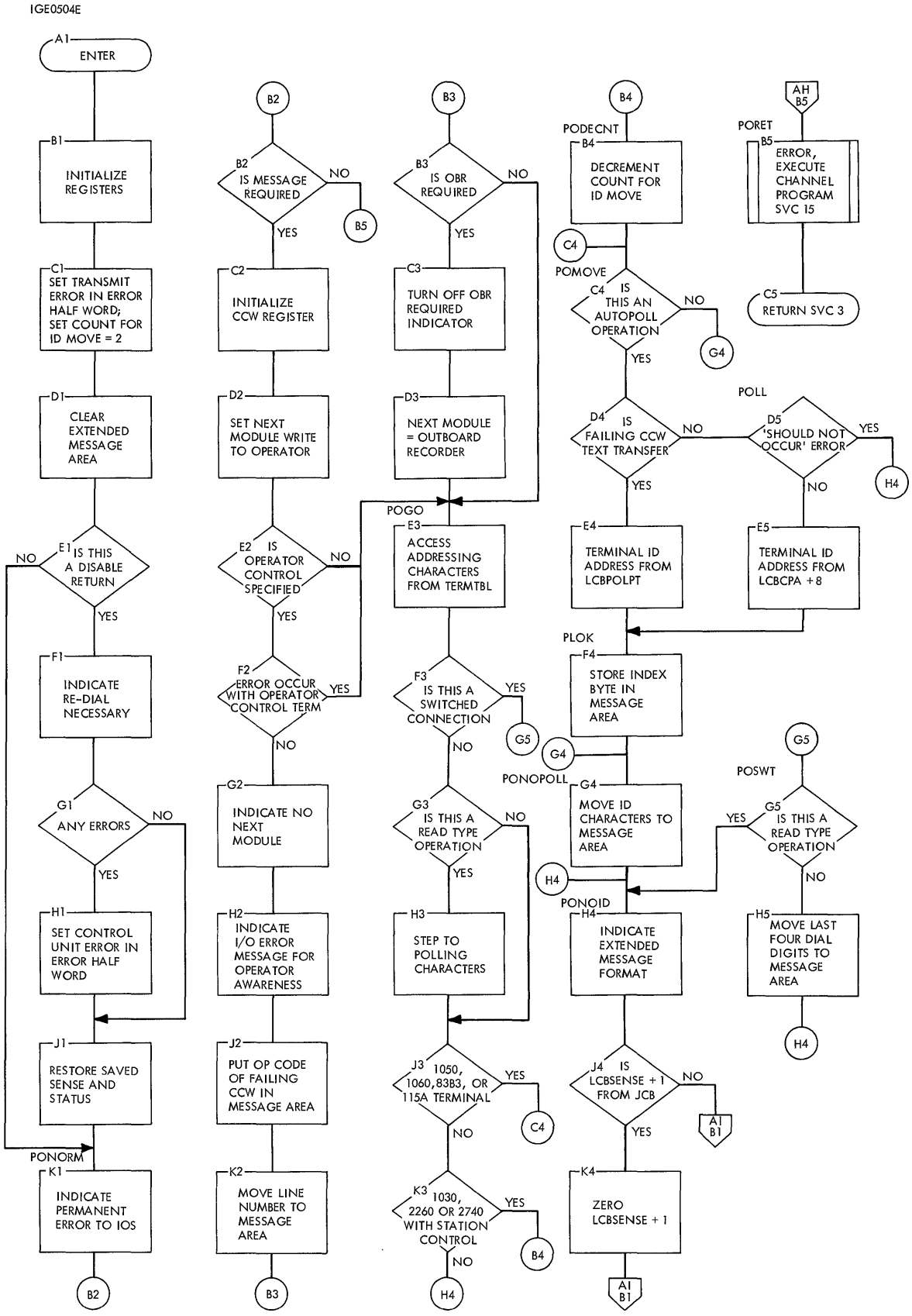


Chart AI. Error Post Routine (Continued)

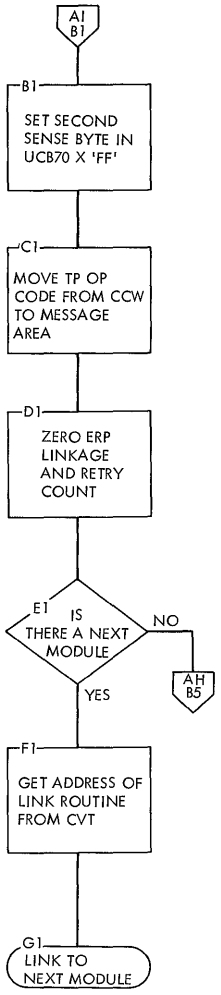


Chart AJ. Bus Out and Overrun Routine

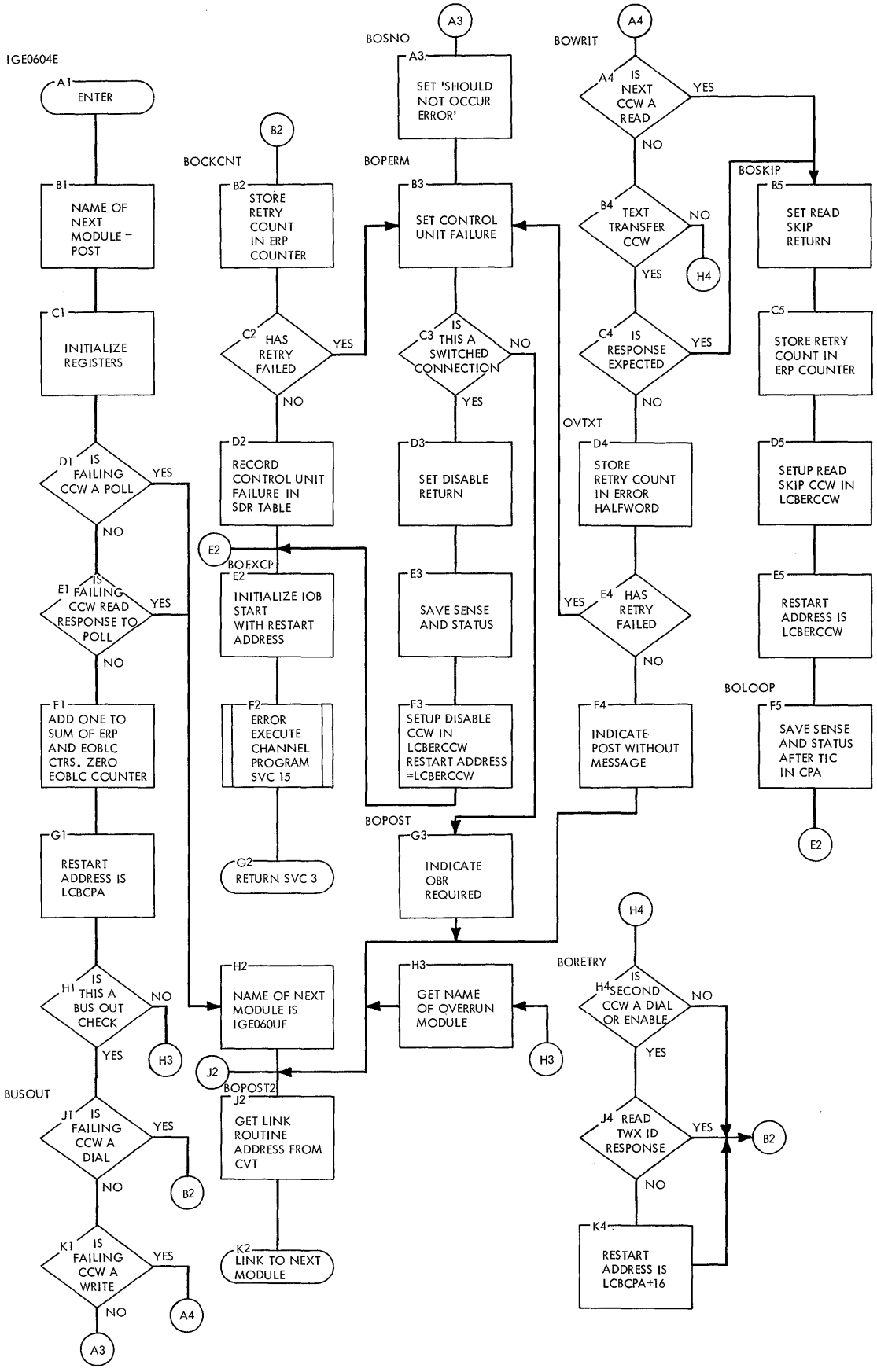


Chart AK. Link Routine

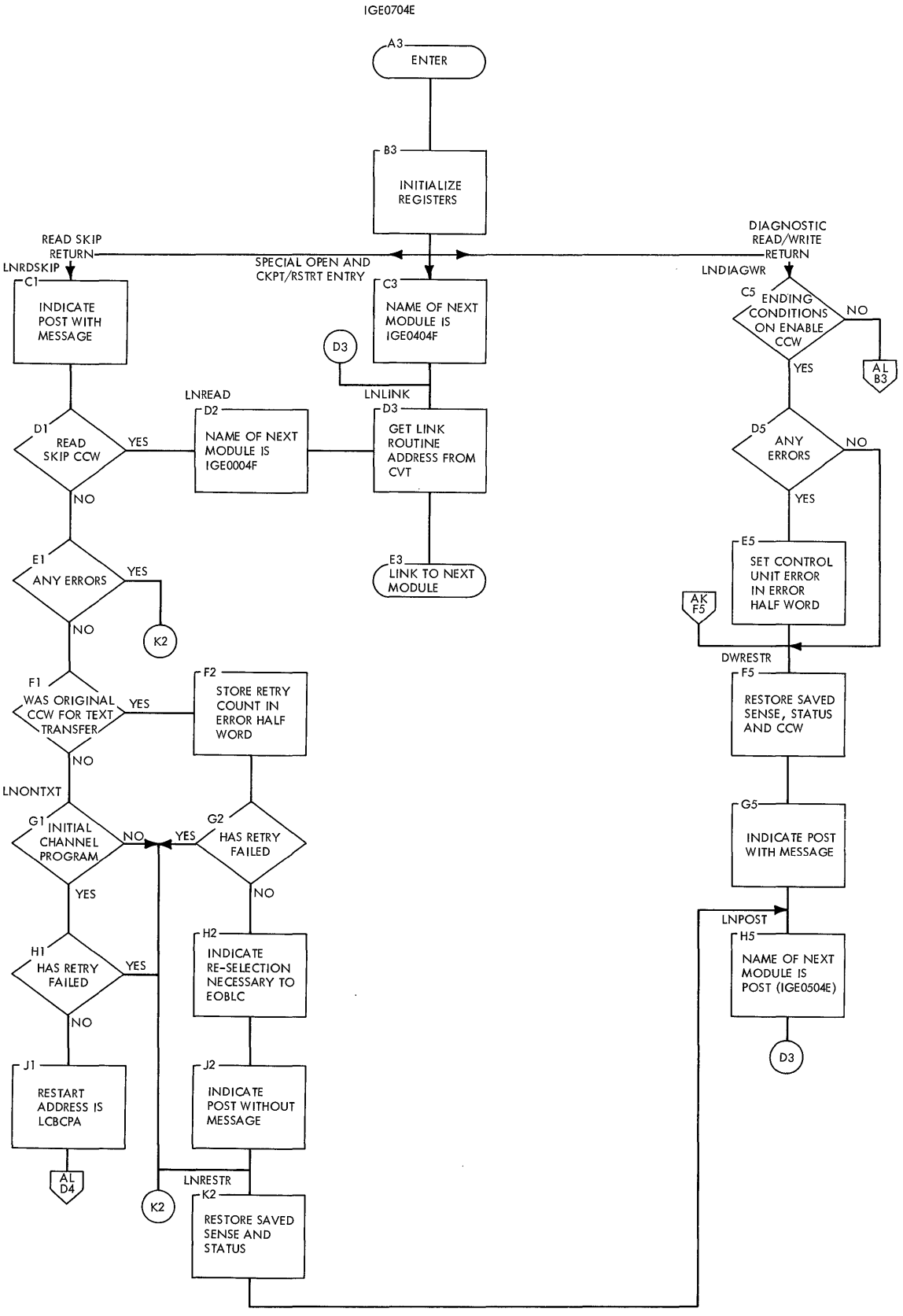


Chart AL. Link Routine (Continued)

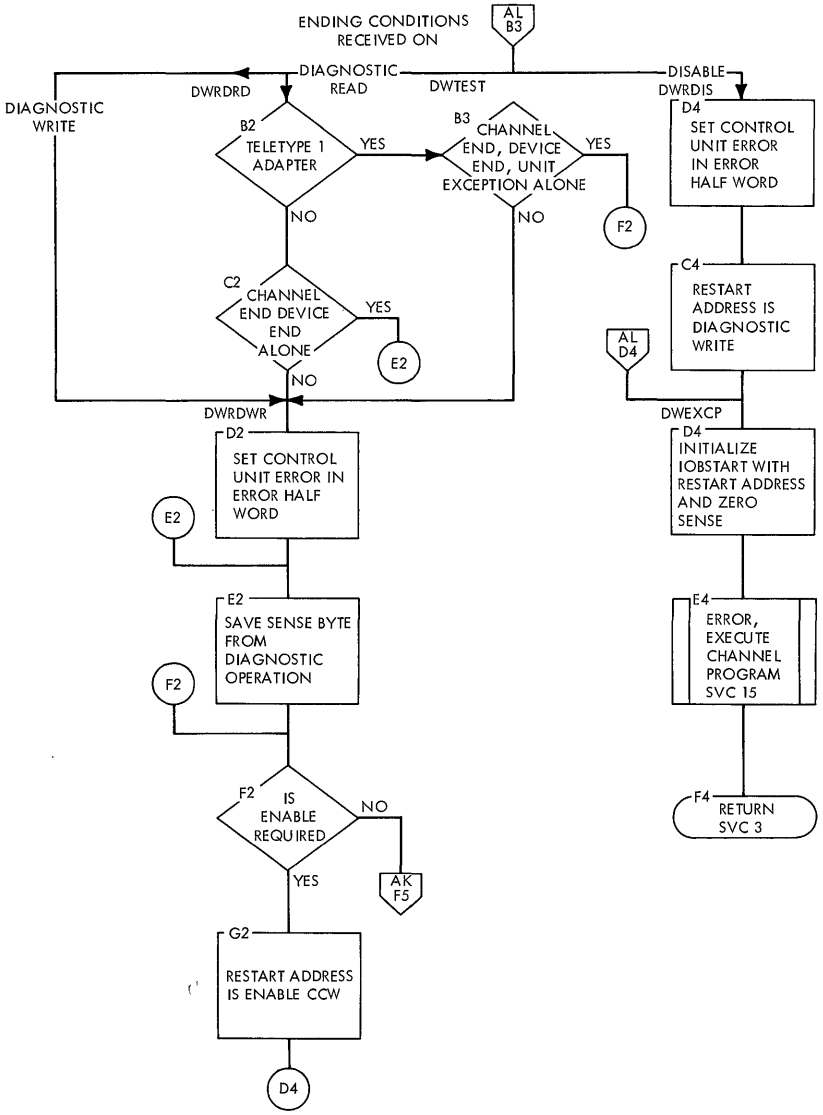


Chart AM. Status Check Routine

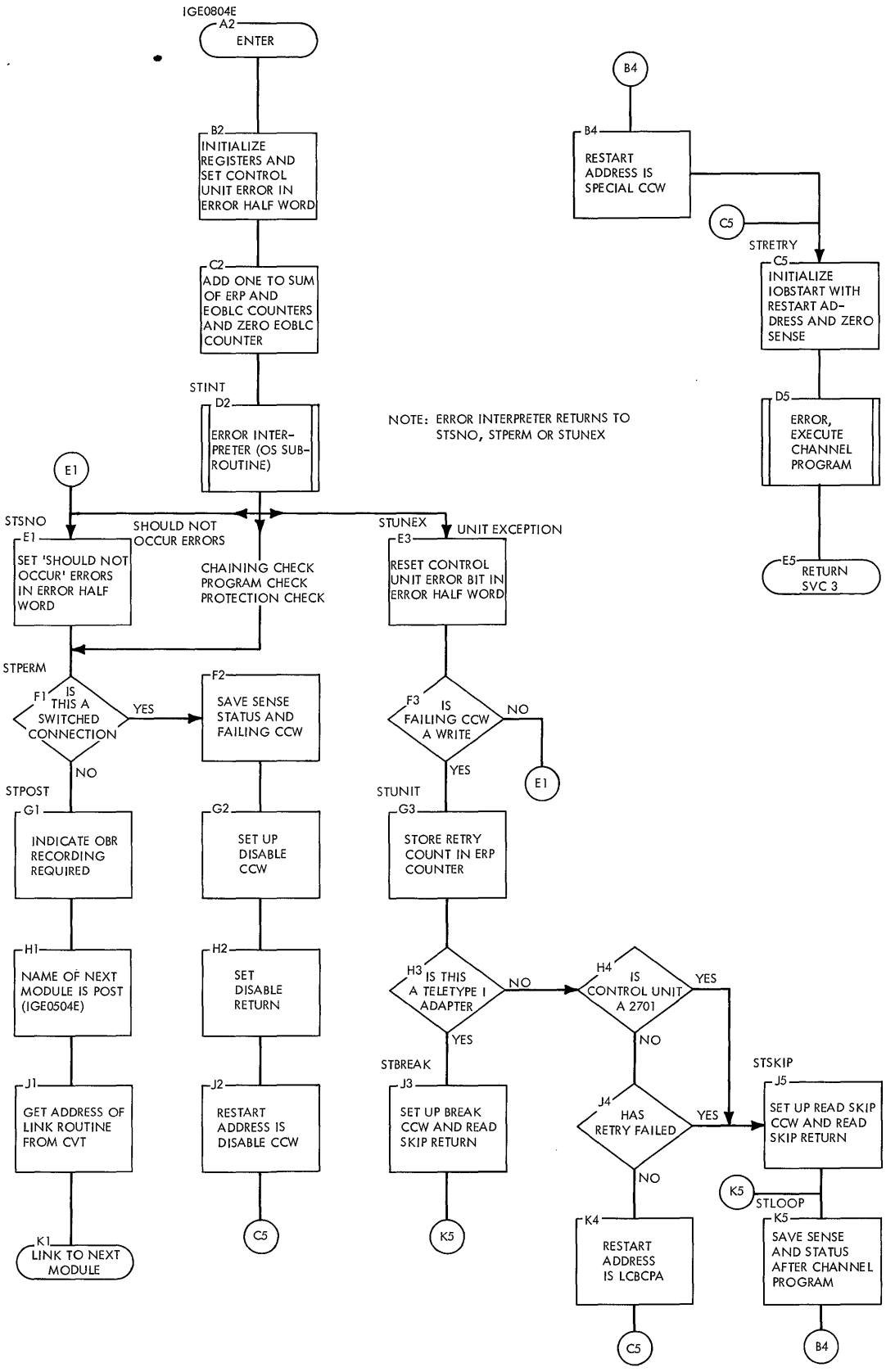


Chart AN. Command Reject, Equipment Check, SIO CC 1, SNO Error Routine

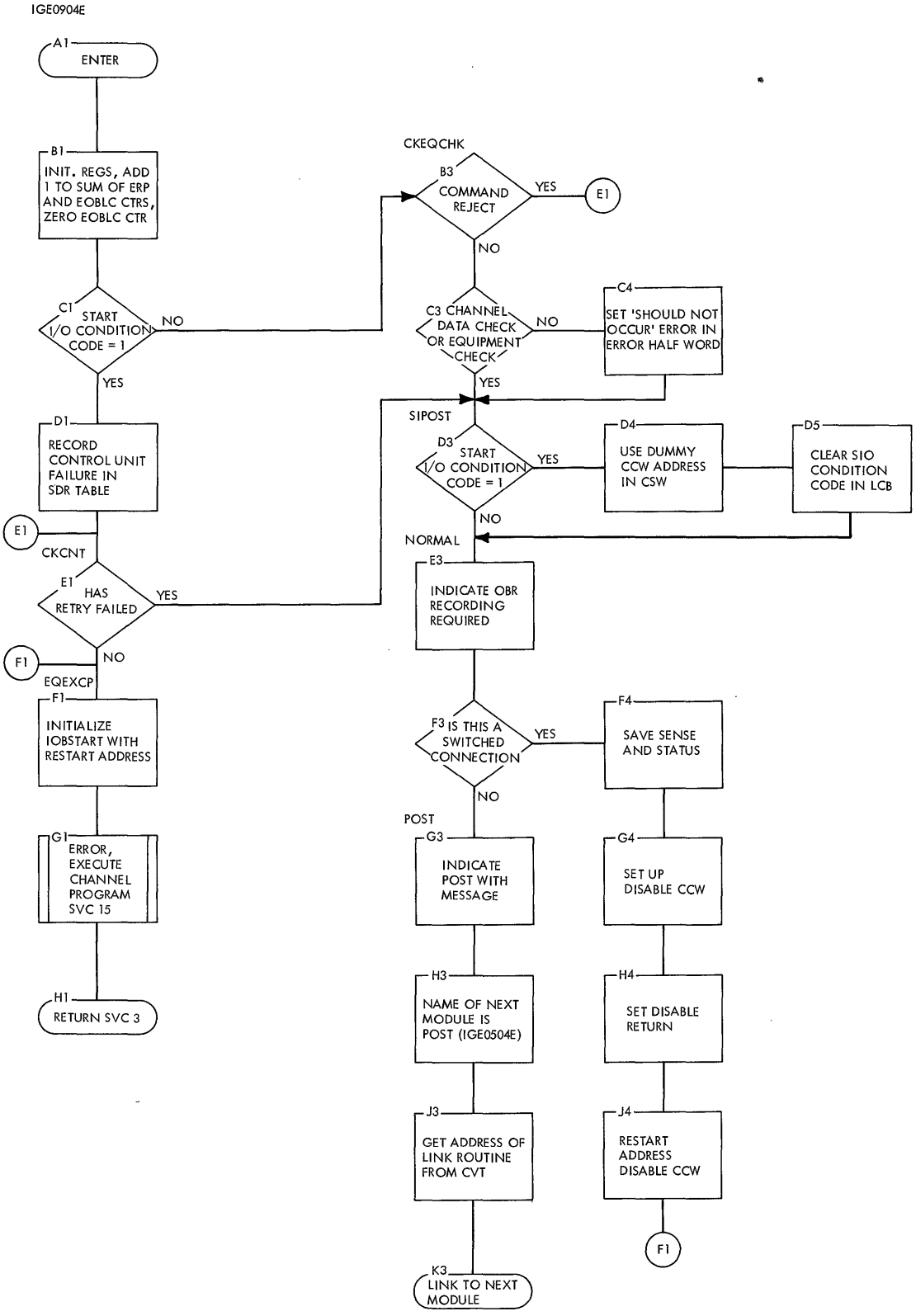


Chart AO. Read Skip Return Routine

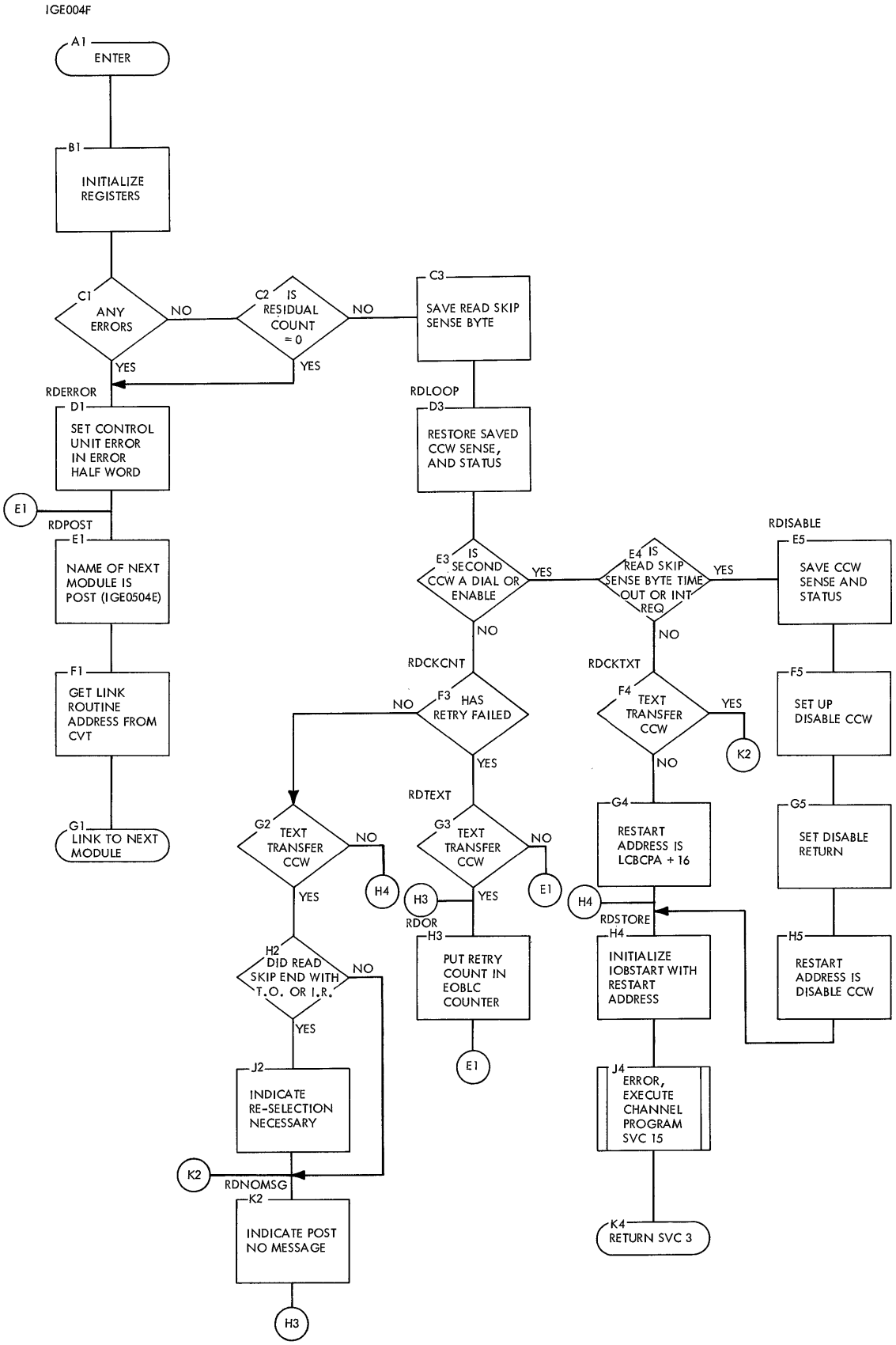


Chart AP. Diagnostic Write/Read Routine

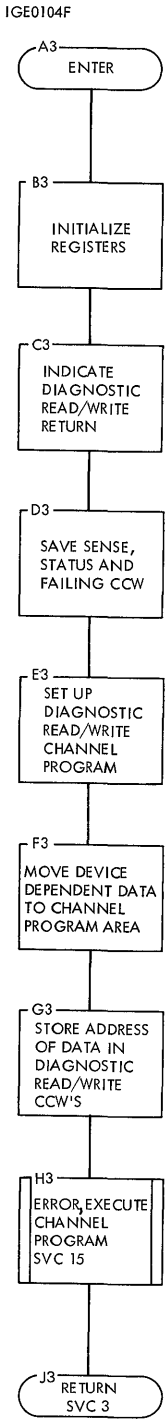


Chart AQ. Line Error Recording Routine

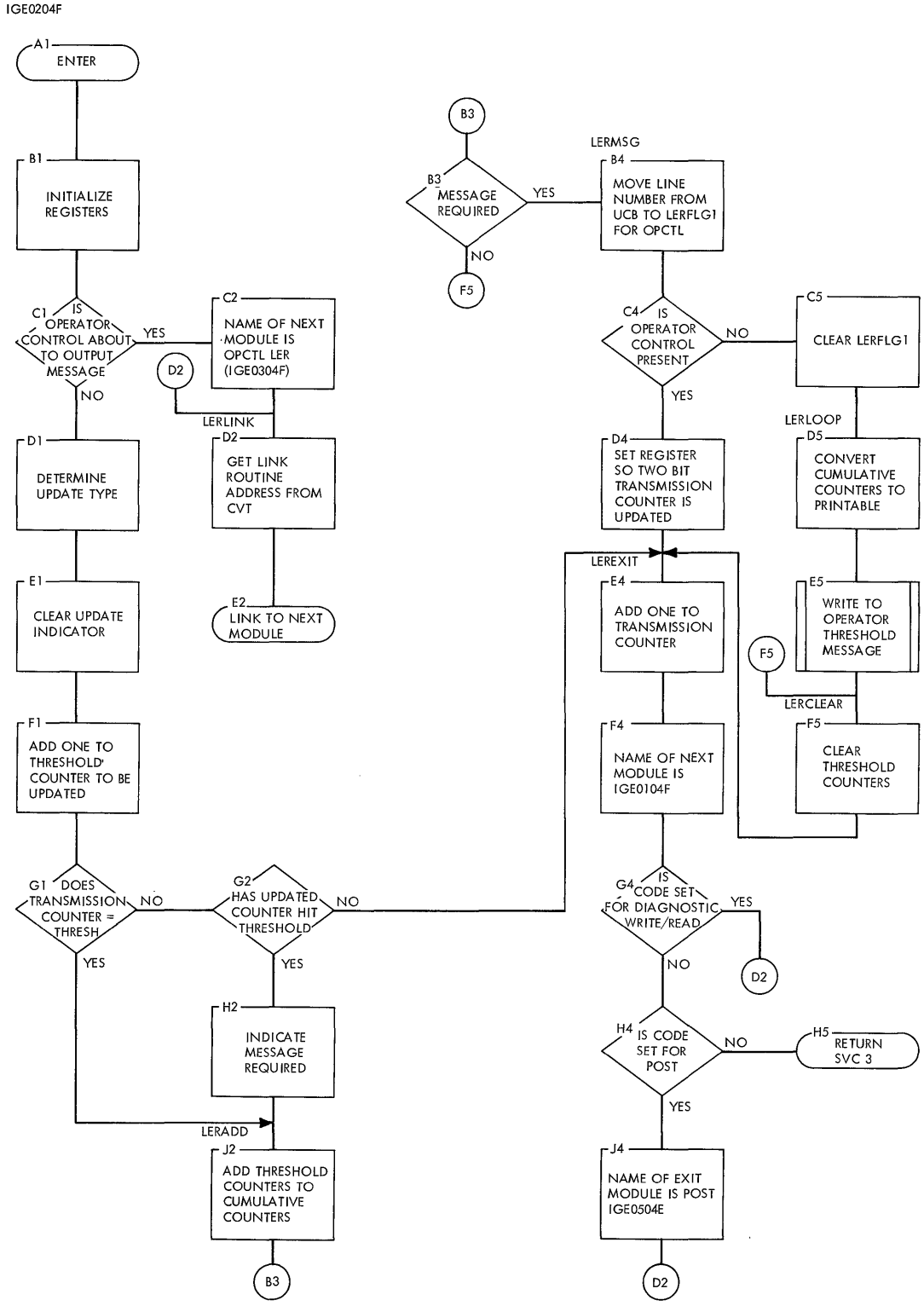


Chart AR. Operator Control LER Addition Routine

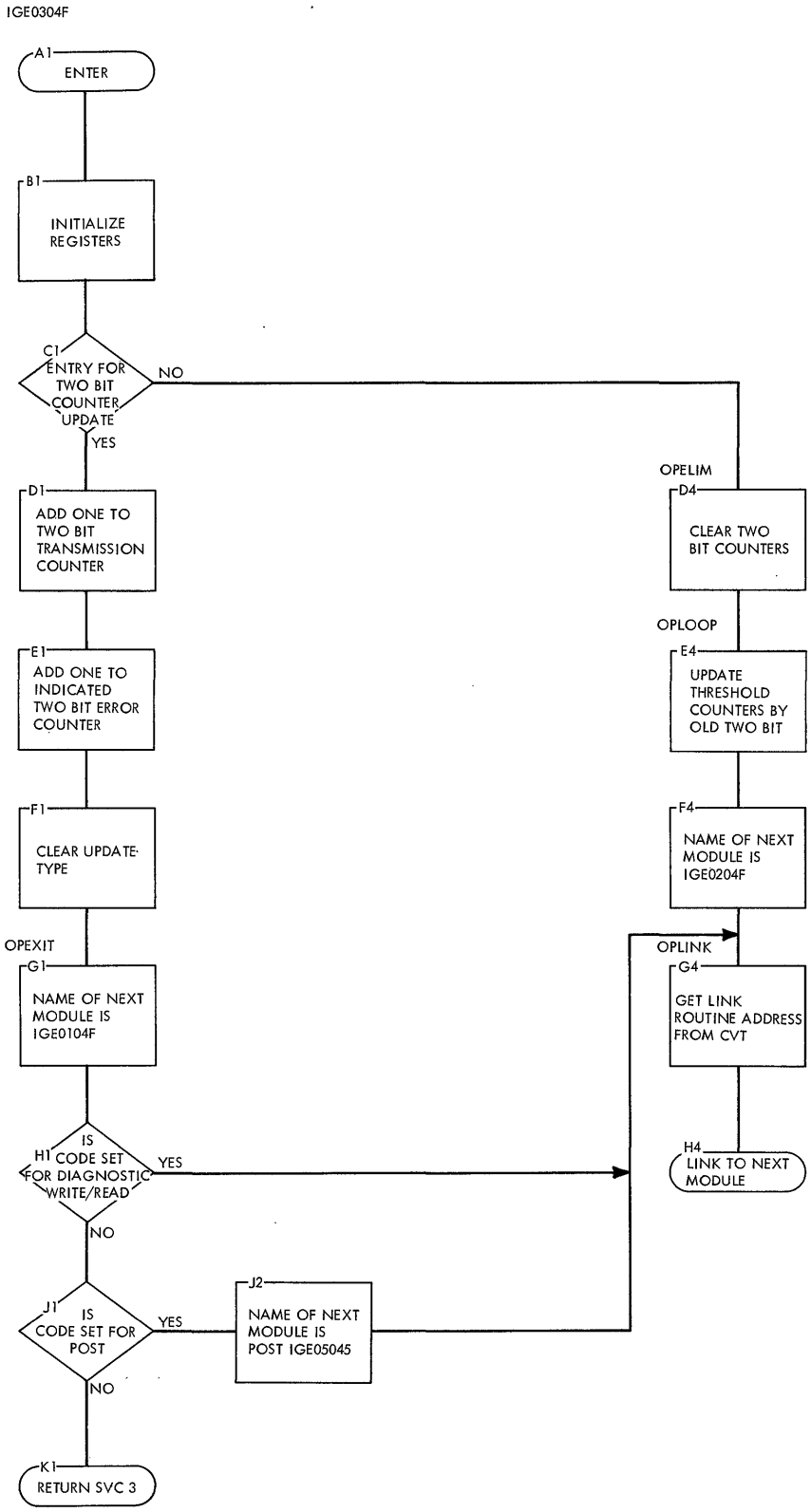


Chart AS. OPEN and Checkpoint/Restart Routine

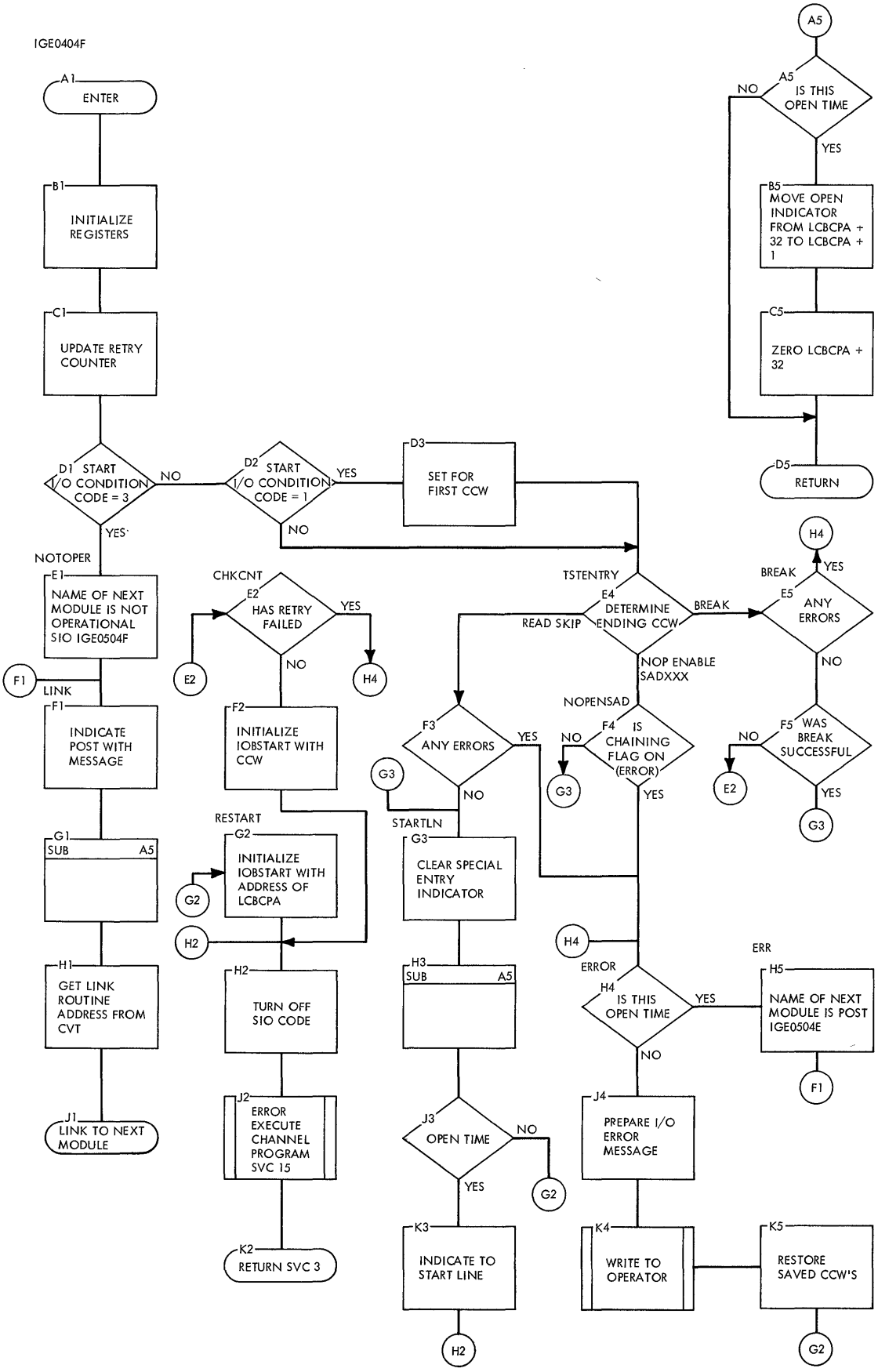


Chart AT. Not Operational Start I/O Routine

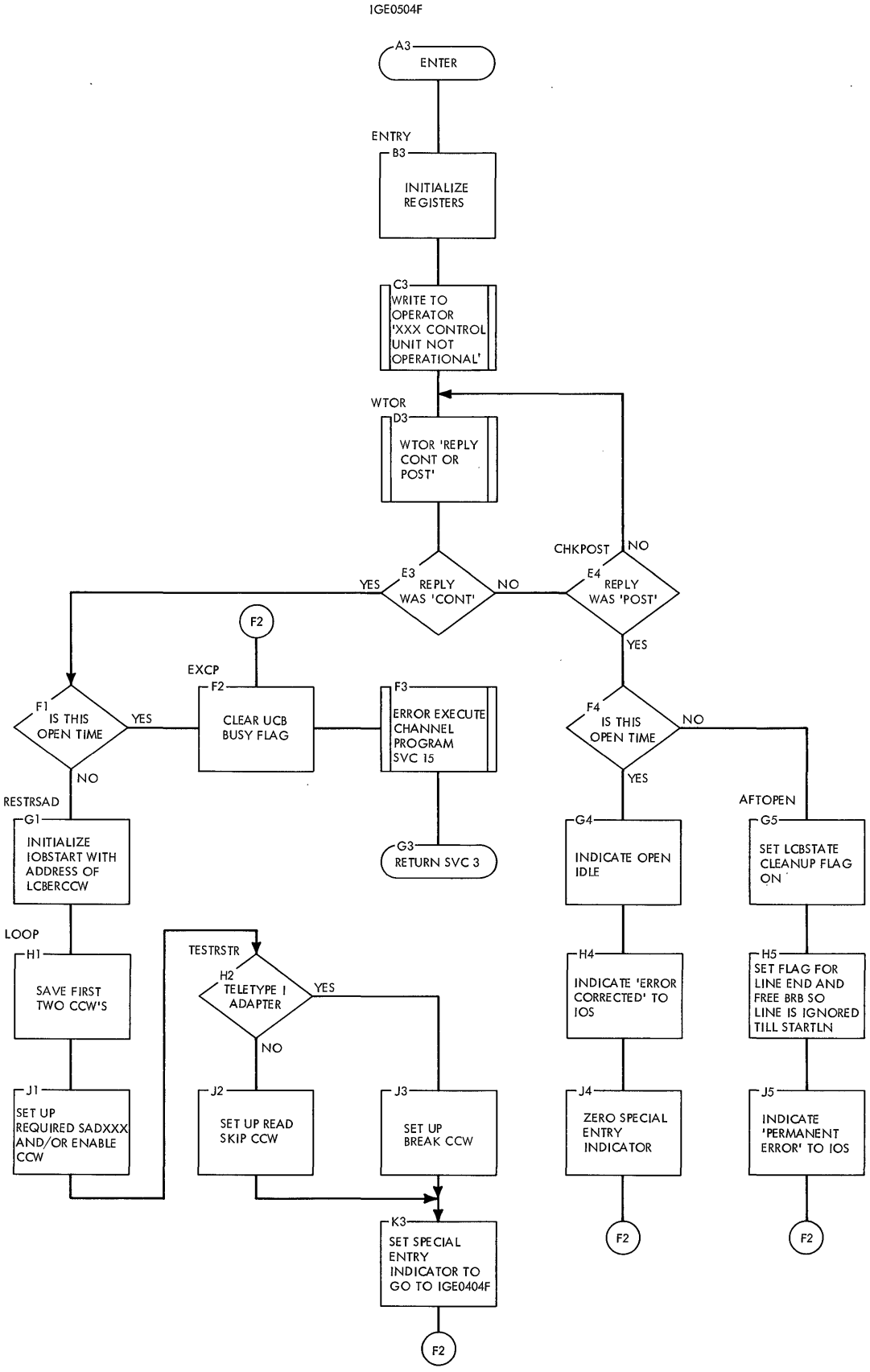


Chart AU. Bus Out and Overrun for Auto Poll Routine

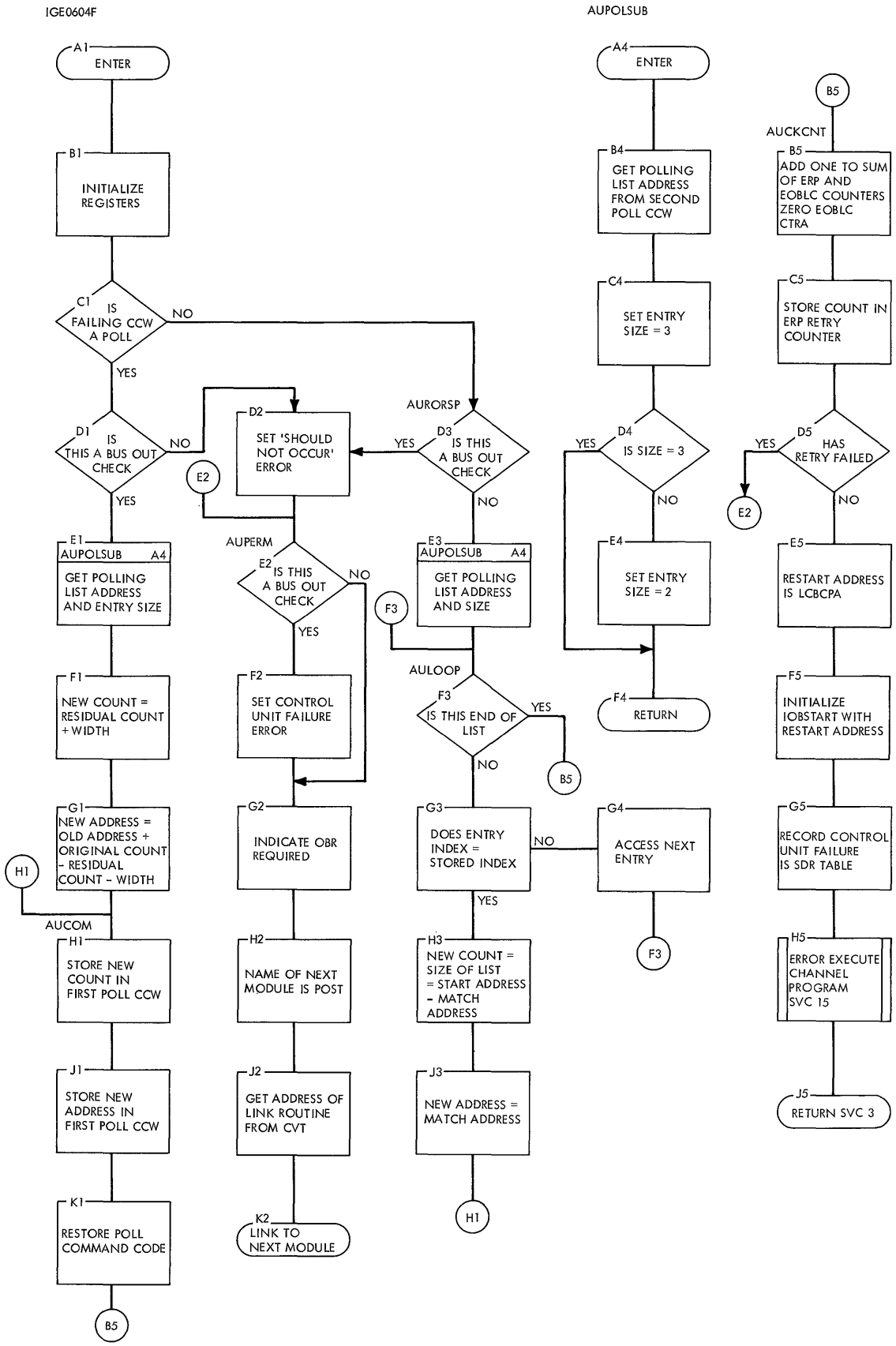


Chart AV. Overrun Routine

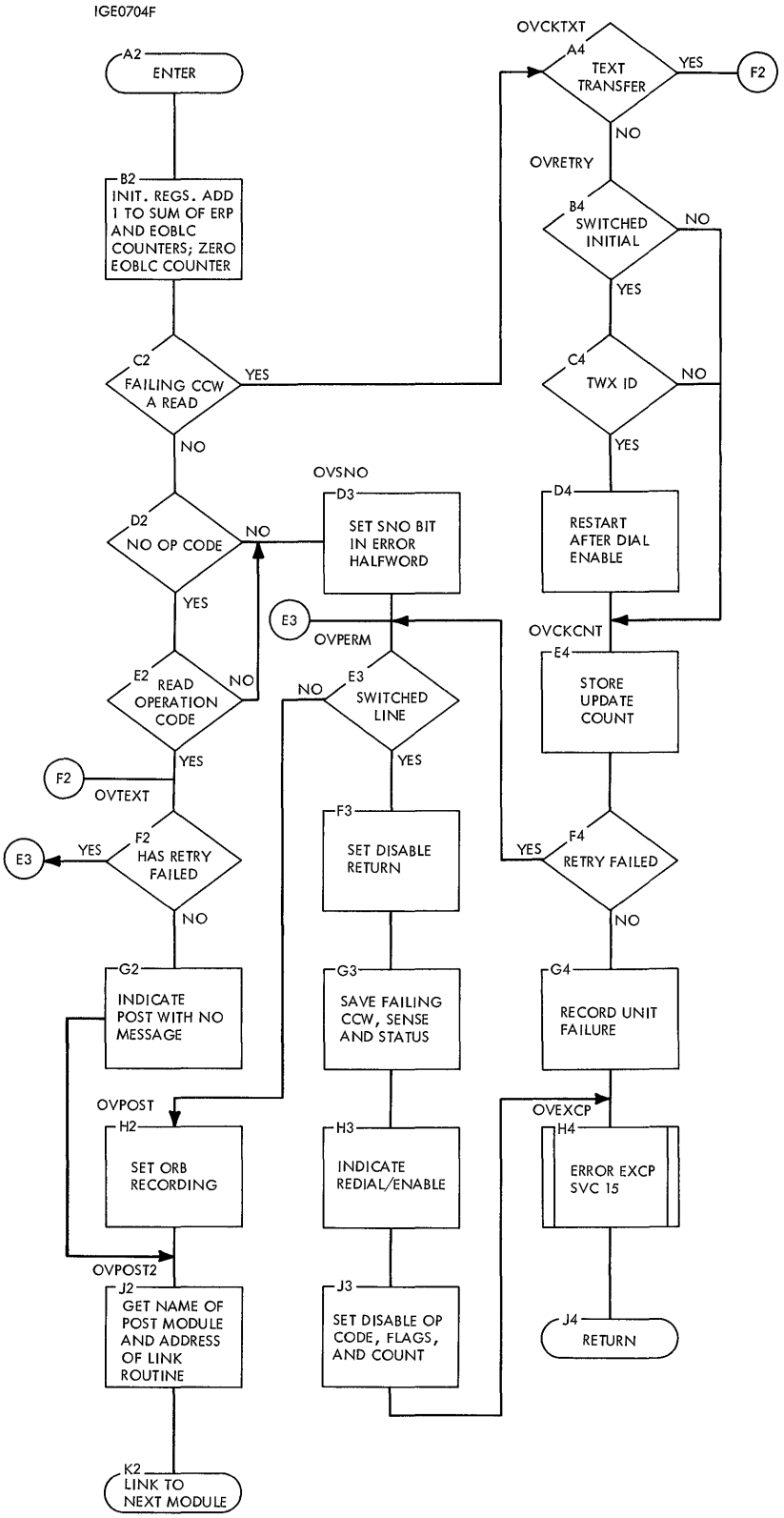


Chart BW. Locate DCB Routine

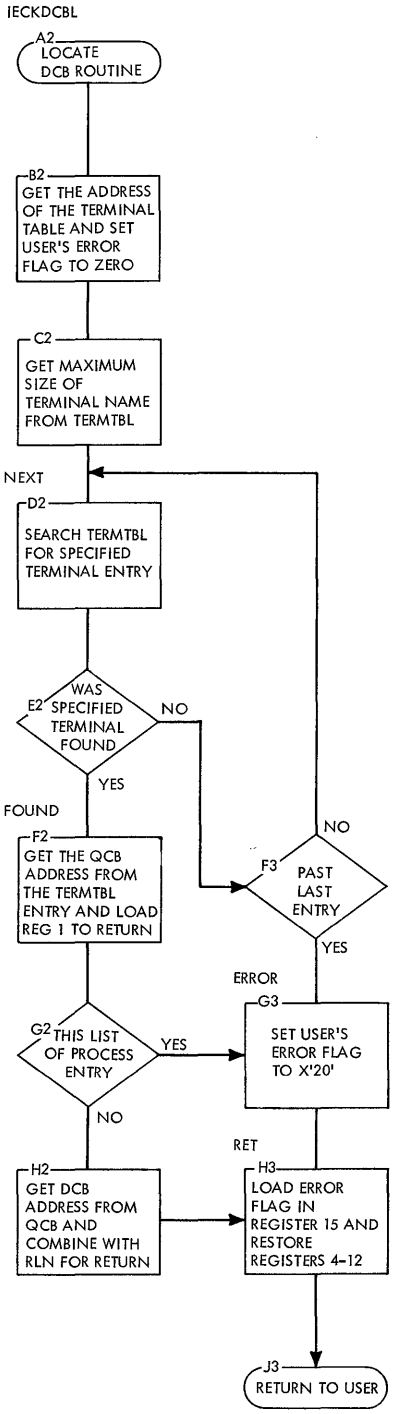
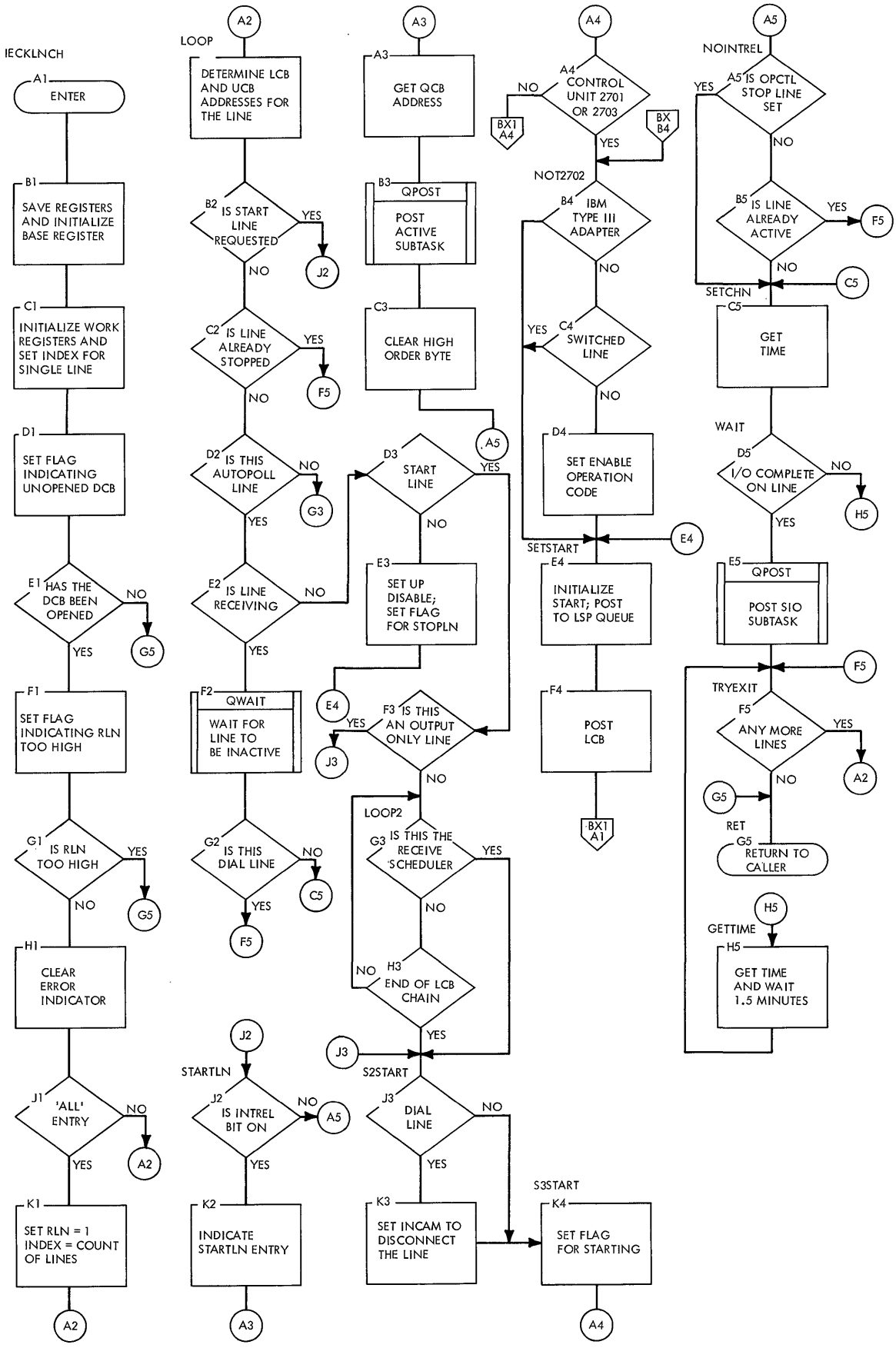


Chart BX. Start Line-Stop Line Routine



● Chart BX1. QTAM Start Line-Stop Line Routine

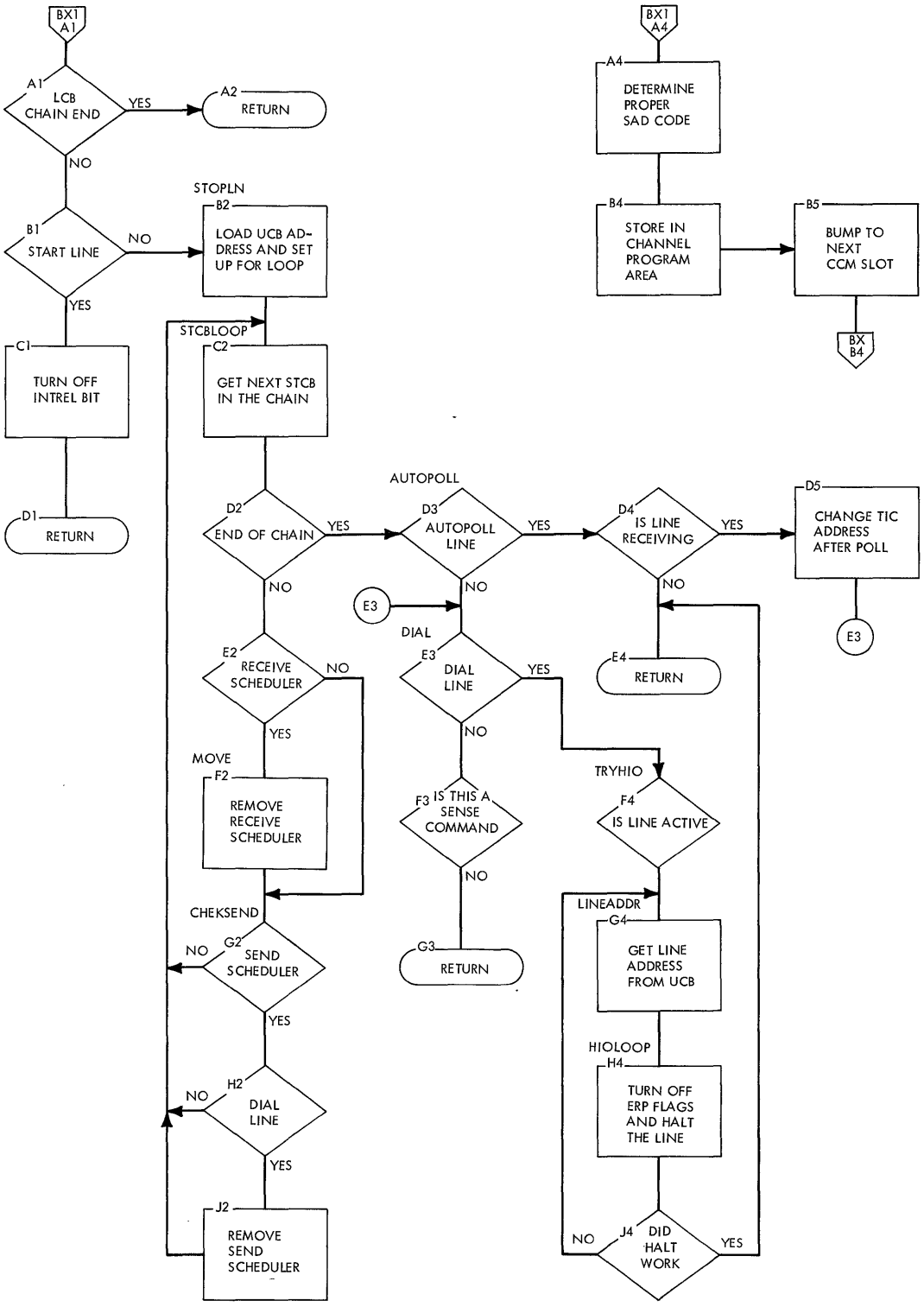


Chart BY. Breakoff Routine

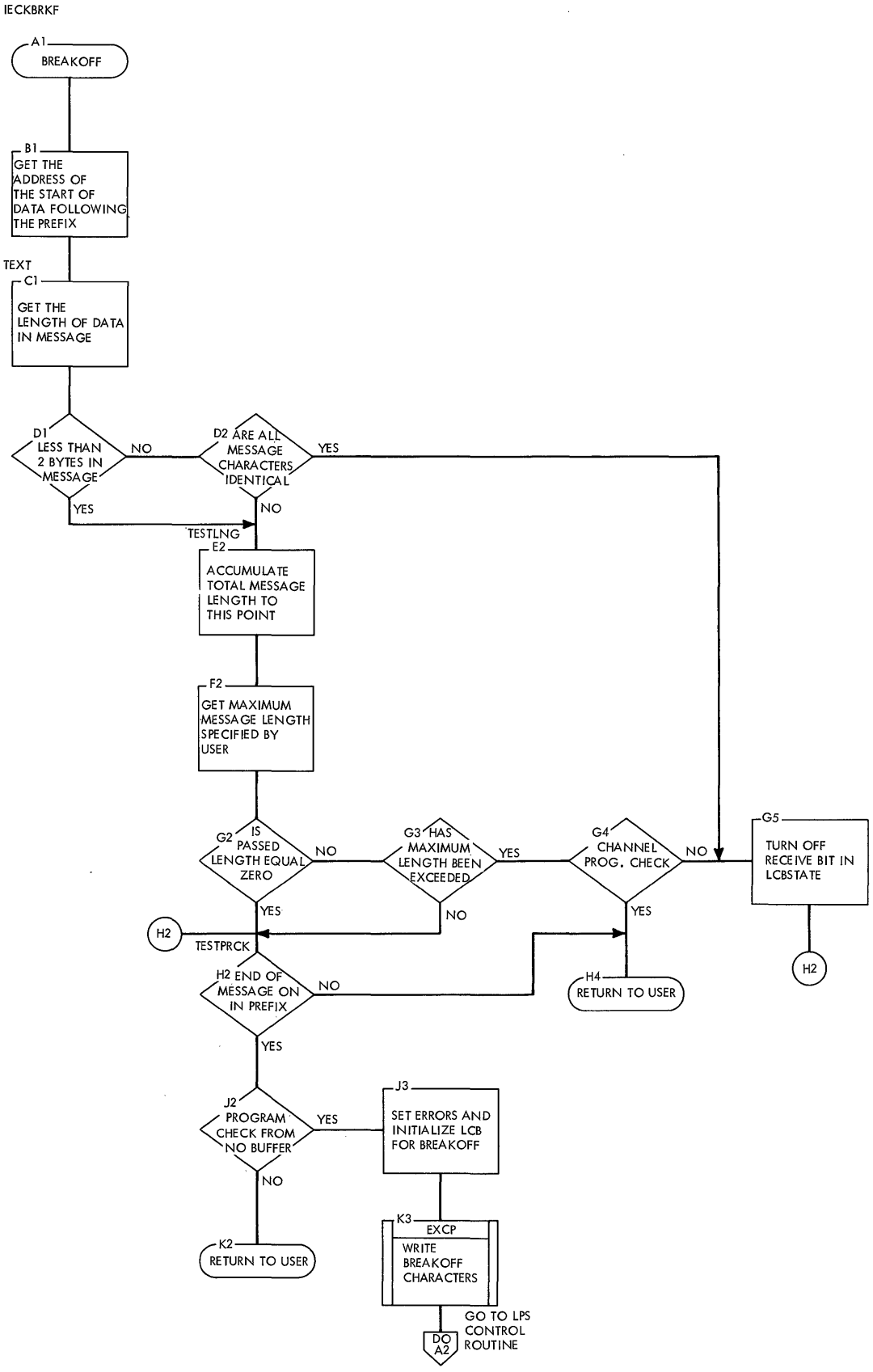


Chart BZ. Release Intercepted Messages Routine

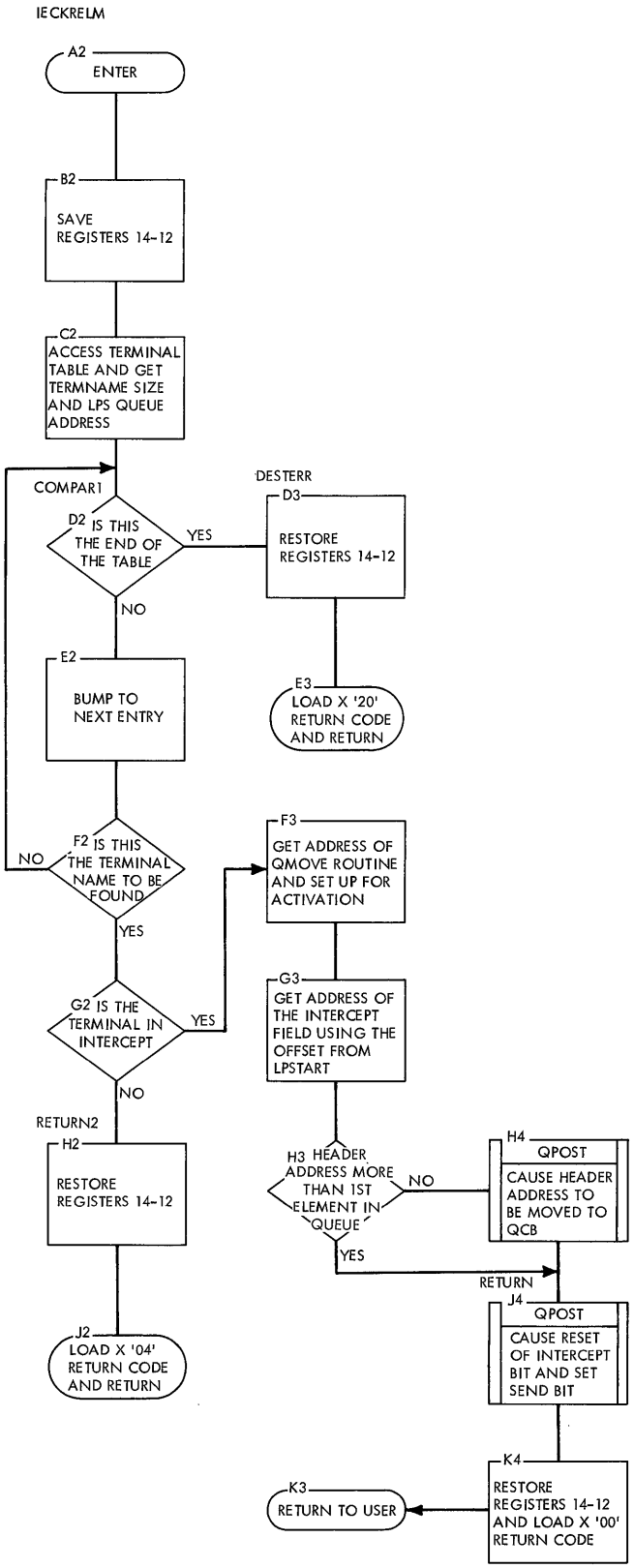


Chart C0. Pause Routine

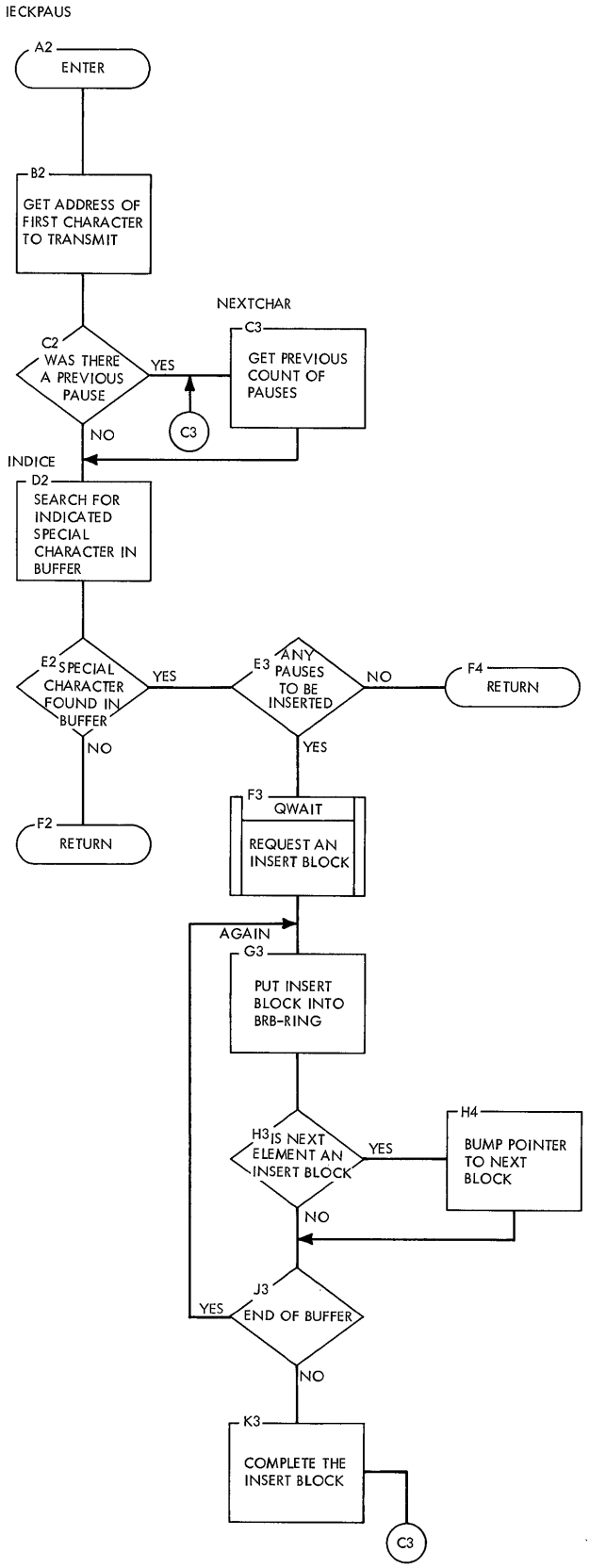


Chart C1. Retrieve - DASD Routine

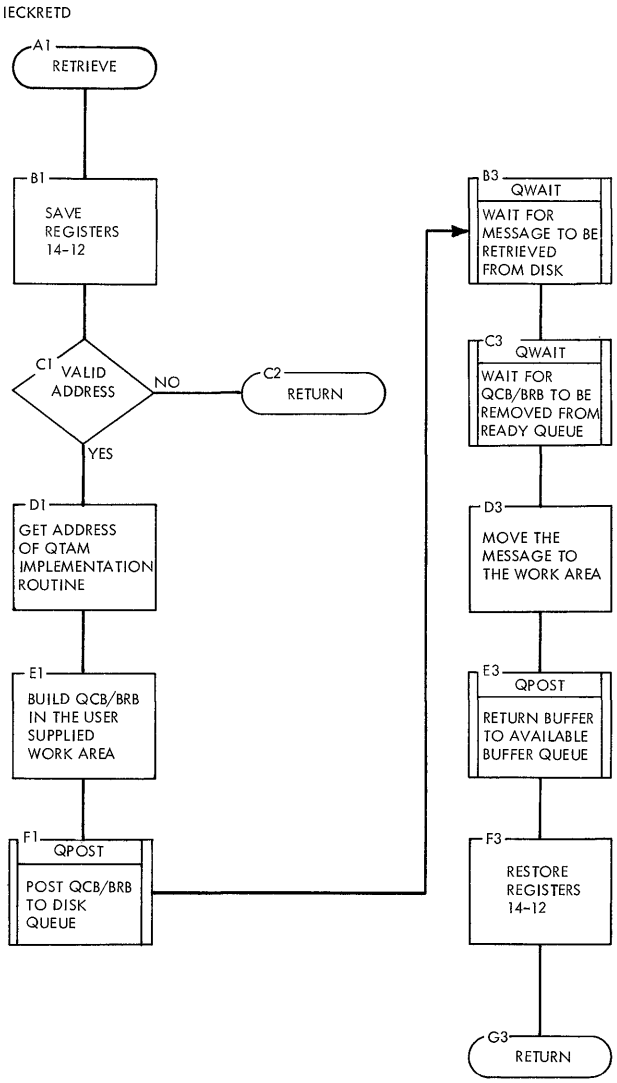


Chart C2. Retrieve by Sequence Number Routine

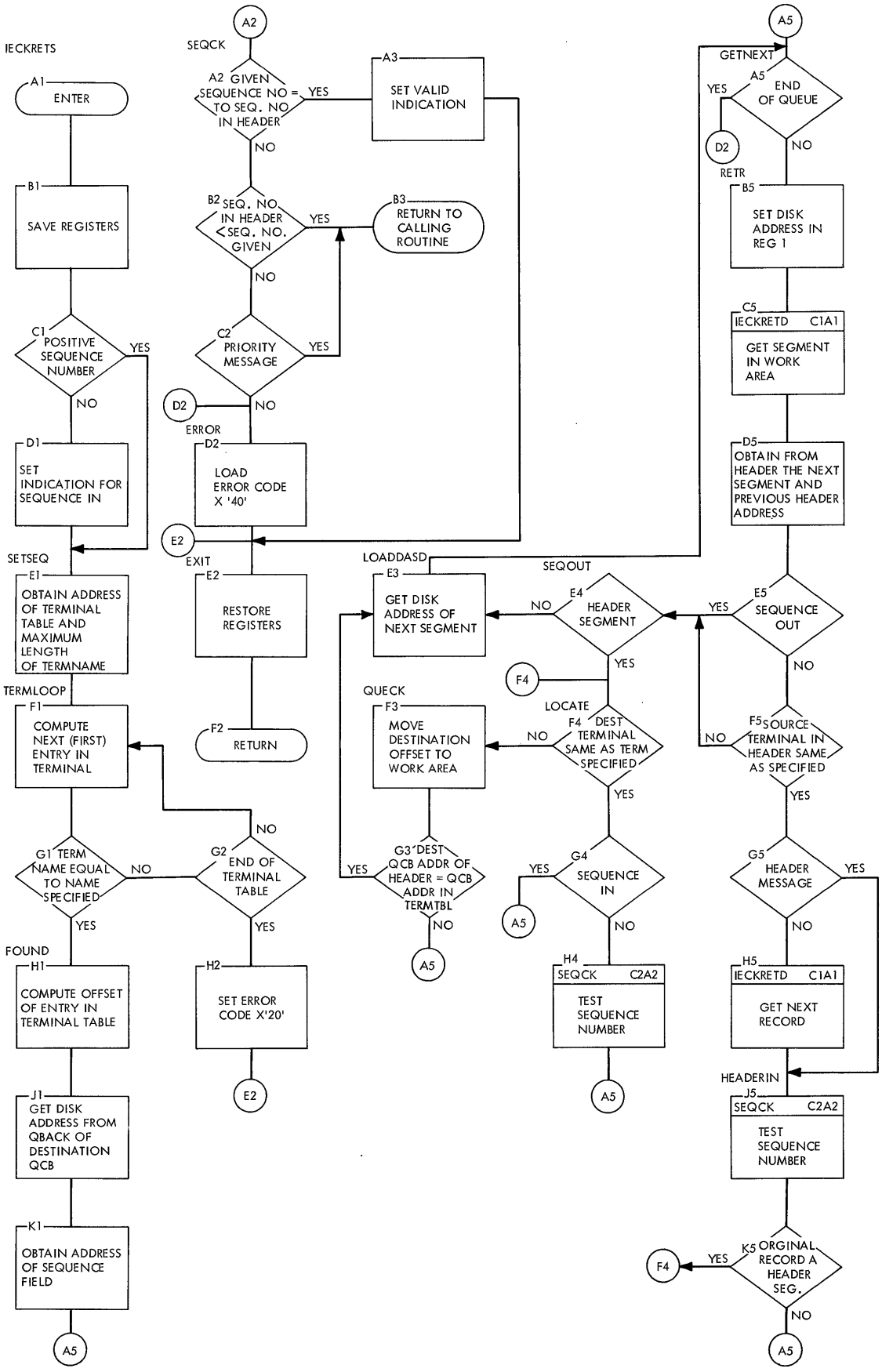


Chart C3. Checkpoint Request Routine

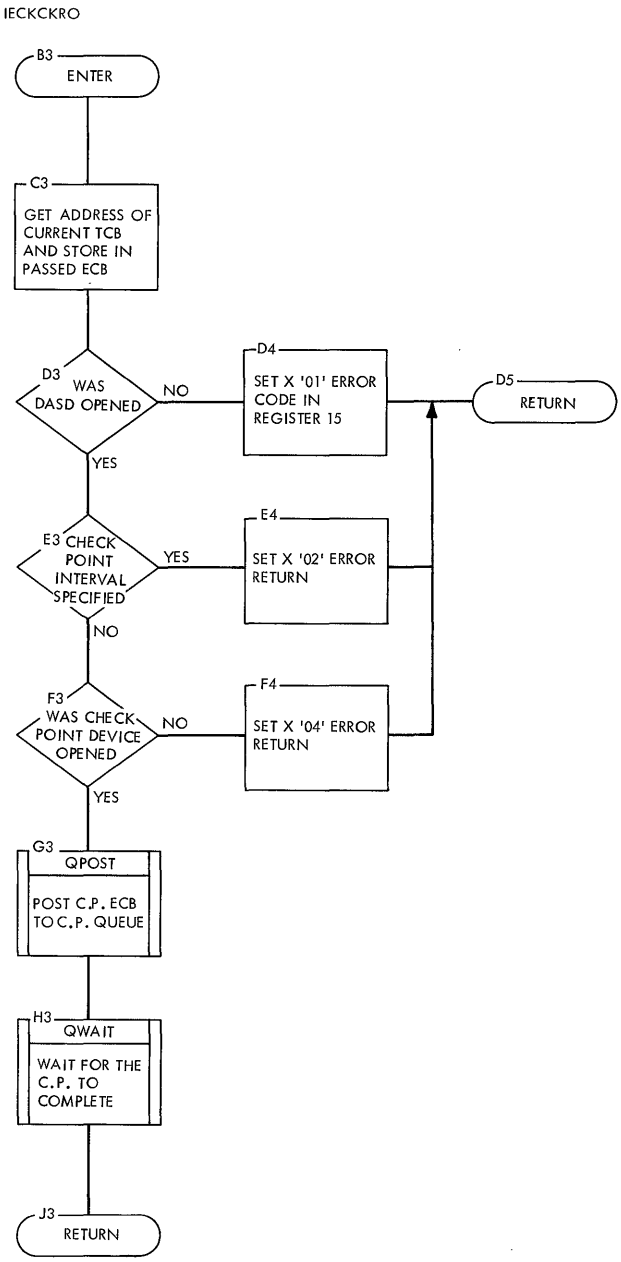


Chart C4. Open Message Process Queue

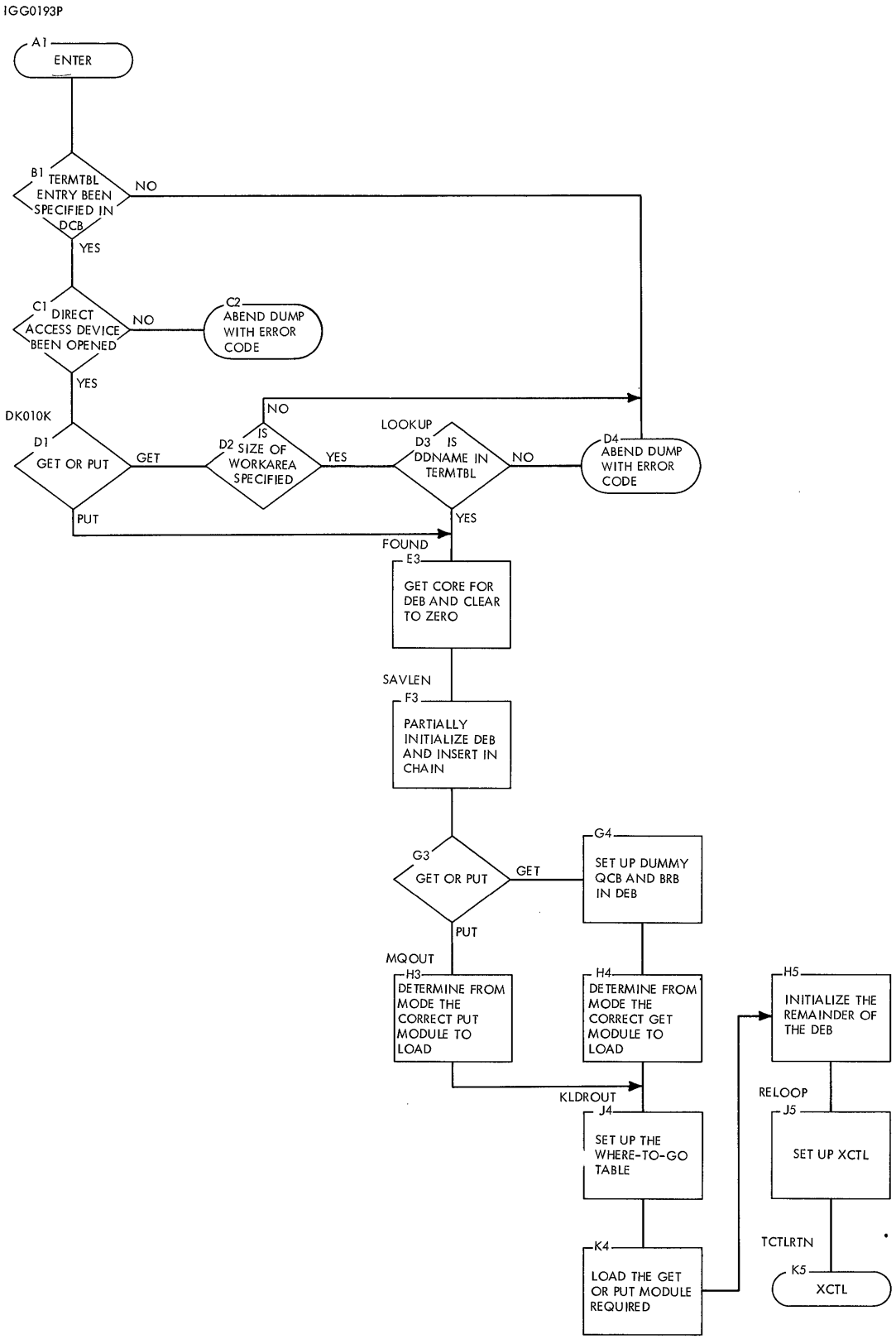


Chart C5. Get Segments Routine

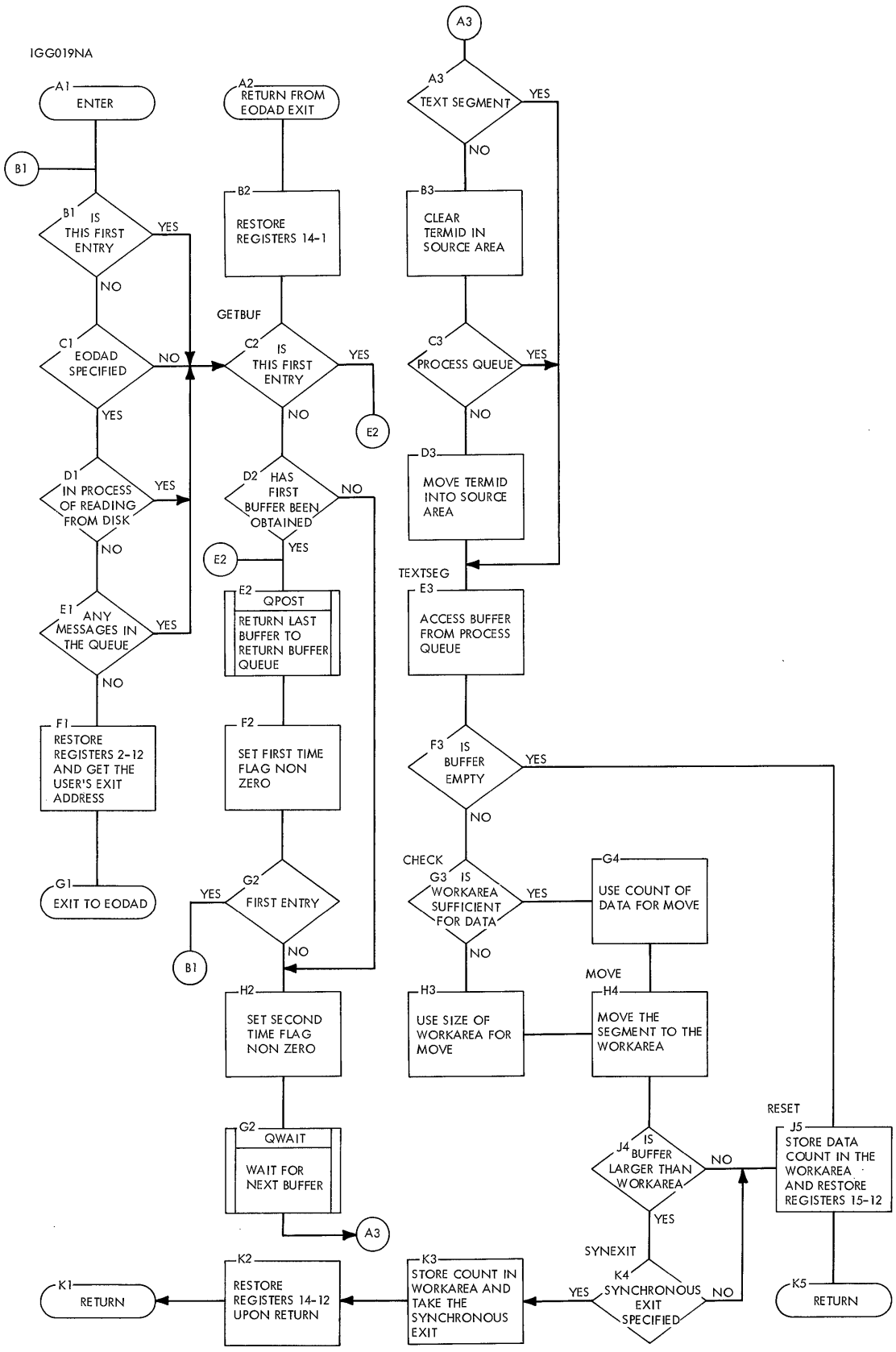


Chart C6. Get Messages Routine

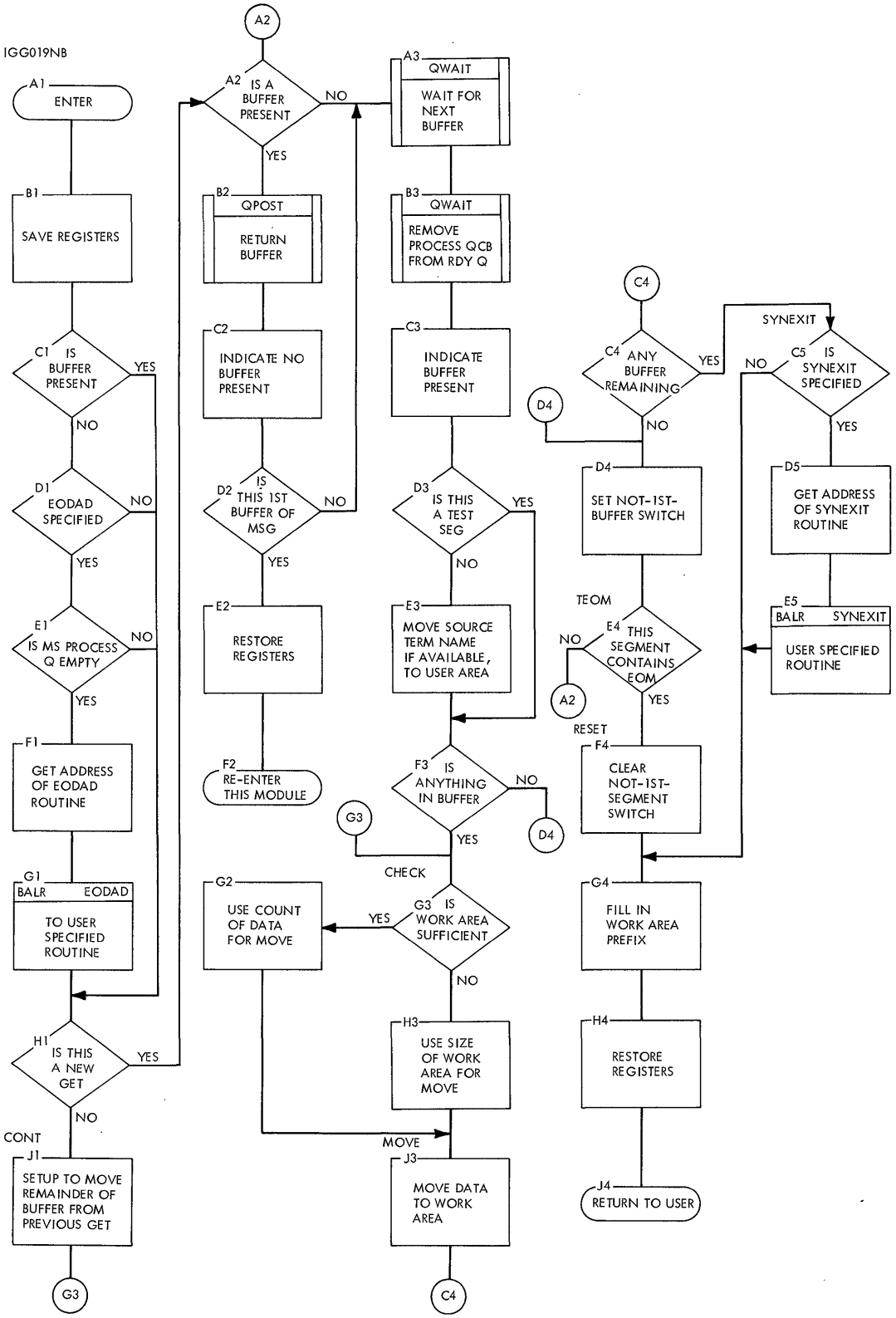


Chart C7. Get Records Routine

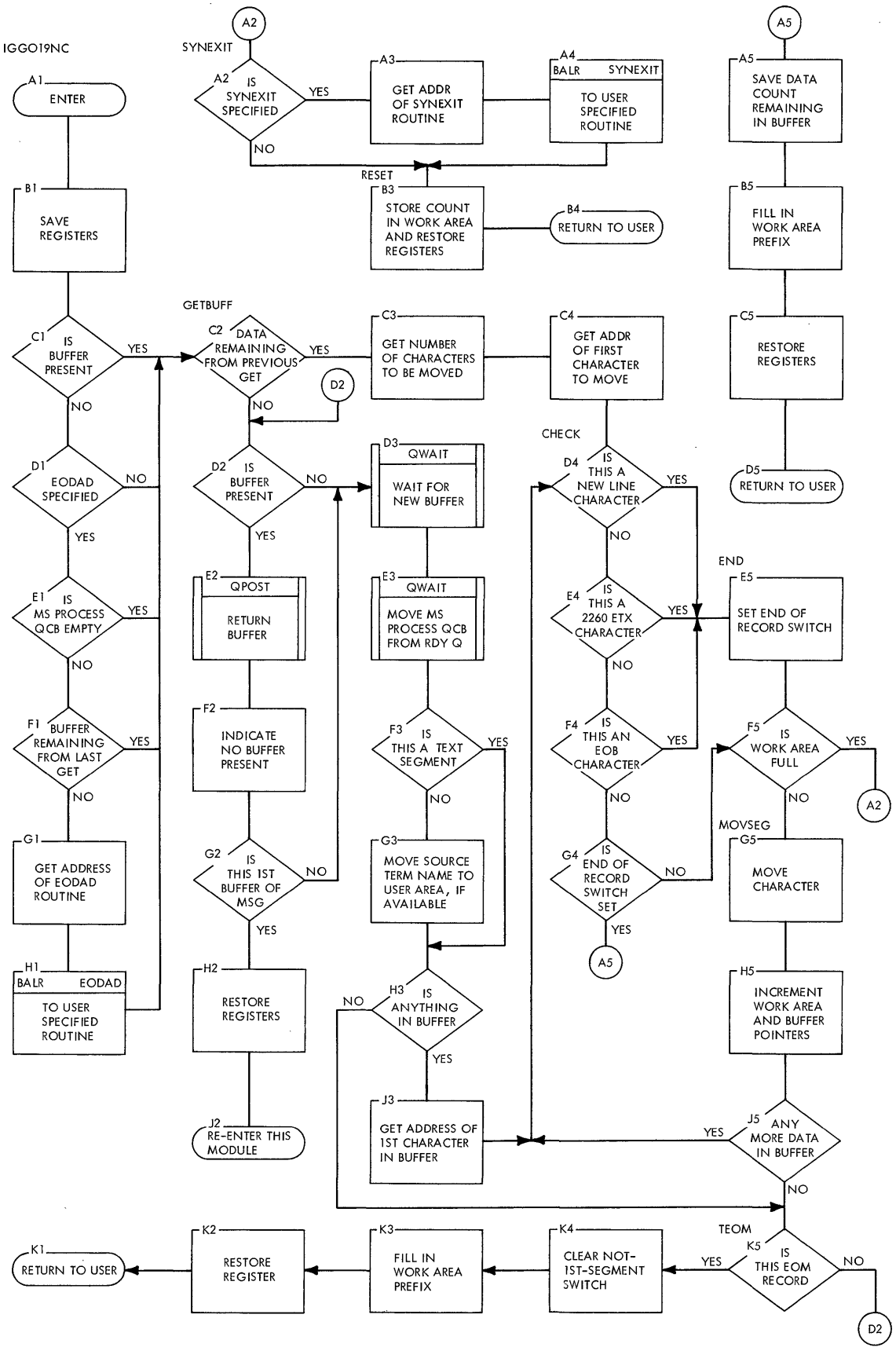


Chart C8. Put Message Segment Routine

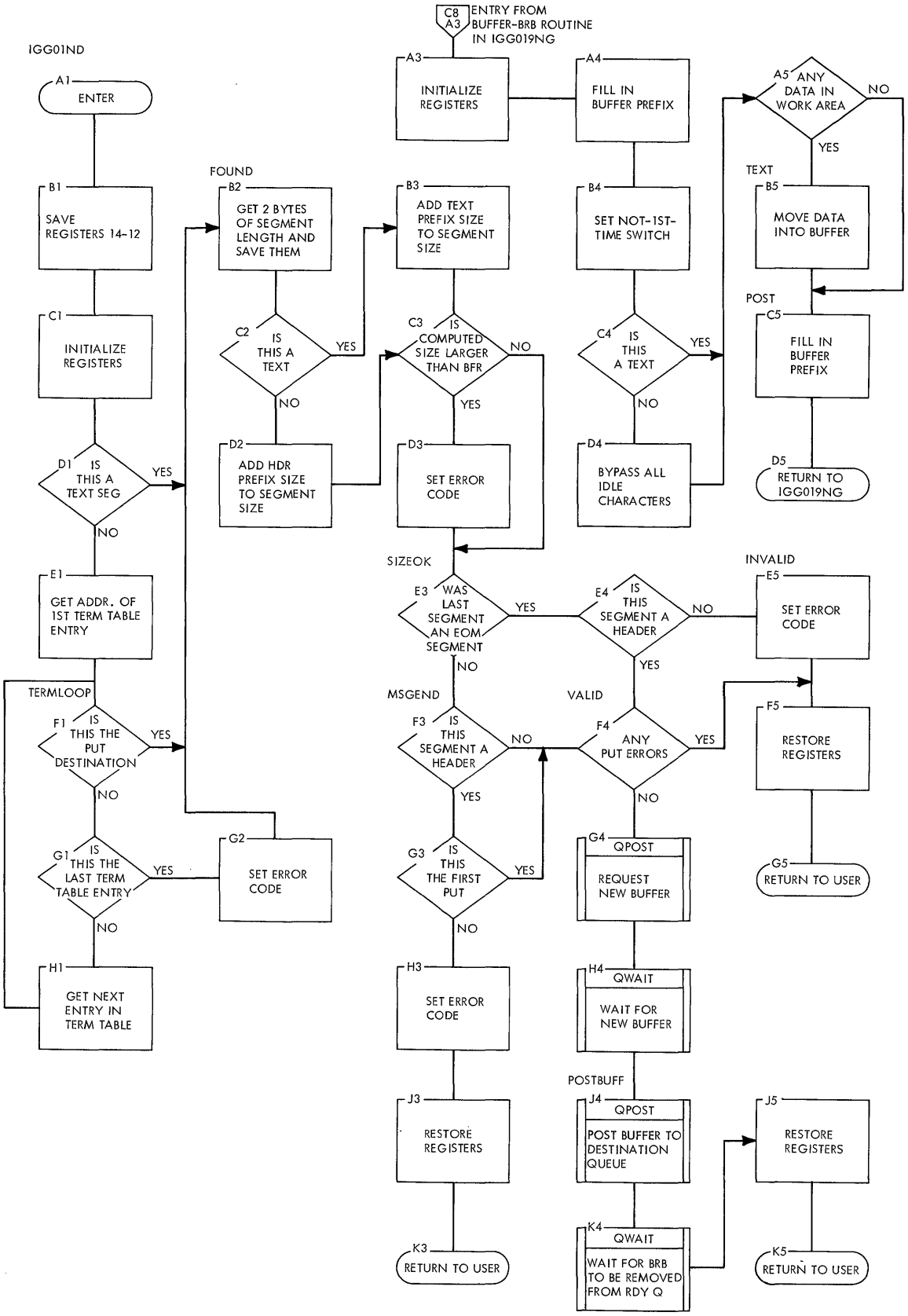


Chart C9. Put Record Routine

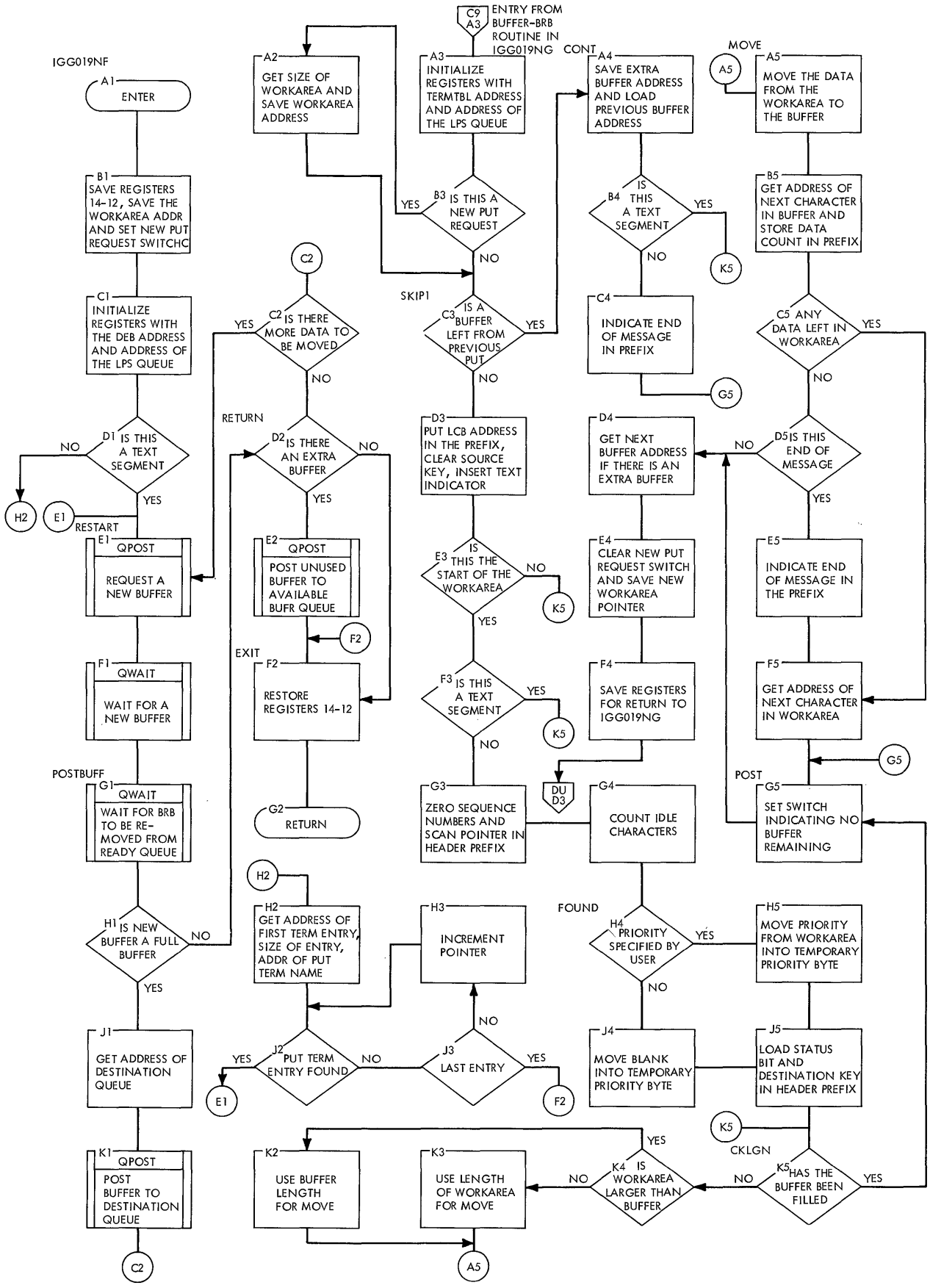


Chart CA. Message Type Routine

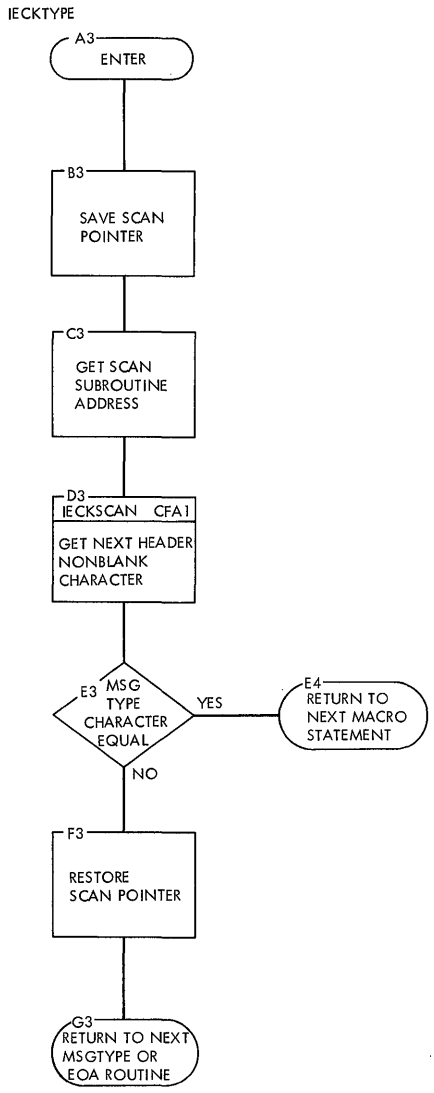


Chart CB. Change Terminal Table Routine

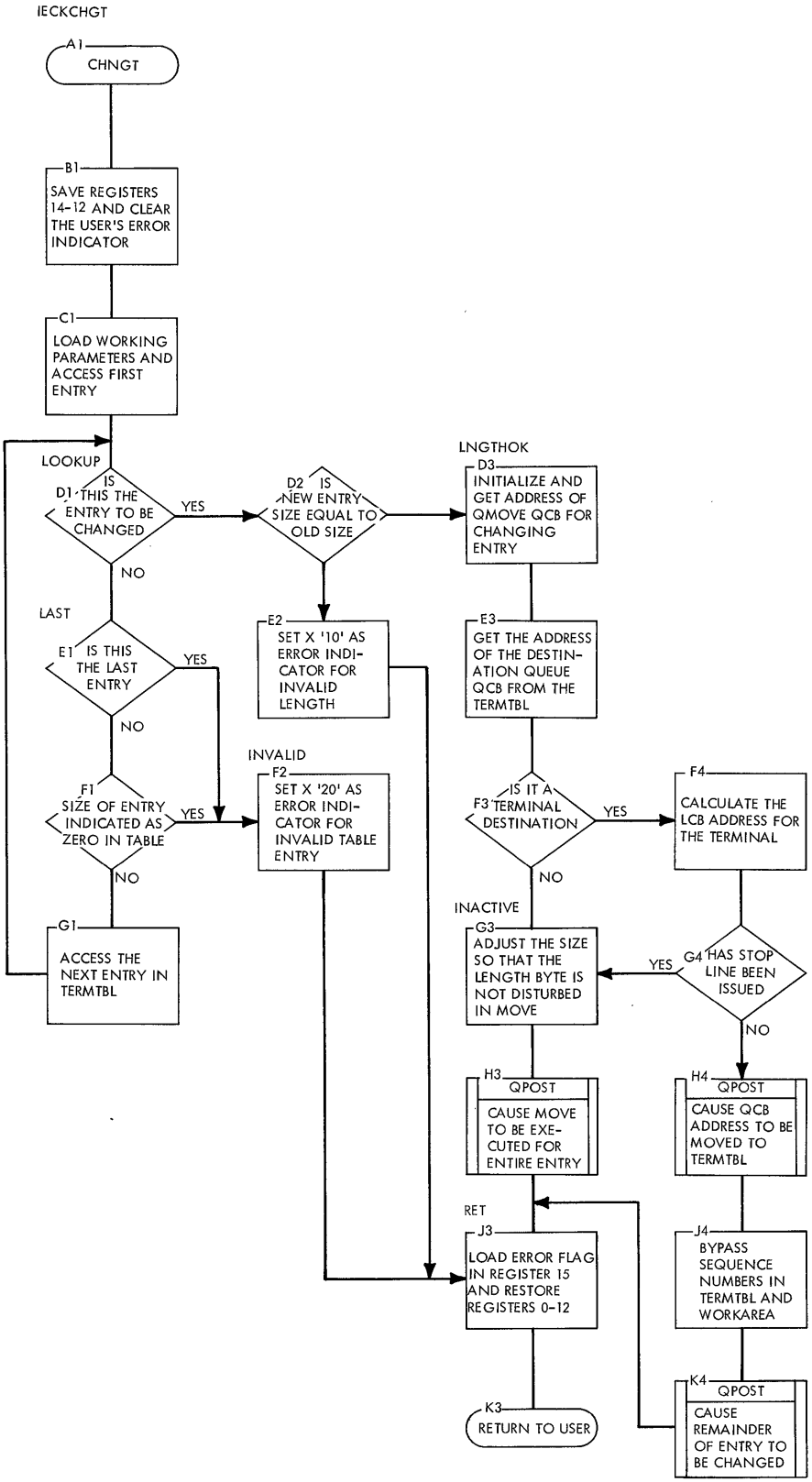


Chart CC. Copy Polling List Routine

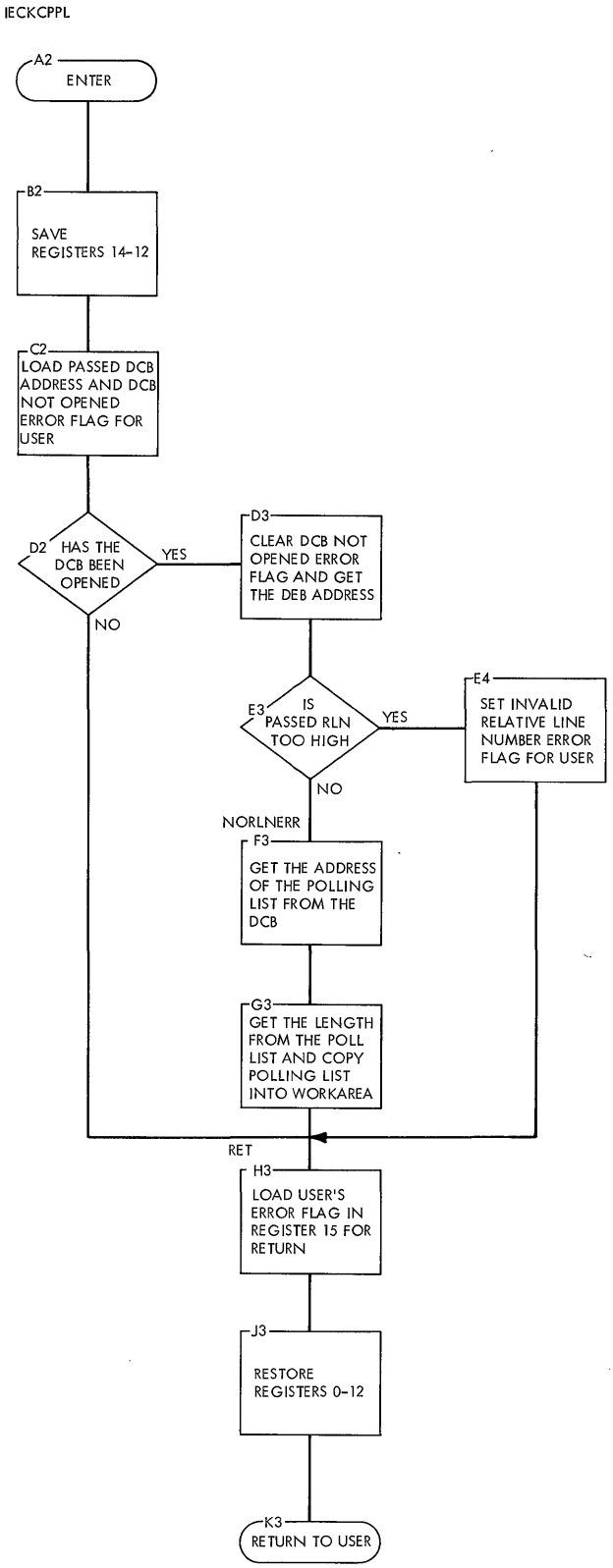


Chart CD. Change Polling List Routine

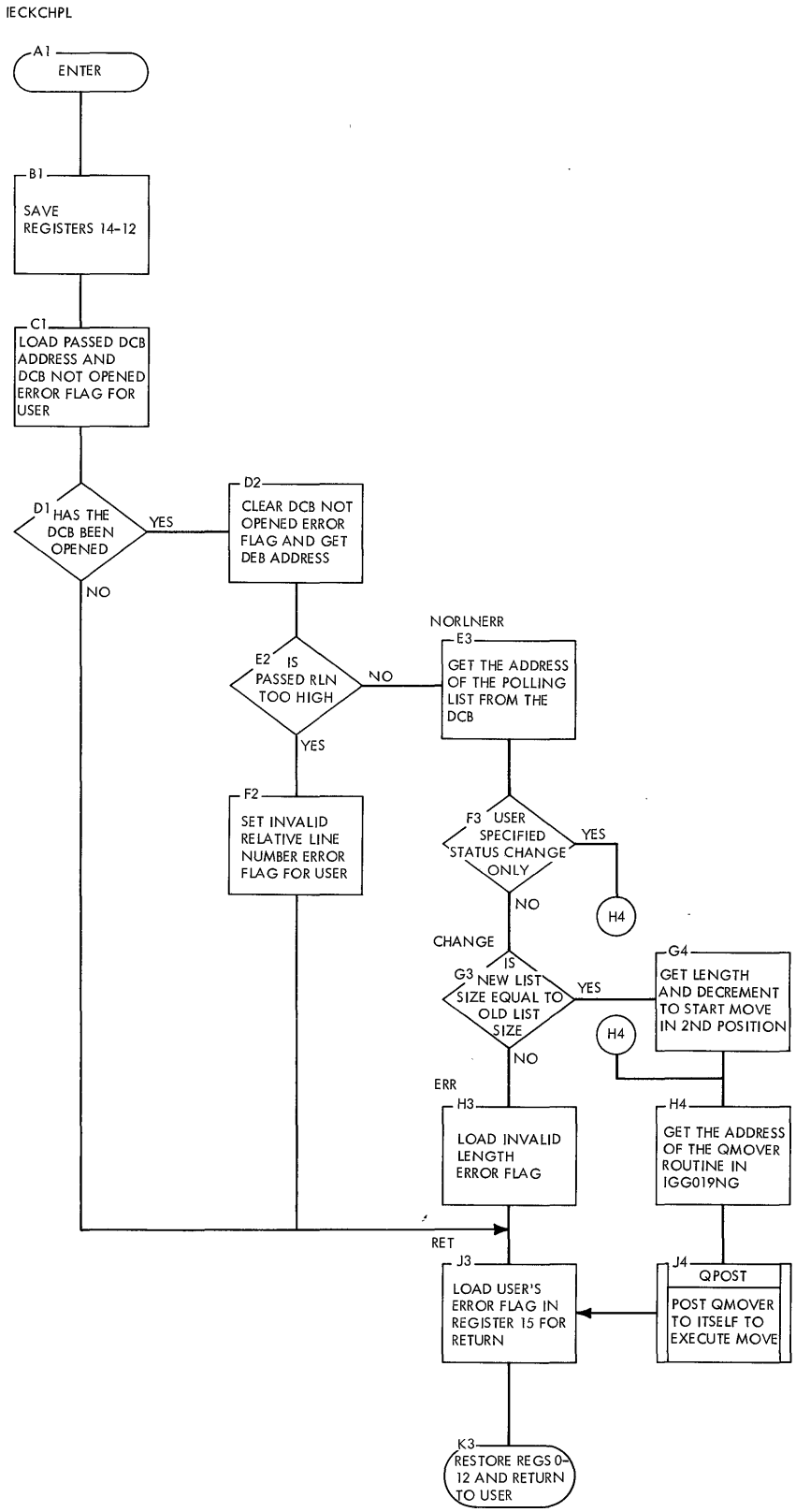


Chart CE. Copy Queue Control Block Routine

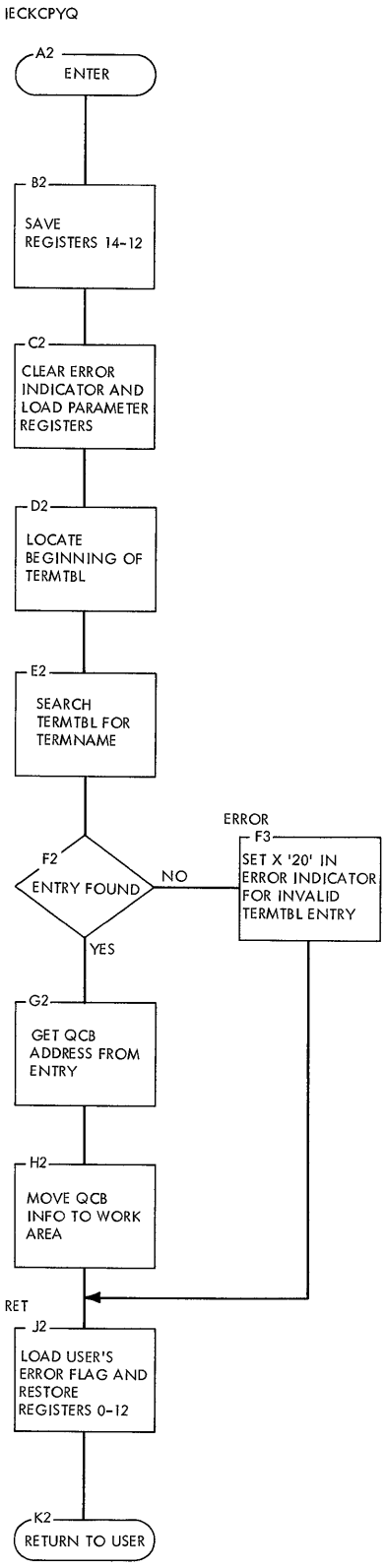


Chart CF. Scan Routine

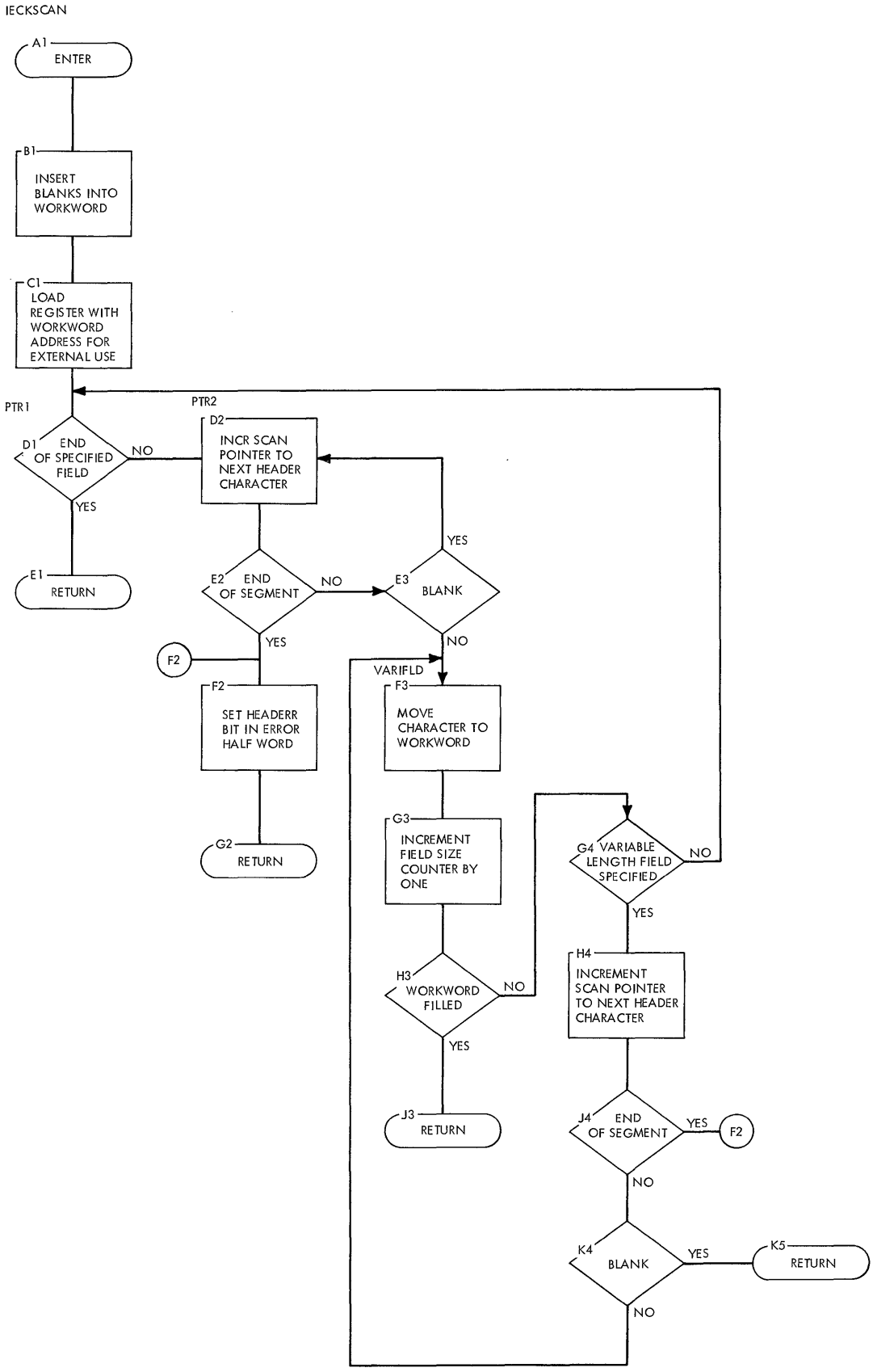


Chart CG. Copy Terminal Table Routine

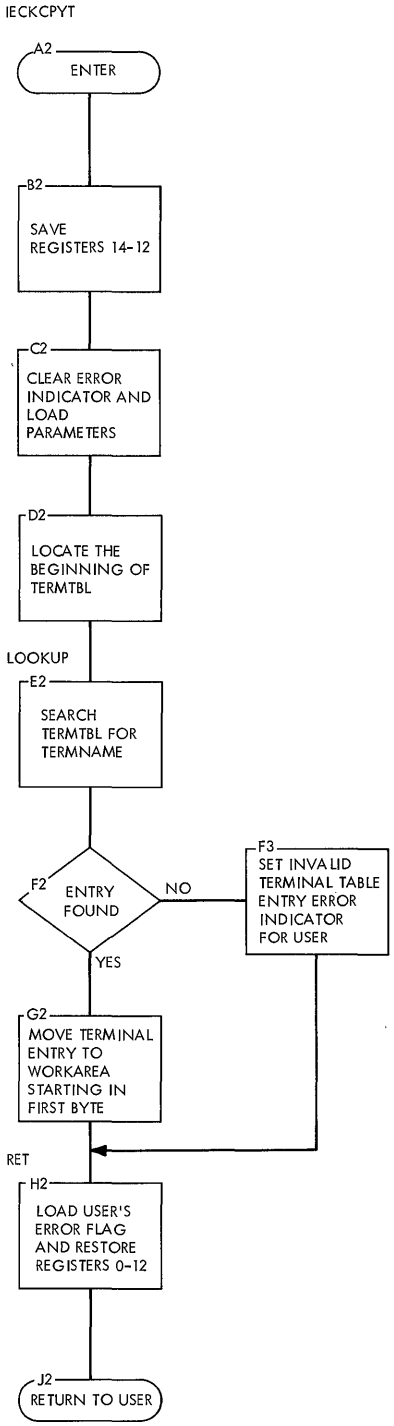


Chart CH. Date Stamp Routine

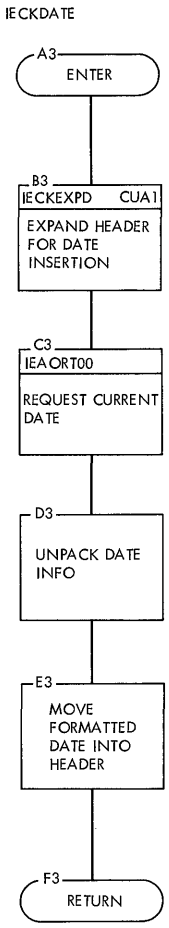


Chart CI. Source Routine

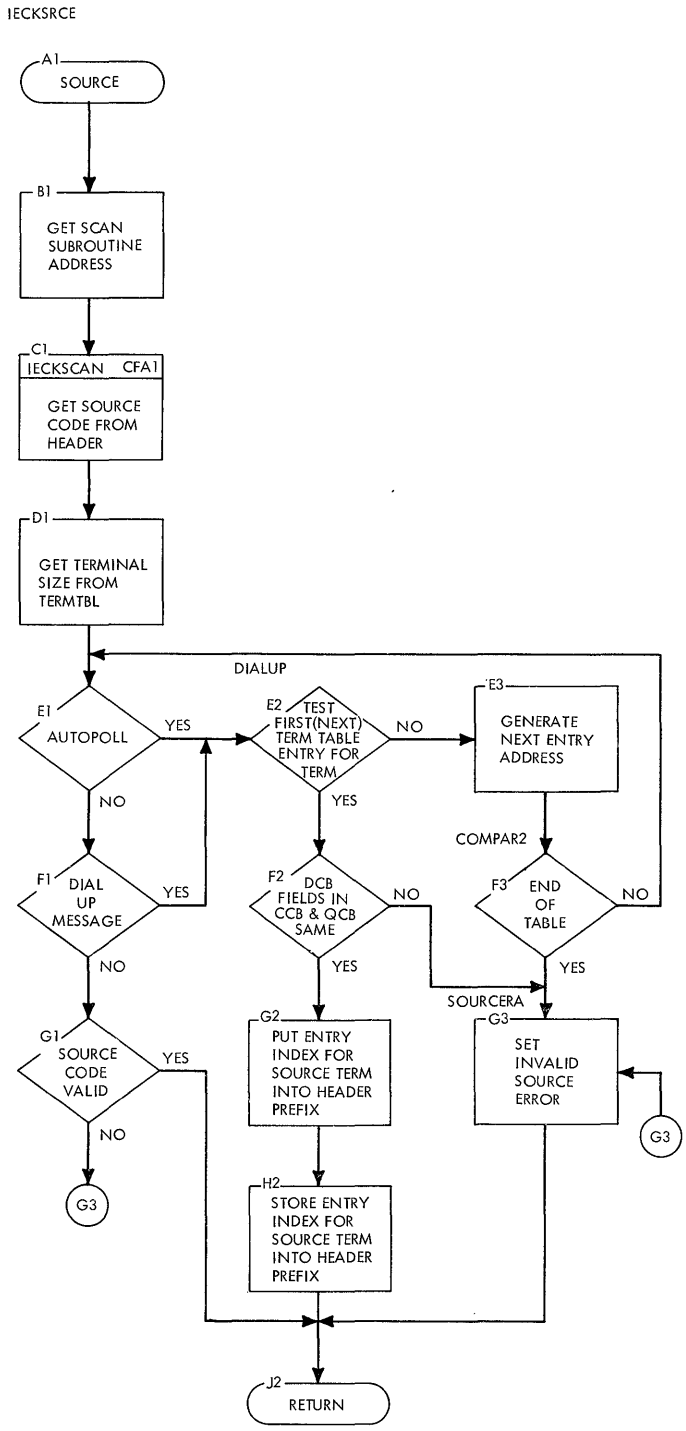


Chart CJ. Skip to Character Set - Skip on Count Routines

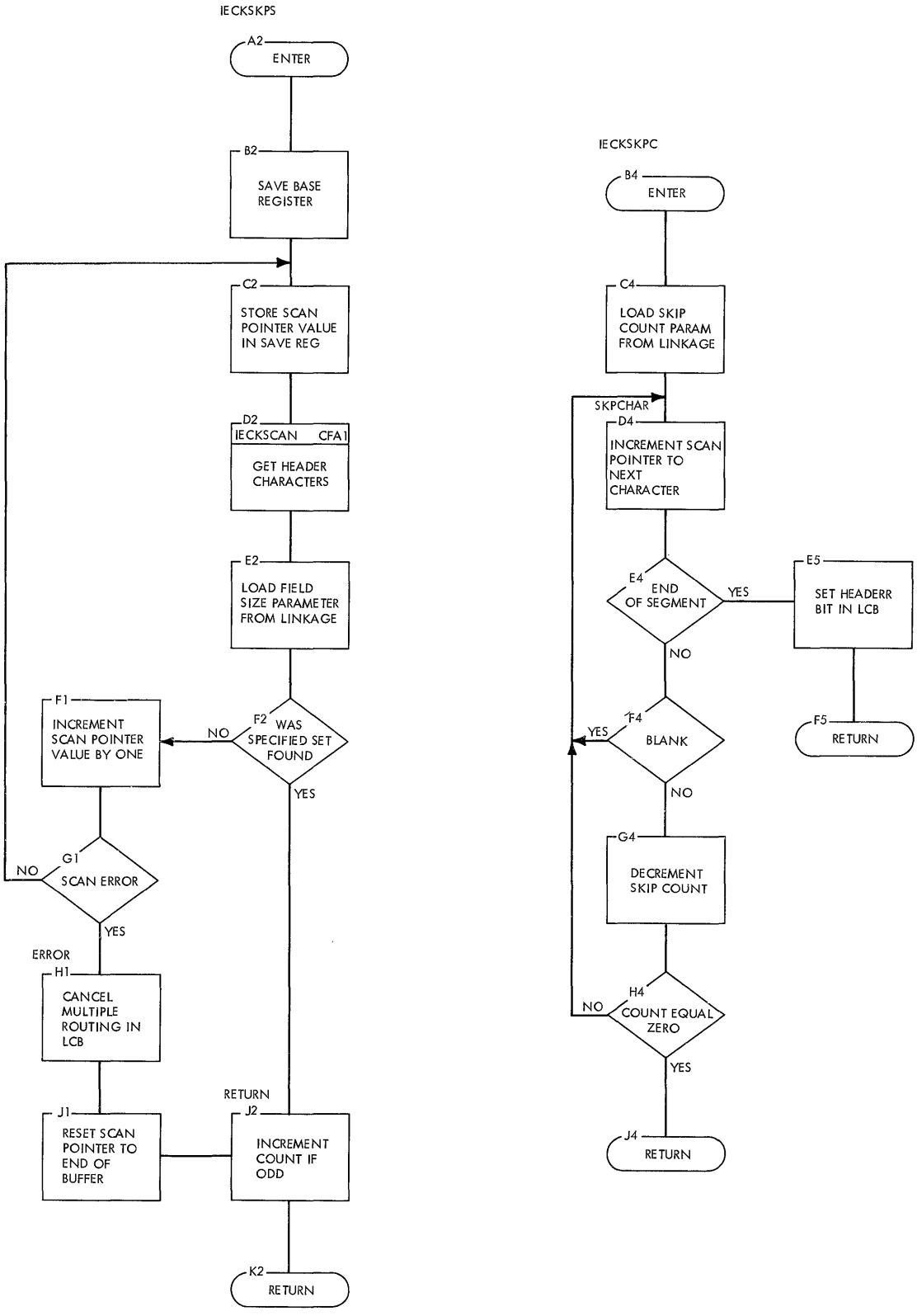


Chart CK. Time Stamp Routine

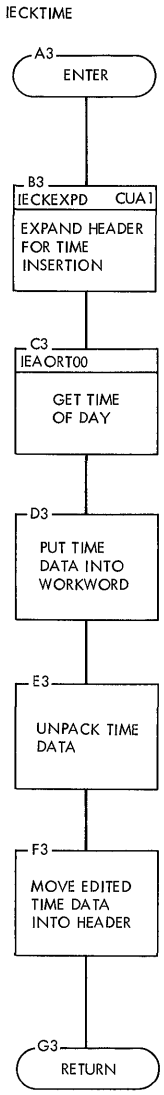


Chart CL. Cancel Message Routine

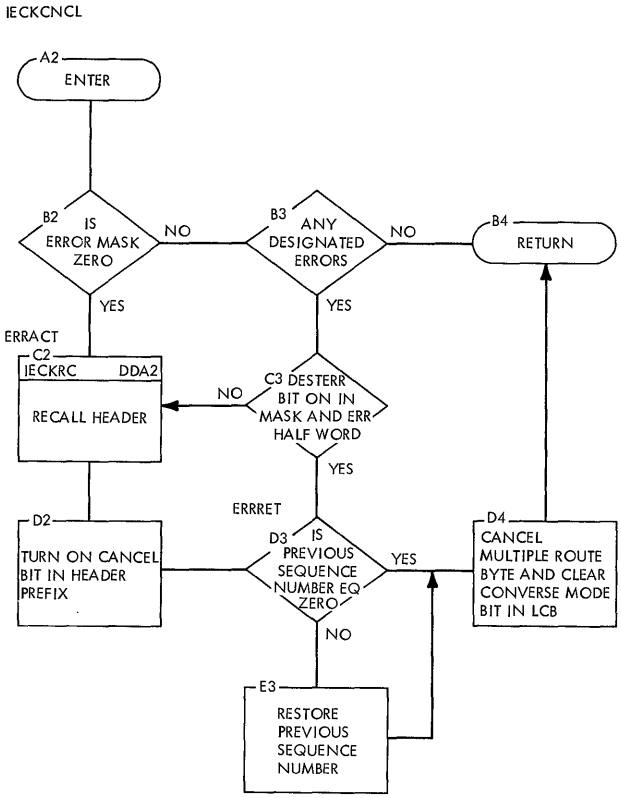


Chart CM. Sequence Out Routine

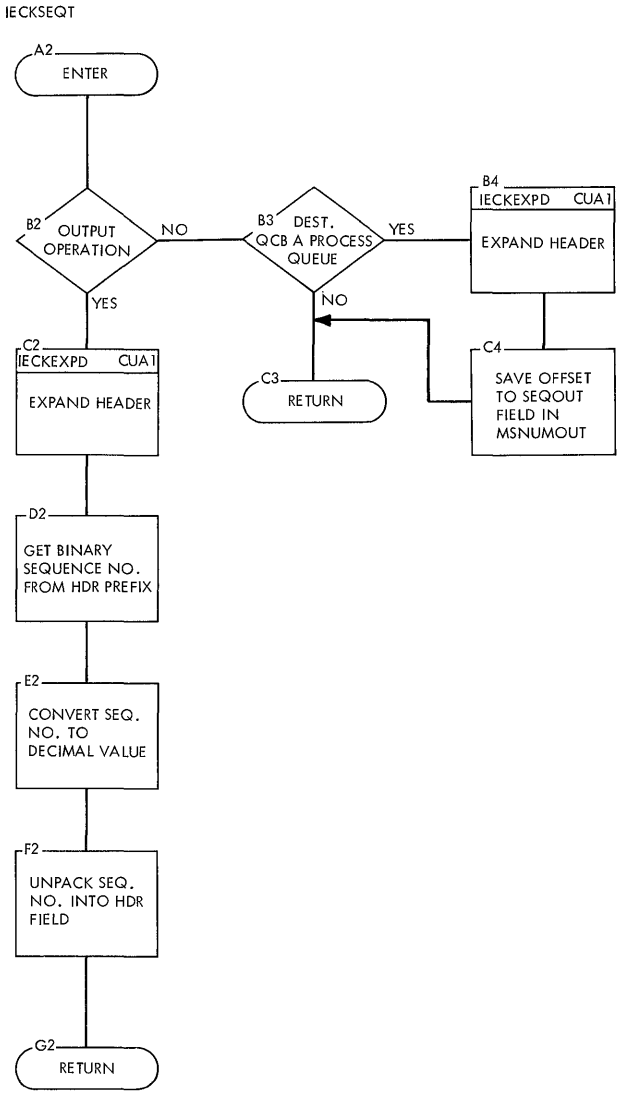


Chart CN. Route Routine

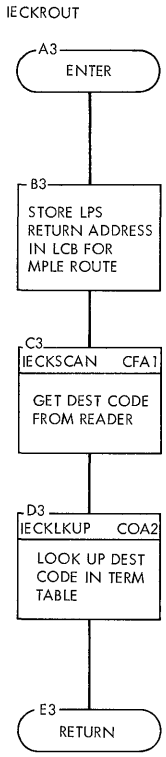


Chart CO. Lookup Routine

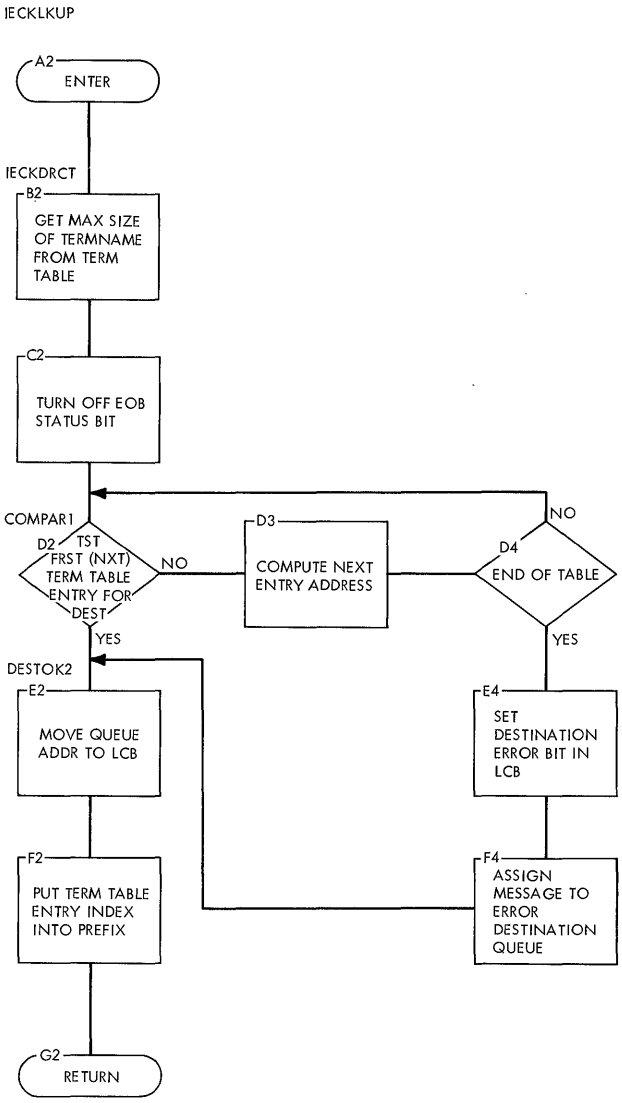


Chart CP. Translate Routine

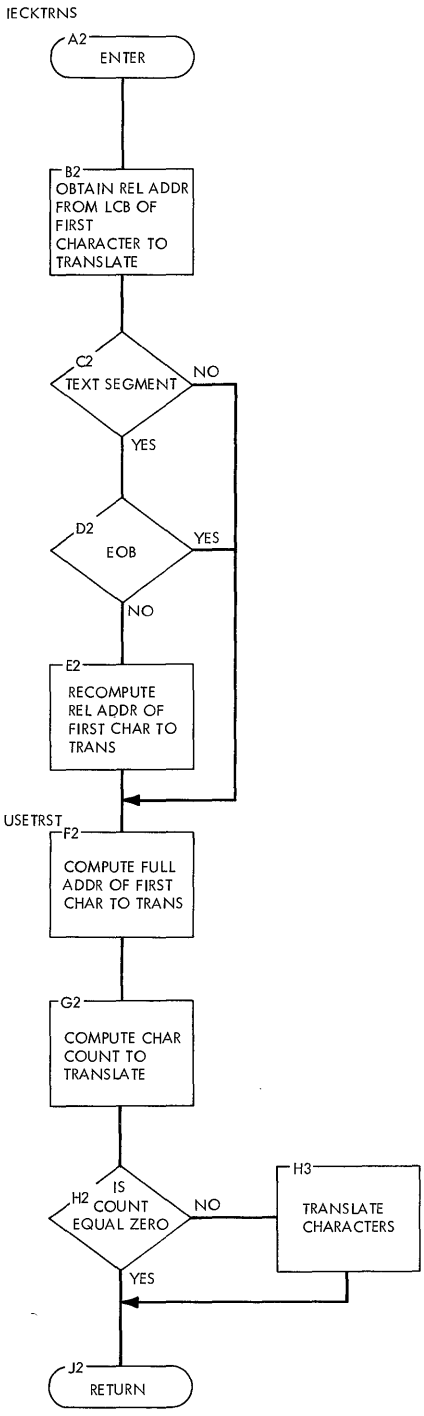


Chart CQ. Error Message Routine

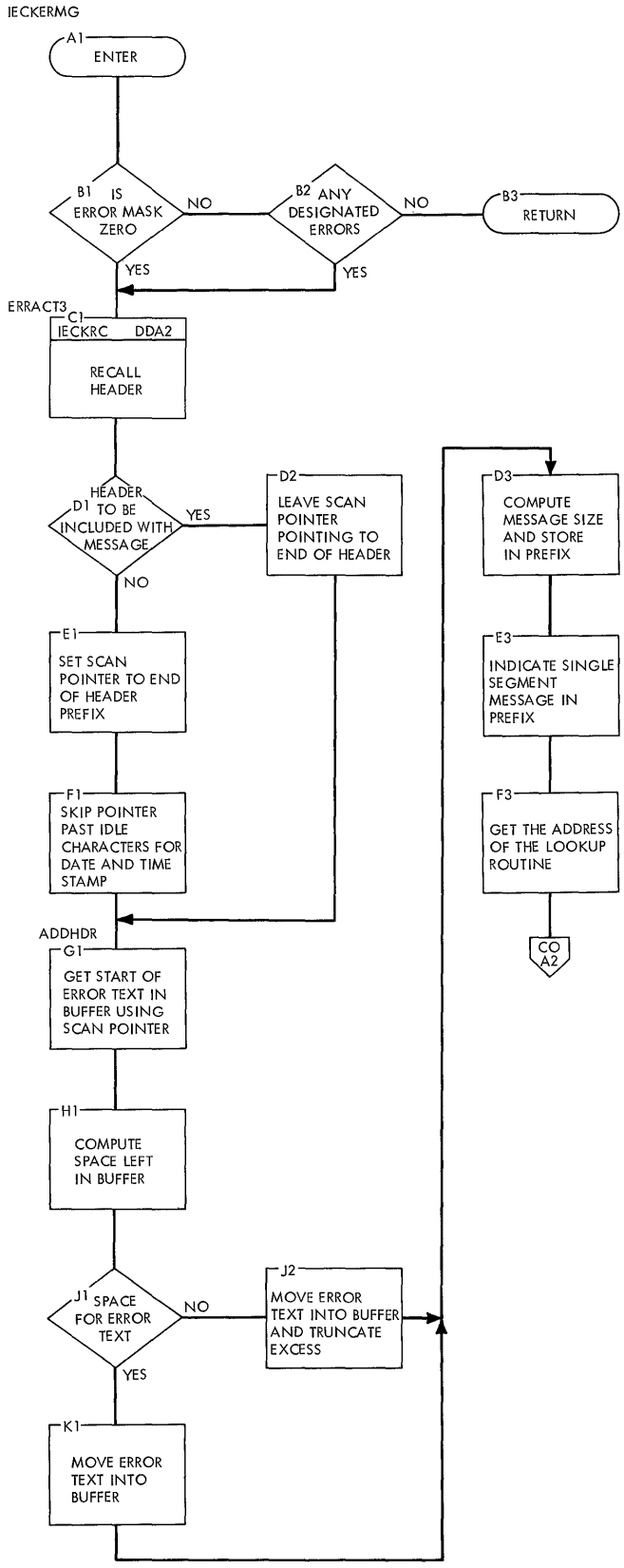


Chart CR. Polling Limit Routine

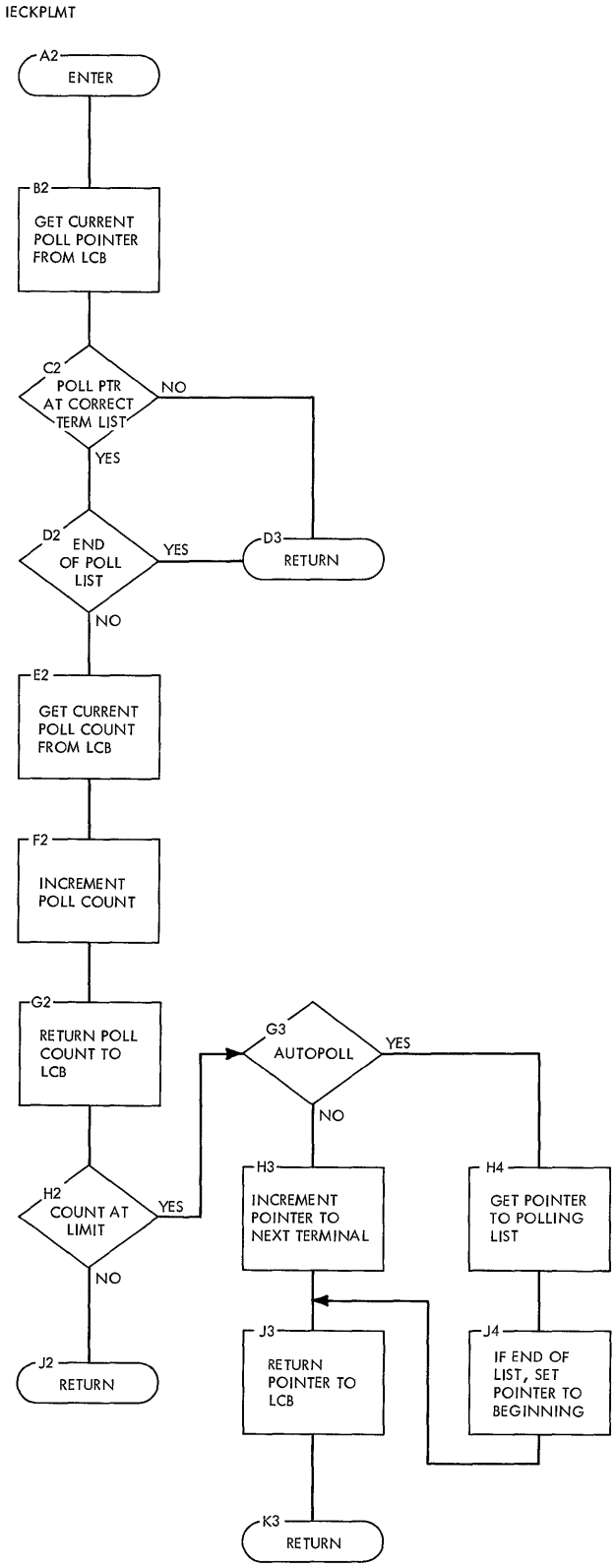


Chart CS. Reroute Routine

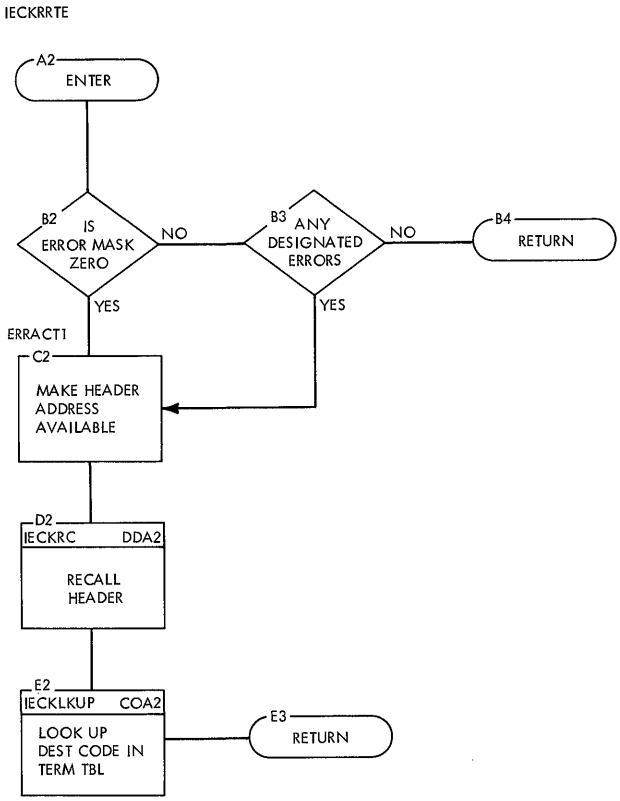


Chart CT. Intercept Routine

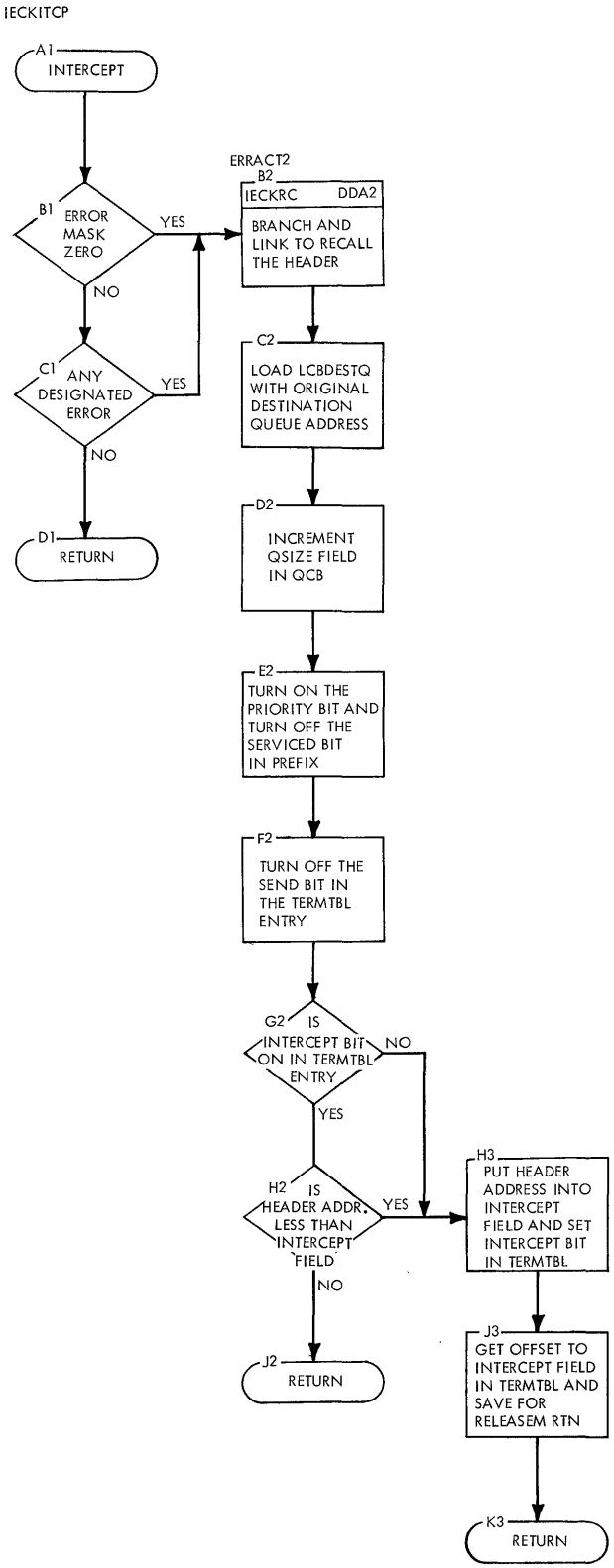


Chart CU. Expand Routine

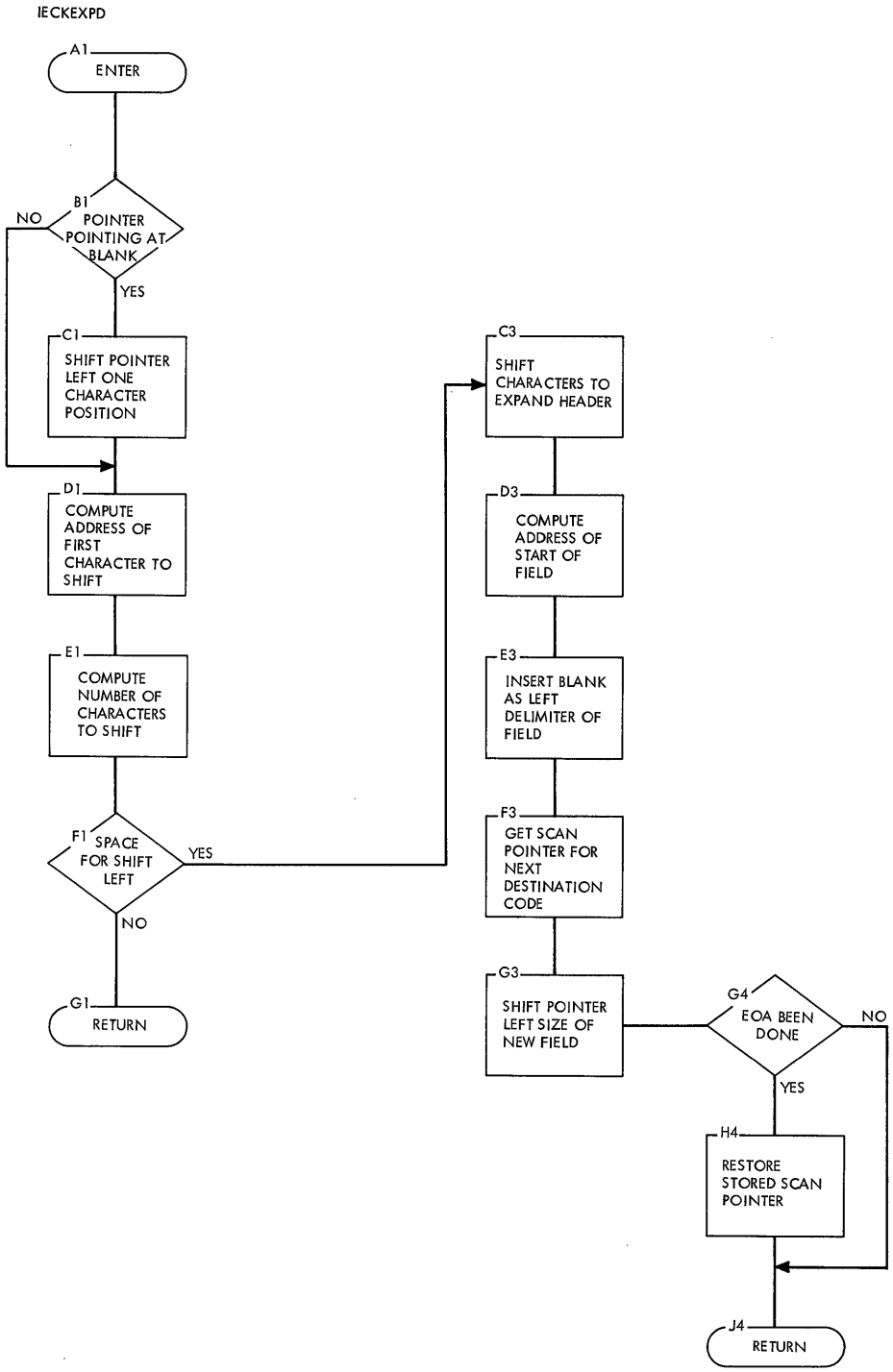


Chart CV. Sequence in Routine

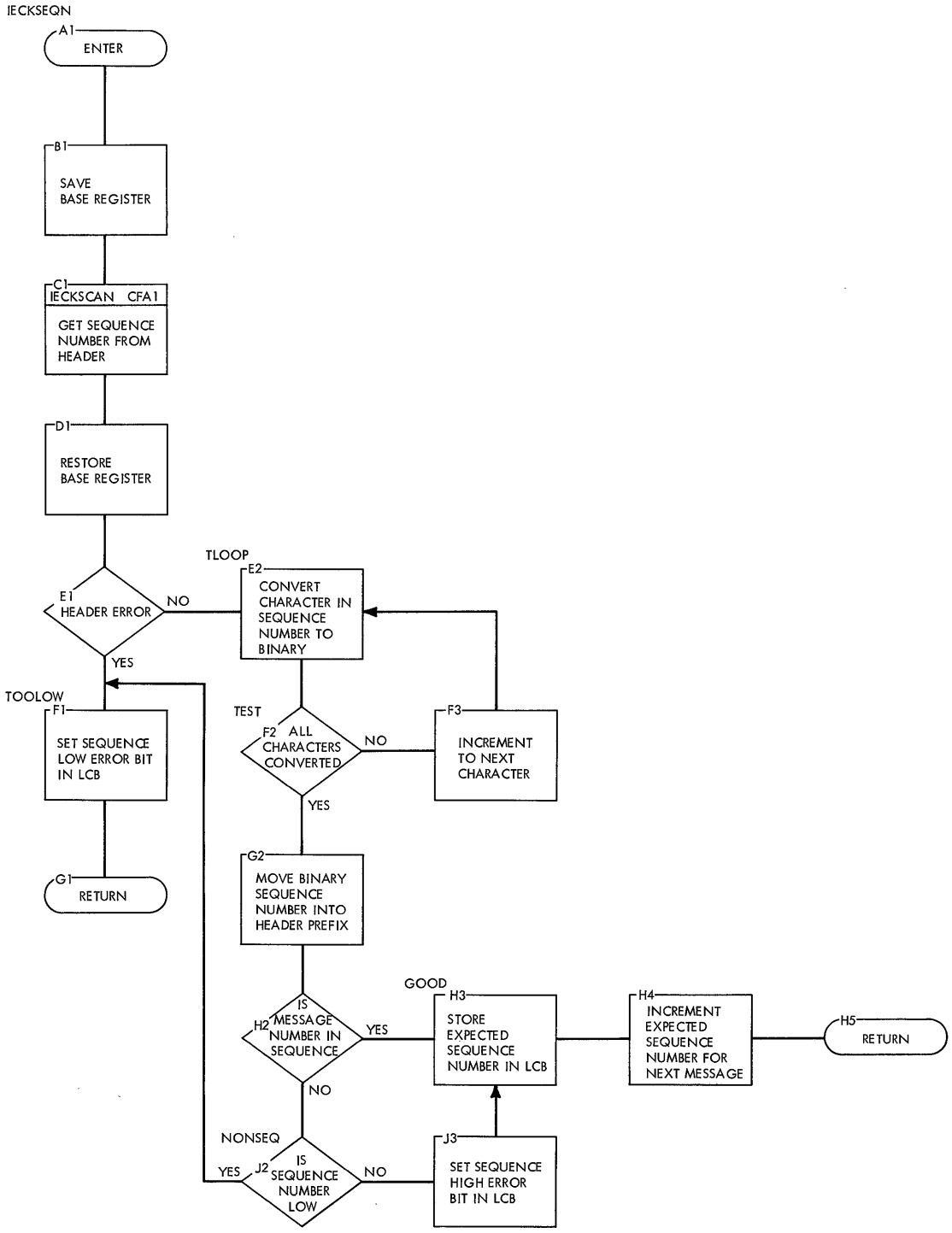


Chart CW. Mode, Initiate, and Priority Routines

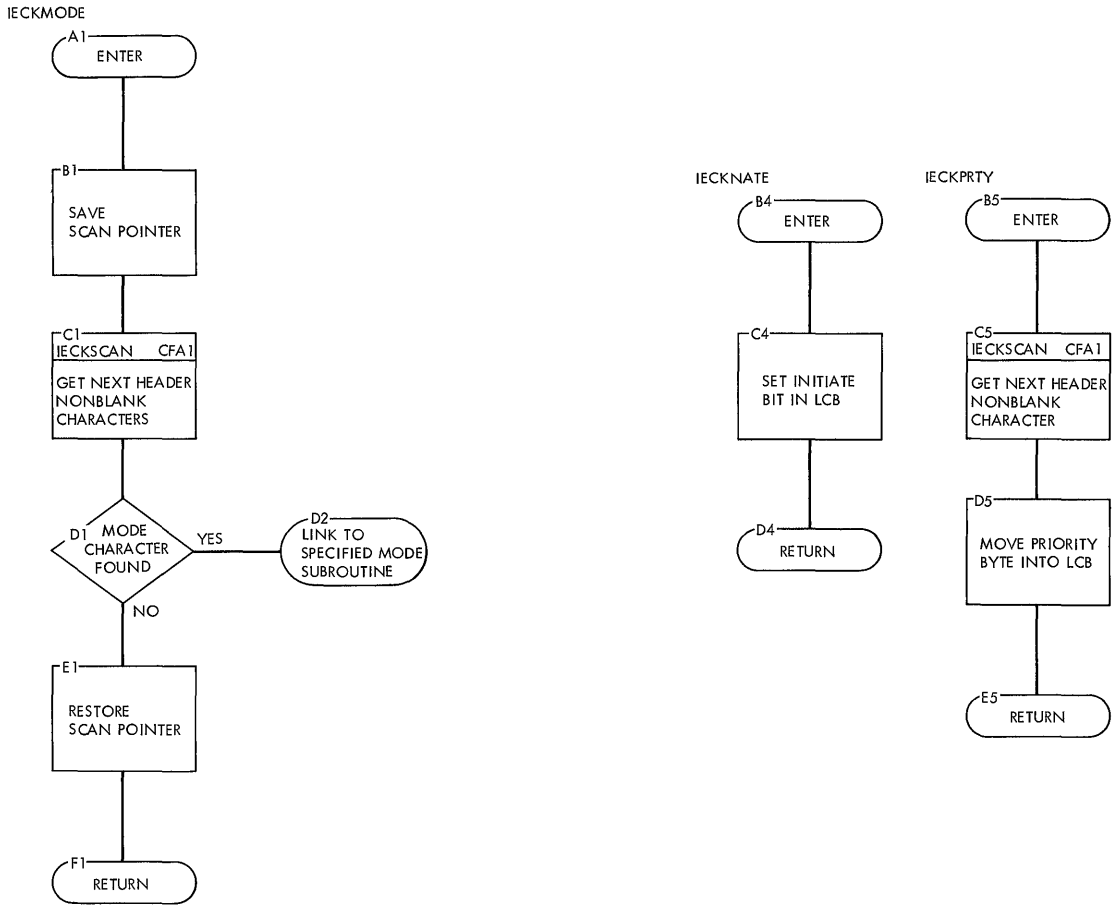


Chart CX. Mode Conversational Routine

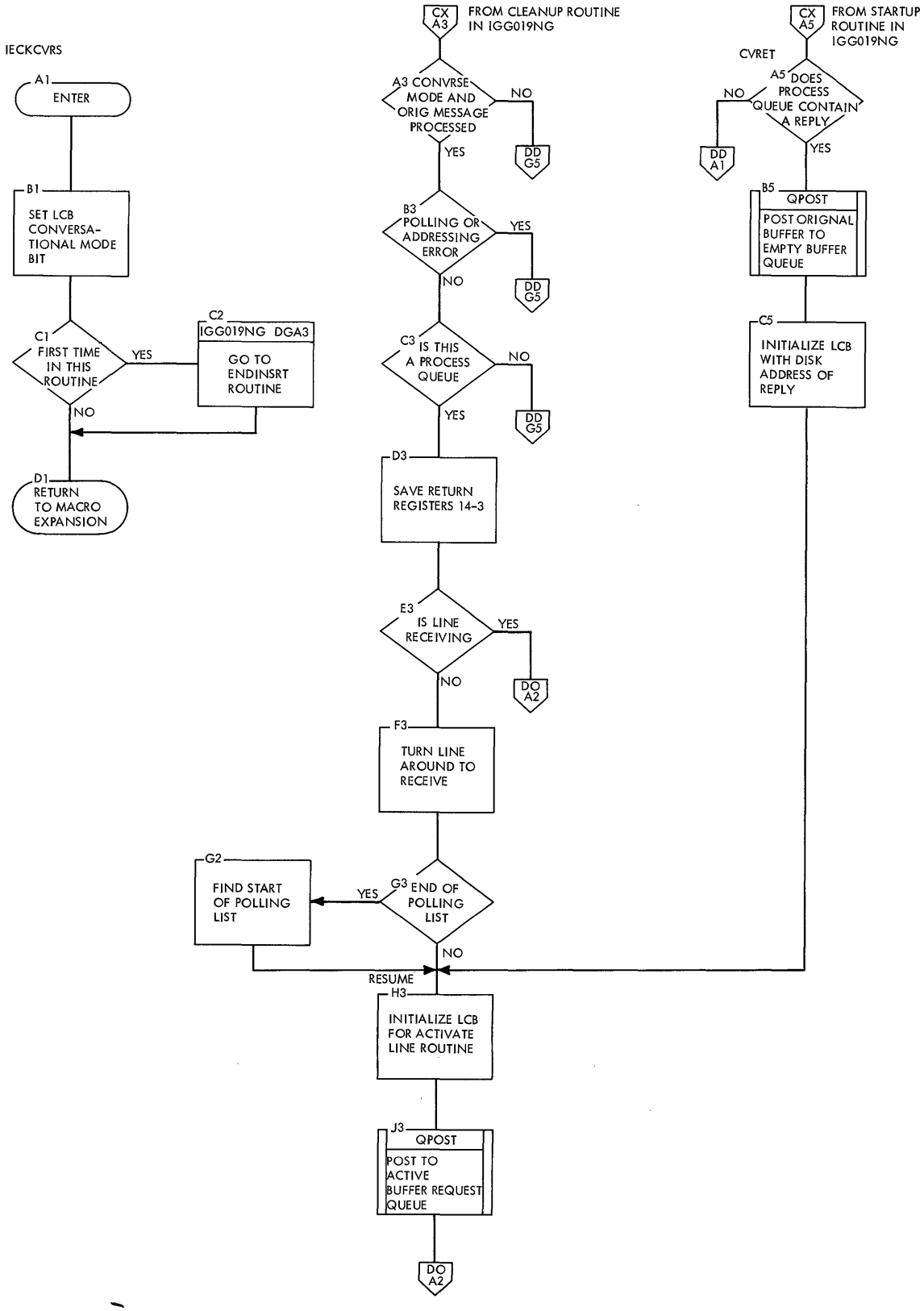


Chart CY. End of Block Routine

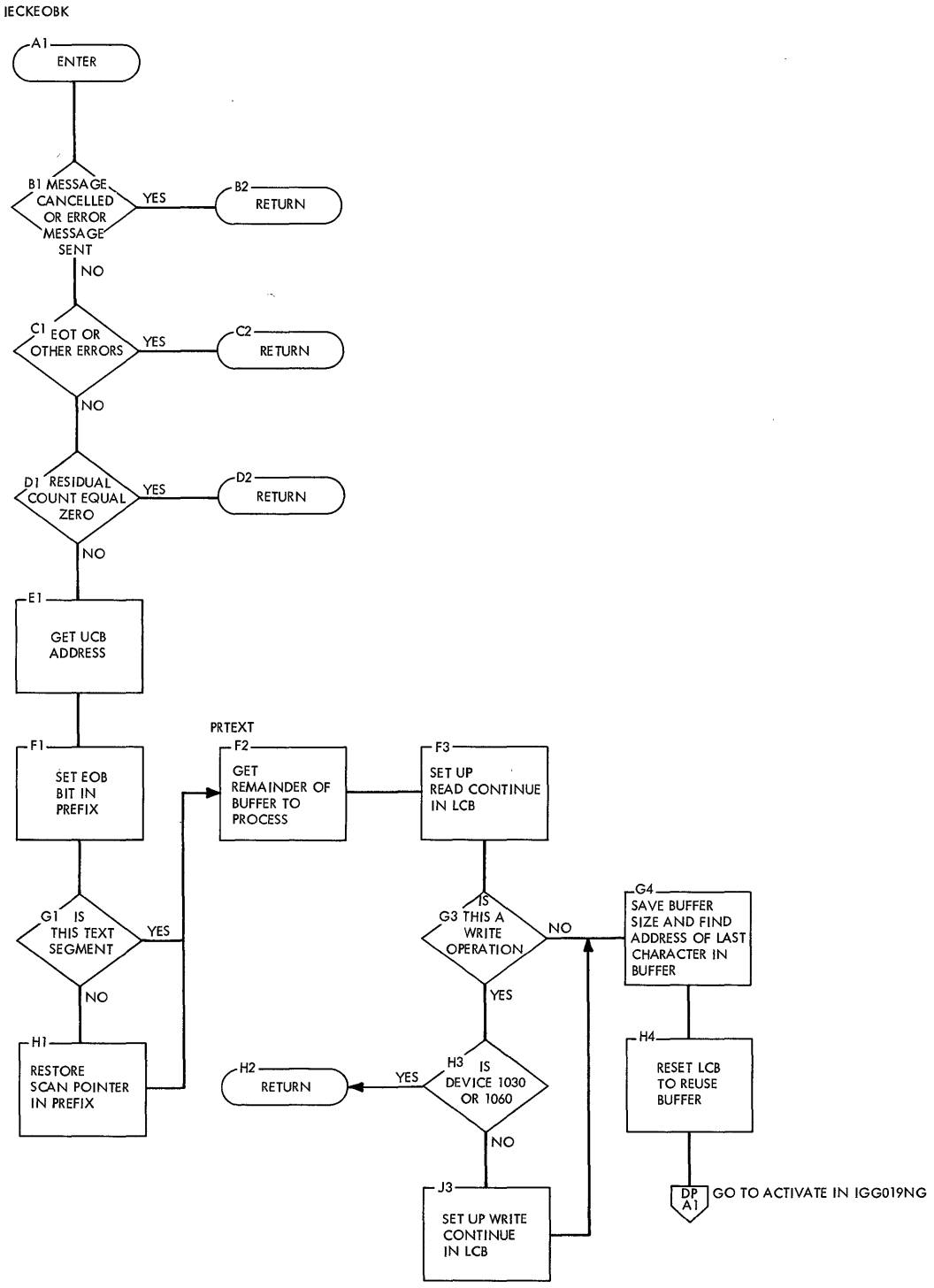


Chart CZ. End of Block and Line Correction Routine

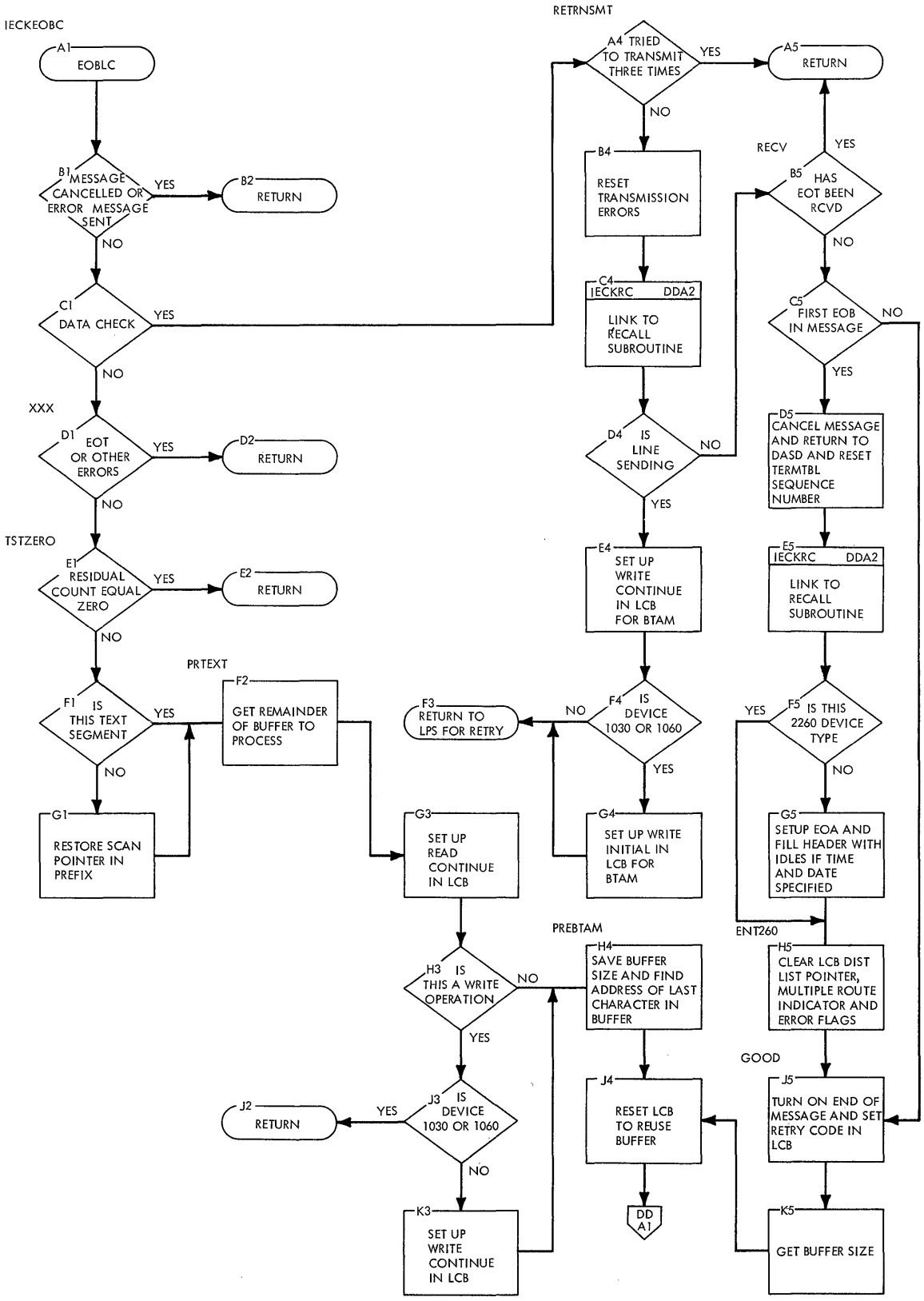


Chart D0. Disk End Appendage Routine

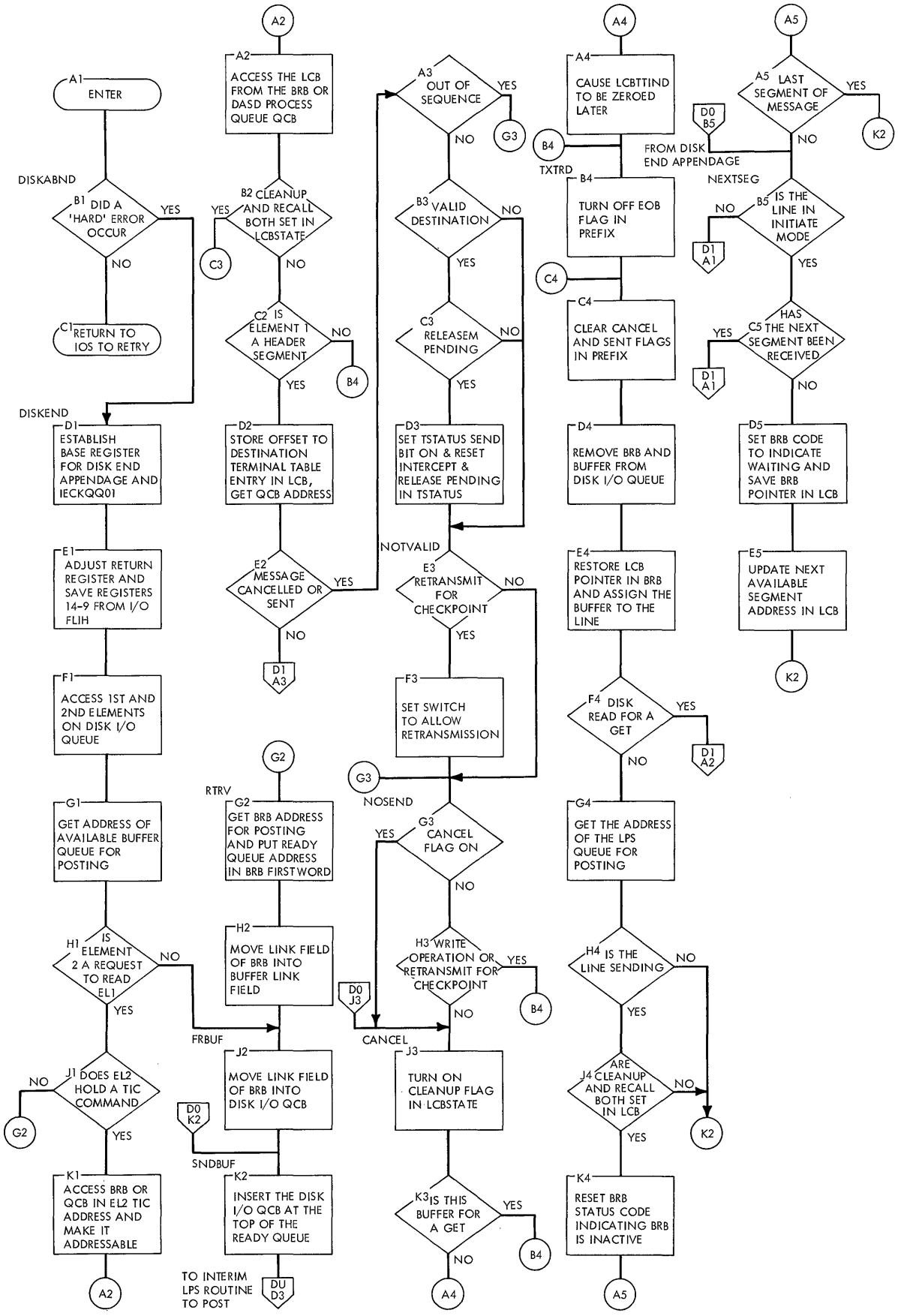


Chart D1. Disk End Appendage Routine (Continued)

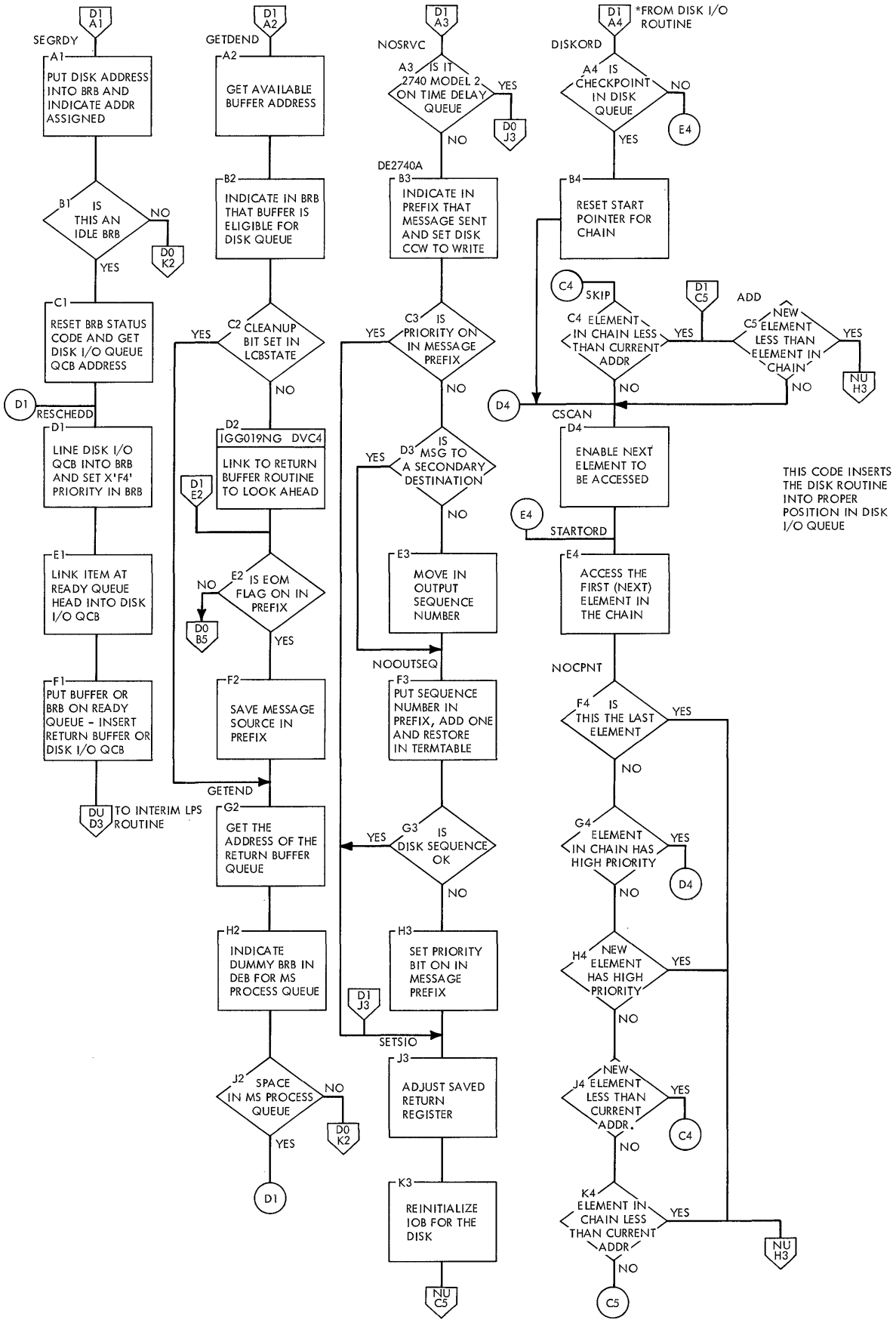


Chart D2. Disk I/O Routine

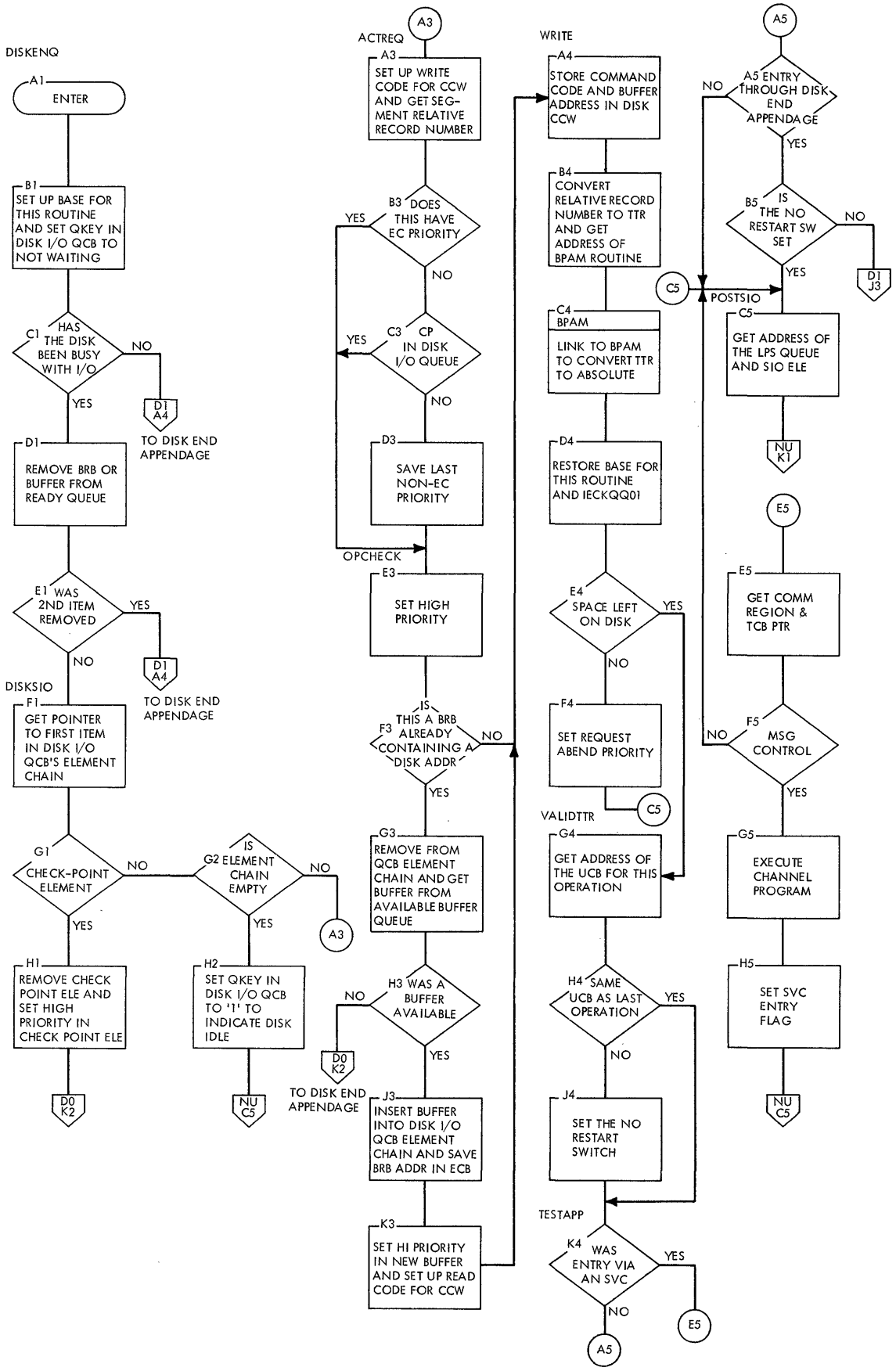


Chart DA. Put Message Routine

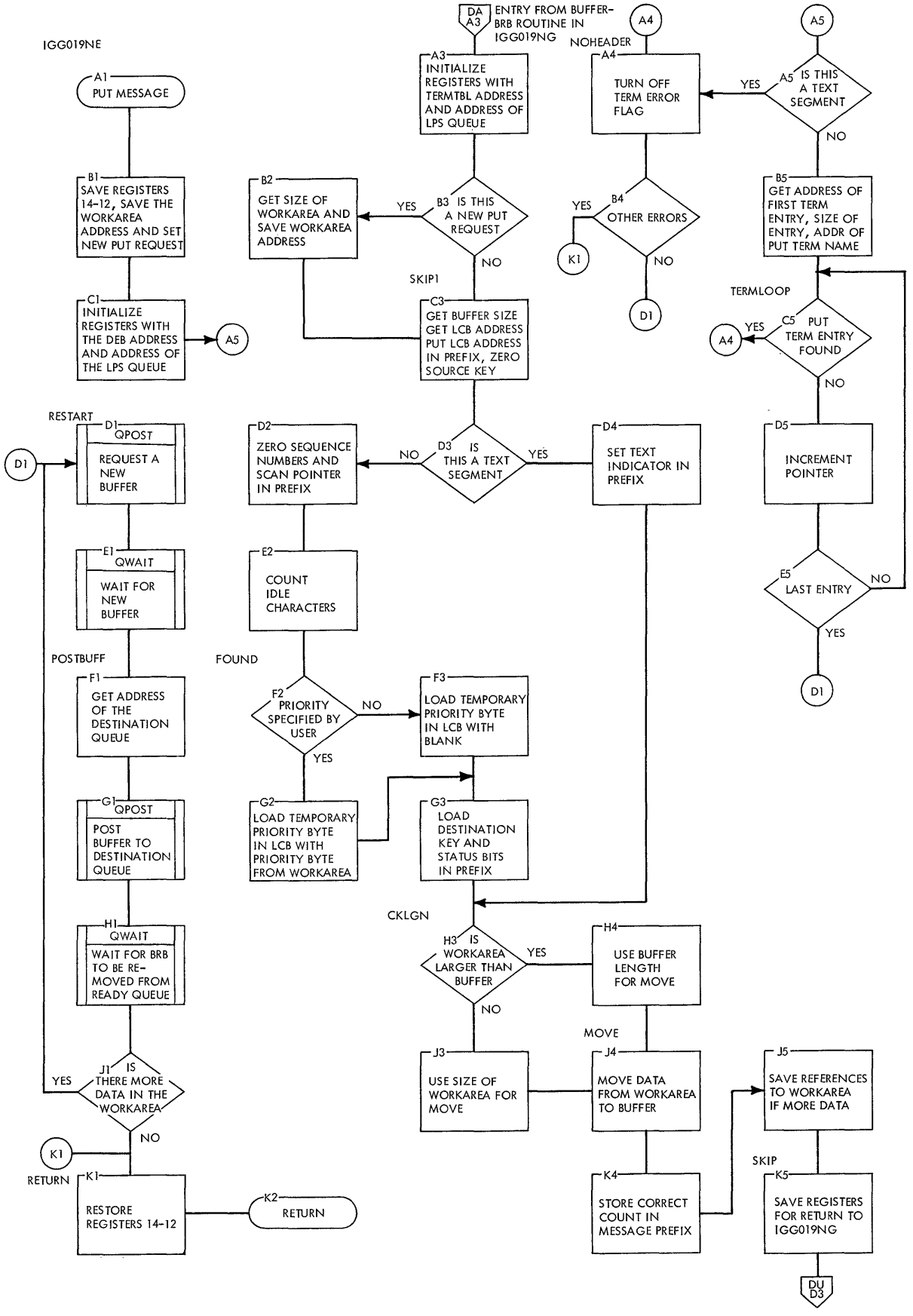


Chart DB. Distribution List Routine

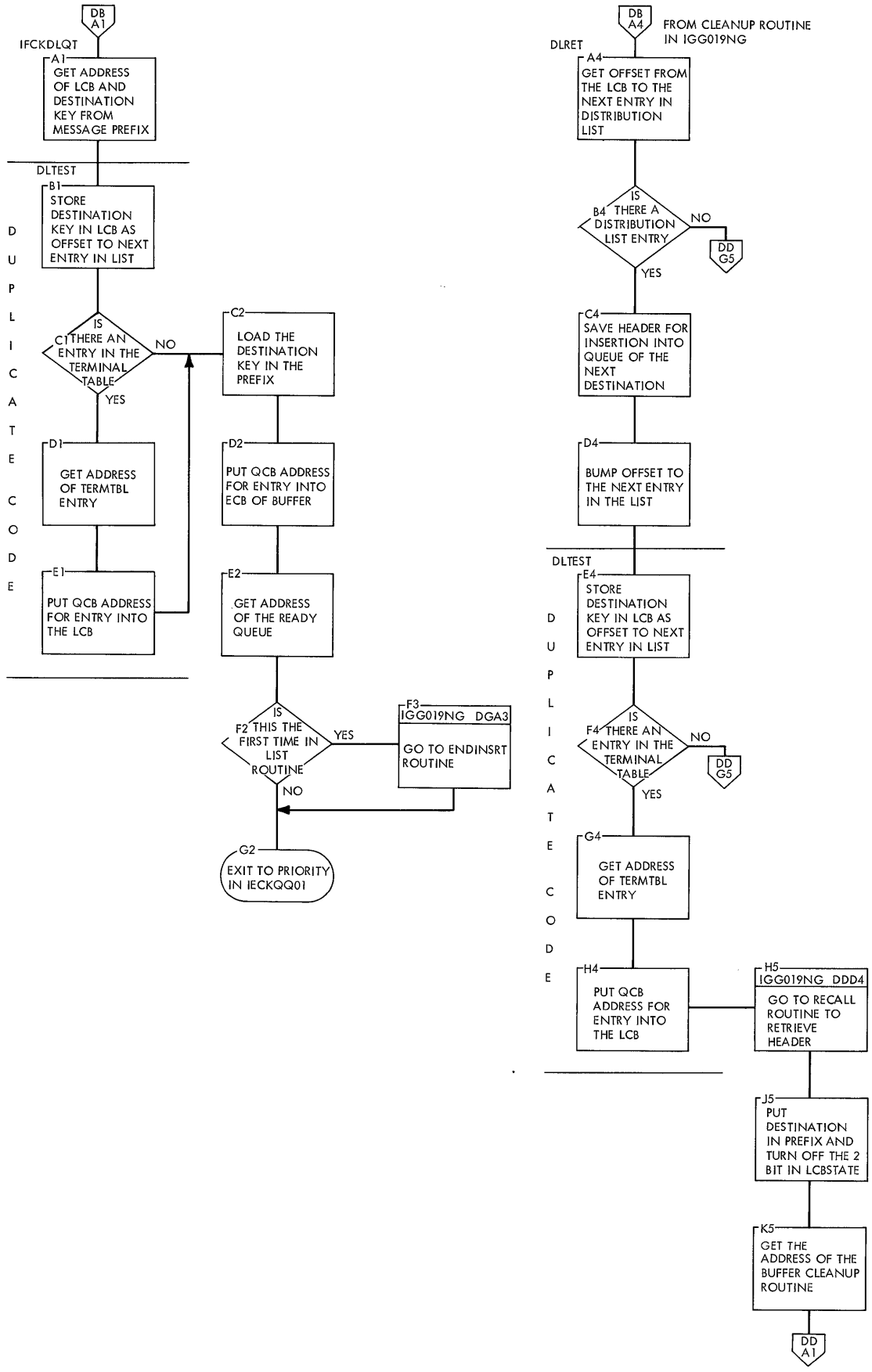


Chart DC. End of Address Routine

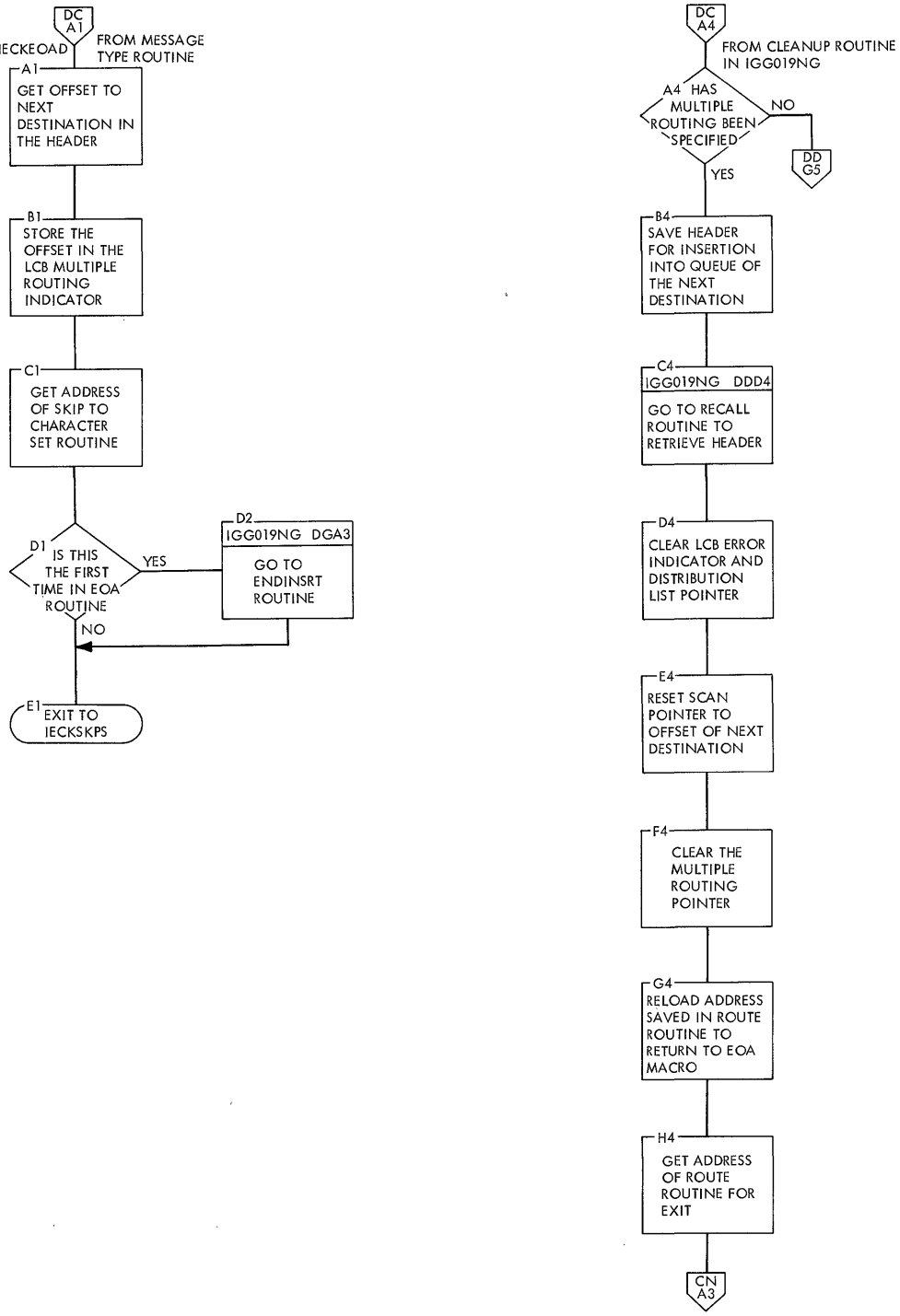


Chart DD. Buffer Cleanup and Recall Routine

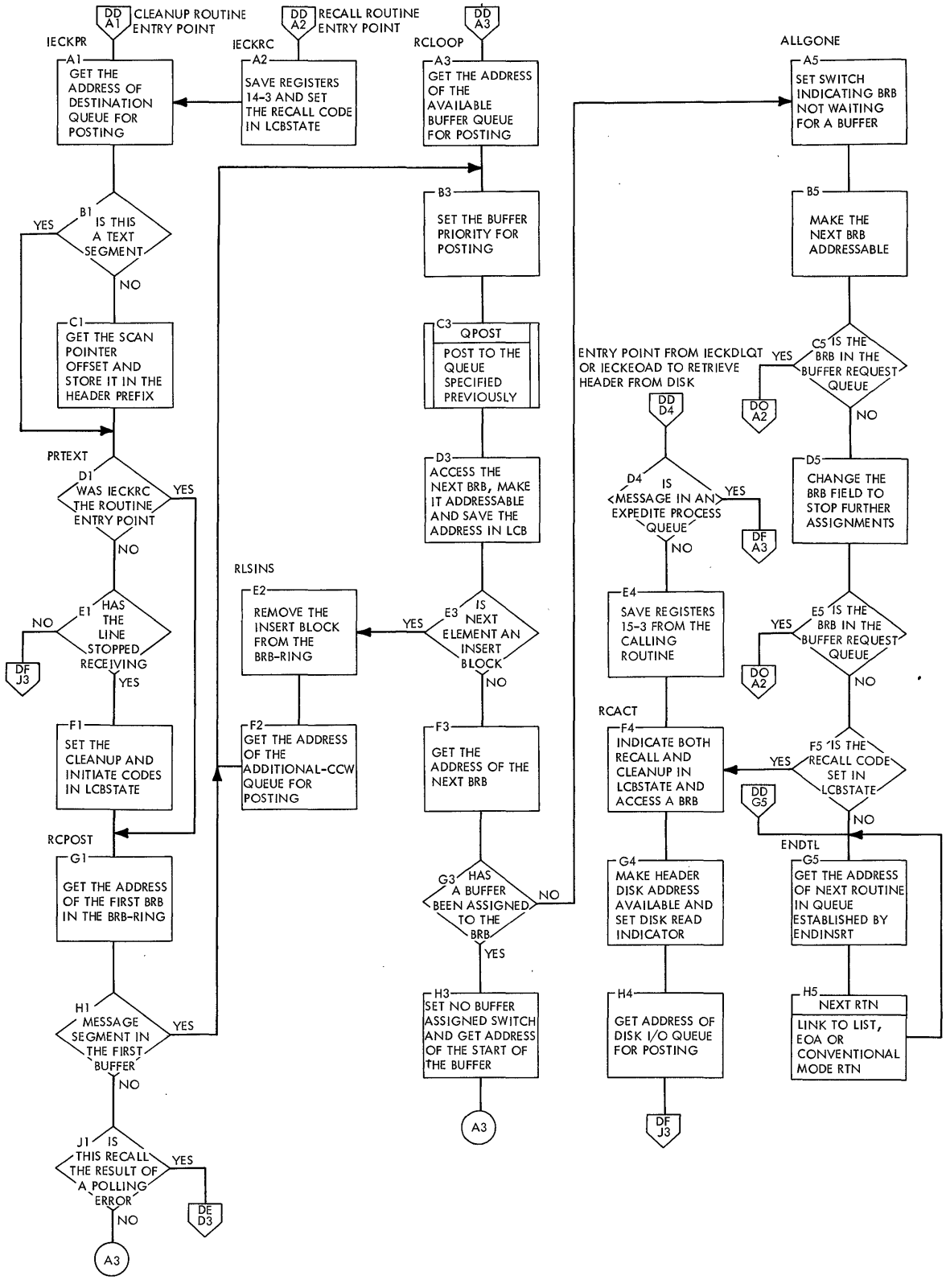


Chart DE. Buffer Cleanup and Recall Routine (Continued)

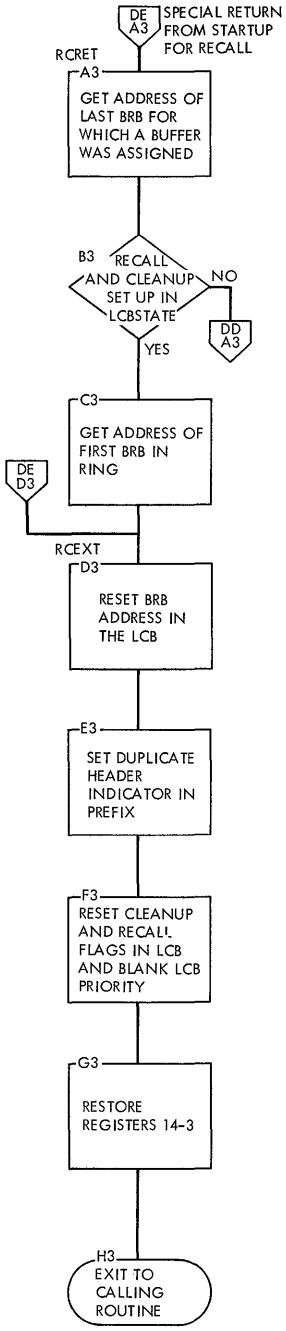


Chart DF. Free BRB Routine

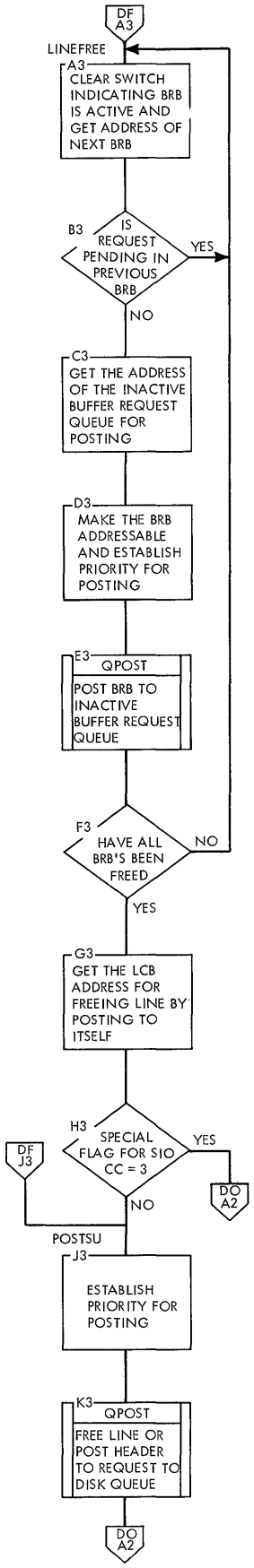


Chart DG. End Insert Routine

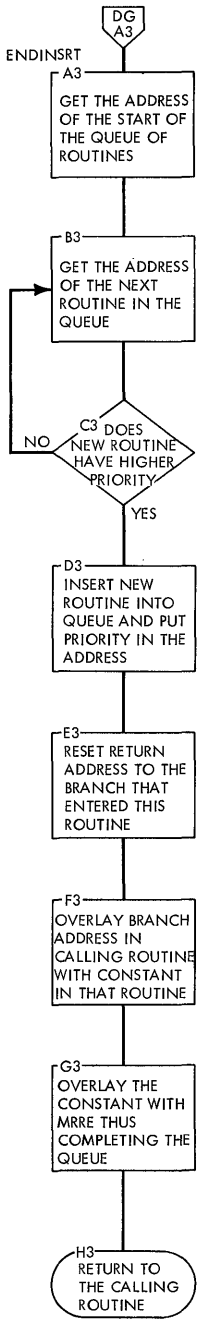


Chart DH. Receive Scheduler Routine

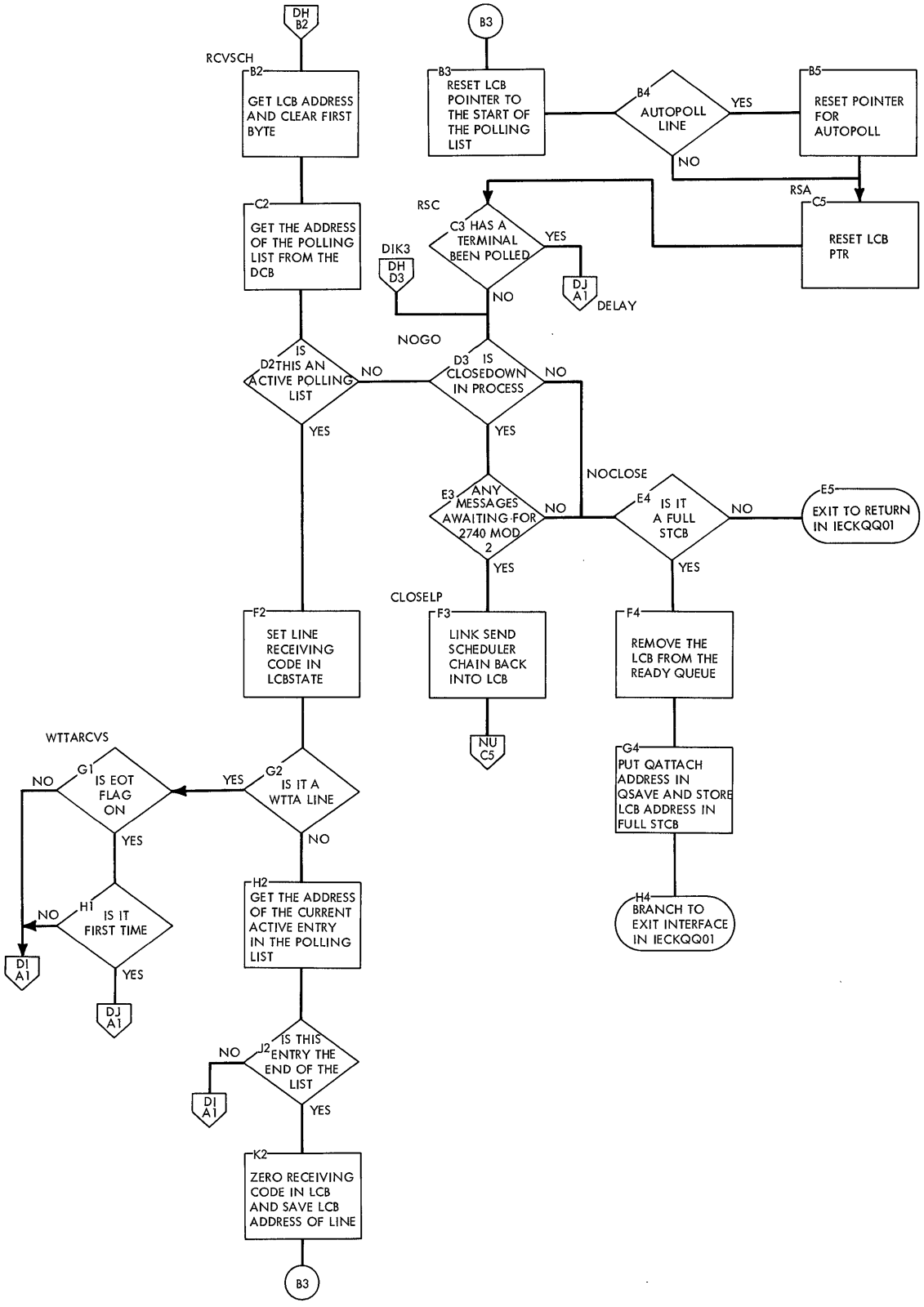
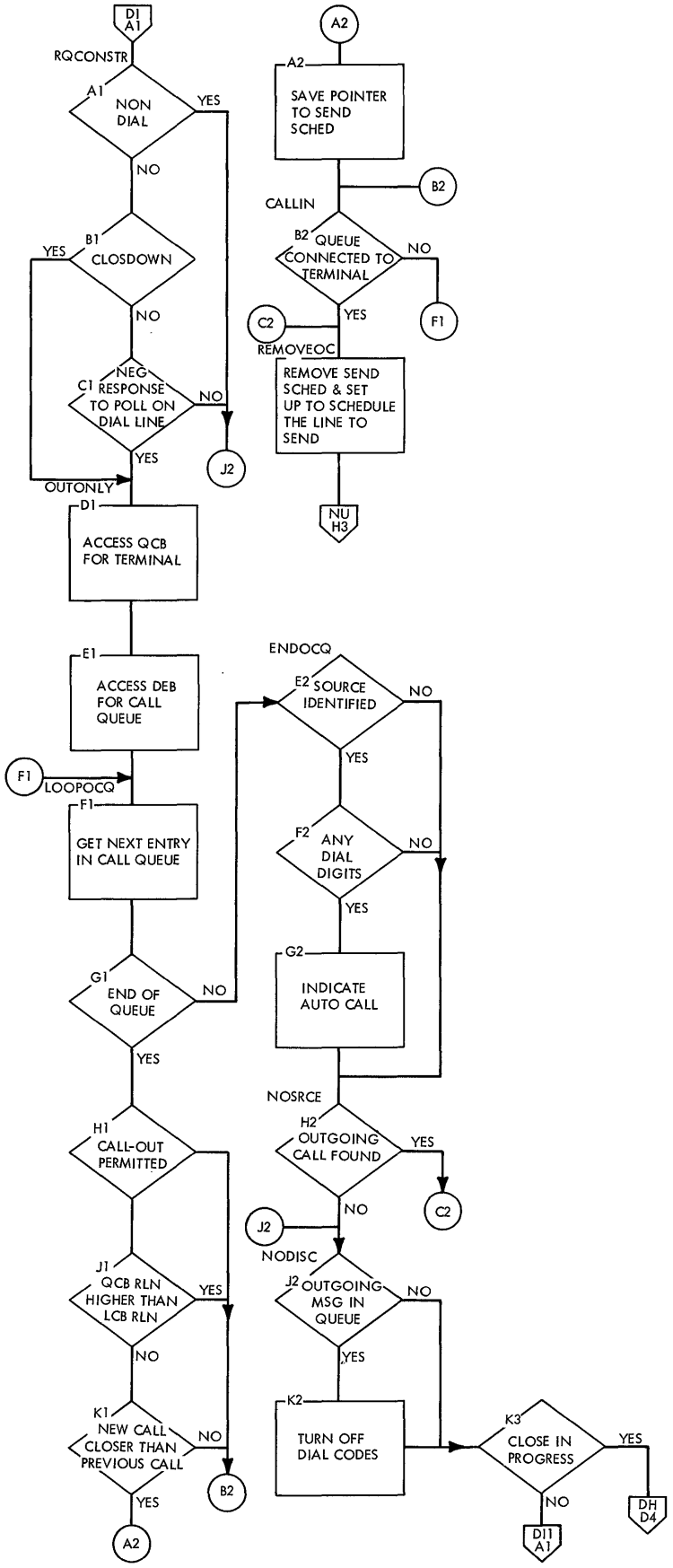


Chart DI. BRB Ring Routine



● Chart DI1. BRB Ring Routine (Continued)

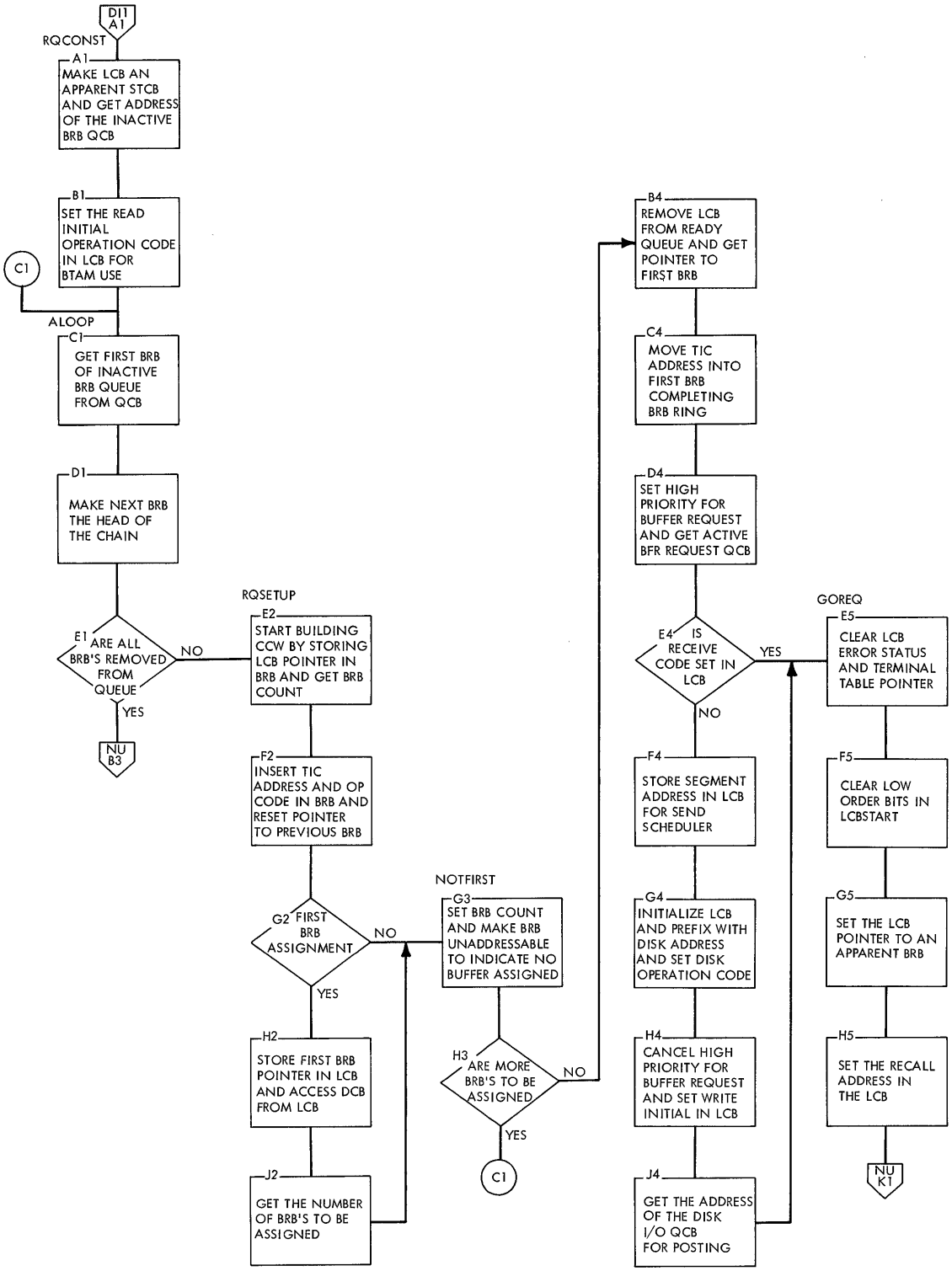


Chart DJ. End of Poll Time Delay Routine

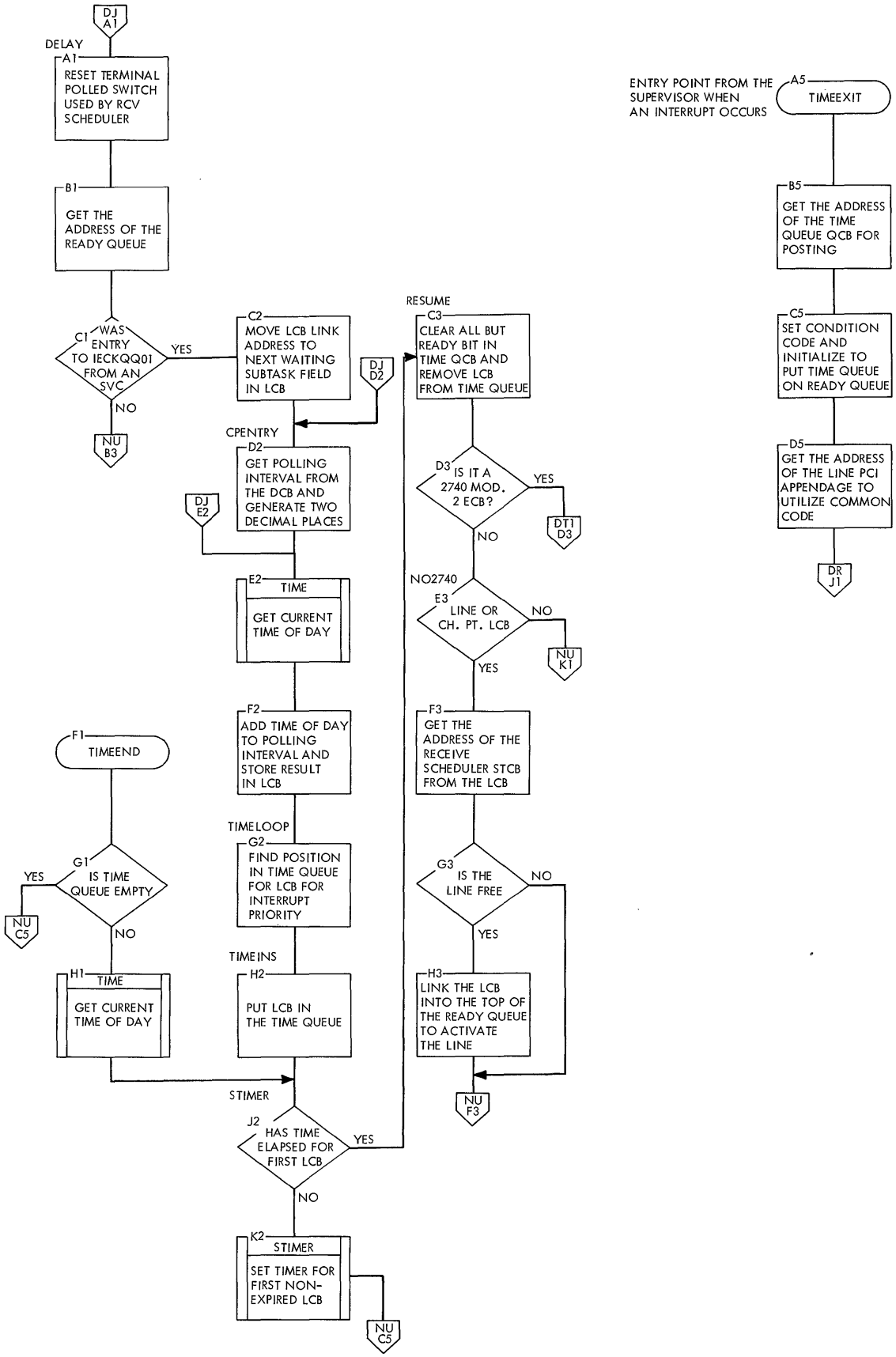


Chart DK. Send Scheduler Routine

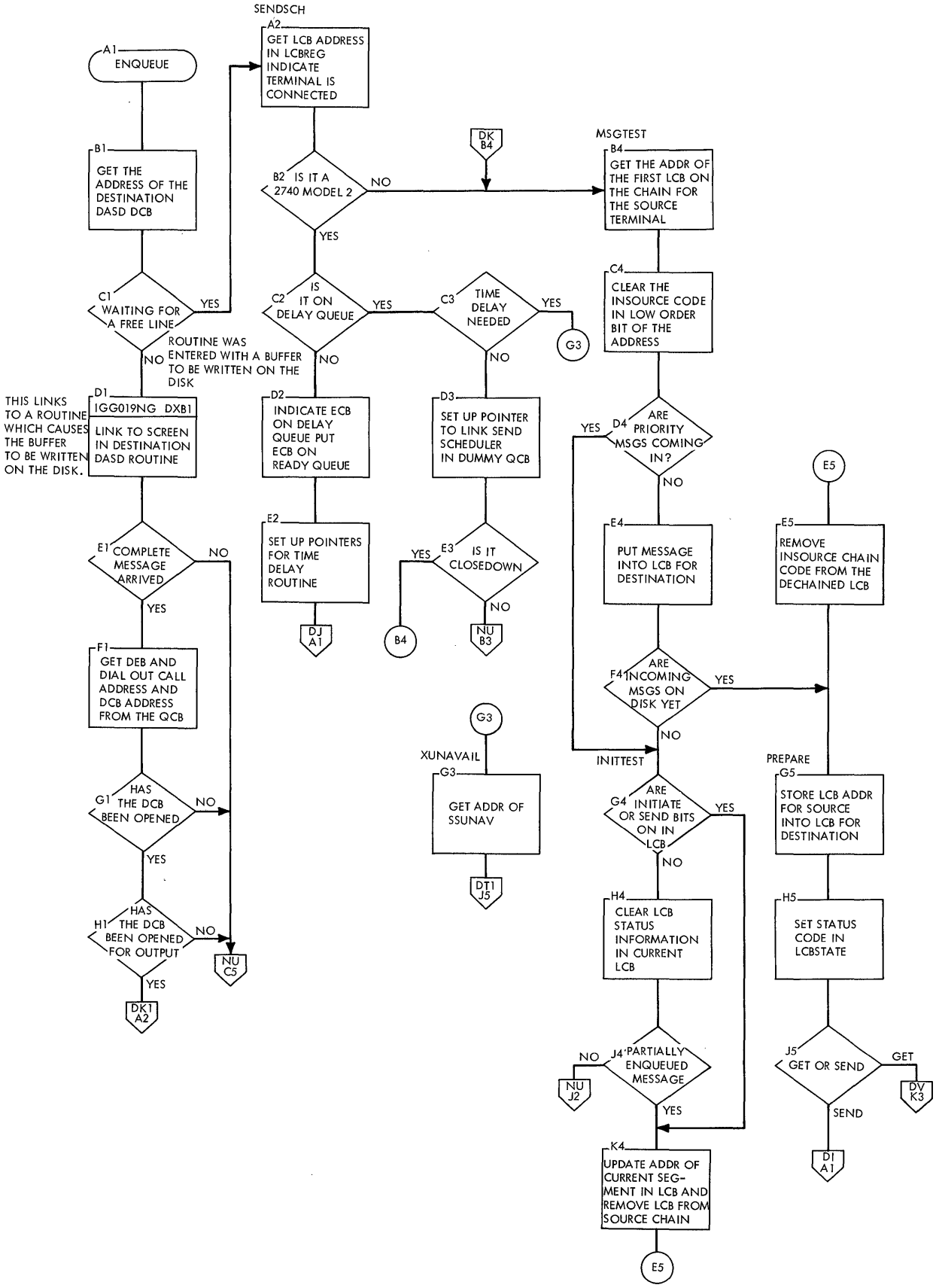


Chart DK1. Send Scheduler Routine (Continued)

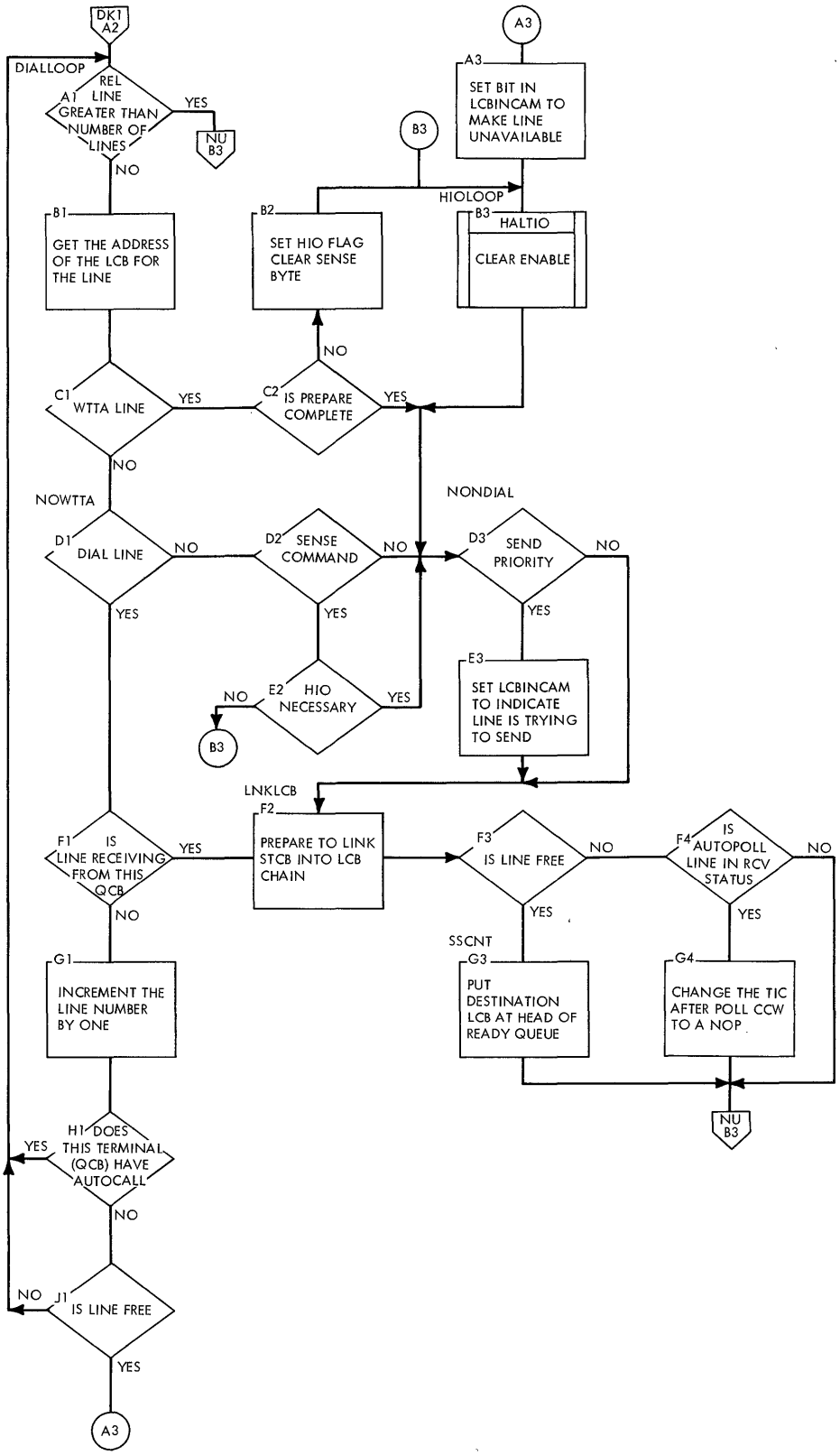


Chart DL. Active Buffer Request Routine

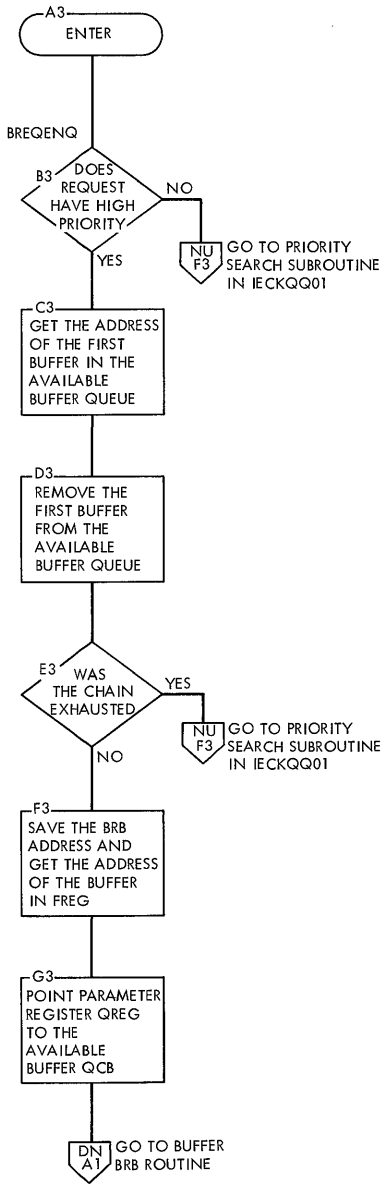


Chart DM. Available Buffer Routine

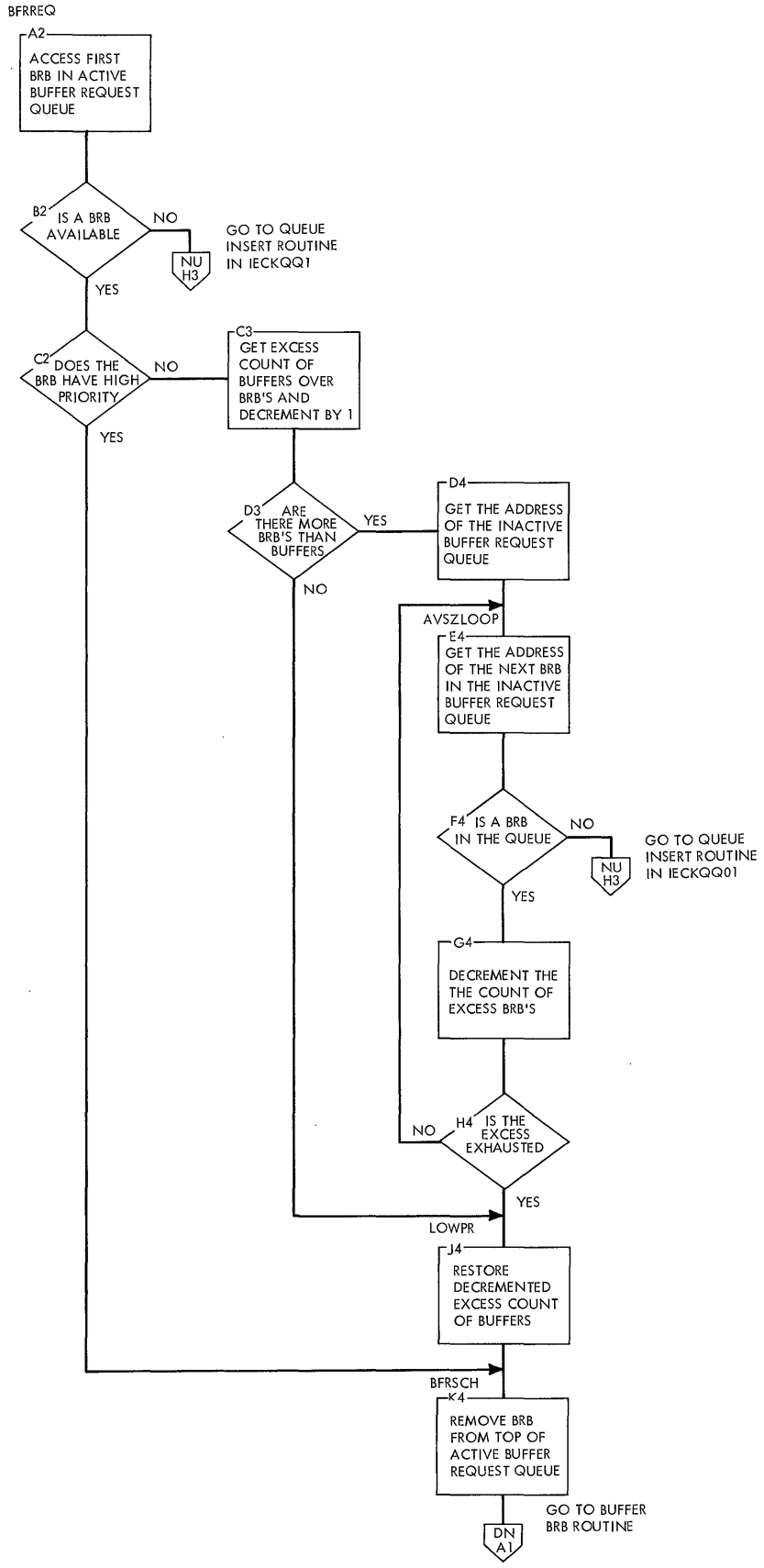


Chart DN. Buffer BRB Routine

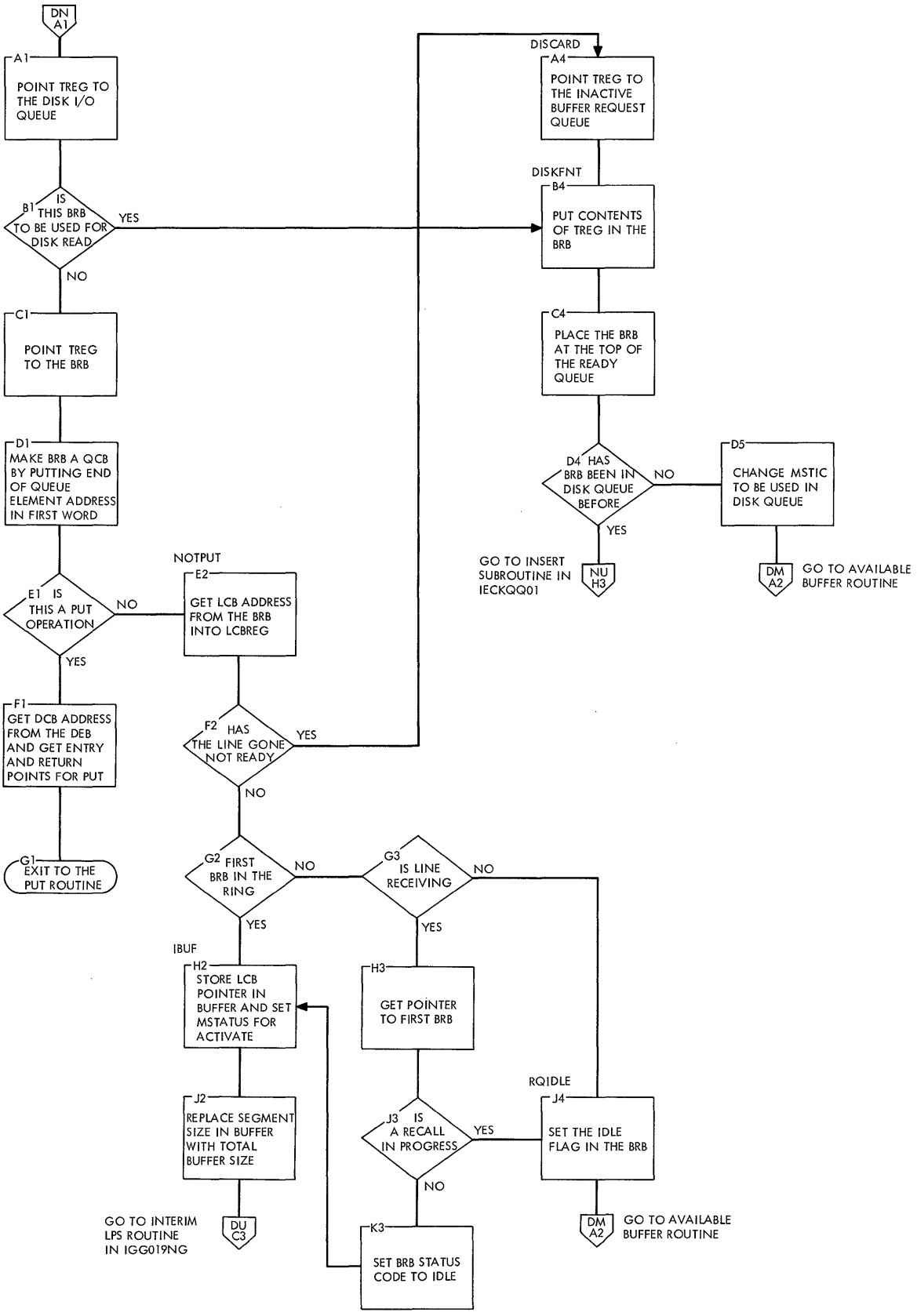


Chart DO. LPS Control Routine

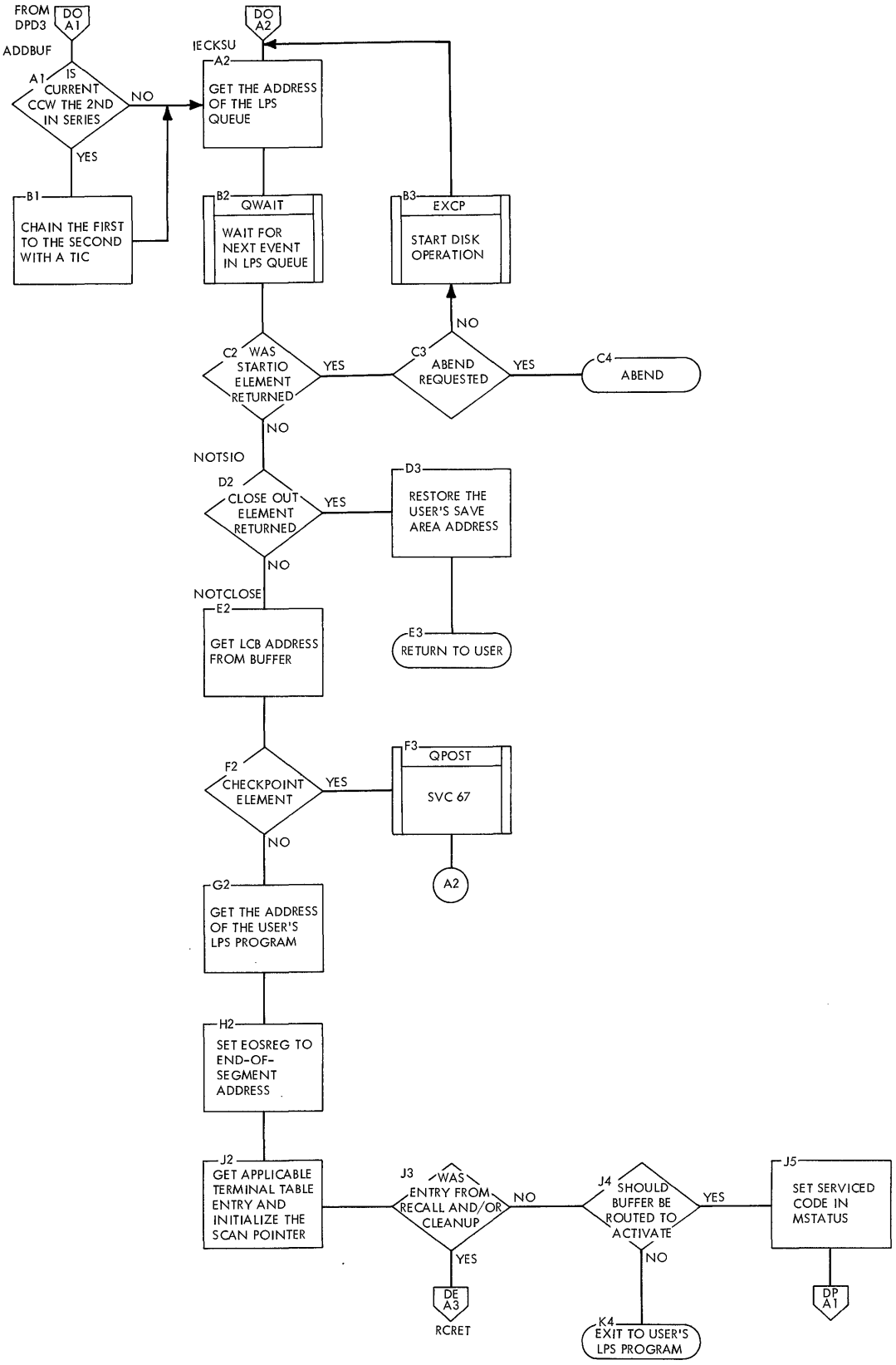


Chart DP. Activate Routine

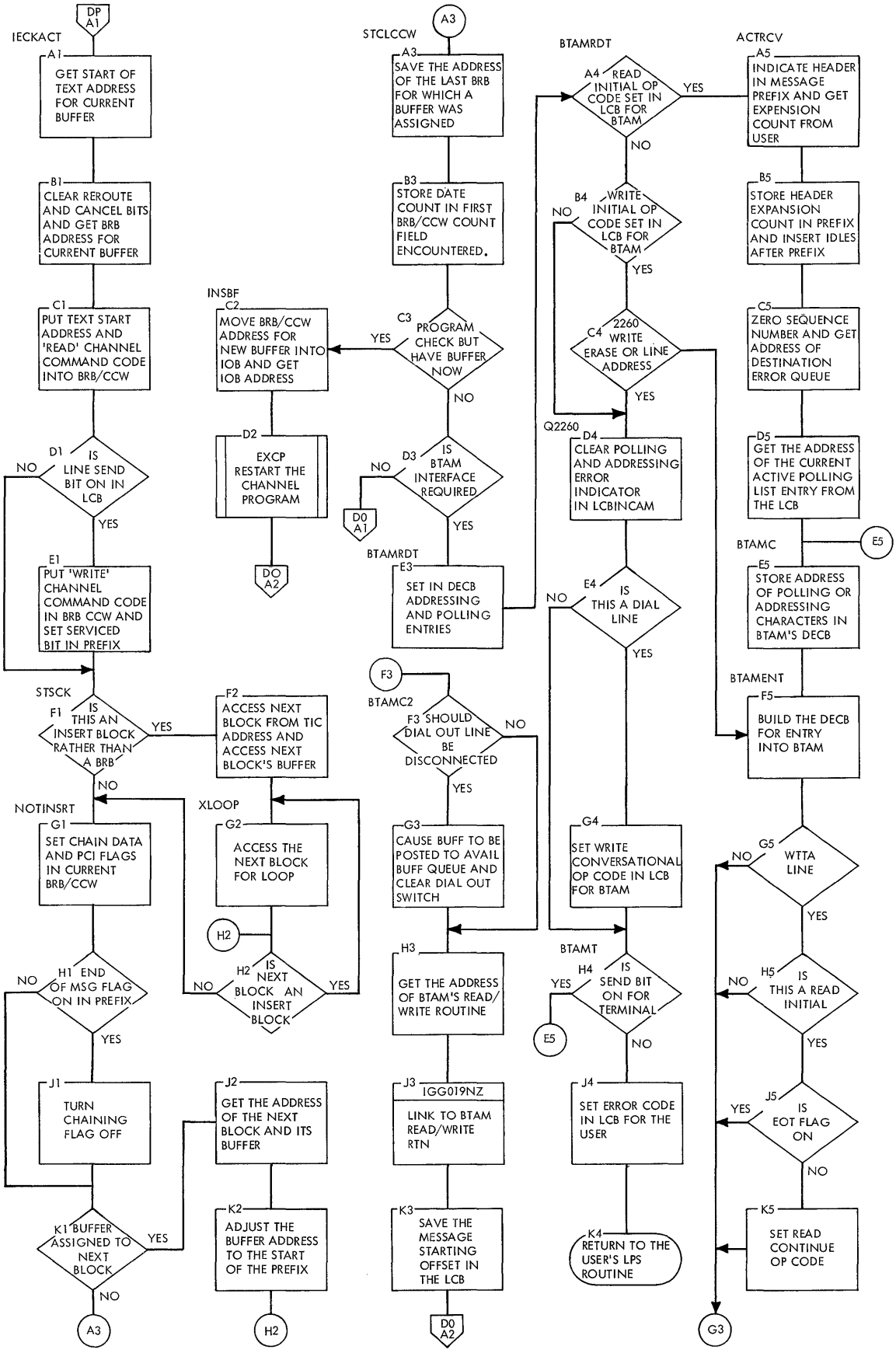


Chart DQ. Line SIO Appendage Routine

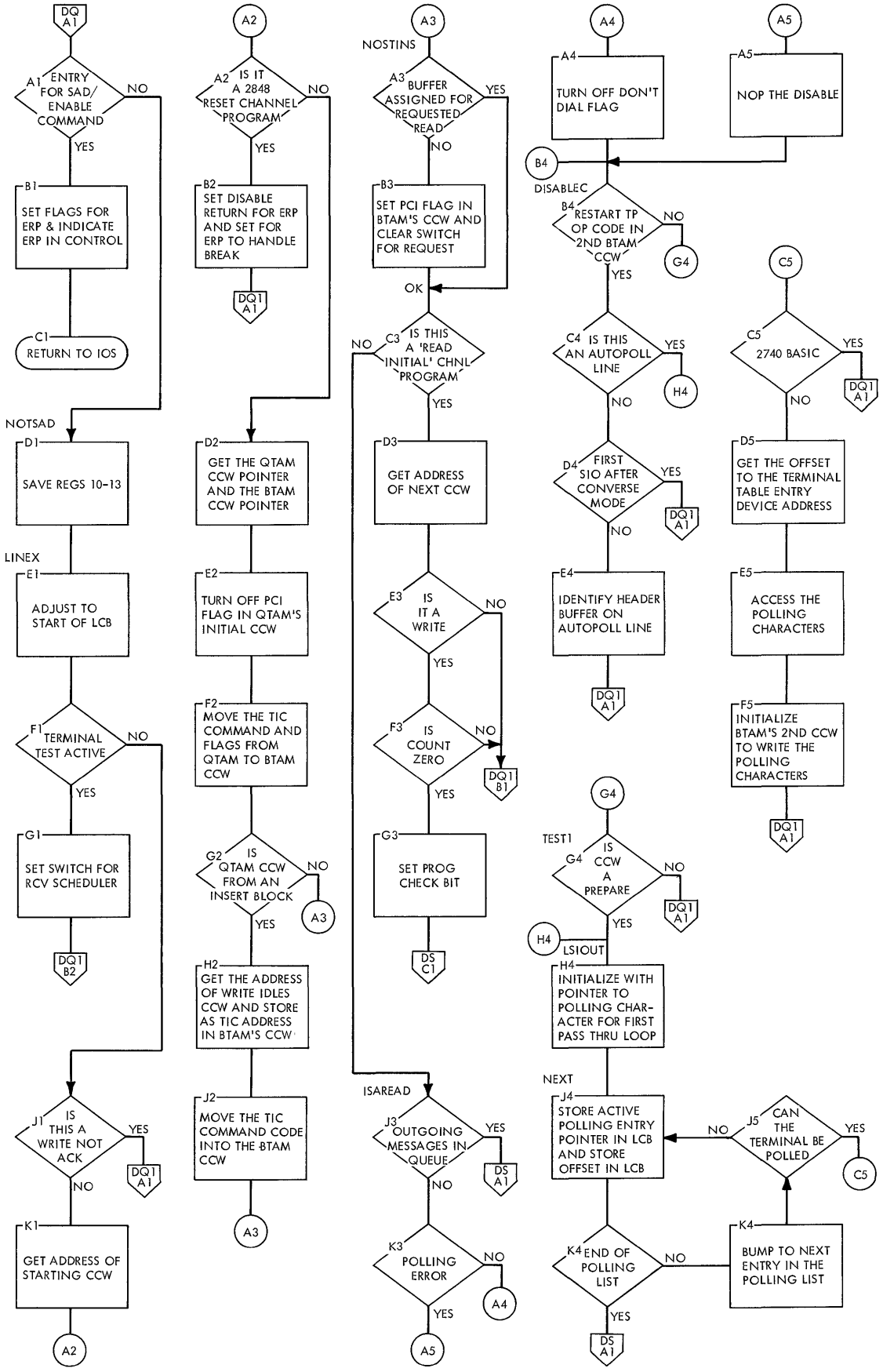


CHART DQ. LINE SIO APPENDAGE RTN

● Chart DQ1. Line SIO Appendage Routine (Continued)

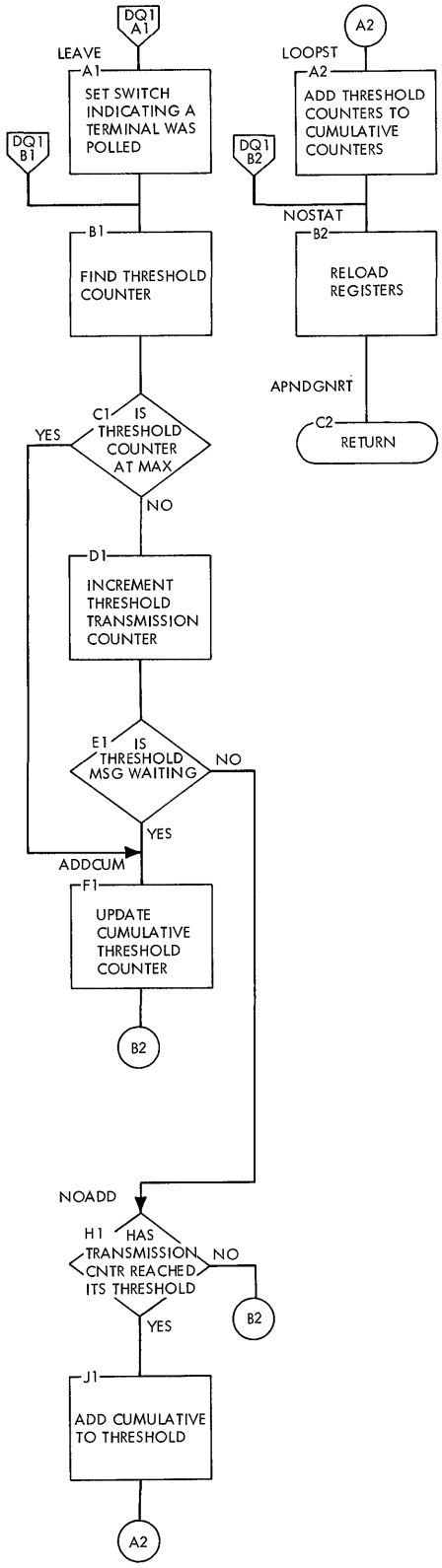


Chart DR. Line PCI Appendage Routine

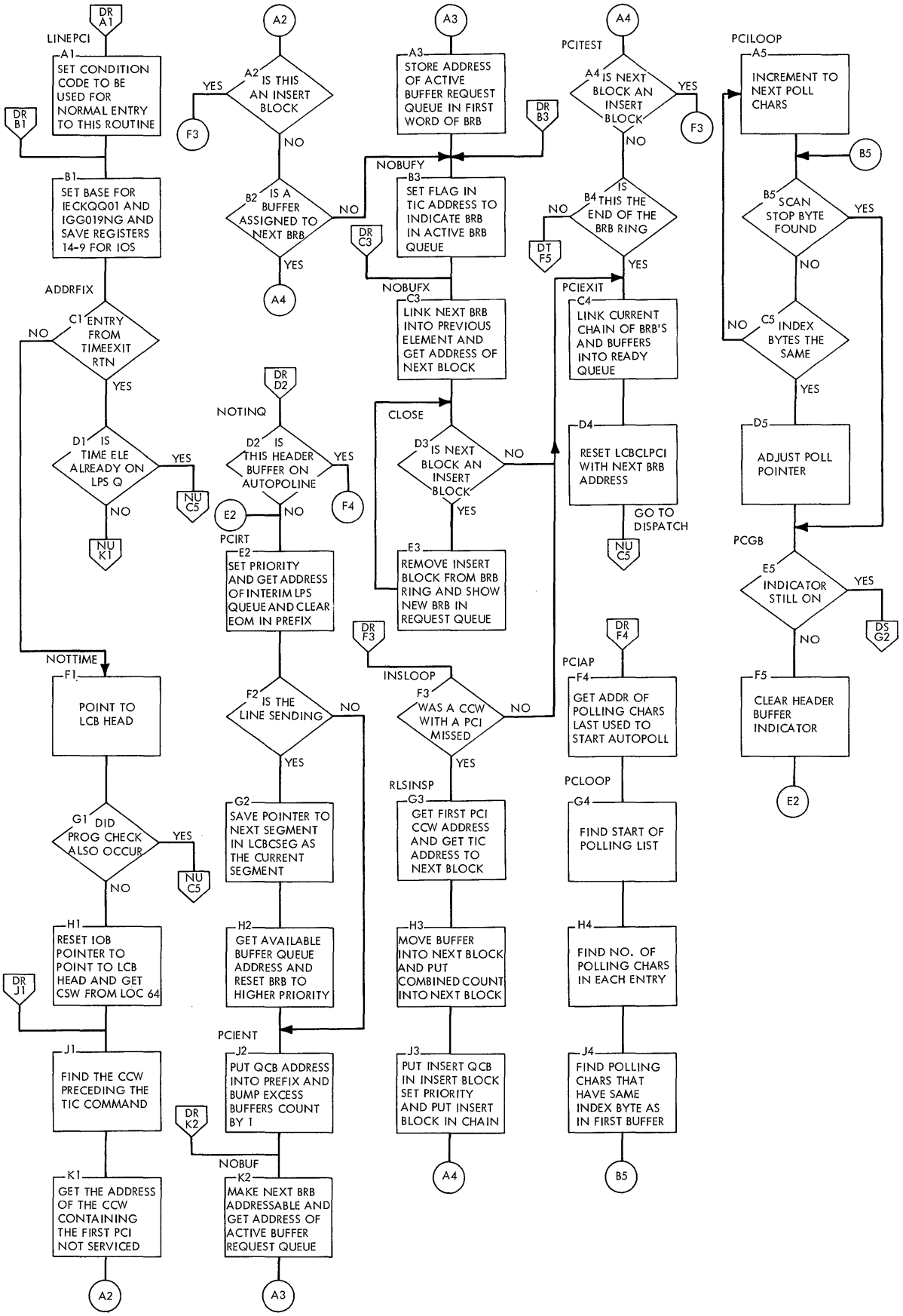


Chart DS. Line End Appendage Routine

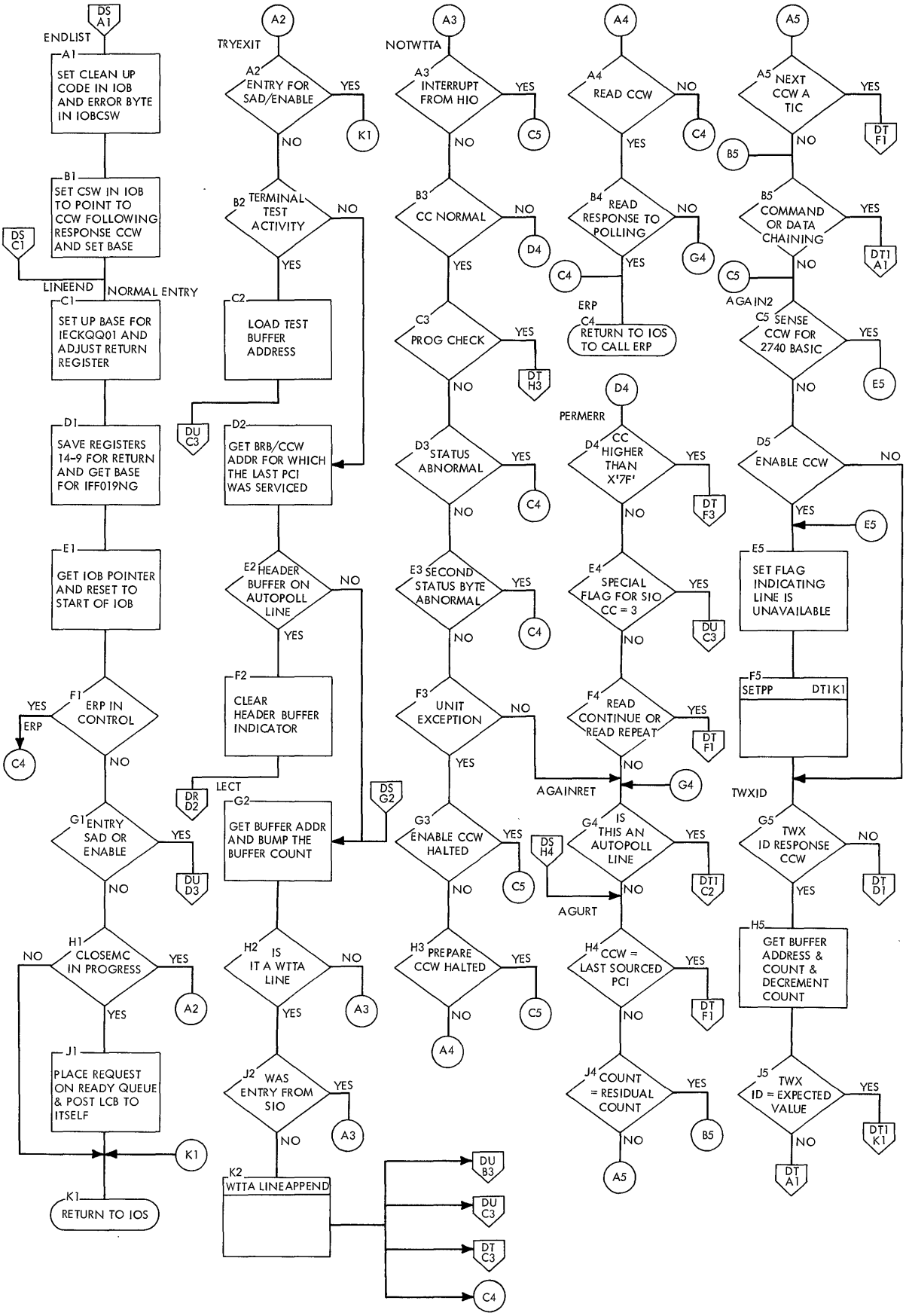


Chart DT. Line End Appendage Routine (Continued)

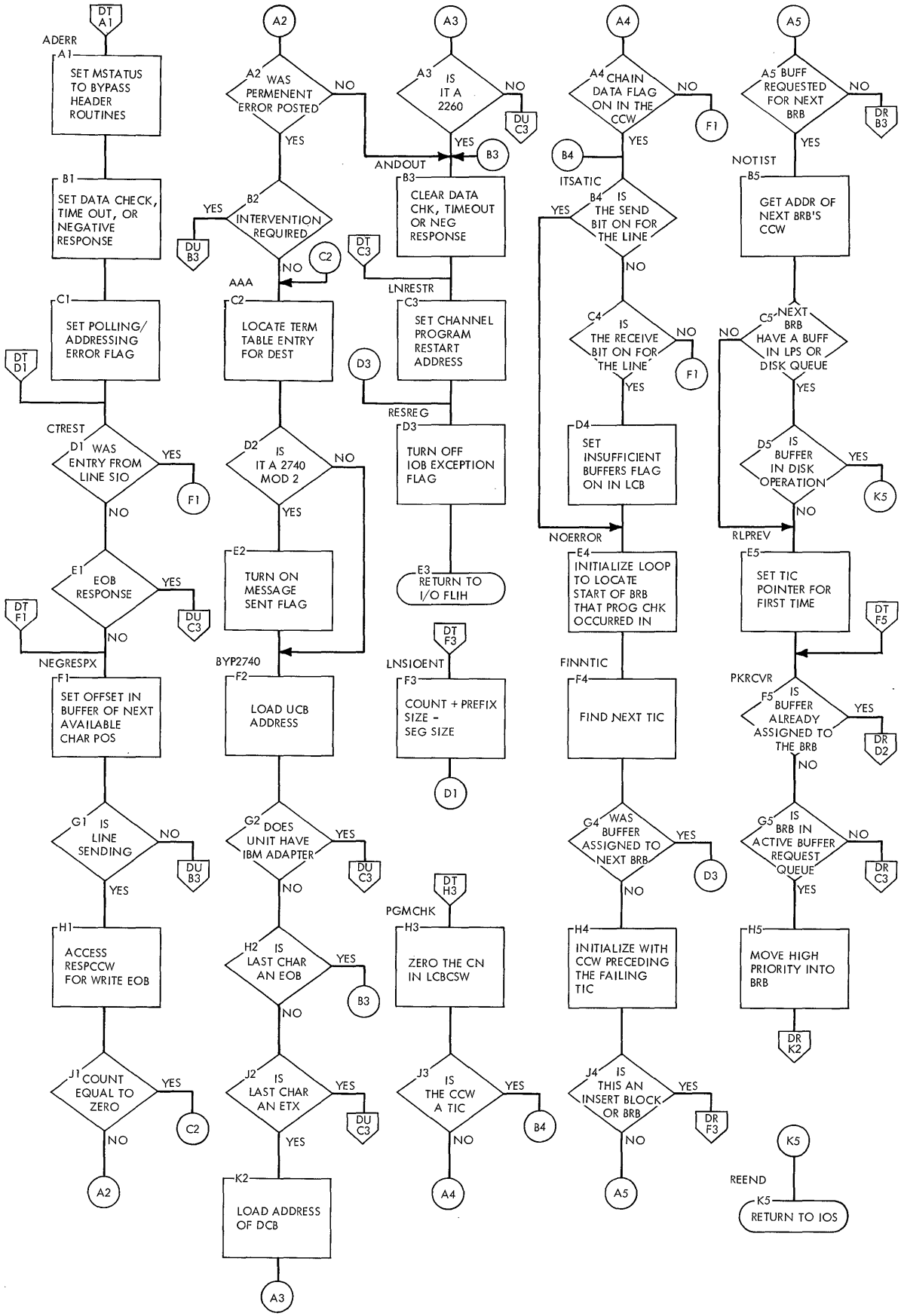


Chart DT1. Line End Appendage (continued)

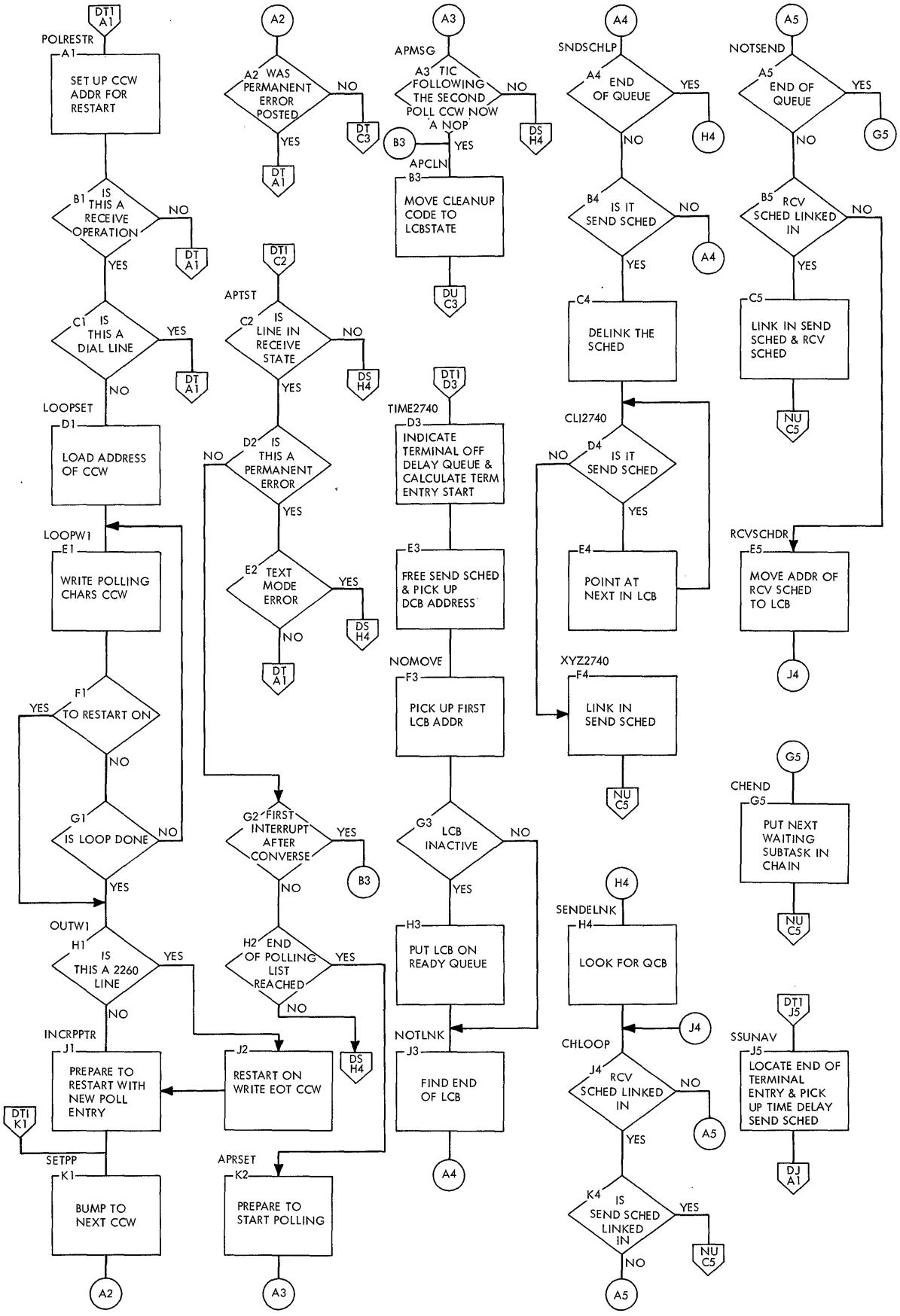


Chart DU. Interim LPS Routine

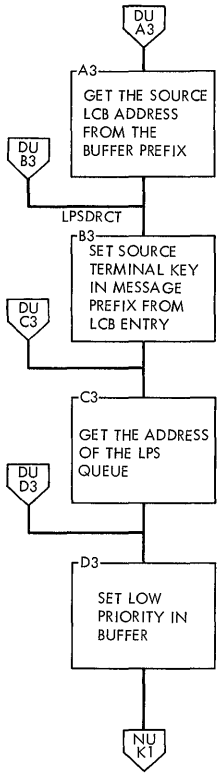


Chart DV. Get Scheduler Routine

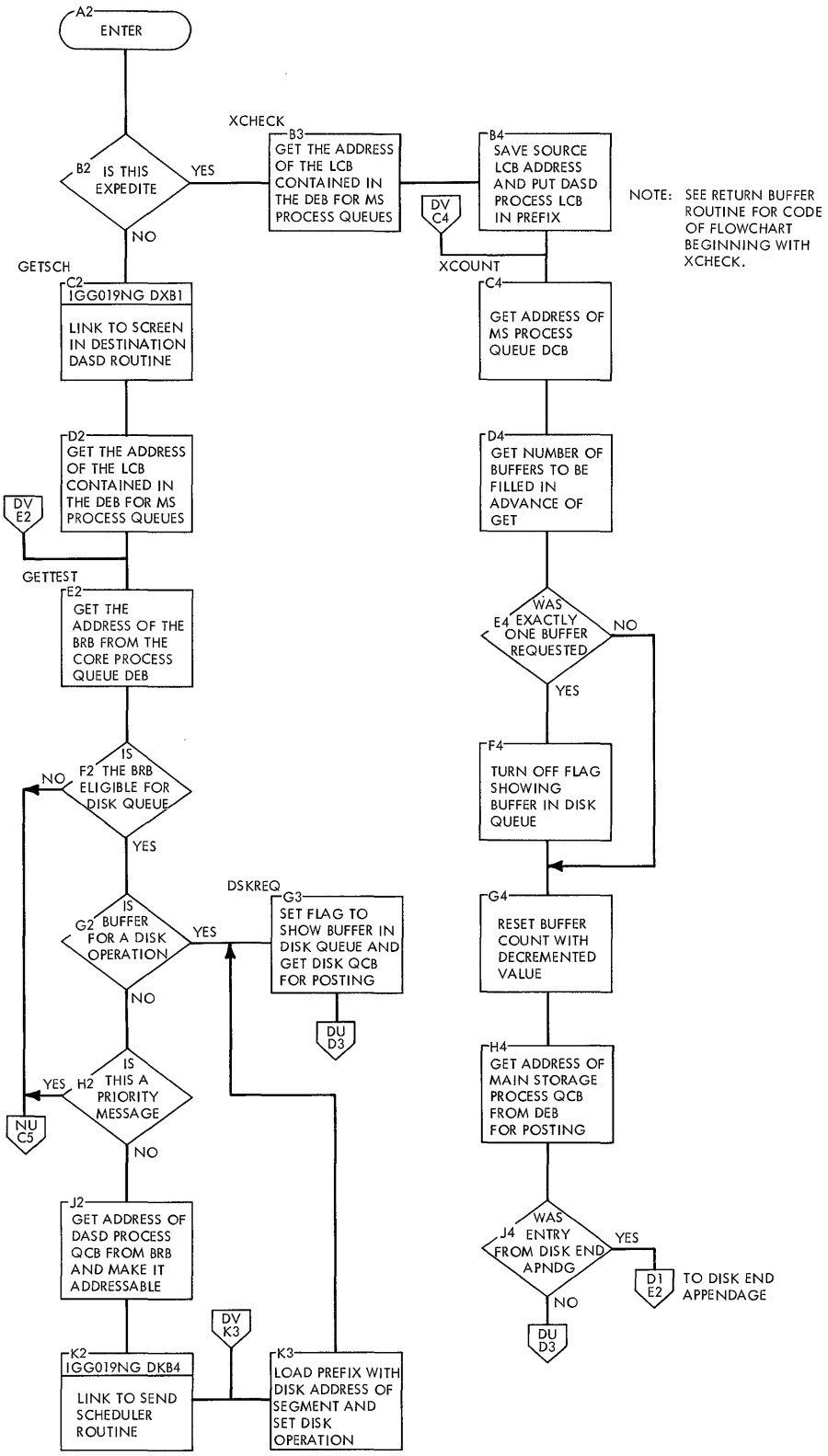


Chart DW. Return Buffer Routine

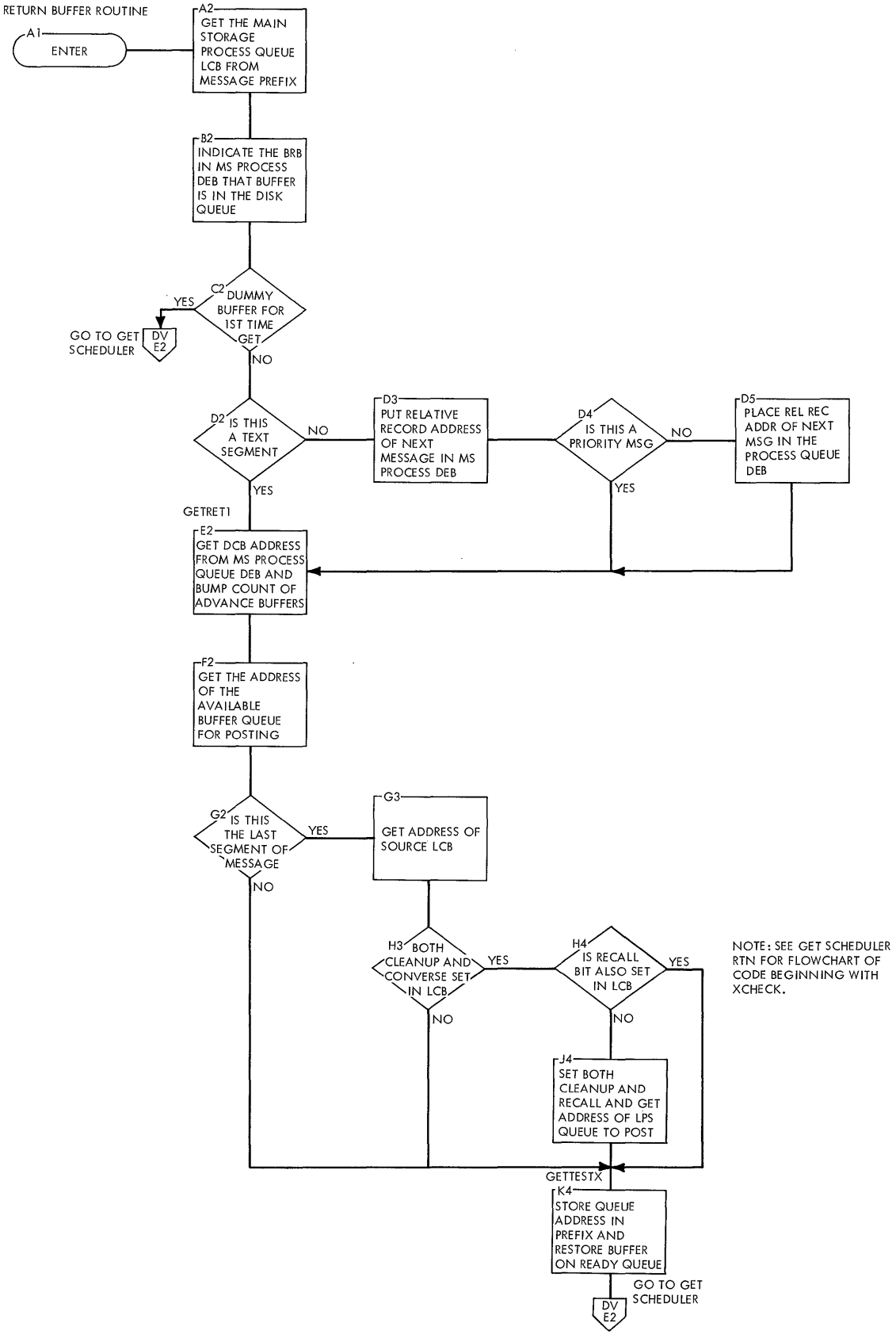


Chart DX. Destination DASD Routine

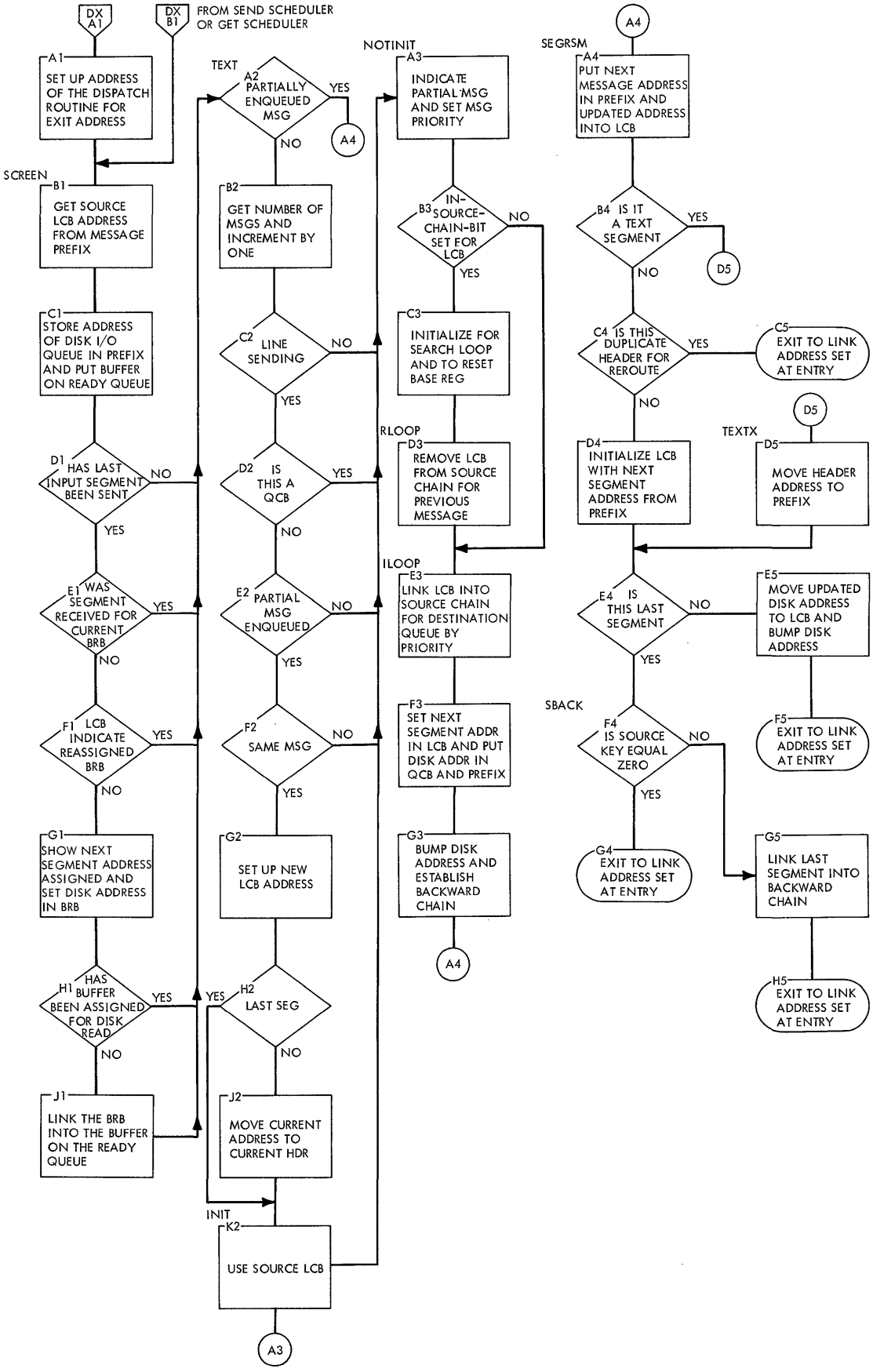


Chart DY. Cross Partition Move Routine

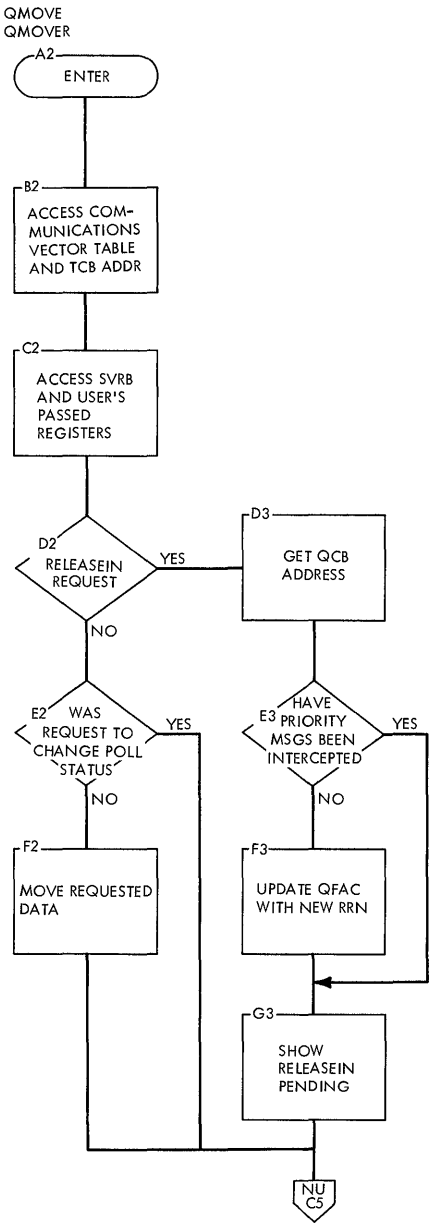
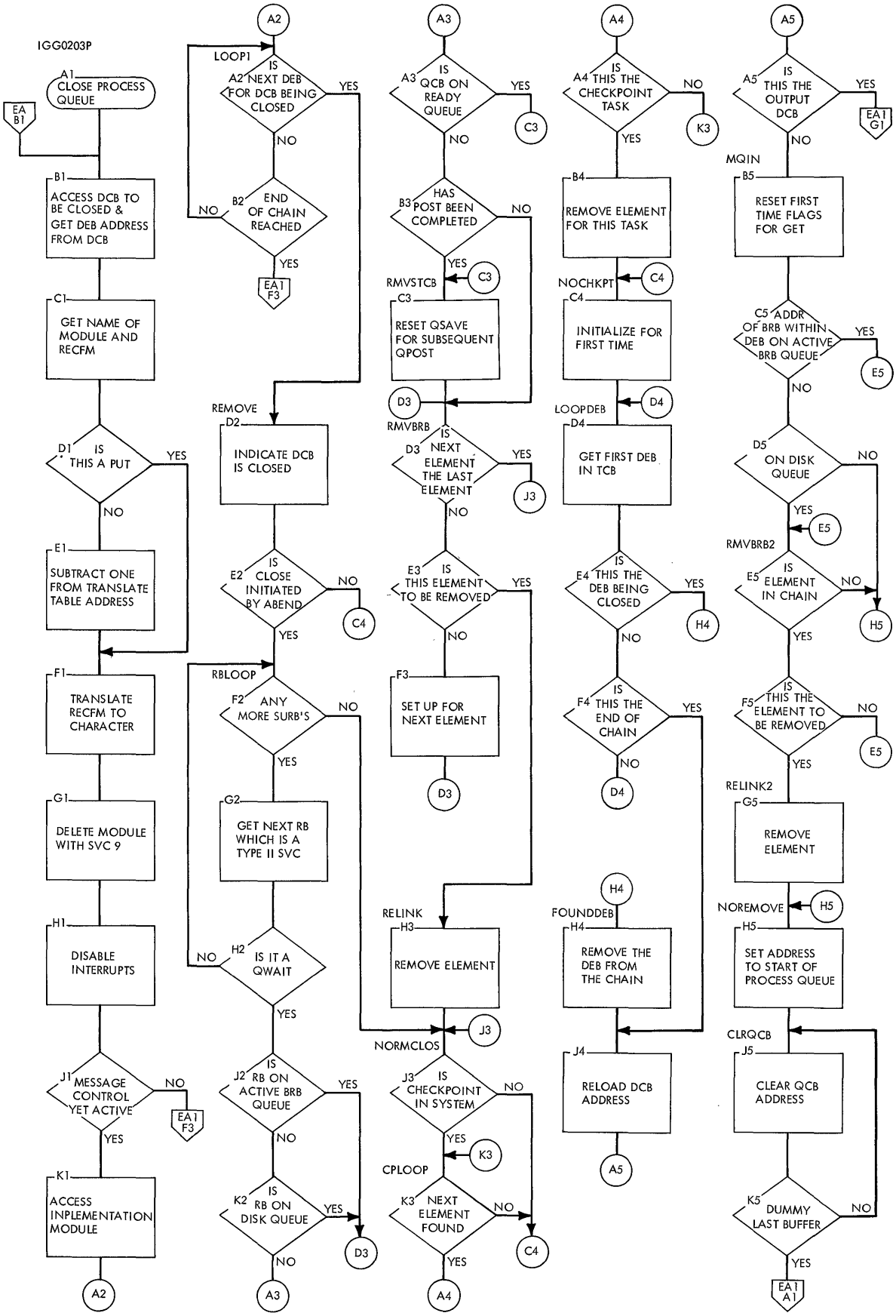


Chart EA. Close Process Queue



● Chart EA1. Close Process Queue (Continued)

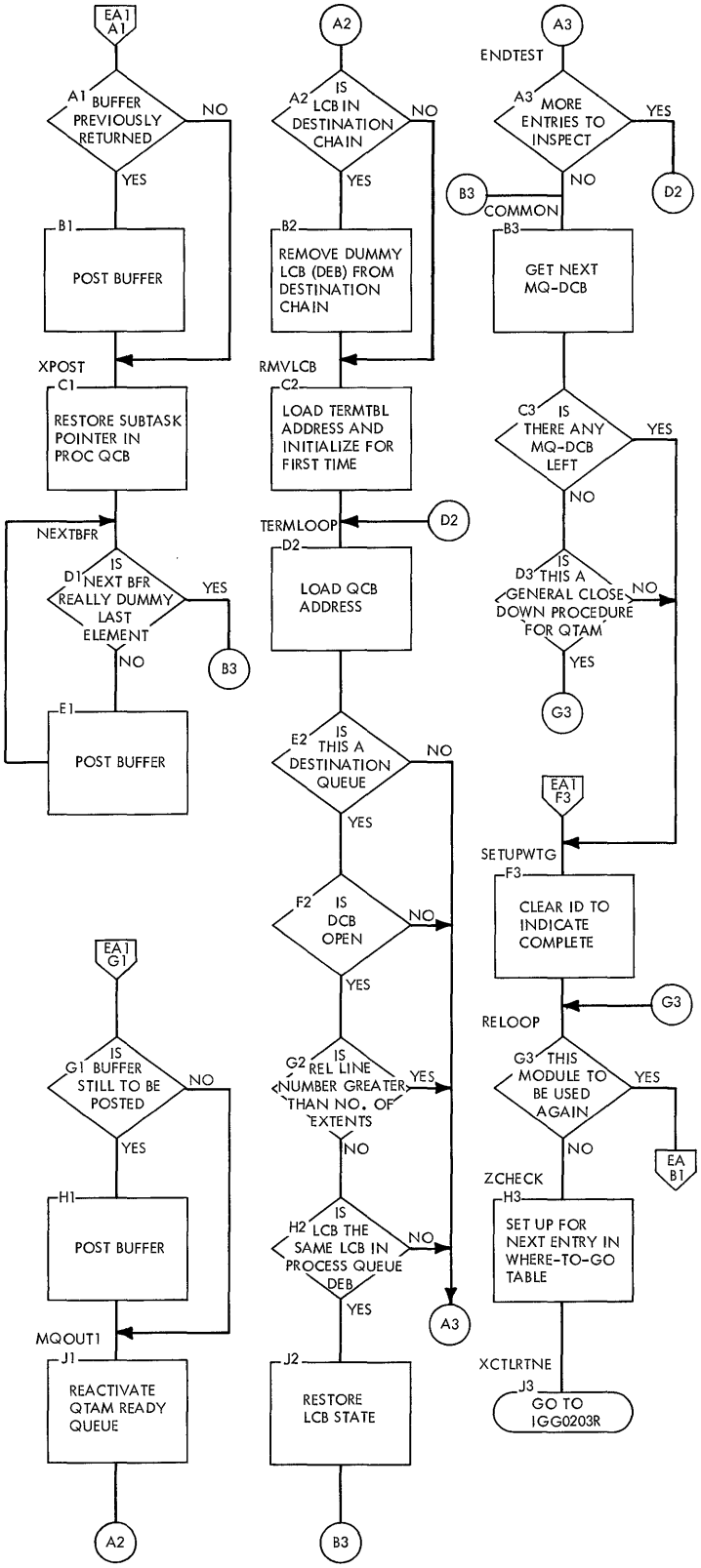


Chart EB. Close Communications Line Group

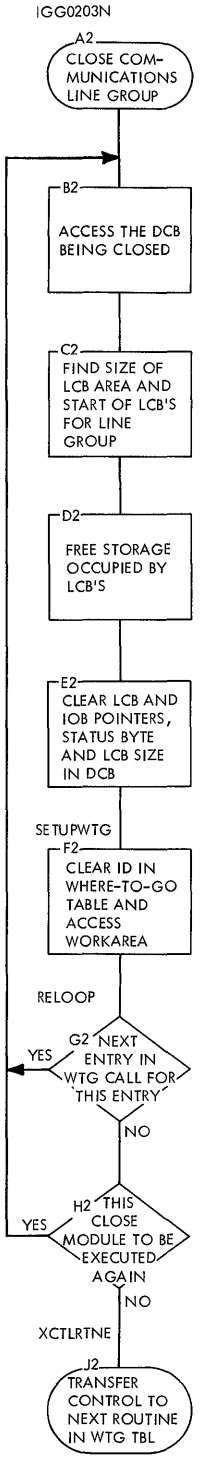


Chart EC. Close Direct Access Message Queue

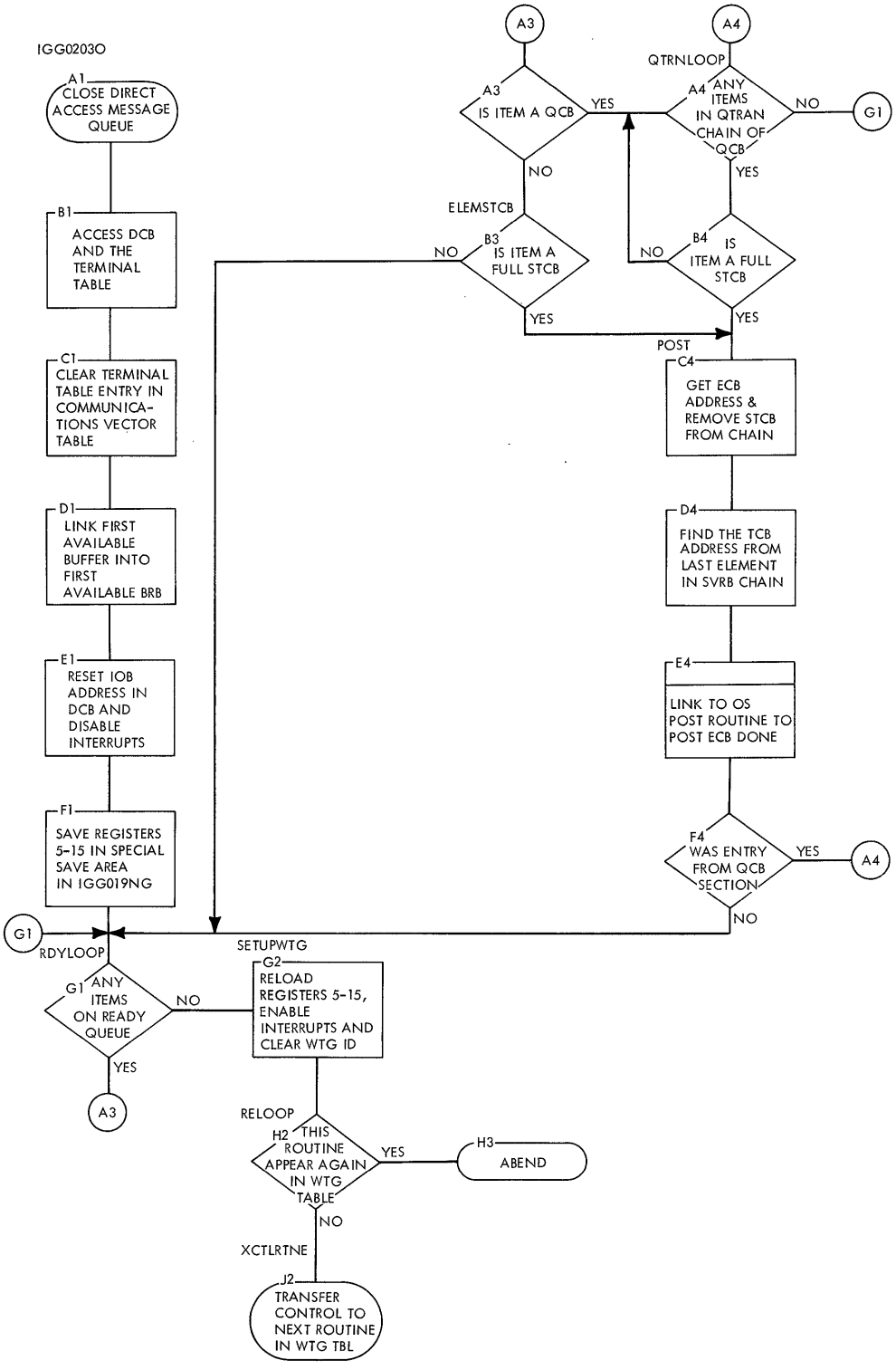


Chart ED. Close Routine

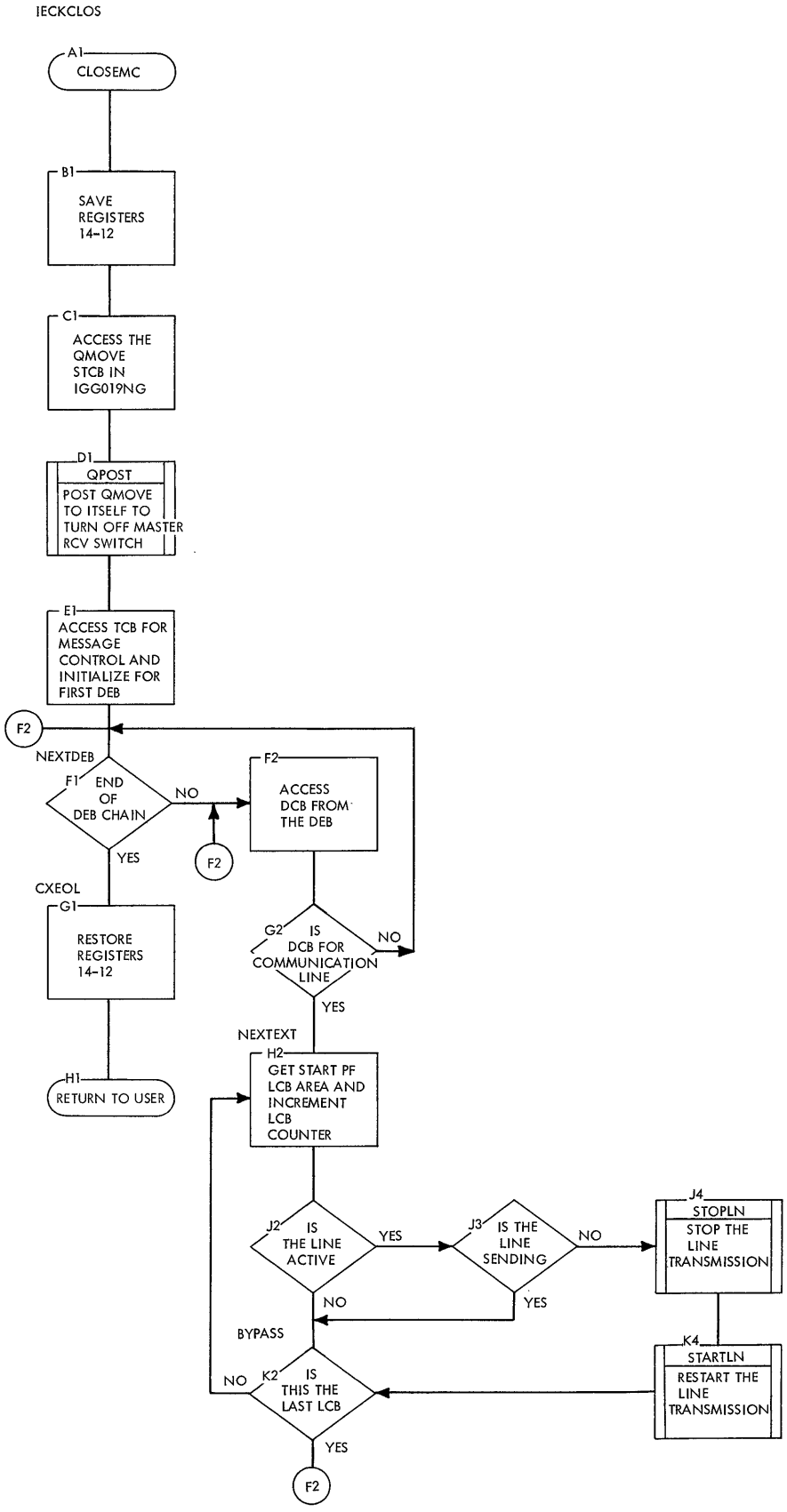


Chart EE. Operator Control Routine

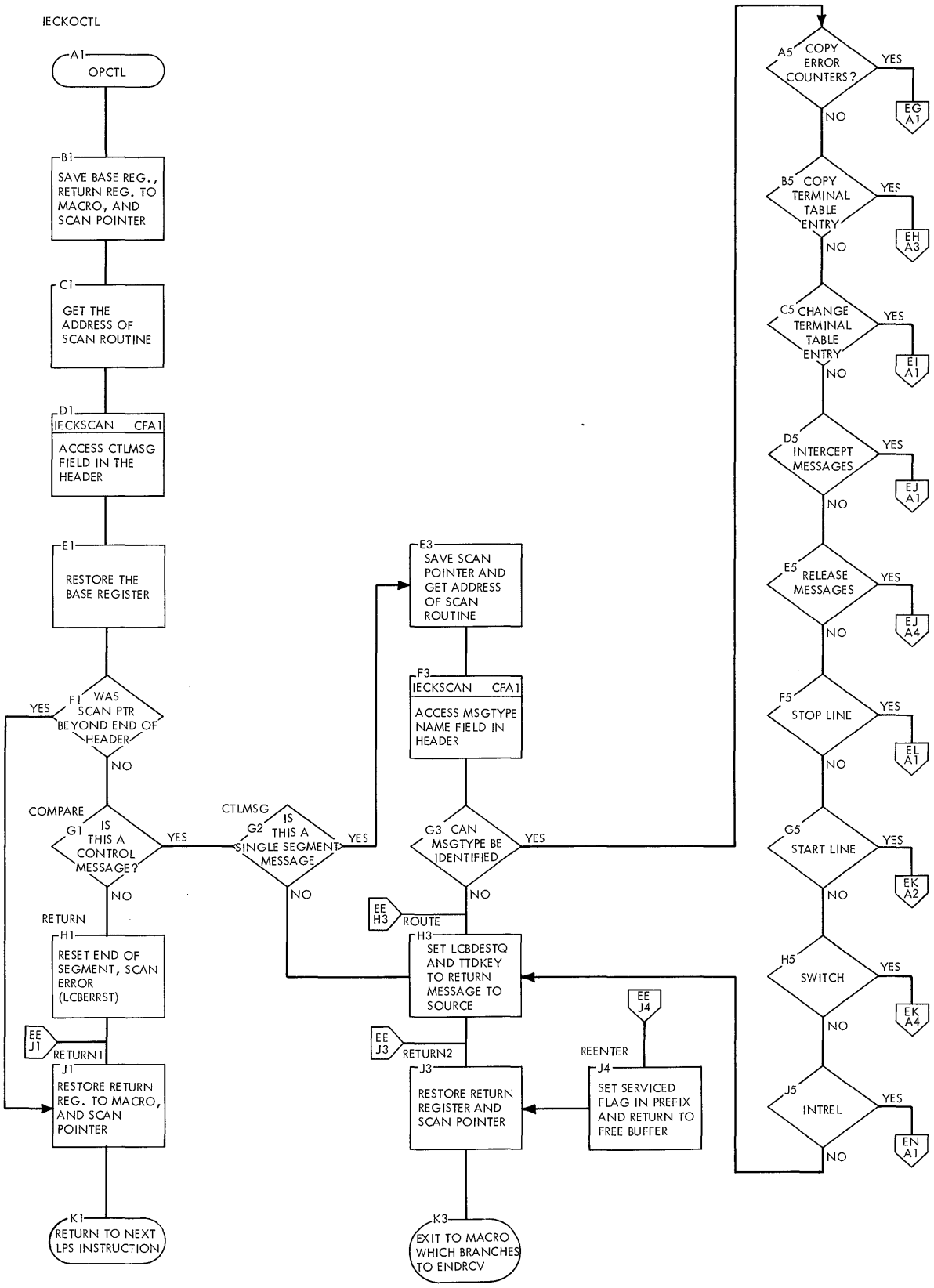


Chart EF. Common Subroutines OPTCL

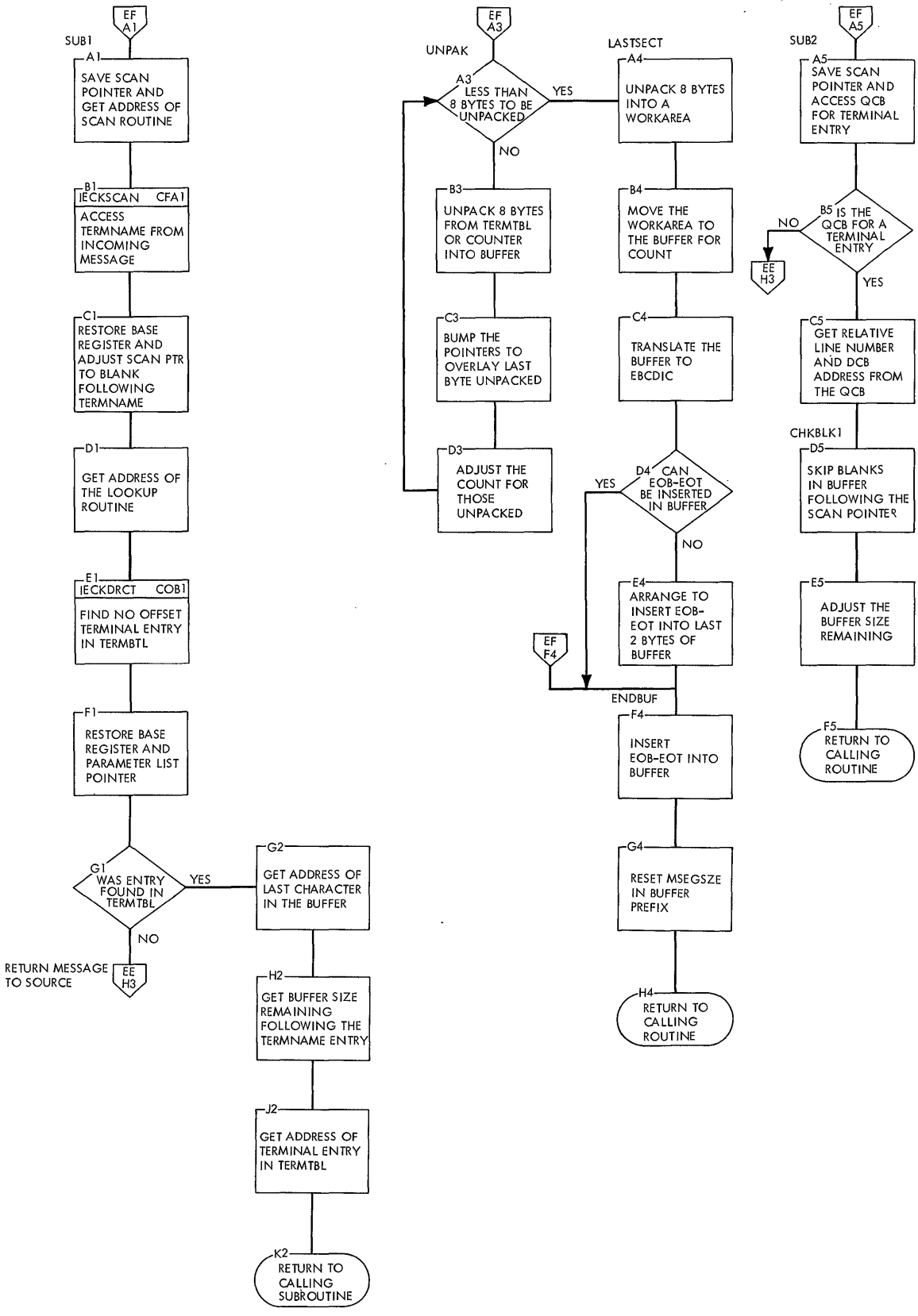


Chart EG. Common Subroutines OPTCL (Continued)

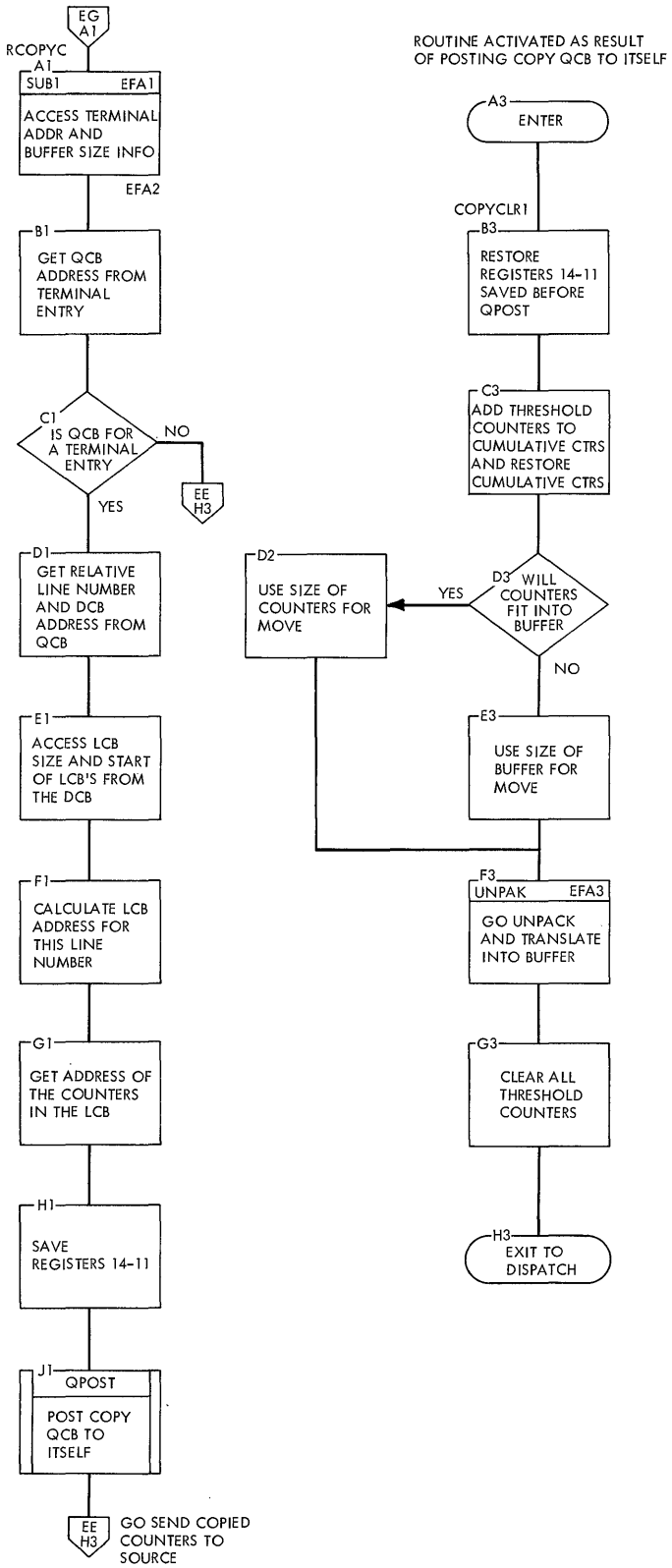


Chart EH. Copy Termtbl Entry OPTCL Routine

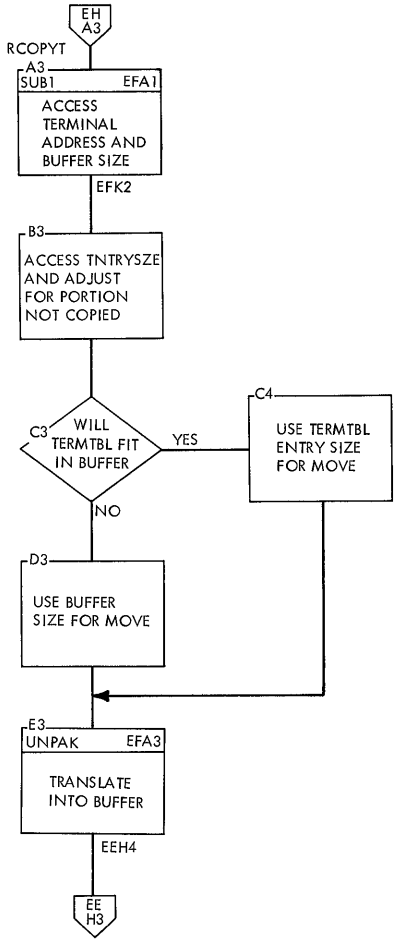


Chart EI. Change Termtbl Entry OPTCL Routine

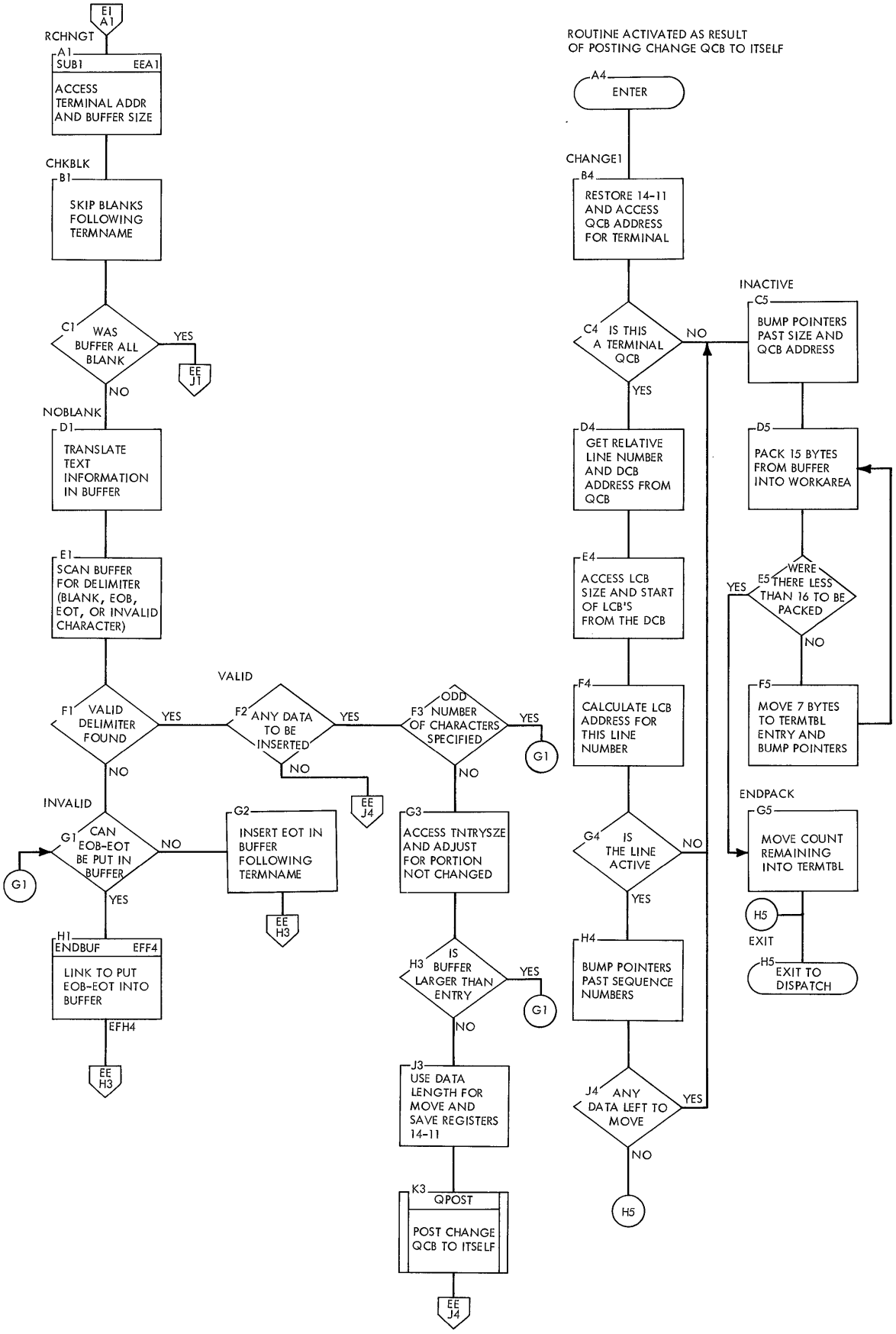


Chart EJ. Intercept and Release OPTCL Routine

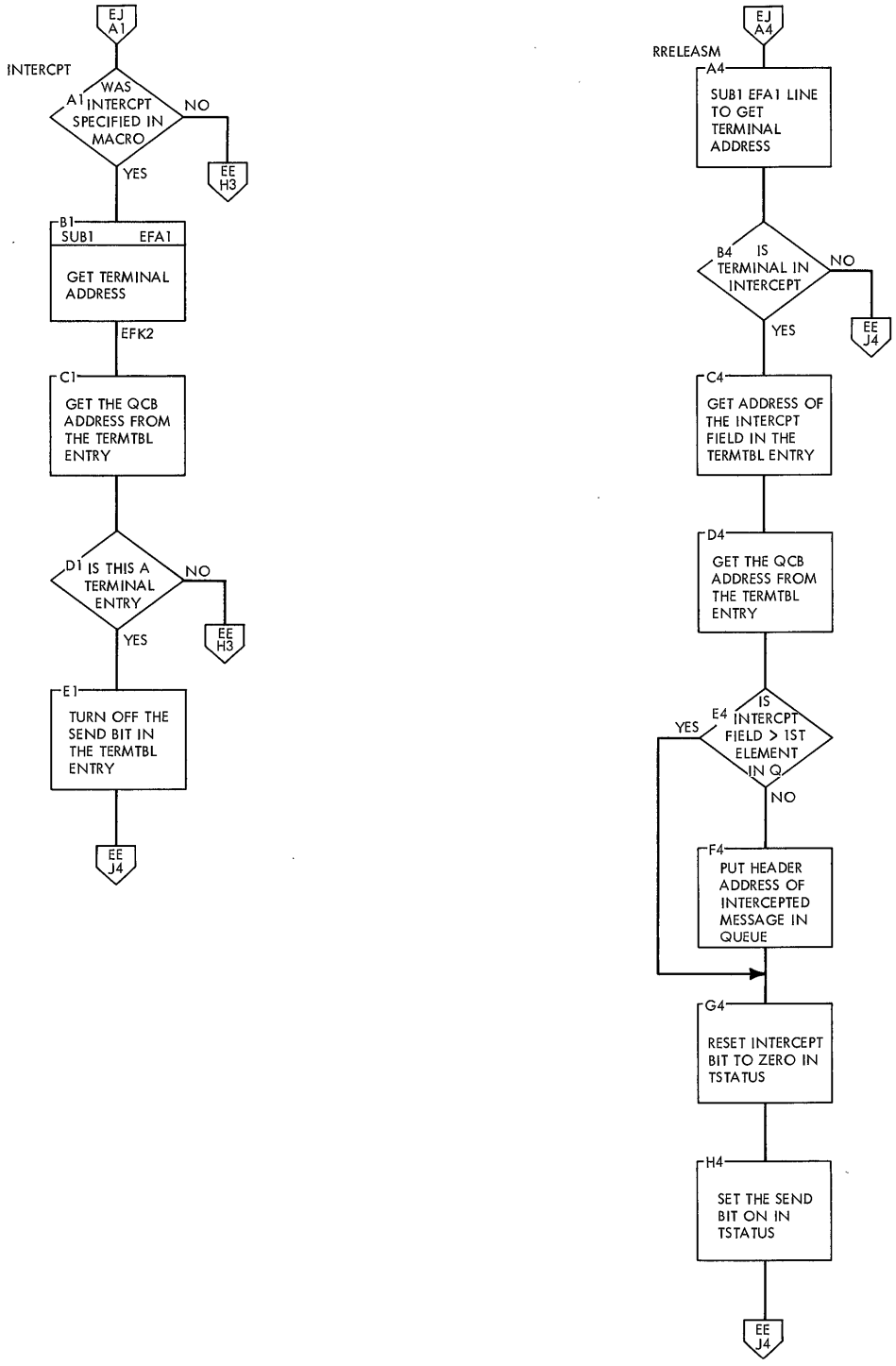


Chart EK. Start Line OPTCL Routine

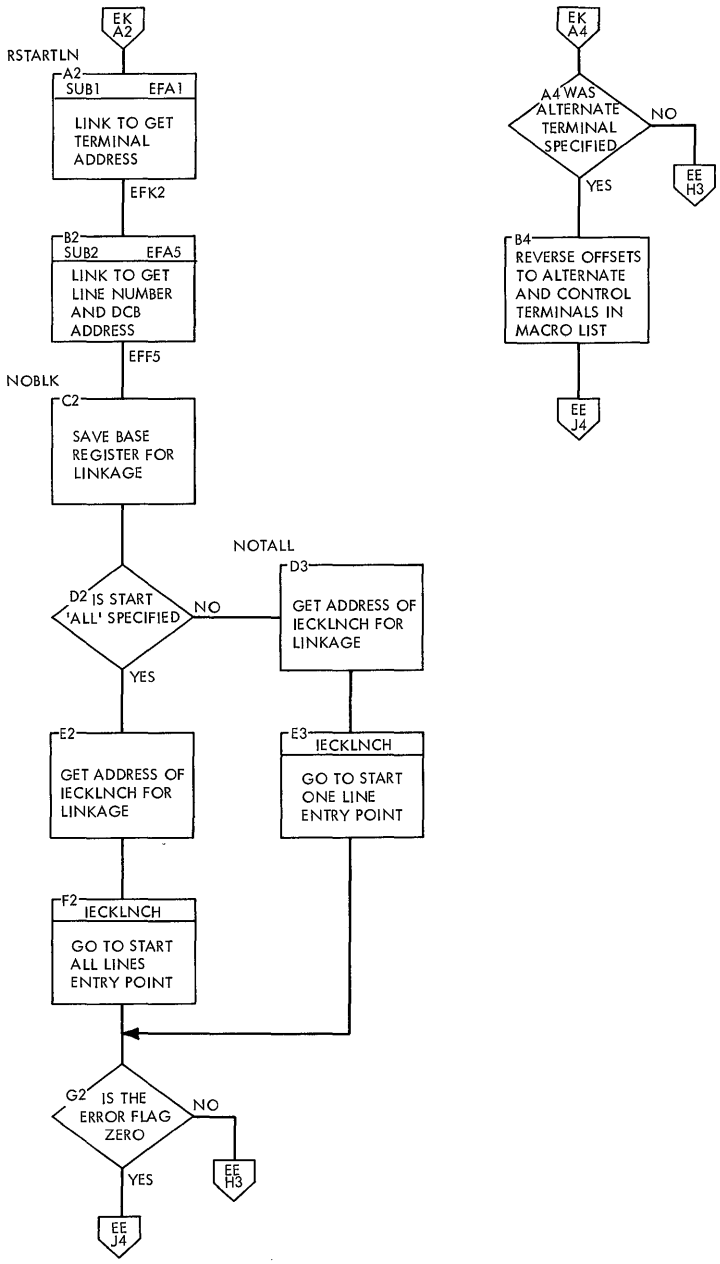


Chart EL. Stop Line OPTCL Routine

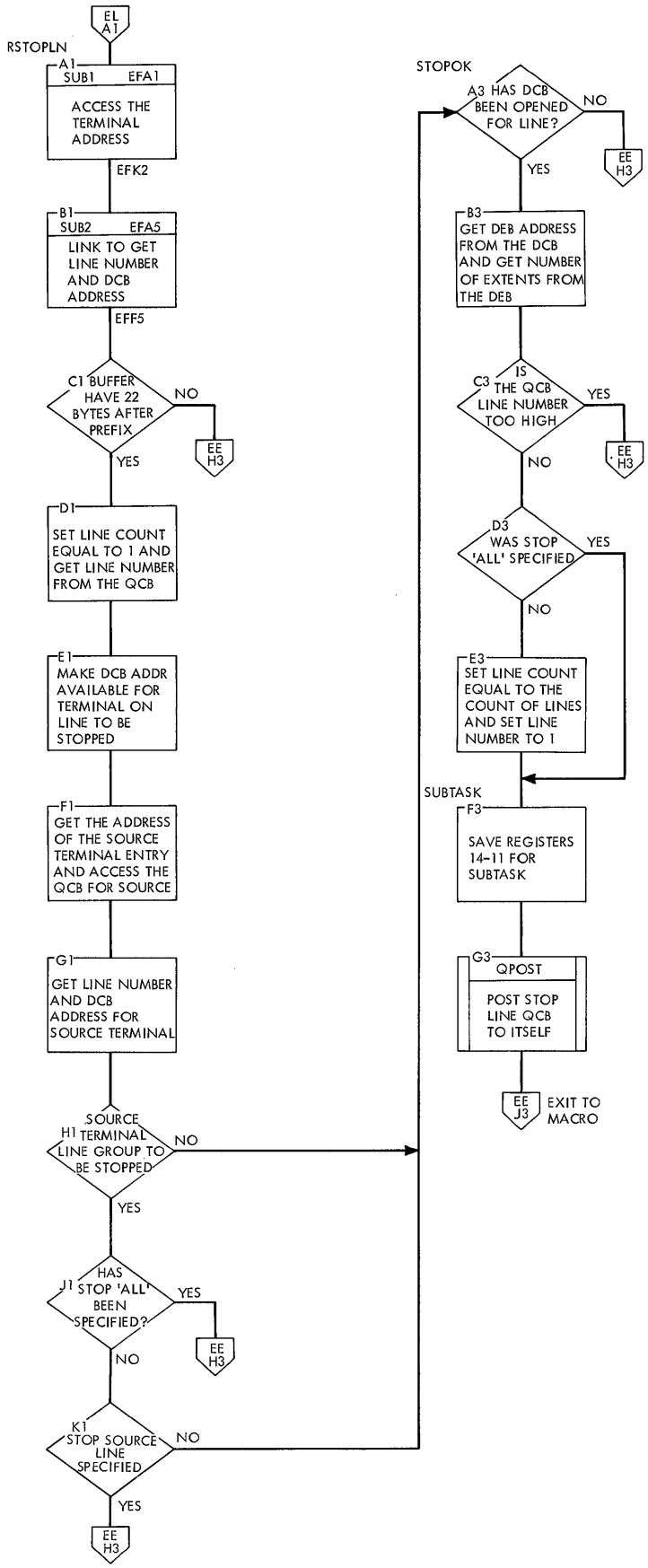


Chart EM. Stop Line OPTCL Routine (Continued)

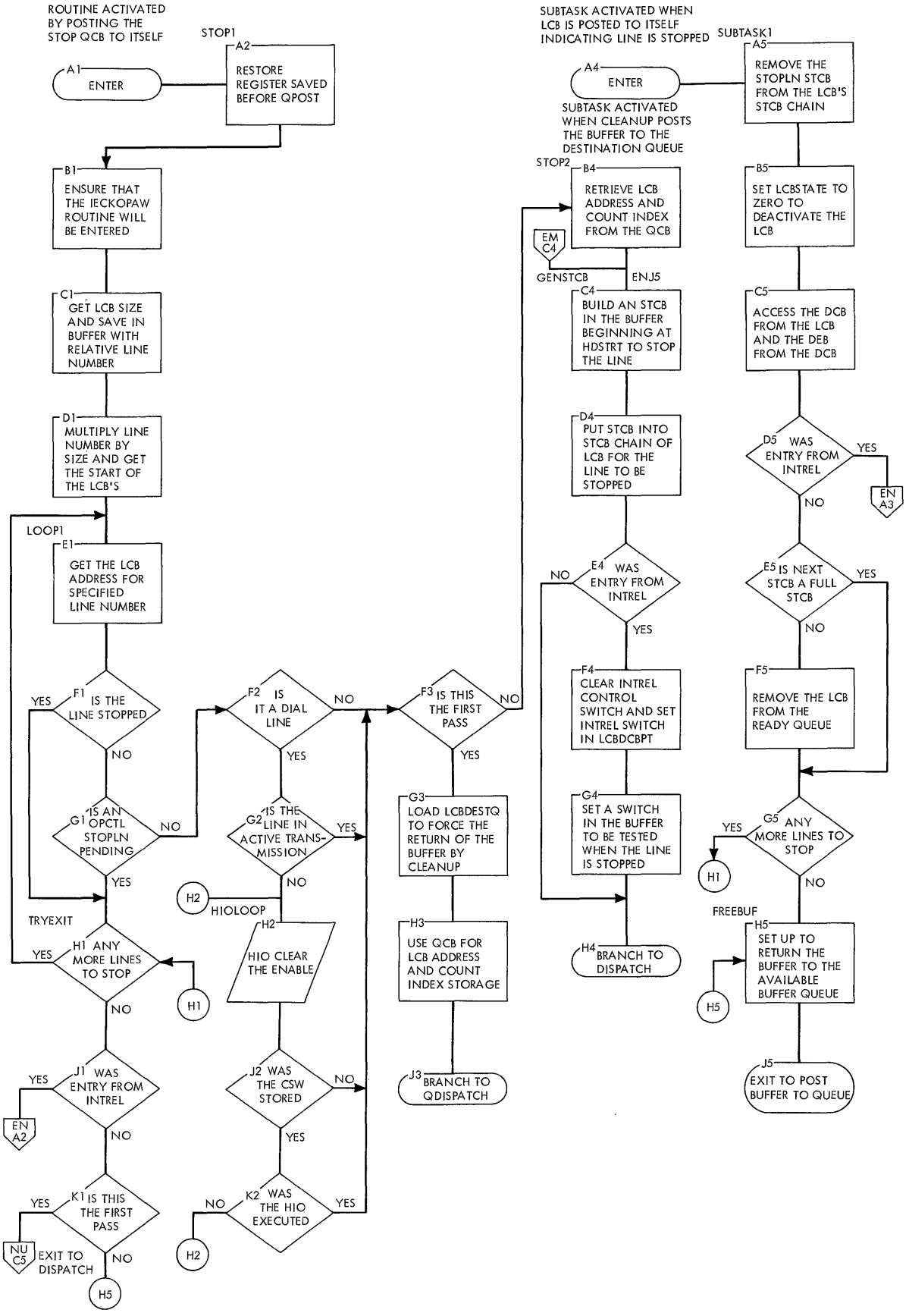


Chart EN. Intrel OPTCL Routine

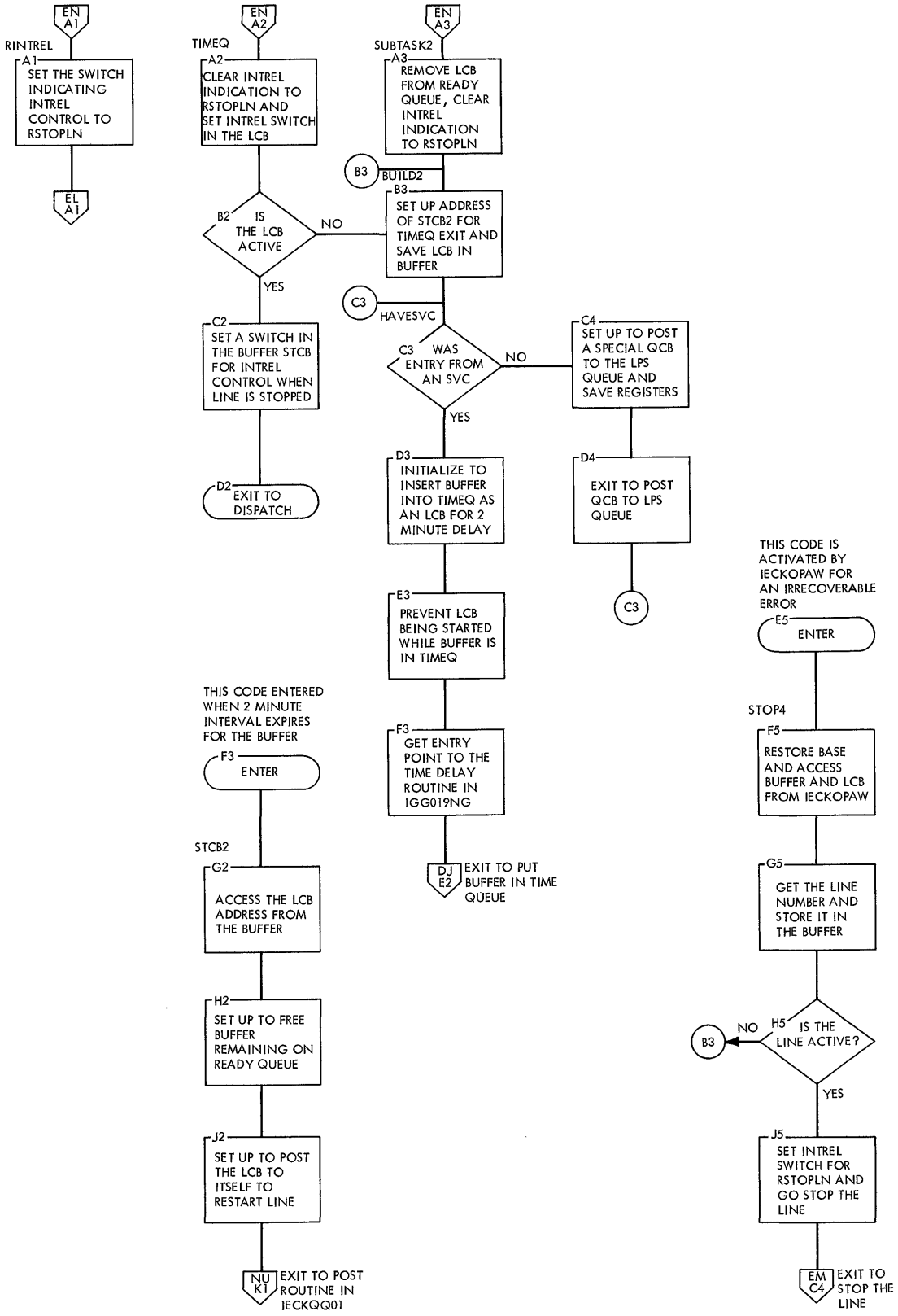


Chart EO. Operator Awareness Routine

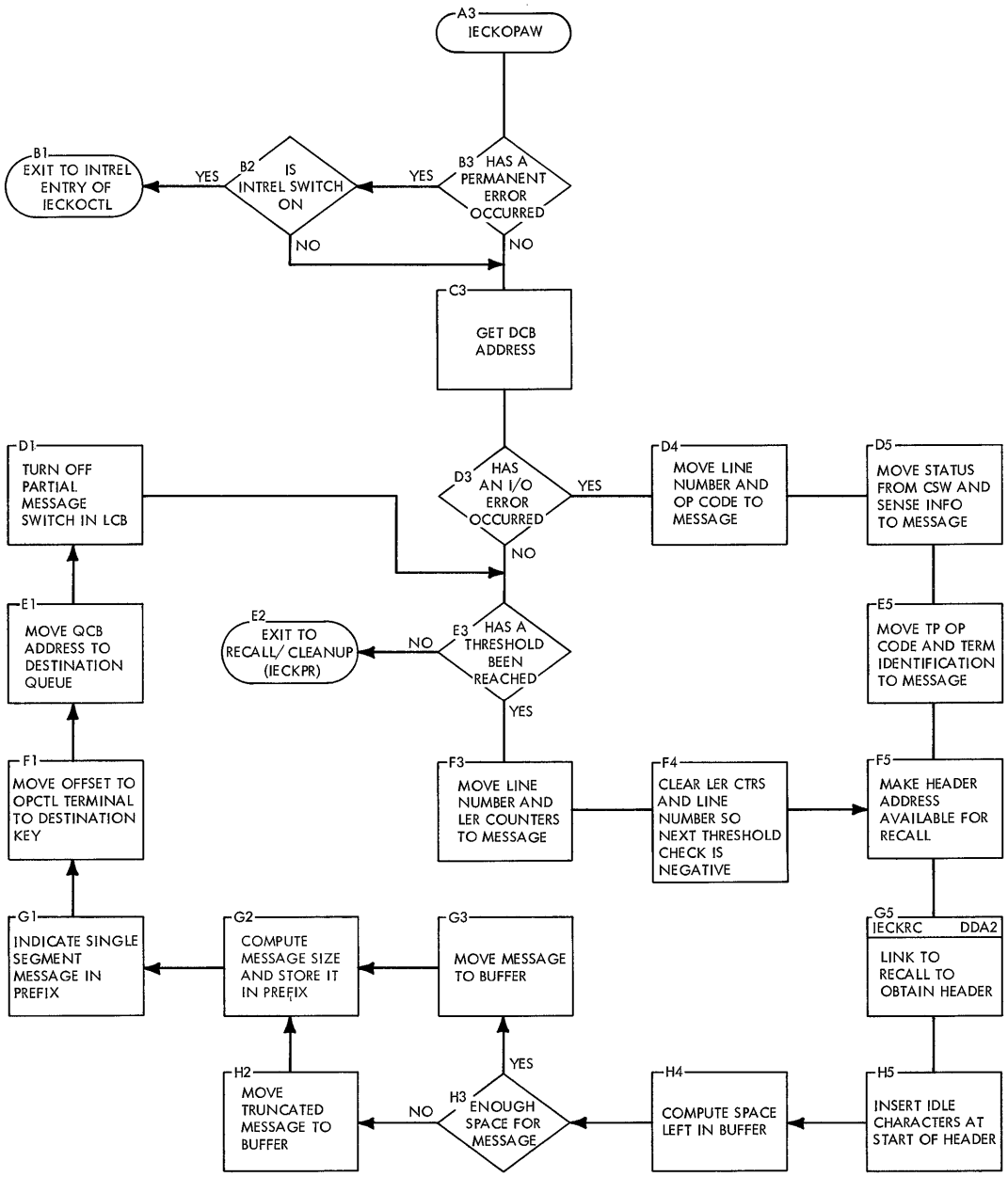


Chart F1. OPEN Line Group Load 1 Executor Routine

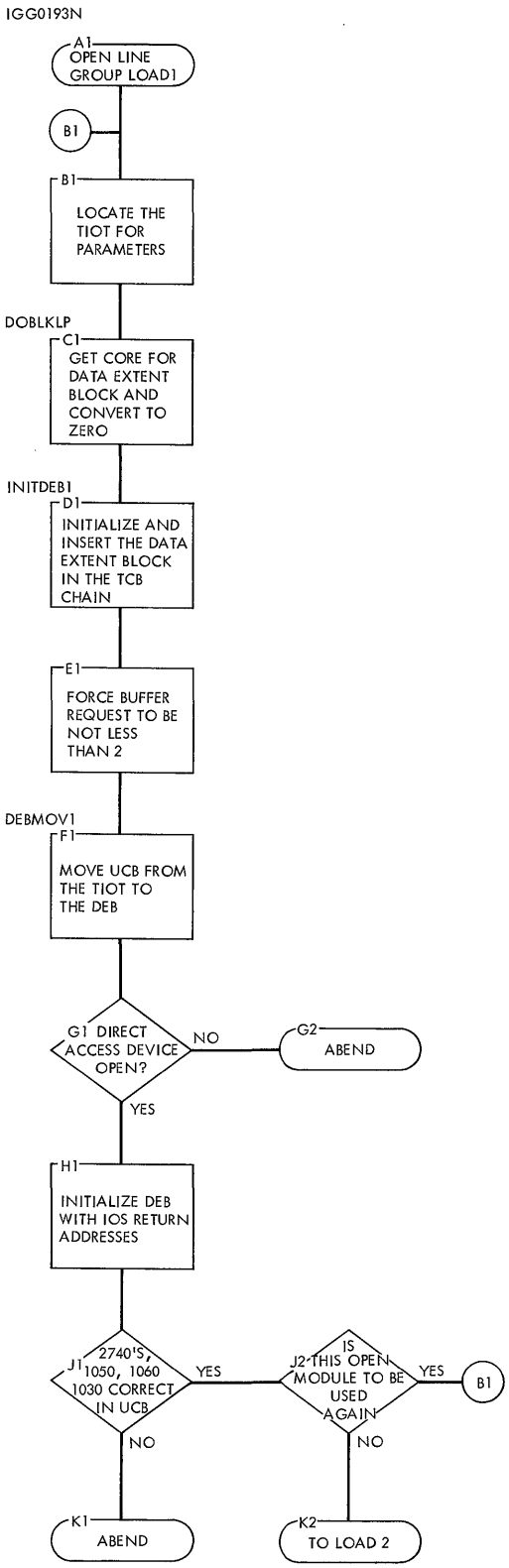


Chart F2. OPEN Line Group Load 2 Executor Routine

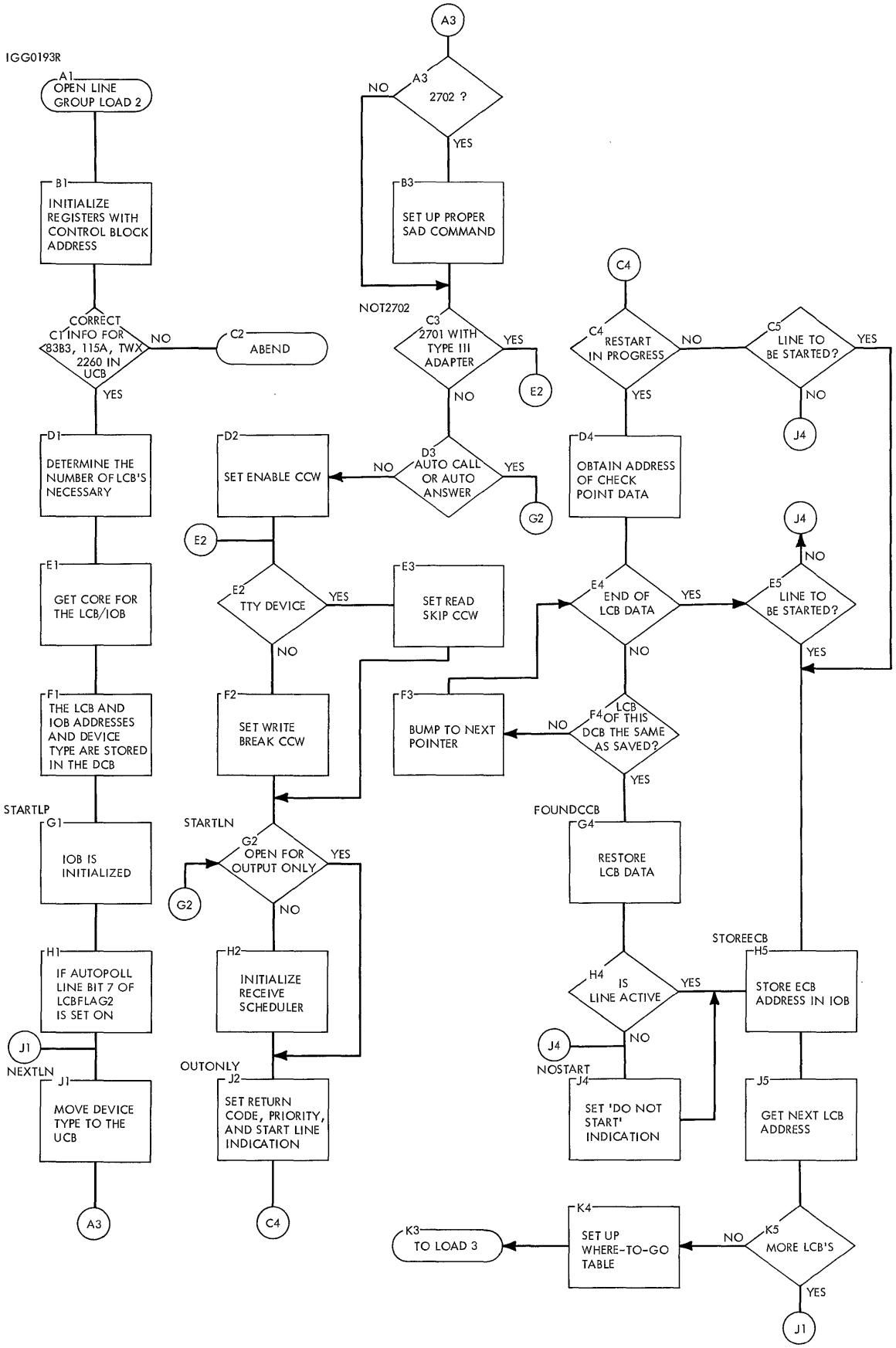


Chart F3. Open Line Group Load 3 Executor Routine

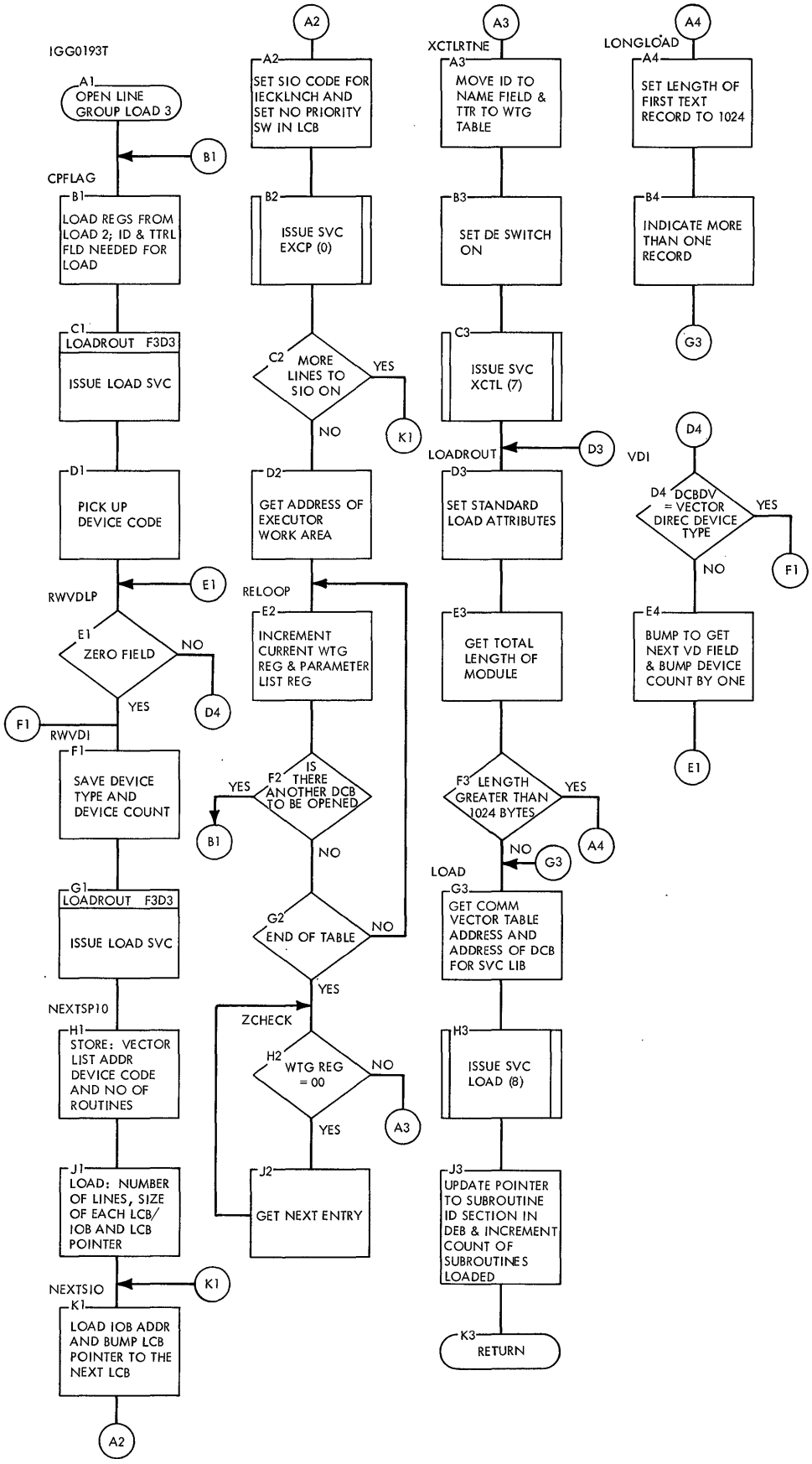


Chart F4. OPEN Direct Access Message Queue Routine

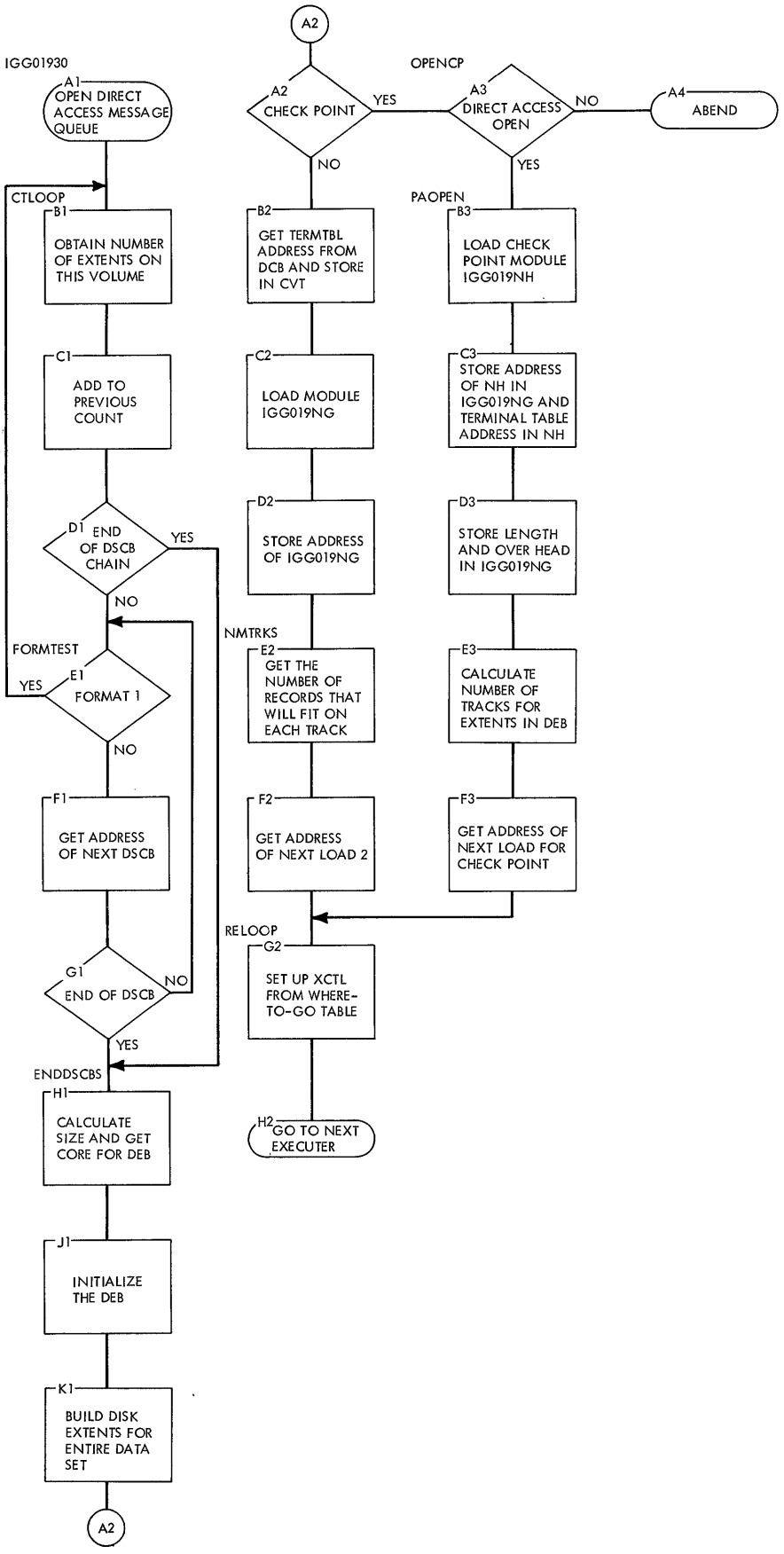


Chart F5. OPEN Direct Access Load 2 Routine

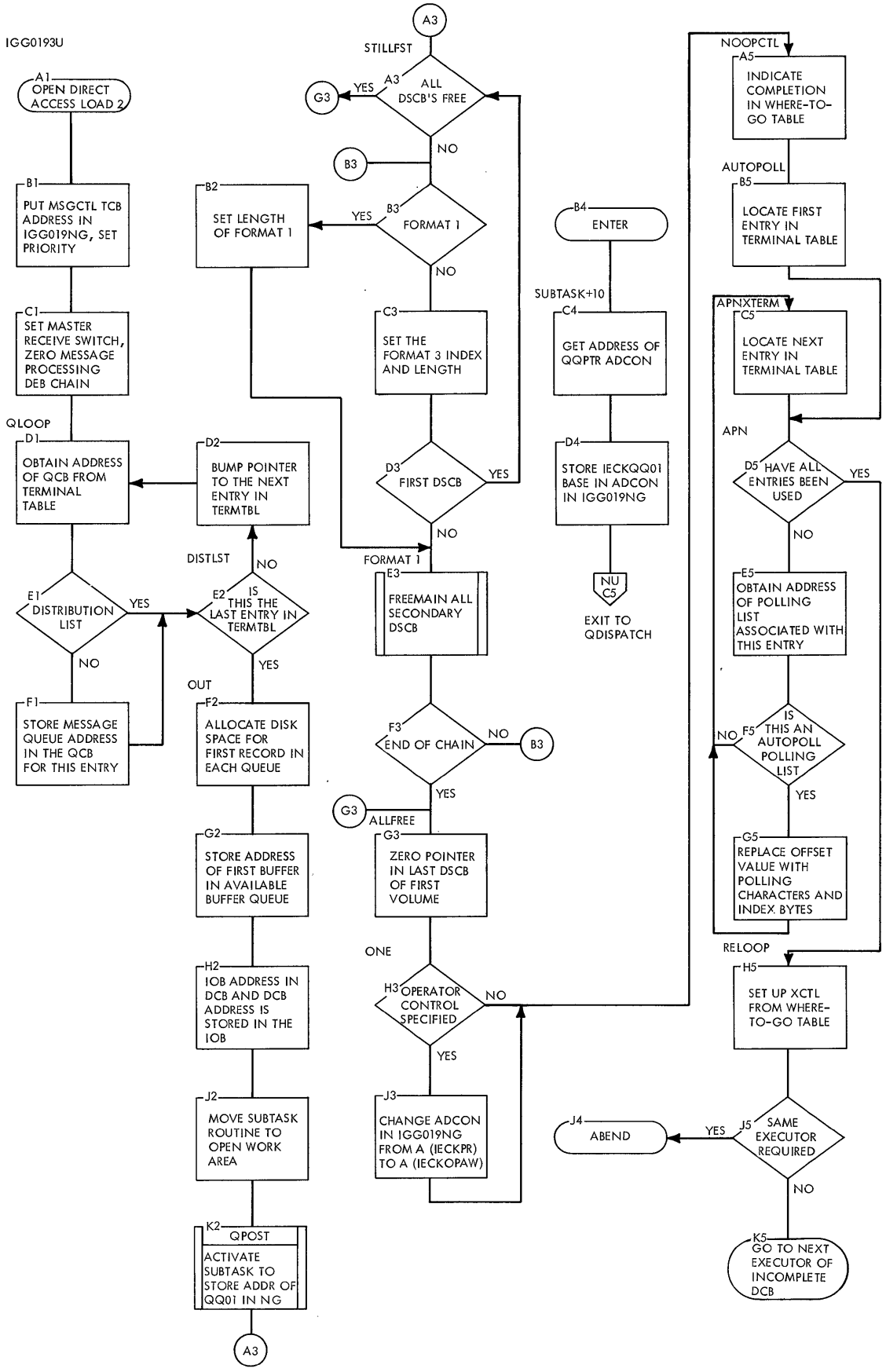


Chart F6. OPEN Checkpoint Data Set Routine

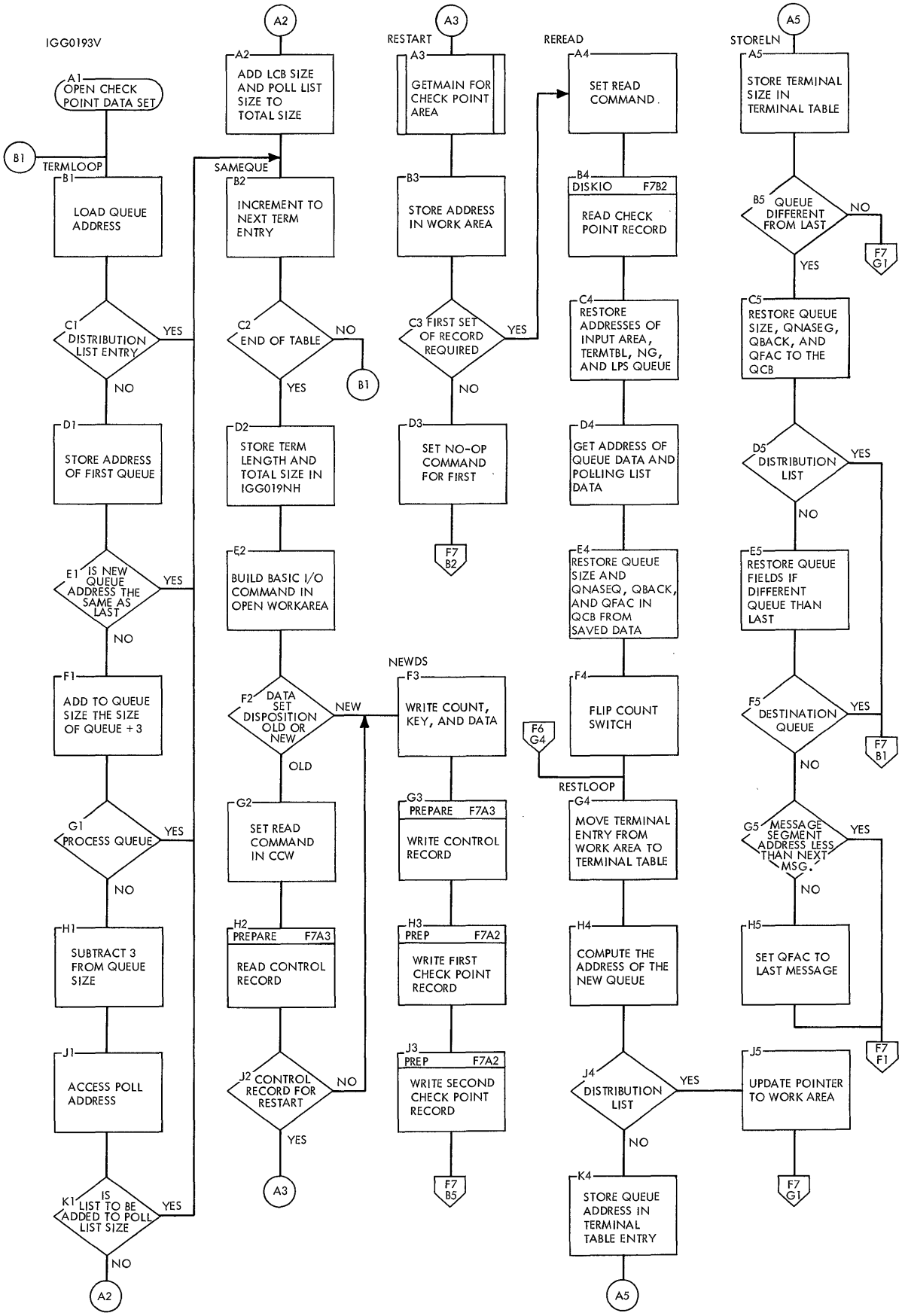
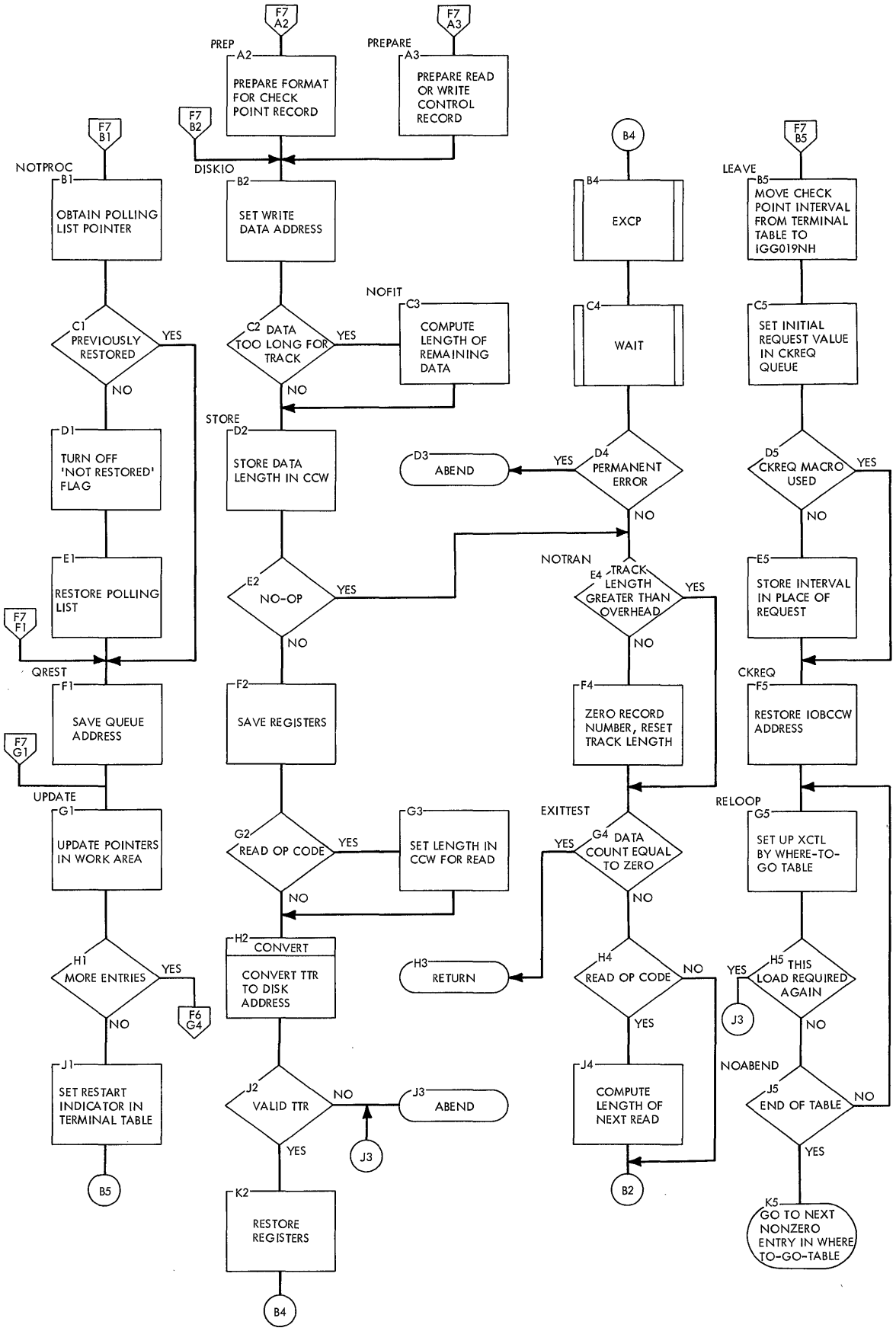
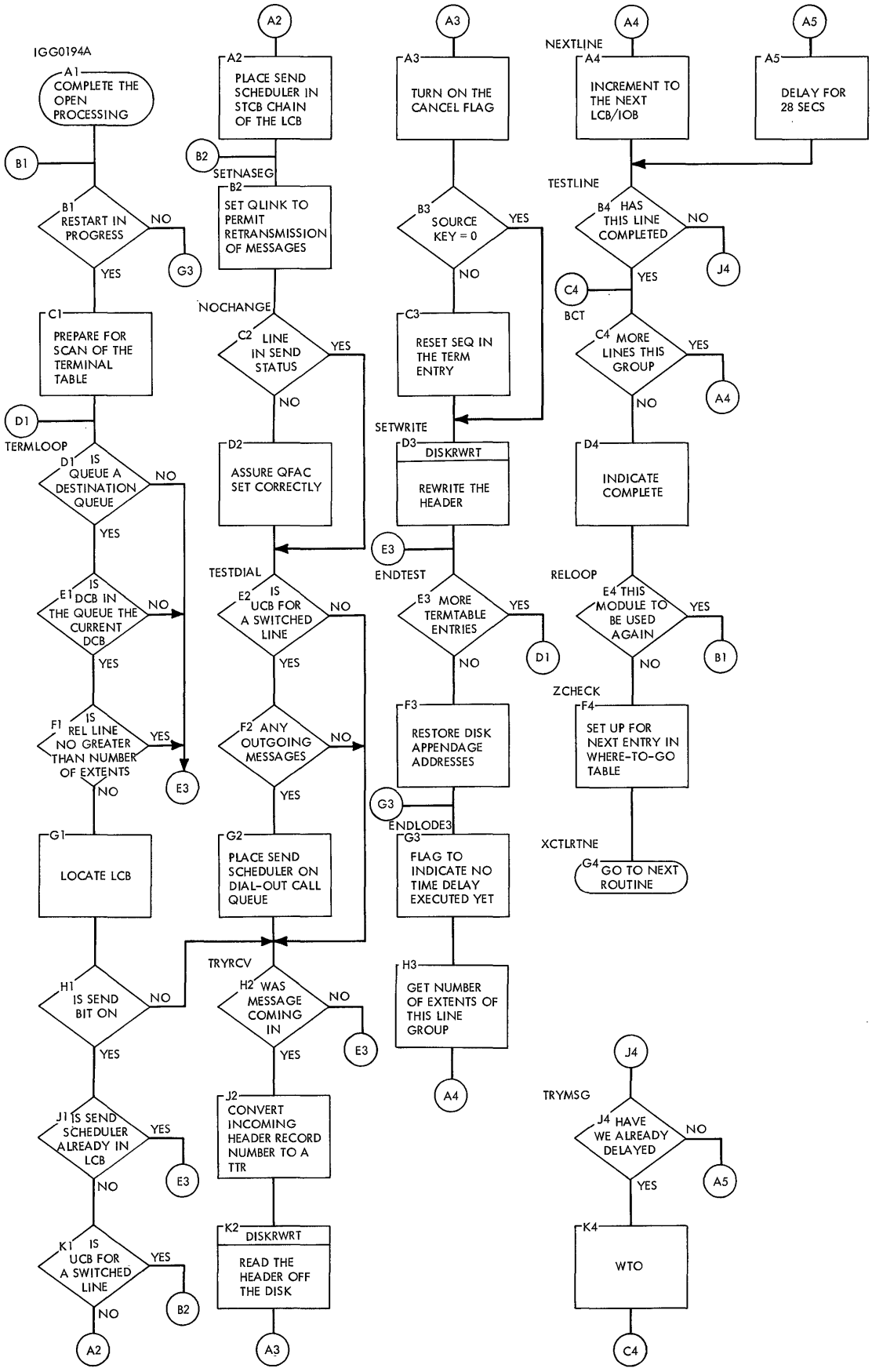


Chart F7. OPEN Checkpoint Data Set Routine (Continued)



● Chart F8. QTAM Open Line Group Load 4



● Chart F9. Close Process Queue Load 2

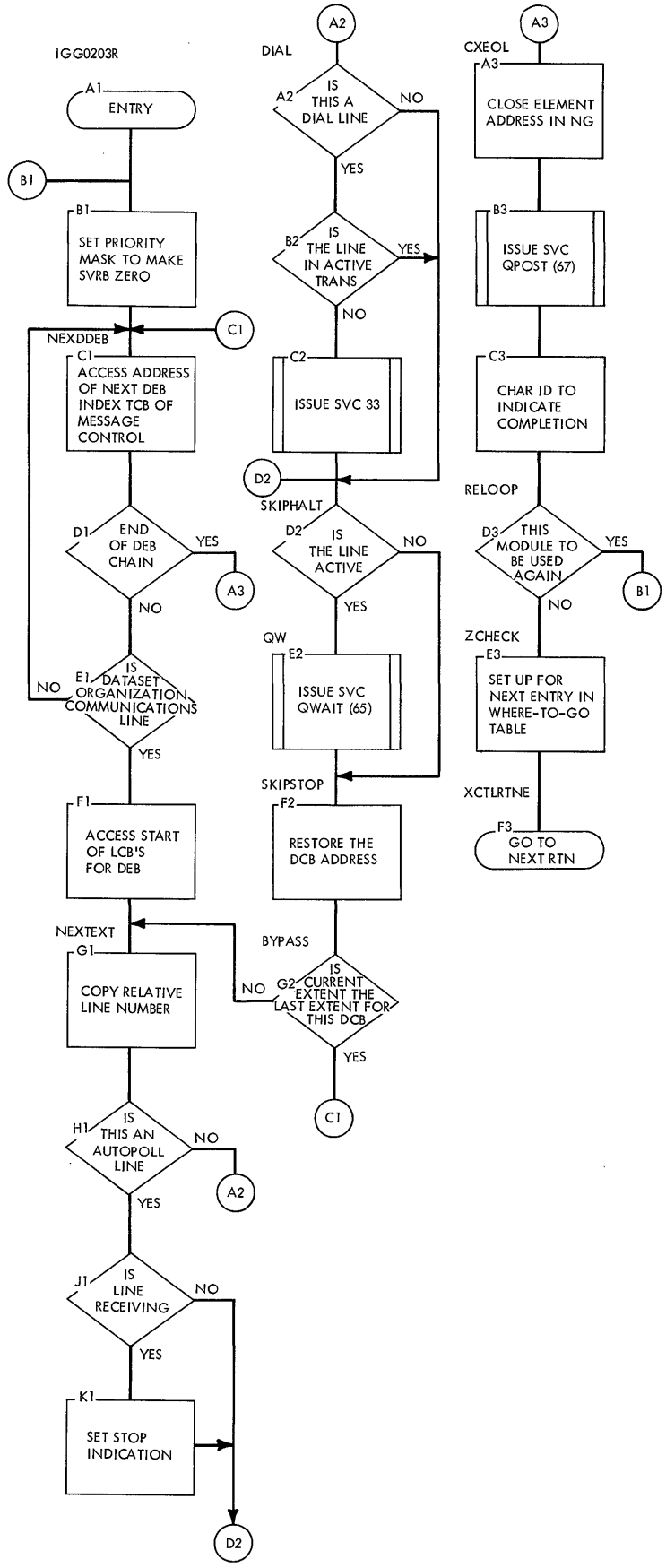


Chart FA. Checkpoint Routine

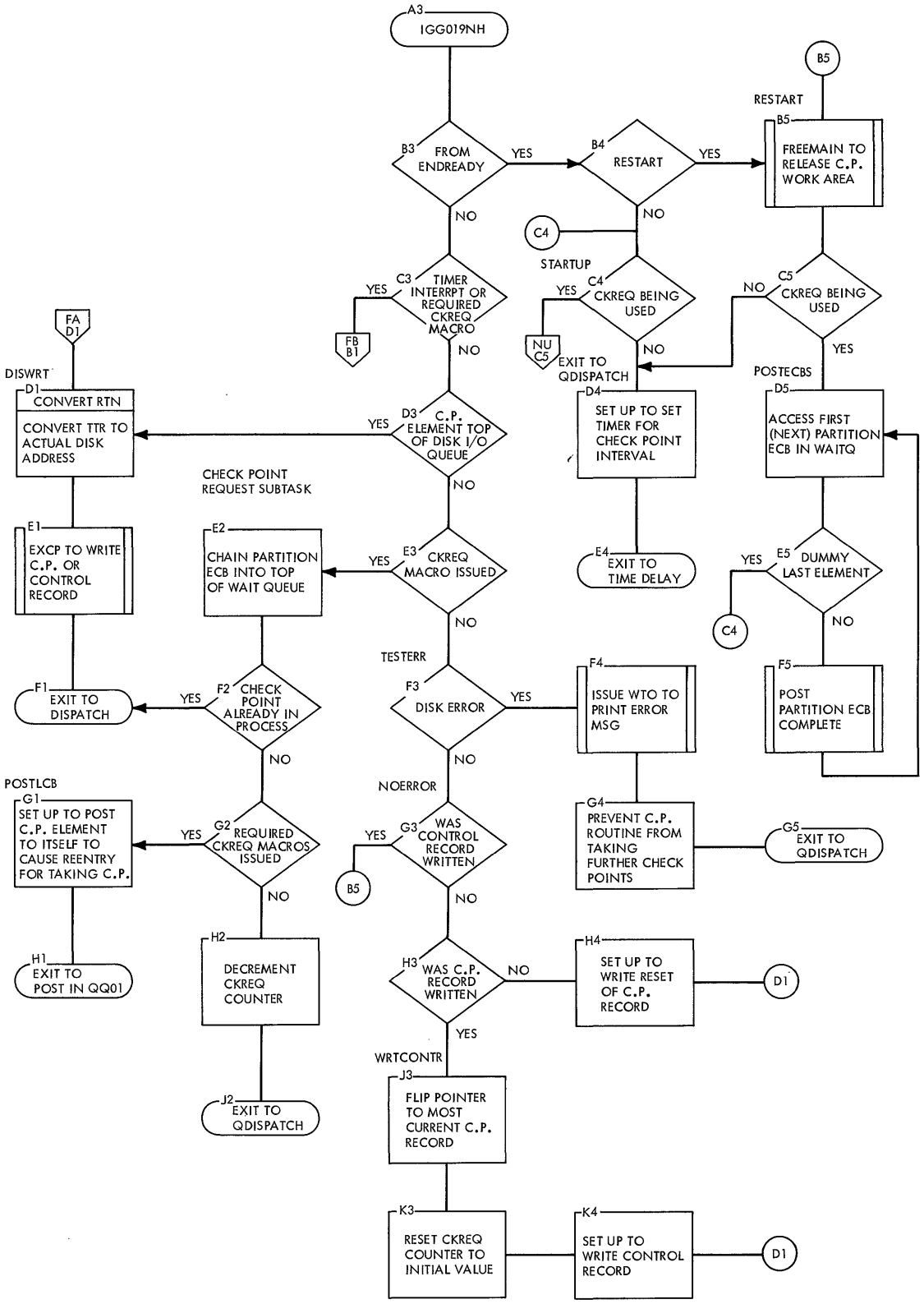


Chart FB. Checkpoint Routine (Continued)

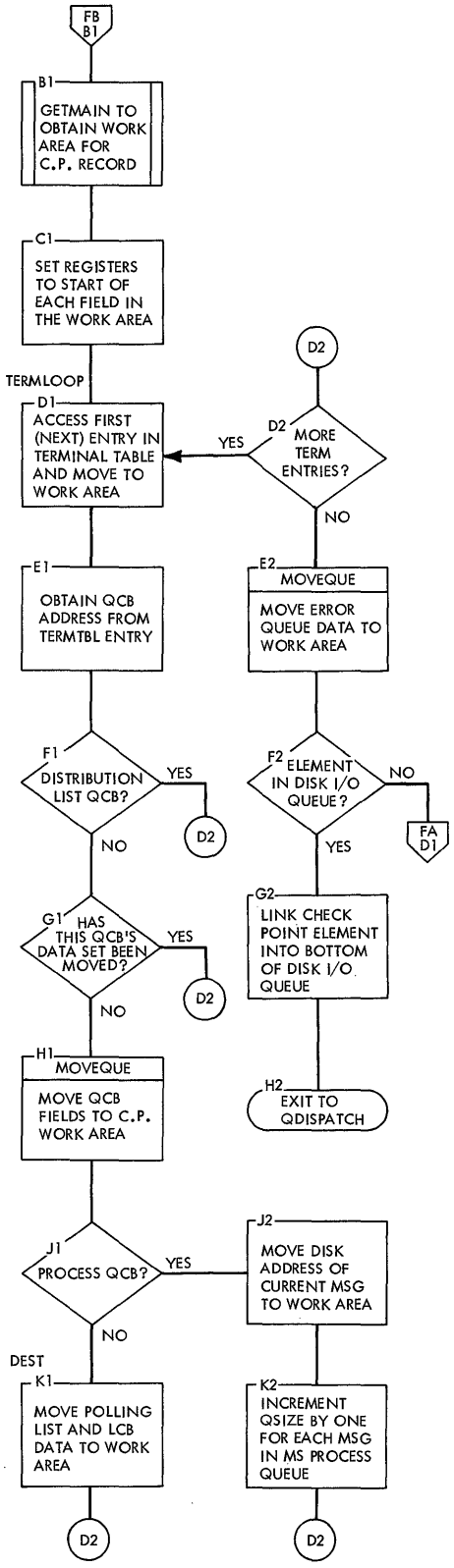


Chart NU. QTAM Nucleus (1 of 2)

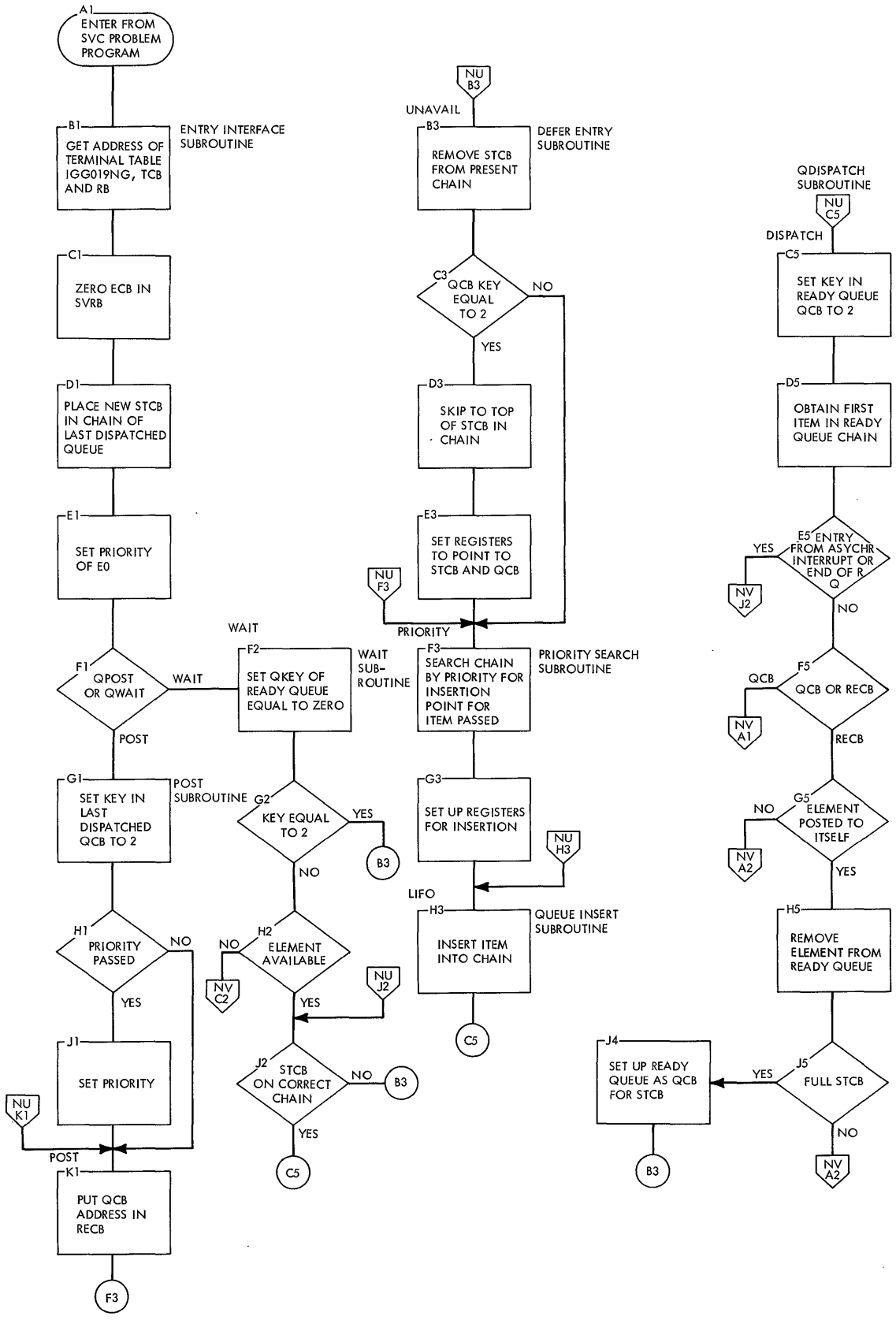


Chart NV. QTAM Nucleus (2 of 2)

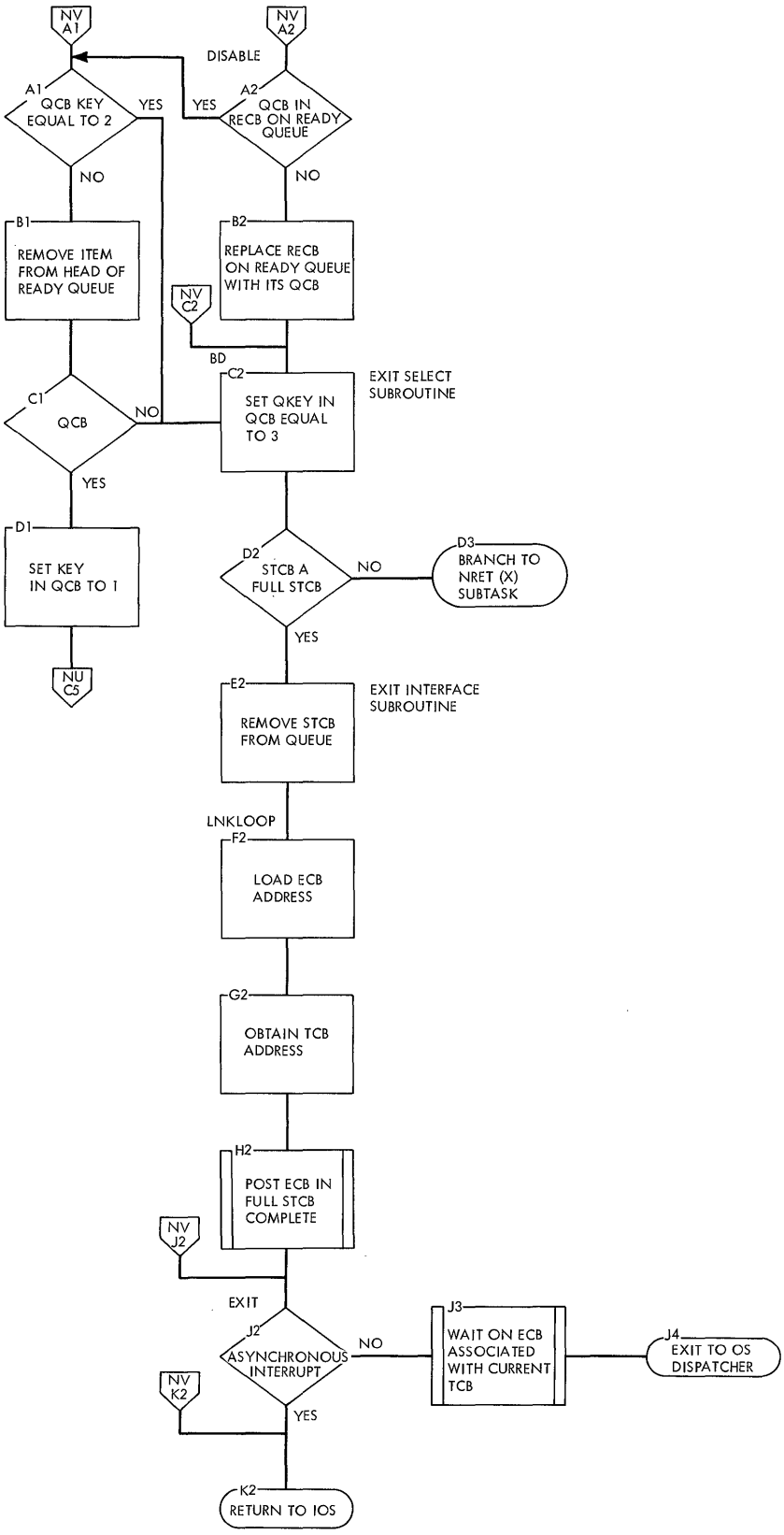


Chart QA. Terminal Test HDR Analysis Module

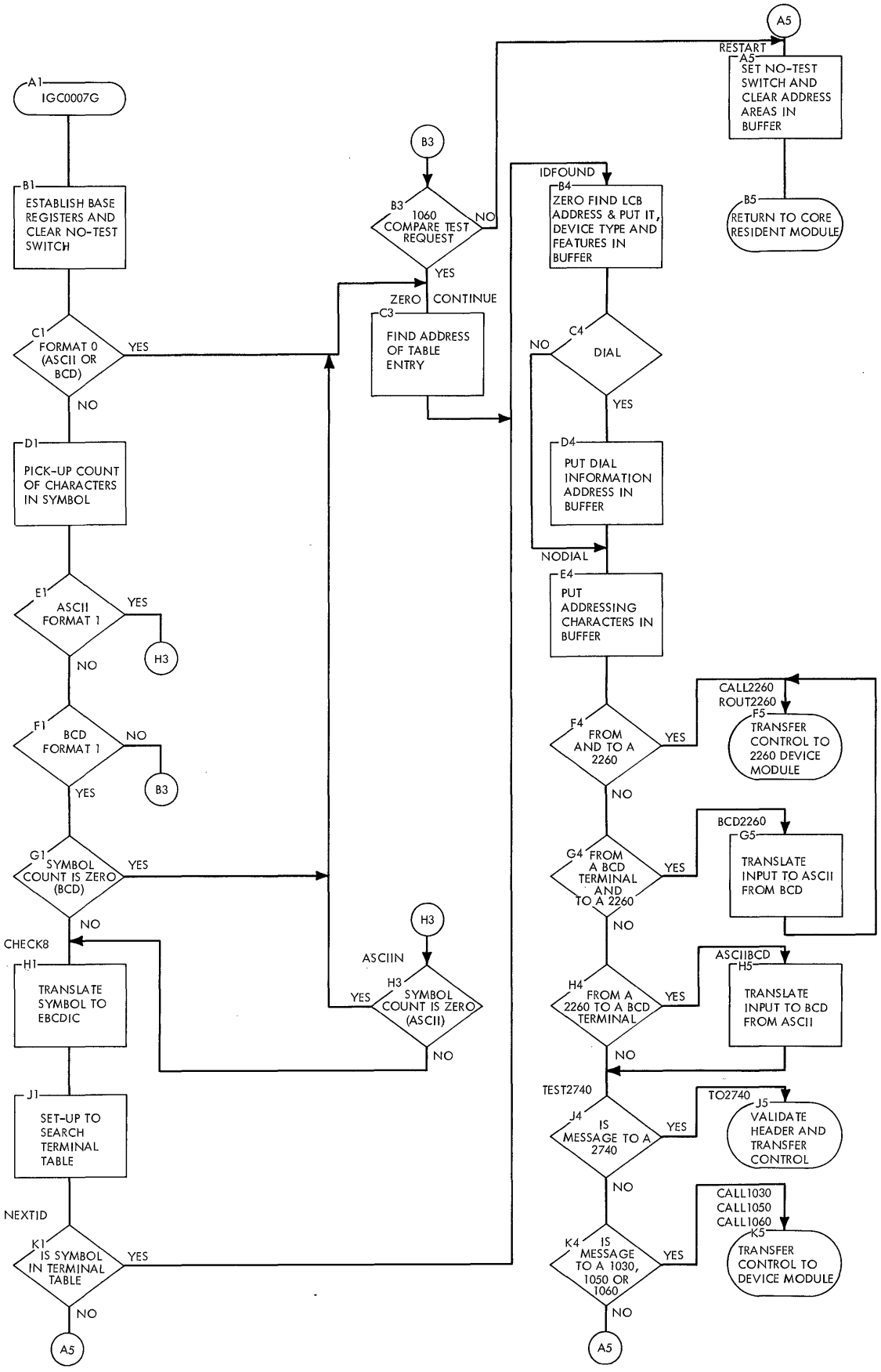
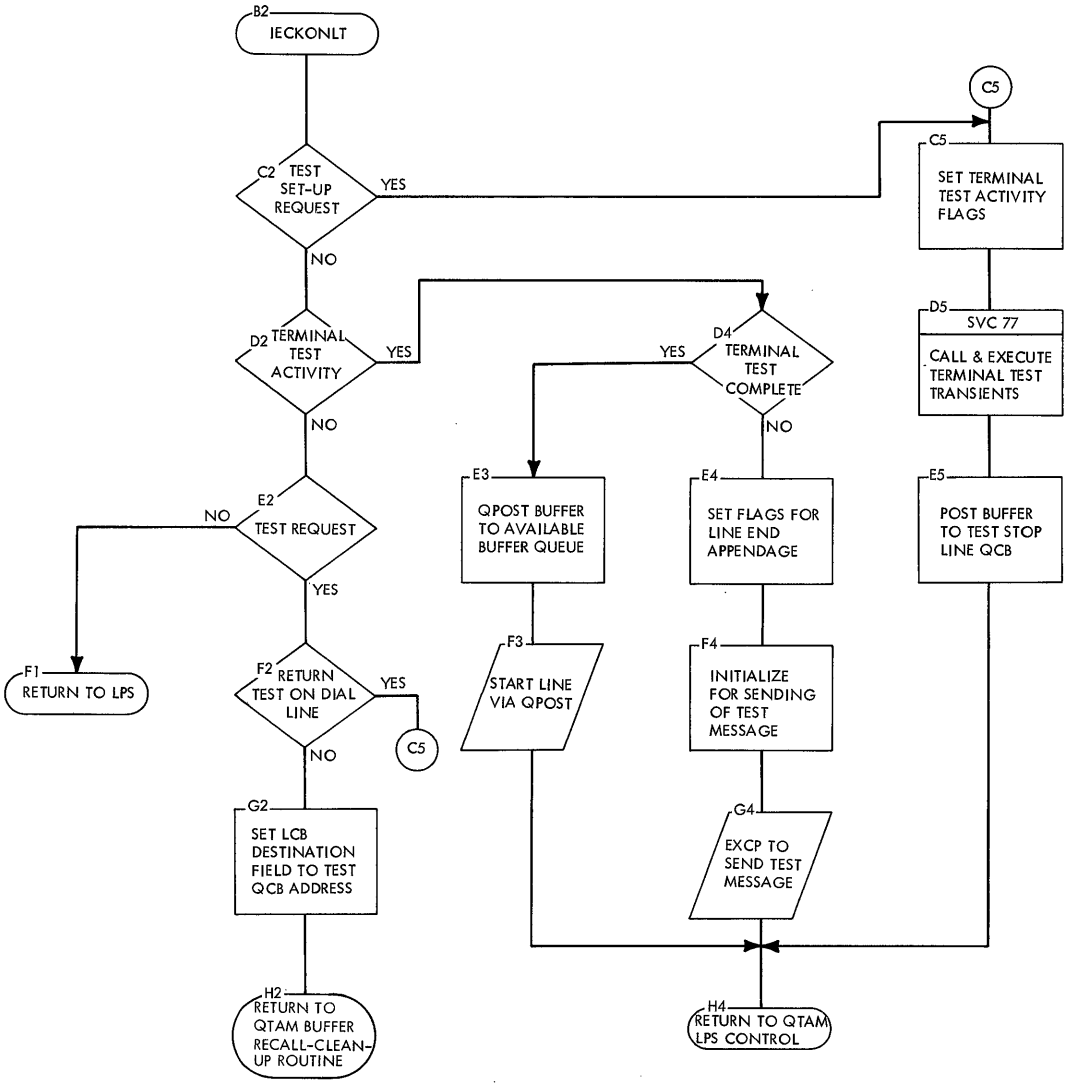


Chart QL. Resident Terminal Test Module



NOTE--THE QTAM BUFFER RECALL-CLEANUP ROUTINE POSTS THE BUFFER TO THE TEST BUFFER ROUTING QCB.

Chart QS. Terminal Subtasks

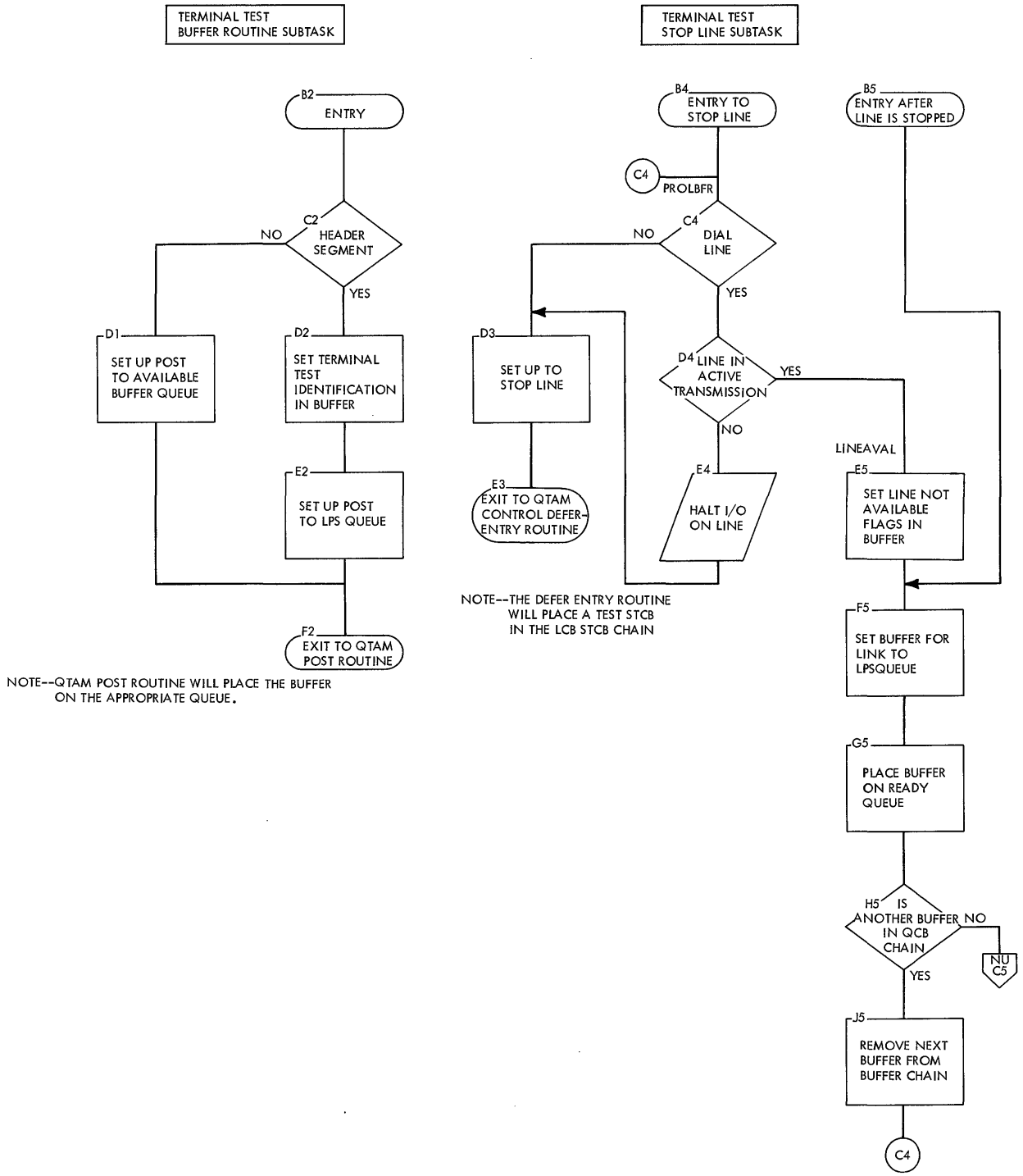


Chart Q3. 1030 Terminal Test Module

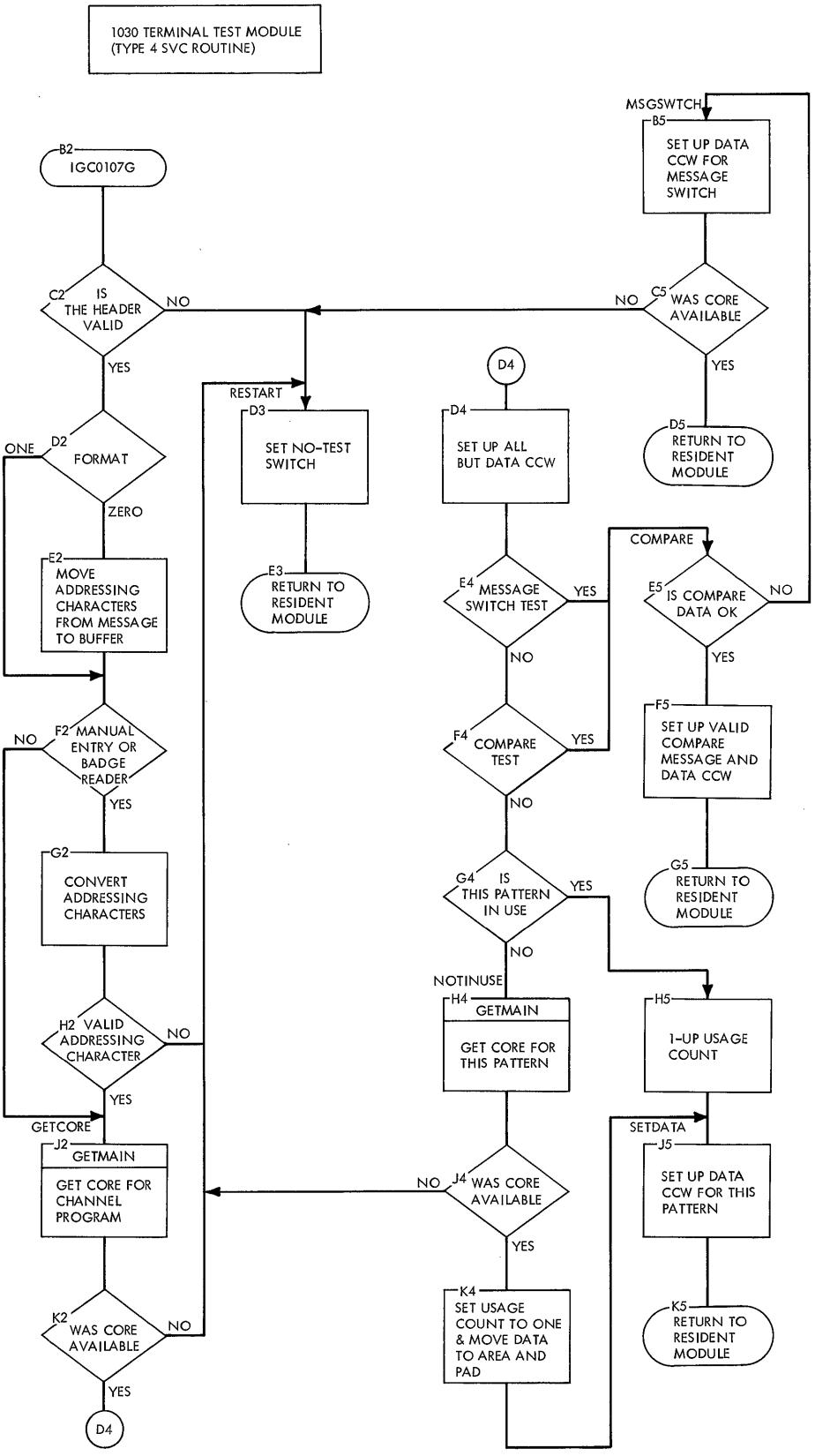


Chart Q4. 2740 Terminal Test Module

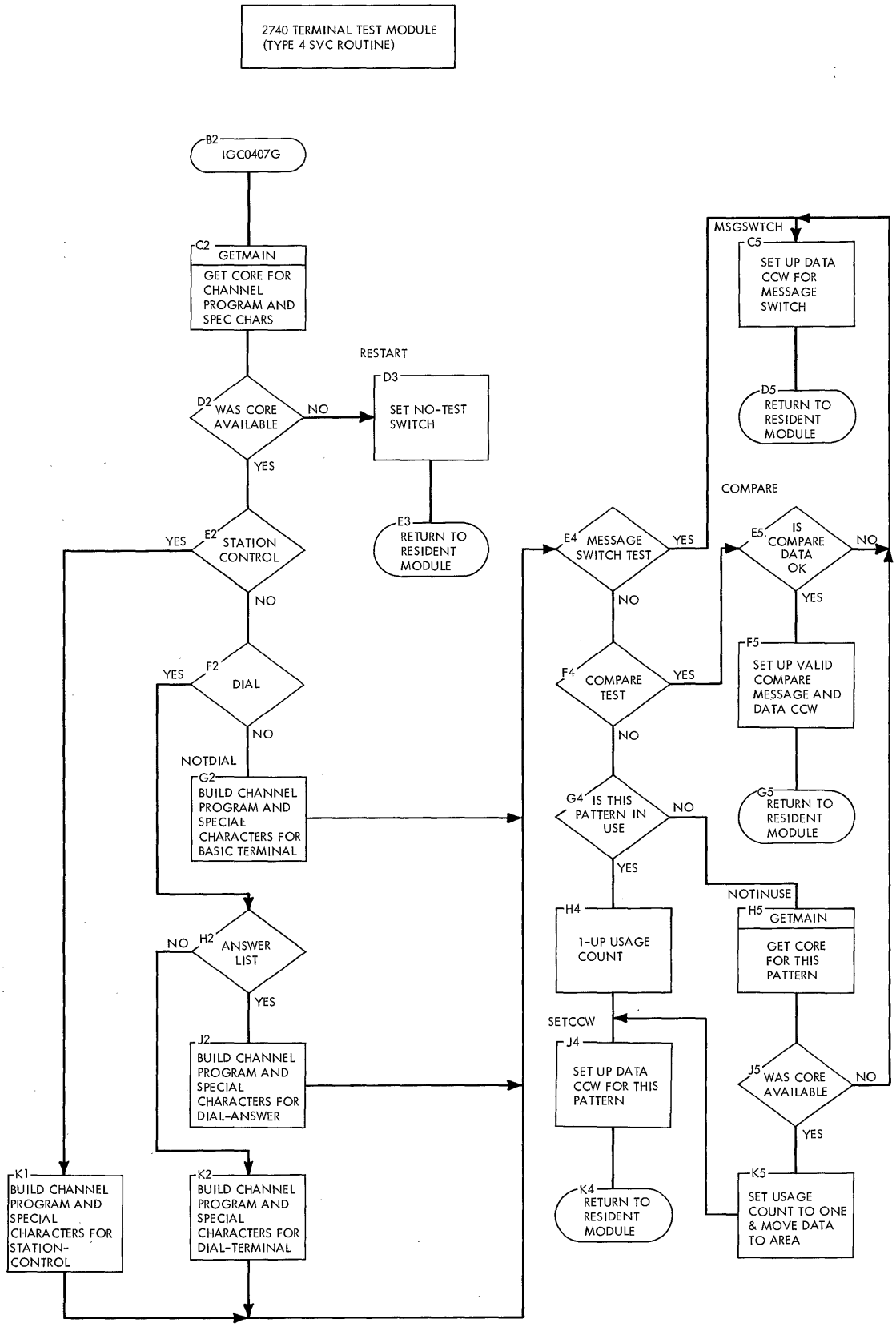


Chart Q5. 1050 Terminal Test Module

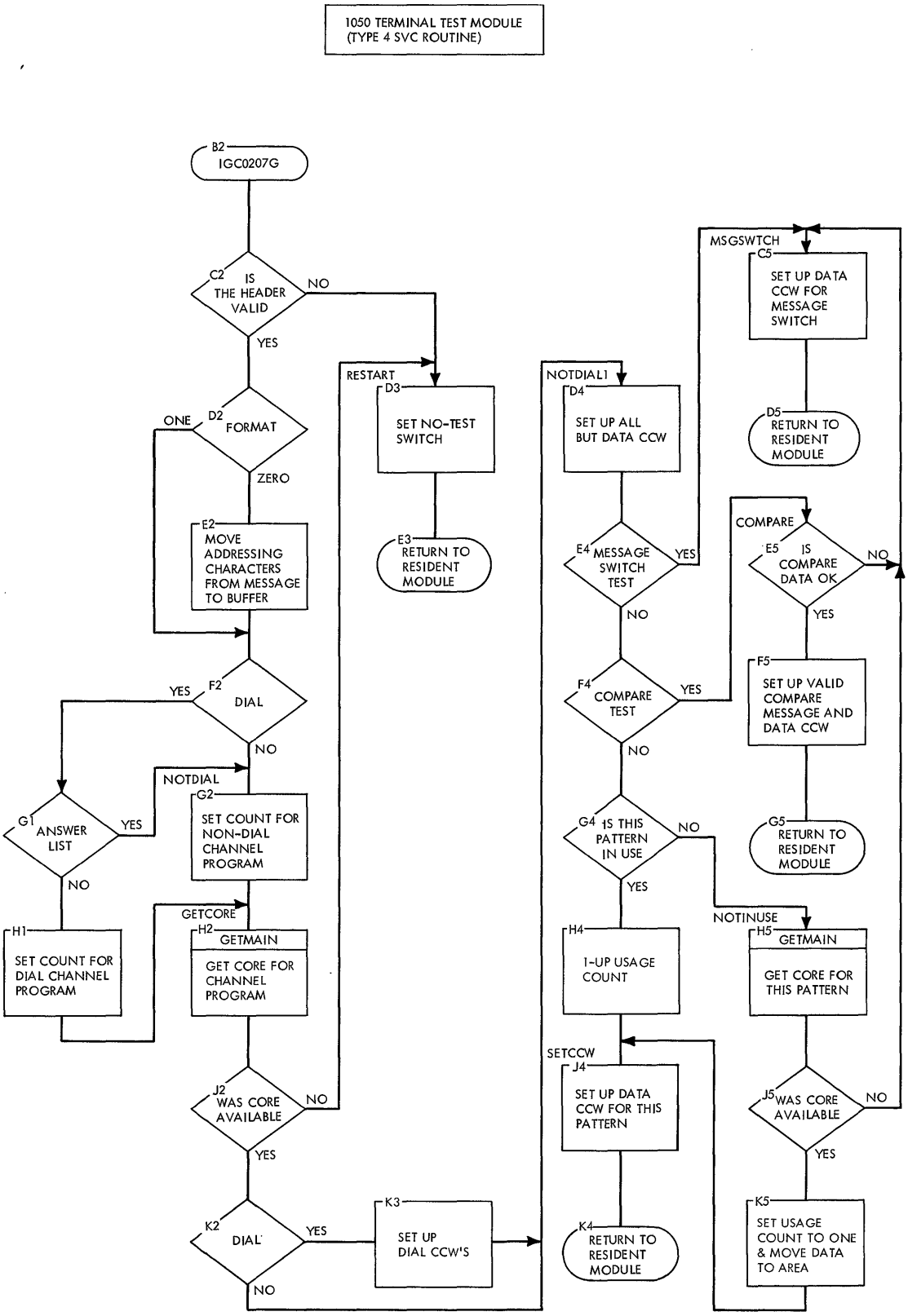


Chart Q6. 1060 Terminal Test Module

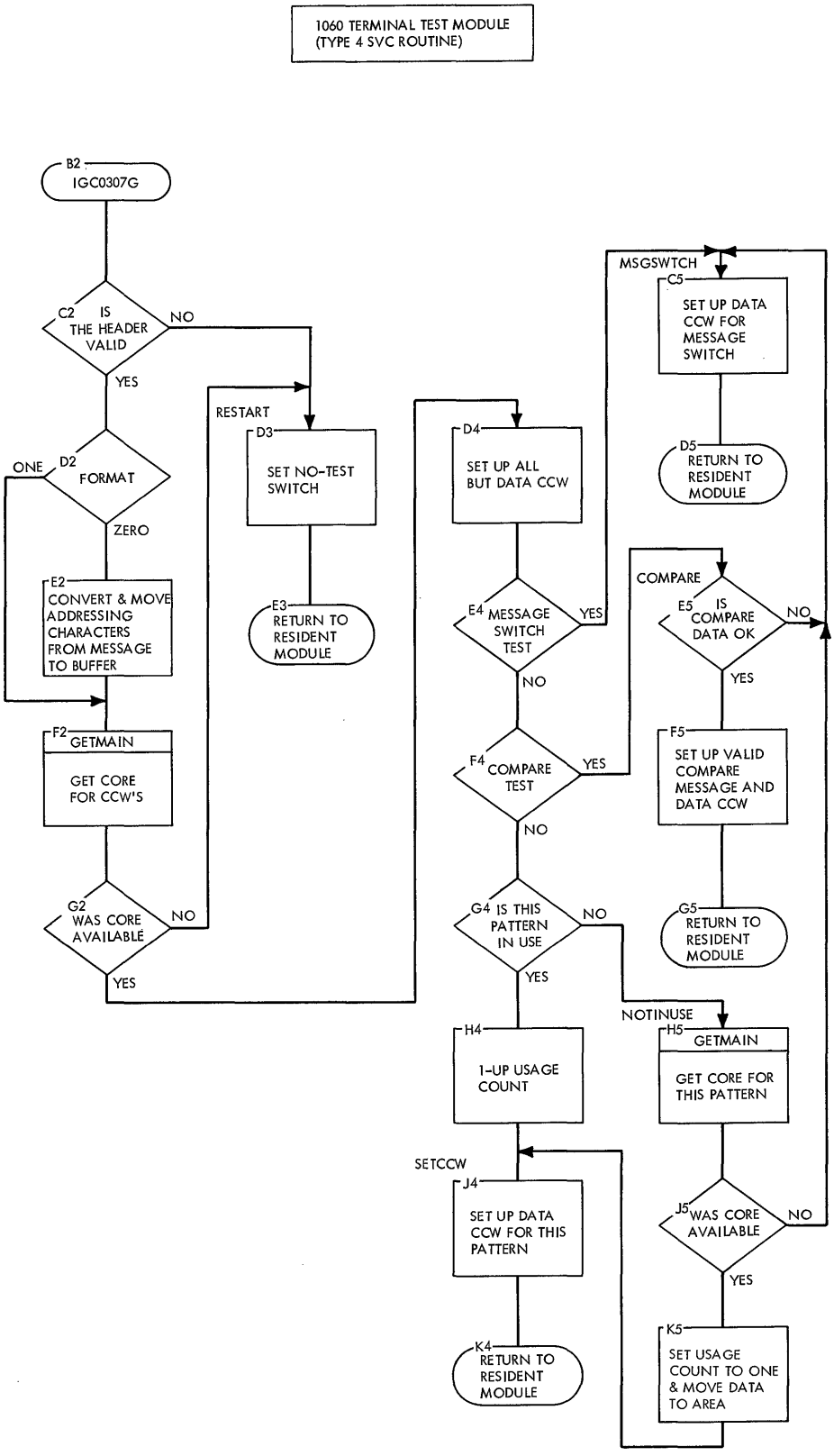
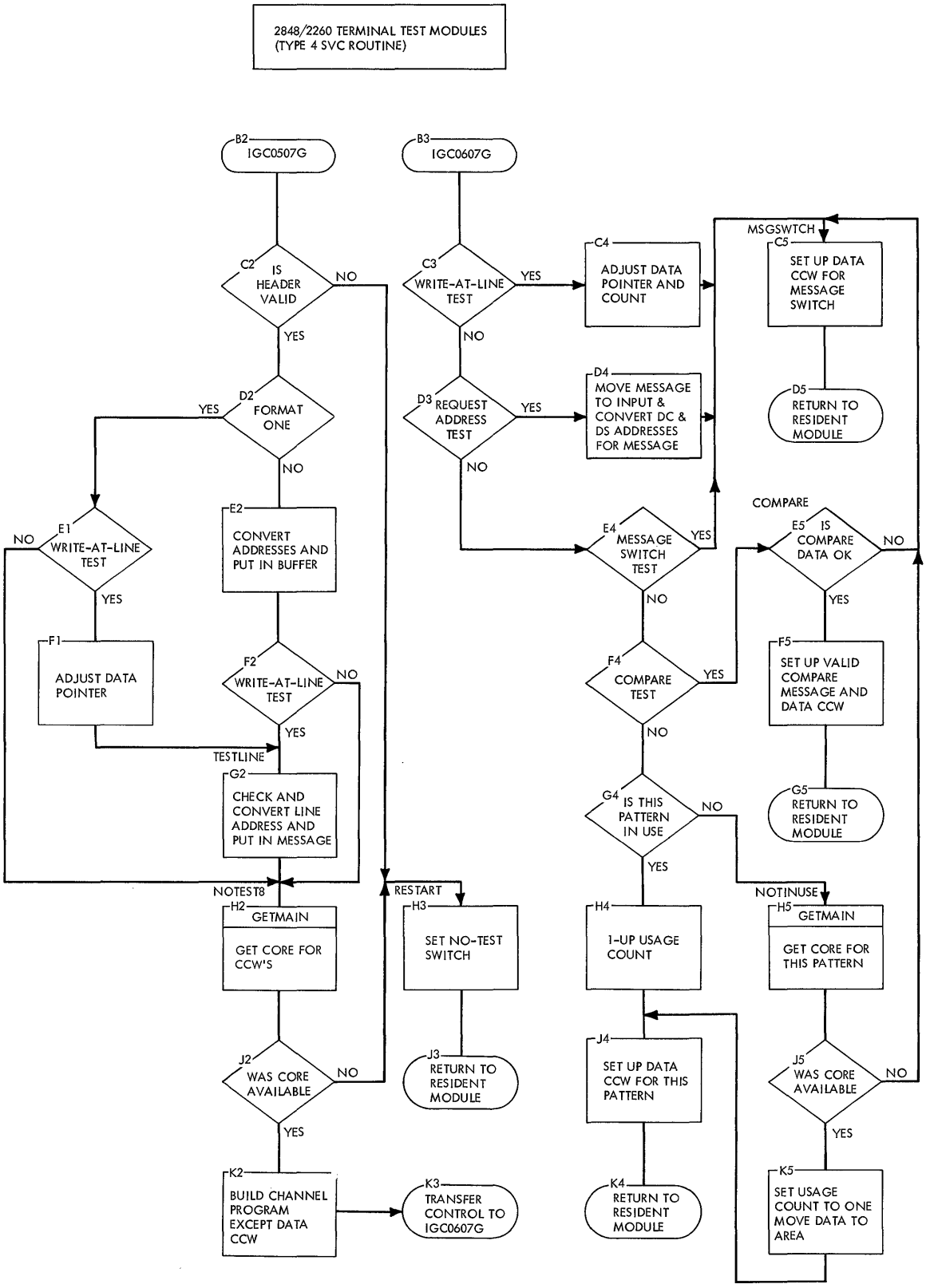
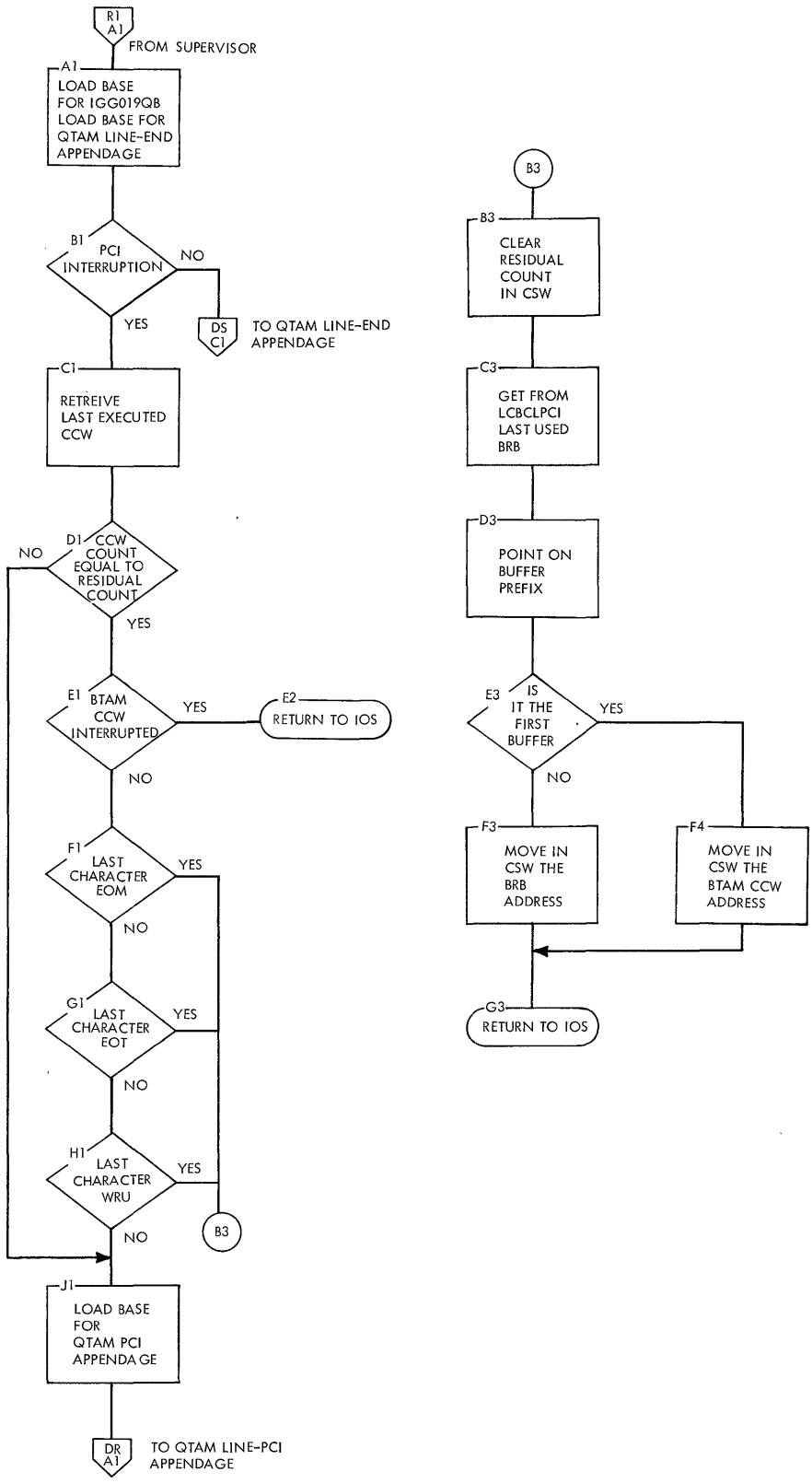


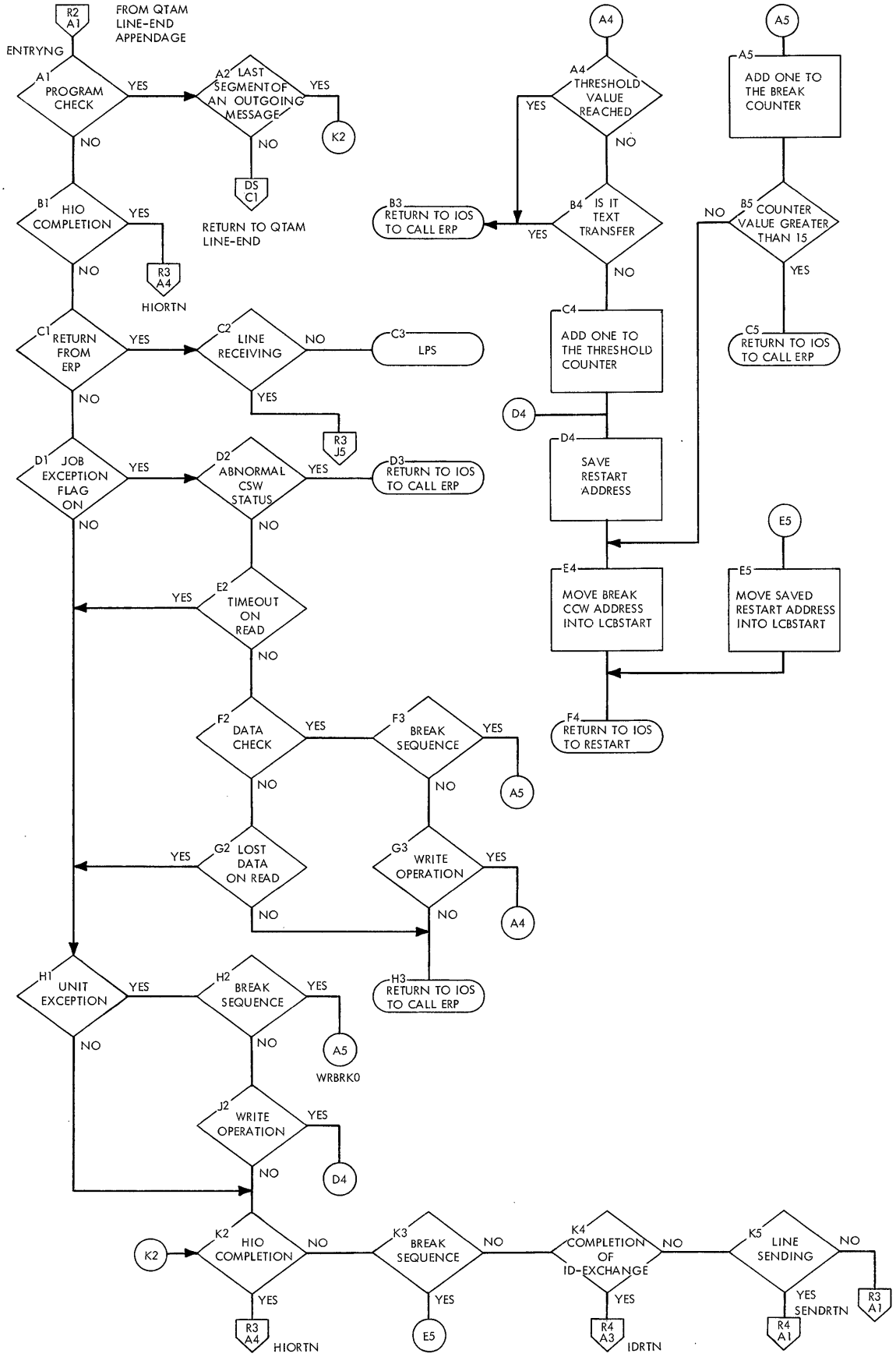
Chart Q8. 2848/2260 Terminal Test Module



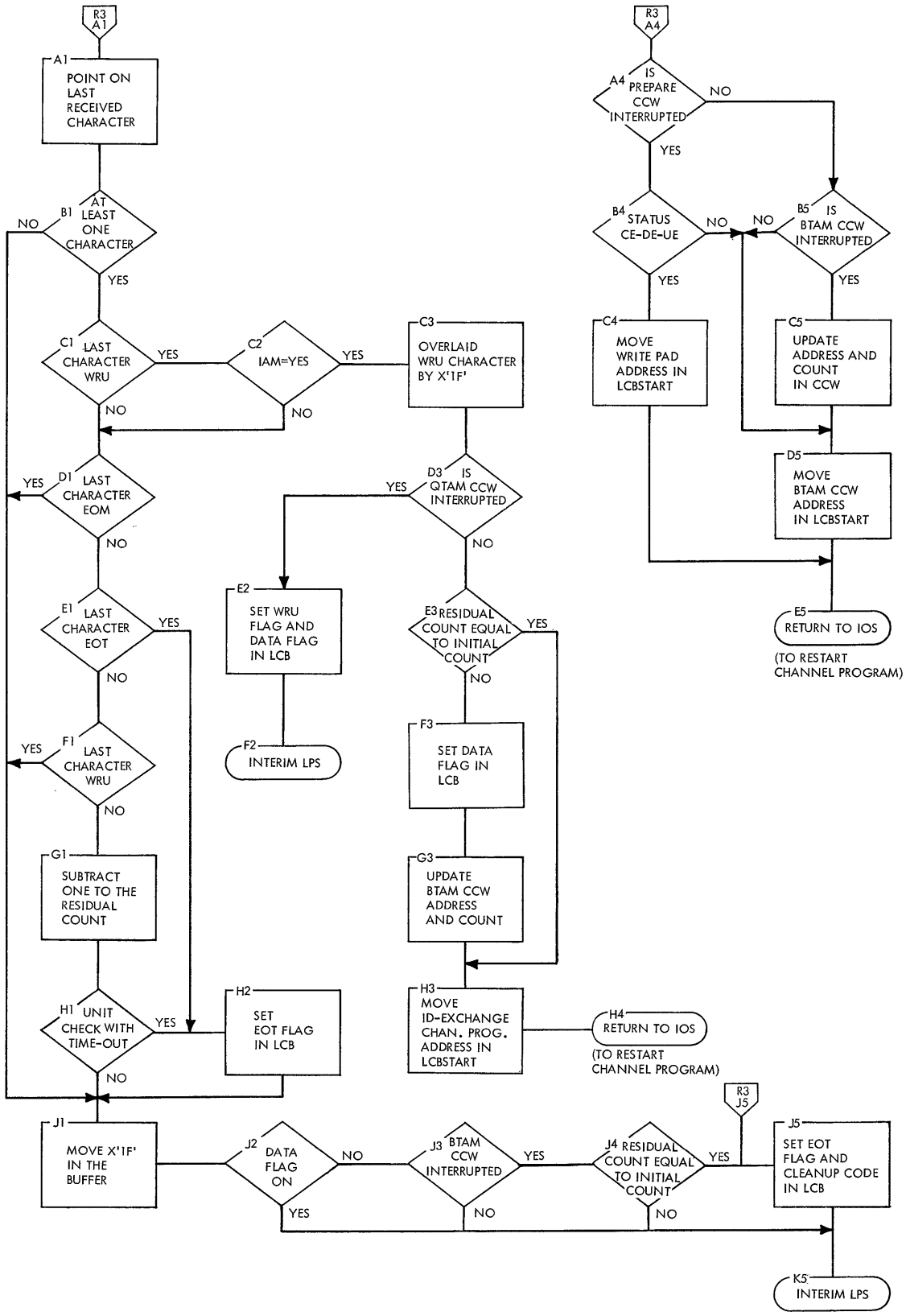
● Chart R1. WTTA Line PCI Appendage Routine



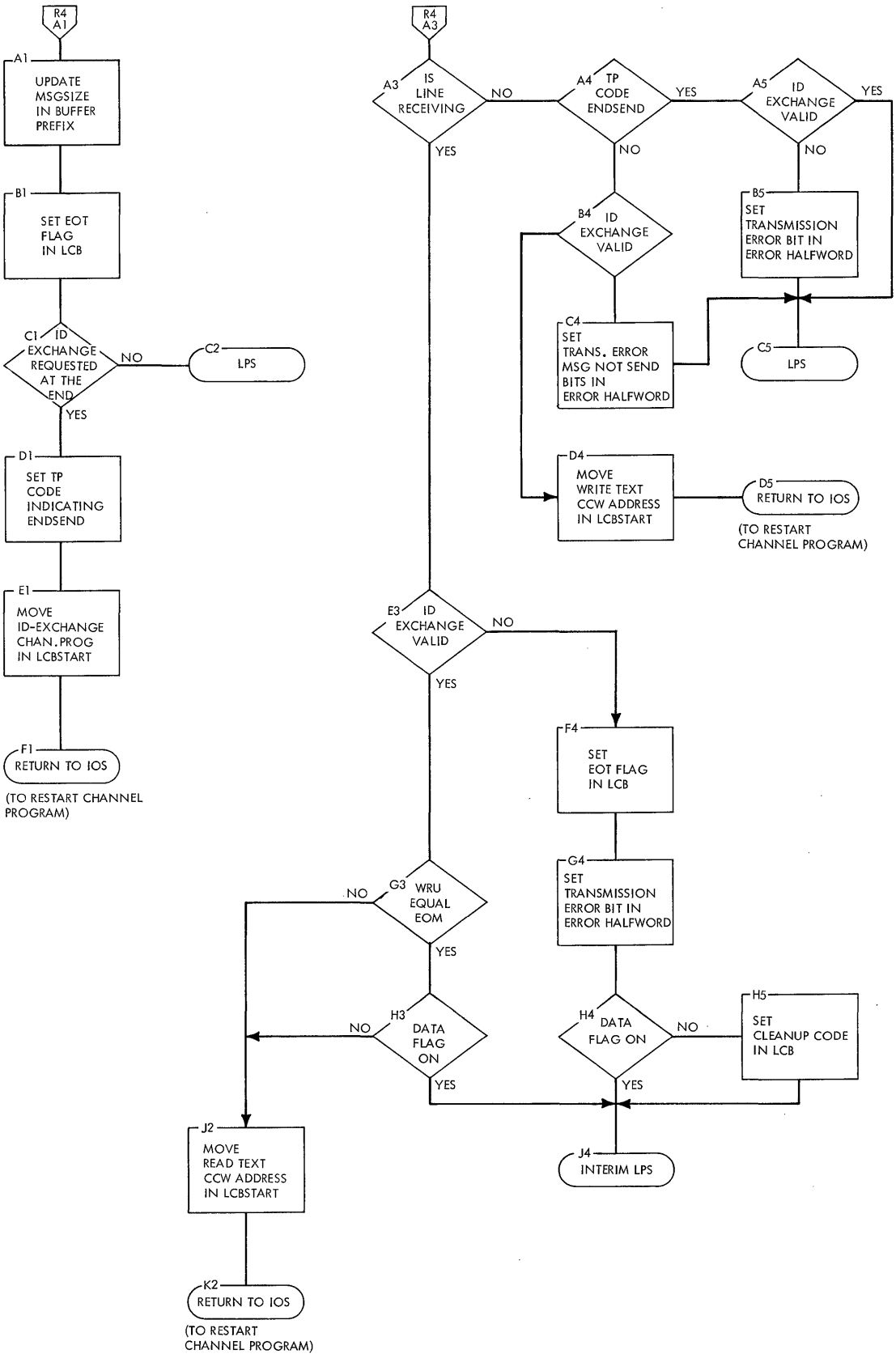
● Chart R2. WTTA Line End Appendage Routine (Part 1 of 3)



● Chart R3. WTTA Line End Appendage Routine (Part 2 of 3)



• Chart R4. WTTA Line End Appendage Routine (Part 3 of 3)



QUEUES

ACTIVE BUFFER REQUEST QUEUE

QCB: Preassembled in the Implementation module; labeled BREQ.

Element Chain: Dynamically created. An element appearing on this chain is an active buffer request block (BRB) representing a BRB ring. The ring is formed by a transfer-in-channel address in each BRB pointing to the next BRB. The element chain, which is distinct from the ring, is formed by the link address in the BRBs in the chain.

STCB Chain: Limited to the STCB for the active buffer request subtask.

ADDITIONAL CCW QUEUE

QCB: Preassembled in the module (IECKPAUS) introduced through the appearance of the PAUSE macro instruction in the message control program; labeled INSERTQ.

Element Chain: Generated in the problem program on expansion of the BUFFER macro instruction; labeled IECKISRT. A chain of special purpose BRBs used to schedule and contain channel commands for the transmission of idle characters.

STCB Chain: May contain the STCB for the LPS subtask. Always ends with the STCB for the queue insert subtask.

AVAILABLE BUFFER QUEUE

QCB: Preassembled in the Implementation module; labeled BUFFER.

Element Chain: Generated in the problem program on expansion of the BUFFER macro instruction; labeled IECKBUFF. A chain of operationally empty buffers.

STCB Chain: Limited to the STCB for the available buffer subtask.

MOVE DATA QUEUE

QCB: Preassembled in the QTAM Implementation module; labeled QMOVE.

Element Chain: Limited to the dummy last element (IECKSTOP).

COMMUNICATIONS LINE QUEUE

QCB: Formed by the first 12 bytes of the line control block, which is created during OPEN.

Element Chain: Limited to a pointer to the LCB itself.

STCB Chain: May contain the STCB for the line's receive scheduling subtask and/or the STCB for the line's send scheduling subtask (or more than one send scheduling STCB if separate queues are maintained for each terminal). Always ends with the STCB for the QEVENT generalized queue handling subtask.

DASD DESTINATION QUEUE

QCB: Generated in the problem program on expansion of a TERM macro instruction; labeled QUEUEn, where "n" is a sequence number reflecting the number of TERM and/or PROCESS macro instructions previously encountered during assembly.

Element Chain: The element chain pointer in a DASD destination QCB is the relative record number of the header segment of the first message in the queue of messages, on the direct access storage device, for the destination. In the message chain, each header segment is linked to the next header segment and the preceding header segment by means of internal control fields. Text segments, which are also on the direct access storage device, are linked to each other and to the header segment to which they relate, through self contained DASD addresses.

Note that the relative record number simply reflects the sequence (1 through n) in which header segments were encountered. This number is subsequently converted to a relative DASD address, which, in turn, is converted to an actual DASD address.

STCB Chain: May contain the STCB for the destination line's send scheduling subtask. Always ends with the STCB for the DASD destination subtask.

DISK INPUT/OUTPUT QUEUE

QCB: Preassembled in the Implementation module; labeled DISK.

Element Chain: Dynamically created. A chain of BRBs (containing channel command words) for direct access Read operations, intermixed with full buffers, to be written on the DASD.

STCB Chain: Limited to the STCB for the disk input/output subtask.

DISTRIBUTION LIST QUEUE

QCB: Preassembled in the module (IECKDLST) introduced by the appearance of the LIST macro instruction in the problem program.

Element Chain: Limited to the dummy last element labeled IECKSTOP. No element chain is developed. Elements related to a distribution list are immediately transferred to the DASD destination queue for the first terminal in the distribution list.

STCB: Limited to the STCB for the distribution list subtask.

INACTIVE BRB QUEUE

QCB: Preassembled in the Implementation module; labeled AVREQ.

Element Chain: Generated in the problem program on expansion of the BUFFER macro instruction; labeled IECKAVRQ. A chain of BRBs of which the third and fourth fullwords are effectively empty.

STCB Chain: May contain the STCB for the line's receive scheduling subtask and/or the STCB for the line's send scheduling subtask (or more than one send scheduling STCB if a separate queue is maintained for each terminal). Always ends with the STCB for the queue insert subtask.

INTERIM LPS QUEUE

QCB: Preassembled in the Implementation module; labeled INTLRM.

Element Chain: Limited to the dummy last element labeled IECKSTOP. Elements are immediately transferred to the LPS queue.

STCB Chain: Limited to the STCB for the interim LPS subtask.

TIME QUEUE

QCB: Preassembled in the QTAM Implementation module; labeled TIMEQ.

Element Chain: Dynamically created. An element appearing on this chain is an LCB waiting for an interrupt from the TIMER.

LPS QUEUE

QCB: Preassembled in the Implementation module; labeled LPS.

Element Chain: Dynamically created. A chain of empty buffers, to be used for messages coming in from terminals, interspersed with message-filled buffers to be processed by the LPS routine.

STCB Chain: May contain the STCB for the LPS subtask. Always ends with the STCB for the queue insert by priority subtask.

DASD PROCESS QUEUE

QCB: Generated in the problem program on expansion of the PROCESS macro instruction; labeled QUEUEn, where "n" is a sequence number reflecting the number of TERM and/or PROCESS macro instructions previously encountered during assembly.

Element Chain: (Refer to the element chain description for the DASD destination QCB).

STCB Chain: May contain the STCB for the process queue's Get scheduling subtask. Always ends with the STCB for the DASD destination subtask.

RETURN BUFFER QUEUE

QCB: Preassembled in the Implementation module; labeled GETRET.

Element Chain: Limited to the dummy last element labeled IECKSTOP. Buffers returned from a GET are immediately transferred to the available buffer queue.

STCB Chain: Limited to the STCB for the return buffer subtask.

COPY CLEAR QUEUE

QCB: Preassembled in the Operator Control routine; labeled COPYCLR.

Element Chain: There is no element chain as this QCB is always posted to itself.

STCB Chain: Limited to the STCB for the Copy Clear subroutine in the Operator Control routine.

CHANGE QUEUE

QCB: Preassembled in the Operator Control routine; labeled CHANGE.

Element Chain: There is no element chain as this QCB is always posted to itself.

STCB Chain: Limited to the STCB for the Change subroutine in the Operator Control routine.

STOP QUEUE

QCB: Preassembled in the Operator Control routine; labeled STOP.

Element Chain: There is no element chain as this QCB is always posted to itself.

STCB Chain: Limited to the STCB for the Stop subroutine in the Operator Control routine.

STOP4 QUEUE

QCB: Preassembled in the Operator Control routine; labeled STOP4.

Element Chain: There is no element chain as this QCB is always posted to itself.

STCB Chain: Limited to the STCB for the Stop 4 subroutine in the Operator Control routine and is used by the Operator Awareness routine.

STOP THE LINE QUEUE

QCB: Preassembled in the Operator Control routine; labeled STOP2.

Element Chain: Dynamically created. The element chain consists of buffers that are used to transmit operator control messages.

STCB Chain: Limited to the STCB for the Stop 3 subtask in the Operator Control routine.

GET SVC 1 QUEUE

QCB: Preassembled in the Operator Control routine; labeled GETSVC1.

Element Chain: Dynamically created. Elements are transferred to the LPS queue.

STCB Chain: Limited to the STCB for the Get SVC 2 subtask in the Operator Control routine.

CHECKPOINT QUEUE

QCB: Preassembled in IGG019NH module in a dummy checkpoint LCB; labeled START.

Element Chain: This QCB has no element chain as it is posted to itself.

STCB Chain: Limited to the STCB for the Checkpoint subtask. Always ends with a dummy end element.

CHECK REQUEST QUEUE

QCB: Preassembled in IGG019NH module; labeled CKQUE.

Element Chain: Dynamically created. The elements are ECBs of partitions waiting for a checkpoint to be taken.

STCB Chain: Limited to the check request subtask in the Checkpoint/Restart routine. Always ends with a dummy end element.

LINE CHANGE QUEUE

QCB: Preassembled in IECKLNCH module; labeled QUEUE.

Element Chain: Dynamically created. An element appearing on this chain is an LCB for a line that is to be started.

STCB Chain: Limited to the STCB for a subtask in the Line Change routine.

DIAL OUT-CALL QUEUE

QCB: Formed in the DEB during OPEN for each line group of dial lines.

Element Chain: None.

STCB Chain: The chain may consist of send scheduler STCBs for messages that were sent to terminals that were busy.

SUBTASKS

ACTIVE BUFFER REQUEST SUBTASK

STCB: Preassembled in the Implementation module; labeled BREQENQ.

Program Entry: Enters the Active Buffer Request routine at BREQENQ+6.

AVAILABLE BUFFER SUBTASK

STCB: Preassembled in the Implementation module; labeled BFRENQ.

Program Entry: Enters the Available Buffer routine at BFRENQ+6 (alternate label BFRREQ).

DASD DESTINATION SUBTASK

STCB: Preassembled in the Implementation module; labeled IECKMQ.

Program Entry: Enters the DASD Destination routine at IECKMQ+6.

DISK INPUT/OUTPUT SUBTASK

STCB: Preassembled in the Implementation module; labeled DISKENQ.

Program Entry: Enters the Disk Input/ Output routine at DISKENQ+6.

DISTRIBUTION LIST SUBTASK

STCB: Preassembled in module IECKDLQT, located at IECKDLQT+8.

Program Entry: Enters the module at IECKDLQT+14.

GET SCHEDULING SUBTASK

STCB: Preassembled within the Implementation module; labeled GETSCH.

Program Entry: Enters the GET Scheduler routine at GETSCH+6.

LPS SUBTASK

STCB: The STCB for the LPS subtask is transient and is dynamically formed within the supervisor request block created on issuance of an SVC 65 or 67 by the subtask.

Program Entry: Activation of the LPS subtask causes the message control program to be re-entered at the instruction following the supervisor call.

QUEUE INSERT SUBTASK

STCB: Preassembled in the Implementation module; labeled QLIFO.

Program Entry: Enters the Implementation module at QLIFO+6 (an unconditional branch to the Queue Insert subroutine (LIFO) in the QTAM control program).

QUEUE INSERT BY PRIORITY SUBTASK

STCB: Preassembled in the Implementation module; labeled QPRIRTY.

Program Entry: Enters the Implementation module at QPRIRTY+6 (an unconditional branch to the Queue Insert subroutine by the Search Priority subroutine (PRIORITY) of the QTAM control program).

QDISPATCH SUBTASK

STCB: Preassembled in the Implementation module; labeled QEVENT.

Program Entry: Enters the Implementation module at QEVENT+6 (an unconditional branch to the Qdispatch subroutine (DISPATCH) of the QTAM control program).

RECEIVE SCHEDULING SUBTASK

STCB: There is one receive scheduling subtask for each line; the STCB for the subtask is contained in the third and fourth fullwords of the corresponding line control block.

Program Entry: All receive scheduling subtasks enter the Receive Scheduler routine at RCVSCH.

RETURN BUFFER SUBTASK

STCB: Preassembled with the Implementation module; located at GETRET+8.

Program Entry: Enters the Return Buffer routine at GETRET+14.

SEND SCHEDULING SUBTASK

STCB: There is one send scheduling subtask for each line or for each terminal, as specified by the user. The STCB for the subtask is contained within the third and fourth fullwords of the QCB for the corresponding DASD destination queue.

Program Entry: All send scheduler subtasks enter the Send Scheduler routine at ENQUEUE.

TIME SUBTASK

STCB: Preassembled within the QTAM Implementation module; labeled TIMEEND.

Program Entry: Enters the End of Poll Time Delay routine at TIMEEND+6.

MOVE DATA SUBTASK

STCB: Preassembled within the QTAM Implementation module; labeled QMOVER.

Program Entry: Enters the Cross Partition Move routine at QMOVER+6.

COPY CLEAR SUBTASK

STCB: Preassembled in the Operator Control routine; labeled COPYCLR1.

Program Entry: Enters the subtask in the Operator Control routine at COPYCLR1+6 to be in supervisory mode.

CHANGE 1 SUBTASK

STCB: Preassembled in the Operator Control routine; labeled CHANGE1.

Program Entry: Enter the subtask in the Operator Control routine at CHANGE1+6 to be in supervisory mode.

STOP 1 SUBTASK

STCB: Preassembled in the Operator Control routine at STOP1+6 to be in the supervisory mode.

Program Entry: Enters the subtask in the Operator Control routine at STOP1+6 to be in supervisory mode.

STOP 3 SUBTASK

STCB: Preassembled in the Operator Control routine; labeled STOP3.

Program Entry: Enters the subtask in the Operator Control routine at STOP3+8.

GETSVC 2 SUBTASK

STCB: Preassembled in the Operator Control routine; labeled GETSVC2.

Program Entry: Enters the subtask in the Operator Control routine at GETSVC2 + 8.

STOP 5 SUBTASK

STCB: Preassembled in the Operator Control routine; labeled STOP5.

Program Entry: Enters the subtask in the Operator Control routine at STOP5+8.

CHECKPOINT SUBTASK

STCB: Preassembled in the Checkpoint/Restart module; labeled TERMTBL.

Program Entry: Enters the subtask in the Checkpoint/Restart routine at CON+2.

CHECK REQUEST SUBTASK

STCB: Preassembled in the Checkpoint/Restart module; labeled CKSTCB.

Program Entry: Enters the subtask in the Checkpoint/Restart routine at CKSTCB+6.

LINE CHANGE SUBTASK

STCB: Preassembled in the Line Change routine; labeled STCB.

Program Entry: Enters the subtask in the Checkpoint/Restart routine at STCB+8.

QDISPATCH SUBTASK

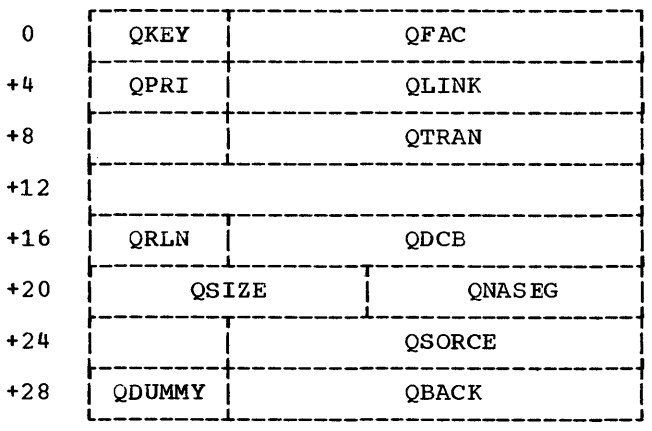
STCB: Preassembled in the Implementation module; labeled QEVENT.

Program Entry: Enters the Implementation module at QEVENT+6. If the LCB indicates a dial line, a switch is set to cause the Activate routine to set up a Write Negative Acknowledgment channel program. A branch is taken to the BRB Ring routine to check the dial out-call queue. If the line is not for a dial line, a branch is taken to the Qdispatch subroutine (DISPATCH) of the QTAM control program.

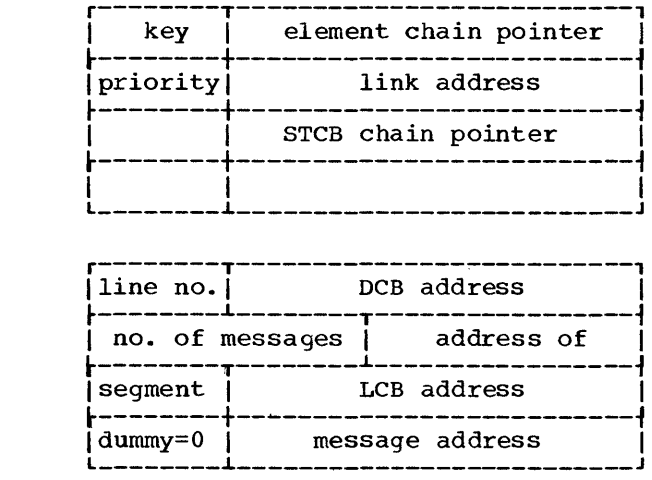
GENERAL CONTROL BLOCK FORMS

QUEUE CONTROL BLOCK

Typical DSECT:



General Form:



Contents:

key: a numeric value (1, 2, or 3) indicating queue status.

1 -- not on ready queue

2 -- not waiting

3 -- waiting

(See Queue Status for more information on key meanings.)

element chain pointer: a pointer to the head of the element control block chain for the queue.

priority: priority of the queue the QCB represents; determines the relative position of the QCB when linked into the ready queue.

link address: a pointer to the next item on the ready queue. This field is meaningful only when the QCB is on the ready queue.

line no.: the relative line number within the line group of the DCB.

DCB address: the address of the DCB associated with this QCB.

no. of messages: the number of messages on the queue to determine the size of the queue.

address of segment: the address of the area into which the next message segment is to be read.

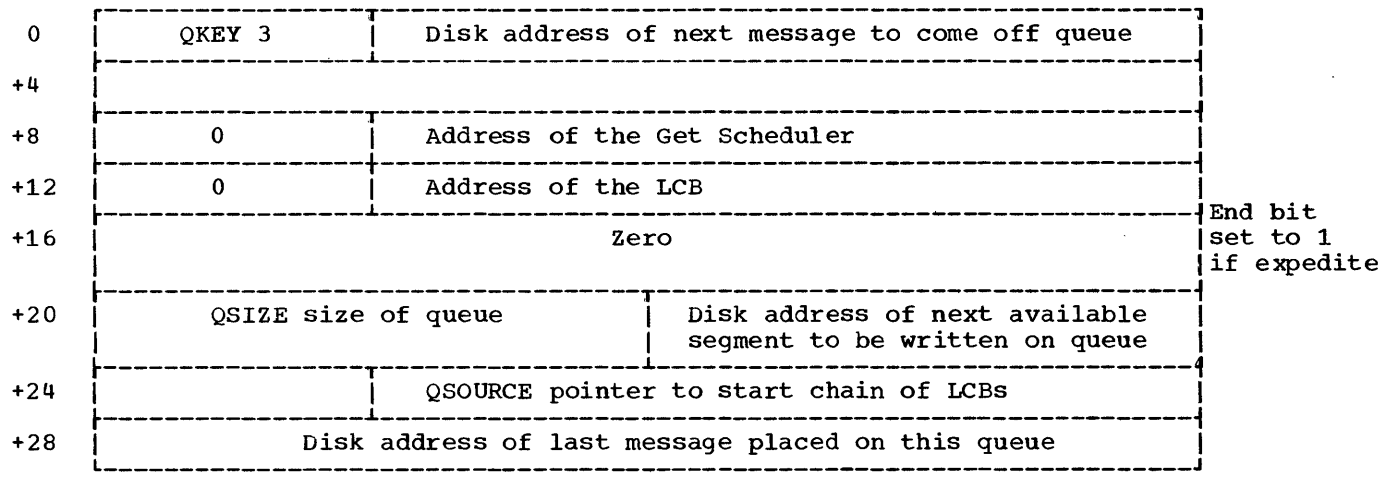
LCB address: the address of the first LCB on the line control block chain.

dummy: always equal to zero.

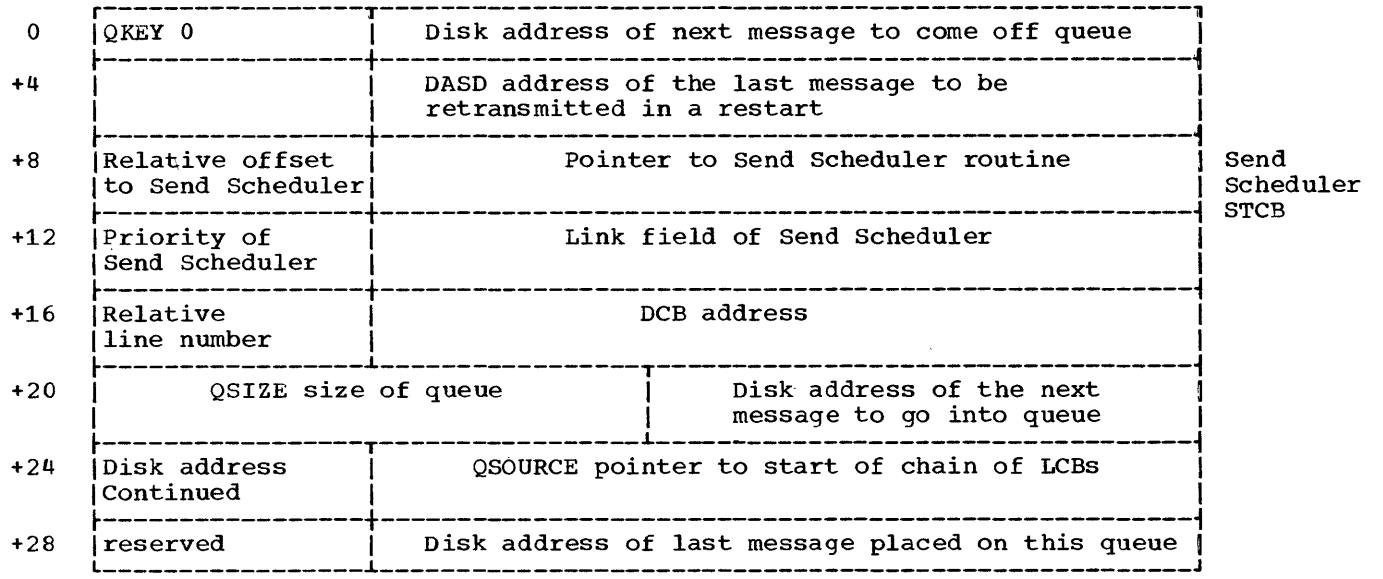
message address: the disk address of the last message placed on this queue.

DASD QCB:

QCB for DASD Process Queue



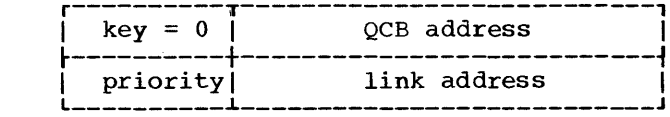
QCB and STCB for DASD Destination Queue



RESOURCE ELEMENT CONTROL BLOCK

Typical DSECT:

General Form:



key: always equal to zero.

QCB address: a pointer to the QCB for the queue to which the element has been posted. This field is meaningful only while the element is on the ready queue, or is being handled by the Qdispatch subroutine after having been encountered on the ready queue.

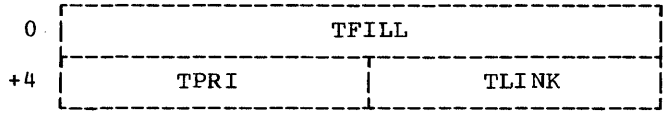
priority: priority of the element that the control block represents. This field determines the relative position of the element when linked into the element chain of a QCB. Priority 255 identifies the last element in a chain; this is a dummy element usable only as an indication that the end of the chain has been reached. QTAM con-

trols element priority as required for internal sequencing.

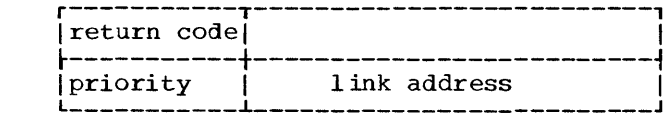
link address: a pointer to the next element control block in the chain; the last element in a chain links to itself. This field is meaningful only when the element control block is linked either into the element chain of a queue control block or into the ready queue.

TRUNCATED SUBTASK CONTROL BLOCK

Typical DSECT:



General Form:



Contents:

return code: branch modifier; a numeric value (a multiple of 2 greater than zero) added to the resolved address of storage location NRET to provide the instruction address to be branched to when the subtask this STCB represents is activated. Commonly appears in the QTAM assembly listing in the form DC AL1 (entry - NRET), where "entry" is the label of the branch address.

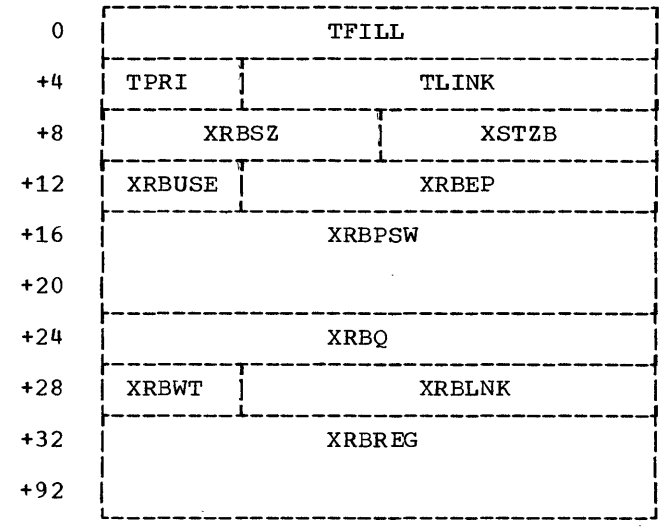
priority: priority of the subtask the STCB represents; determines the relative position of the STCB when linked into the STCB chain of a queue control block. Priority 255 identifies the last STCB in a chain.

QTAM sets the priority value of STCBs for send scheduling subtasks as required to support the send versus receive priority specified by the user in the DCB for the data set.

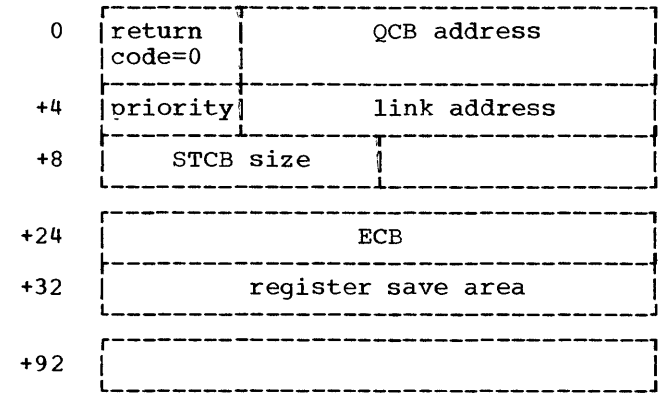
link address: a pointer to the next STCB in the STCB chain; the last STCB in a chain links to itself. This field is meaningful only when the STCB is linked into the STCB chain of a queue control block. If the STCB is not linked in a chain, the last two bytes of the link address field are truncated.

FULL SUBTASK CONTROL BLOCK

Typical DSECT:



General Form:



Contents:

Return code: always zero.

QCB address: the address of QATTACH.

priority: same as for a truncated STCB, but never 255.

link address: same as for a truncated STCB.

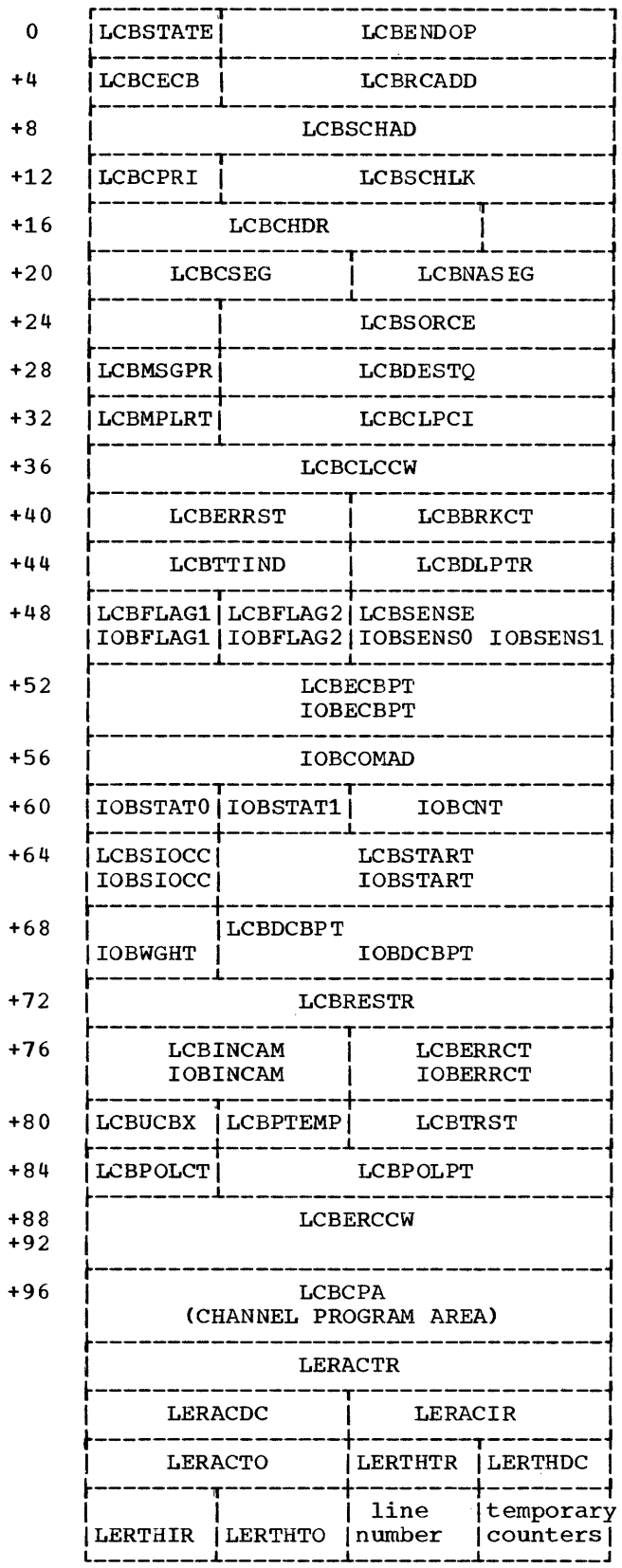
STCB size: 96. The size of the STCB is 96 bytes since the STCB is created within an SVRB including a register save area.

ECB: Event Control Block. This word is used for waiting or posting completion of the task.

LINE CONTROL BLOCK

The LCB contains the IOB, which can be referred to as prefixed by the LCB or ICB depending upon the DSECT issued.

Typical DSECT:



Contents:

Field	Bytes	Description
LCBSTATE	1	00 = inactive 01 = free 02 = partial message in queue 04 = send 08 = receive 10 = initate 20 = converse 40 = recall 80 = cleanup (all numbers given in hex notation)
LCBENDOP	3	For an incoming message, contains the contents of the return register (14) from the ROUTE macro. For an outgoing message, contains the address of the LCB for the originating line.
LCBCECB	1	BTAM opcode for current segment of current message.
LCBRCADD	3	Disk address of the last correctly transmitted segment in current message.

(The receive scheduler STCB is bytes 8-15.)

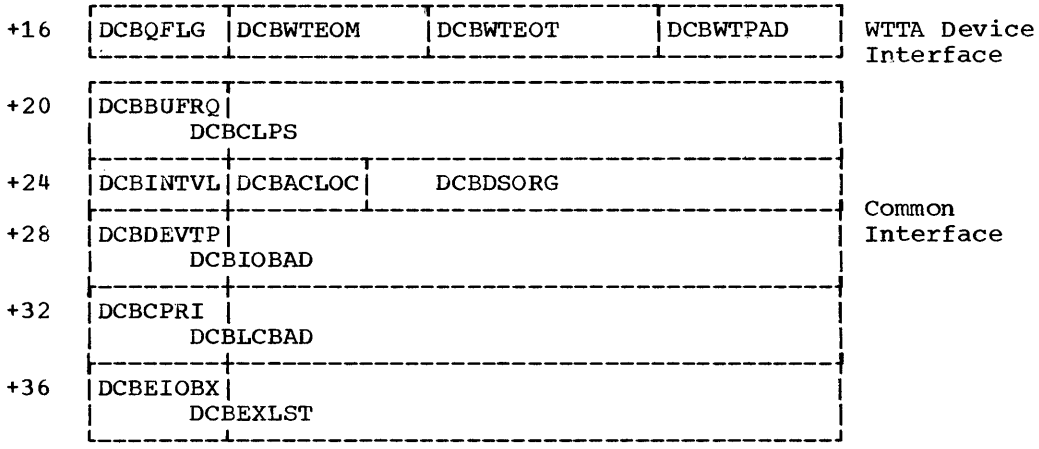
LCBSCHAD	4	Address of the first waiting QTAM subtask for the LCB.
LCBCRPI	1	Priority of the receive scheduler.
LCBSCHLK	3	Link field of the receive scheduler.
LCBCHDR	3	Disk address of the current message header.
LCBCSEG	3	Disk address of the current message segment.
LCBNASEG	3	Pointer to the first segment of the last message received on this line that is to be transmitted.
LCBSORCE	3	Address of the chain of LCBs for source lines currently sending to the same destination (low order bit = "in-source chain" flag).
LCBMSGPR	1	Priority of the current incoming message.

LCBDESTQ	3	Address of the QCB for destination terminal.
LCBMPLRT	1	Scan pointer for next destination.
LCBCLPCI	3	Address of last CCW for which PCI was received.
LCBCLCCW	4	Address of the last BRB for which a buffer was assigned.
LCBERRST	2	Error halfword.
LCBBRKCT	2	If receiving, contains the last status of the SEQUIN (terminal table). If not receiving, contains the time of the requested interrupt.
LCBTTIND	2	Pointer to terminal table entry for current message.
LCBDLPTR	2	Pointer to next entry in distribution list.
LCBFLAG1	1	Status bits used by the I/O supervisor.
LCBFLAG2	1	Status bits used by the I/O supervisor.
LCBSENSE	2	Sense information stored by the I/O supervisor.
LCBECBPT	4	Not used by QTAM.
LCBCSW	8	Channel status word.
LCBSIOCC	1	Start I/O condition code.
LCBSTART	3	Pointer to the first CCW executed in the channel program.
LCBDCBPT	4	Pointer to the DCB.
LCBRESTR	4	Used by ERP to send error messages. Contains terminal ID and TP Op code.
LCBINCAM	2	byte 1: 01 = Tells Poll routine the line is trying to send. 02 = Dial line not available. 04 = Polling or addressing error. 08 = Halt I/O issued on WTTA line. 10 = EOT received on WTTA line. 40 = WRU received on WTTA line. byte 2 = Used by ERP.
LCBERRCT	2	Number of retries.
LCBUCBX	1	Index to the UCB in the DEB.
LCBPTEMP	1	Temporary storage for message priority.
LCBTRST	2	Address of EOB character relative to the address of the last correctly transmitted segment of current message.
LCBPOLCT	1	Count of messages received from terminal.
LCBPOLPT	3	Pointer to currently active entry in polling list.
LCBERCCW	8	Channel Command Word for ERP.
LCBCPA	variable	Channel program area.
LERACTR	4	Cumulative counter for number of transmissions.
LERACDC	2	Cumulative counter for number of data checks.
LERACIR	2	Cumulative counter for number of intervention required.
LERACTO	2	Cumulative counter for number of time-outs.
LERTHTR	1	Threshold counter for number of transmissions.
LERTHDC	1	Threshold counter for number of data checks.
LERTHIR	1	Threshold counter for number of intervention required.
LERTHTO	1	Threshold counter for number of time-outs.

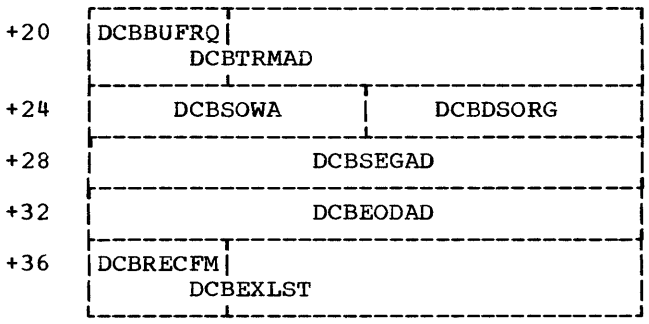
DATA CONTROL BLOCK

Typical DSECT:

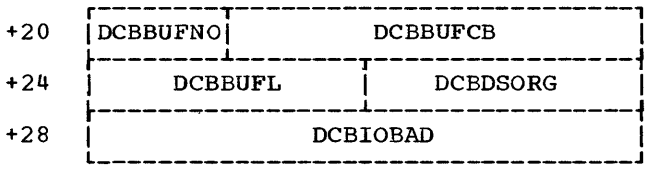
QTAM Line Group Interface



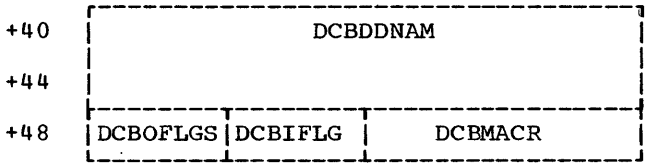
QTAM Processing Program Message Queue Interface



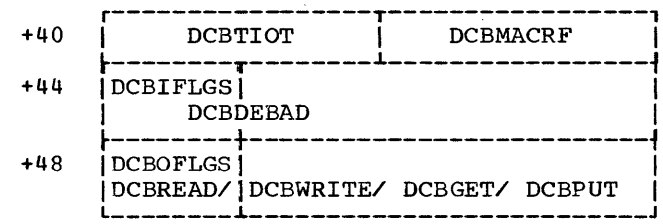
QTAM Direct Access Message Queue Interface



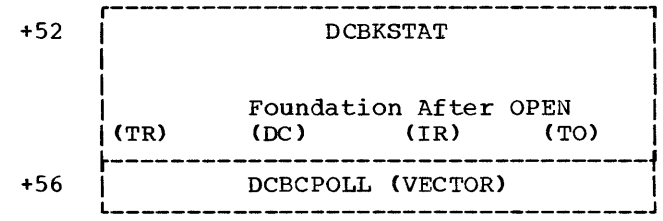
Foundation Before OPEN



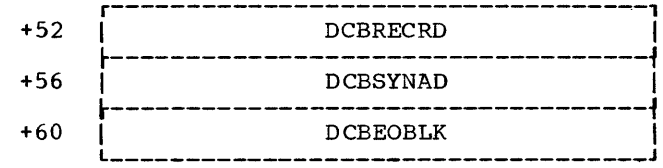
Foundation after OPEN



QTAM Polling List Origin



QTAM Processing Program Message Queue Interface



Contents:

Field	Bytes	Description
		QTAM LINE GROUP
DCBBQFLG	1	Bit Setting / Meaning x1xxxxxx — WRU=YES in the DCB macro instruction. xx1xxxxx — IAM=YES in the DCB macro instruction. xxx1xxxx — WRU macro instruction in the Send Header subgroup of the LPS. xxxx1xxx — WRU macro instruction in the End Send subgroup of the LPS.
DCBWTEOM	1	Hexadecimal representation of the EOM character.
DCBWTEOT	1	Hexadecimal representation of the EOT character. When EOT=2EOM, this byte contains X'00'.
DCBWTPAD	1	Number of padding characters when MON=NO is coded or omitted in the DCB macro instruction.
DCBBUFRQ	1	The number of buffers to be requested for a Read or Write operation in advance of actual transmission.
DCBCLPS	4	Address of the line procedure specification for the line group.
DCBINTVL	1	The number of seconds of intentional delay between passes through a polling list for nonswitched lines.
DCBACLOC	1	The offset, relative to zero, of the device access field for each terminal table entry.
DCBDSORG	2	Data set organization. DSORG= CX — Byte 1: xx01xxxx — Byte 2: reserved
DCBDEVTP	1	Device type pointer. A one-byte value calculated during OPEN and used in the BTAM Read/Write module to calculate the appropriate Device I/O module.
DCBIOBAD	4	The first IOB address.
DCBCPRI	1	Communication priority. Indicates the relative priority to be given to sending and receiving operations. CPRI= R — Bit settings: xxxxx1xx CPRI= E — Bit settings: xxxxxx1x CPRI= S — Bit settings: xxxxxxx1
DCBLCBAD	4	Line control block address. The first LCB address minus the length of an LCB.
DCBEIOBX	1	Extended IOB index; the size of an LCB.
DCBEXLST	4	Address of the exit list.
		QTAM PROCESSING PROGRAM MESSAGE QUEUE
DCBBUFRQ	1	Specifies the number of buffers to be filled with data from the direct access queue before they are requested by a GET macro instruction. Used only in process queue DCB.
DCBTRMAD	4	Address of a user-provided area in which the terminal name is stored.
DCBSOWA	2	The size of the user-provided work area. Used only in process queue DCB.
DCBDSORG	2	Data set organization. Byte 1 has MQ=xxxxx1xx. Byte 2 is reserved.
DCBSEGAD	4	Address of current segment.
DCBEODAD	4	Address of a user provided routine to be entered if no messages are available (the process queue is empty) when a GET macro is issued. Used only in process queue DCB.
DCBRECFM	1	Record Format / Bit Settings G — xxxxx1xx S — xxxx1xxx R — xxxxxx1x

DCBEXLST 4 Address of the exit list.

QTAM DIRECT ACCESS MESSAGE QUEUE

DCBBUFNO 1 Not used.

DCBBUFCB 3 Address of the terminal table, TERMTBL.

DCBBUFL 2 Size of the data in the buffer equated to 1ECKBUFL.

DCBDSORG 2 Data set organization

DSORG=	Byte 1	Byte 2
CQ	xxxx1xxx	reserved

DCBIOBAD 4 Input/output block address.

FOUNDATION BEFORE OPEN

DCBDDNAM 8 Data set name as used in data definition statement. Used by OPEN to locate job file control block (JFCB) address.

DCBOFLGS 1 Flags used by OPEN.

Bit setting	Meaning
xxx1xxxx	OPEN has been successfully completed.
xxxxxxx1	This bit is set to 1 by a I/O support function if the DCB is to be processed by that function.

DCBIFLG 1 Used by IOS in communicating error conditions and in determining error procedures.

Bit Setting	Meaning
00xxxxxx	Not in error procedure.
01xxxxxx	Error correction in process
11xxxxxx	Permanent error condition.
xx10xxxx	Channel 9 printer carriage.
xx01xxxx	Channel 12 printer carriage.
xxxx00xx	Always use IOS error routine.
xxxx01xx	Reserved.
xxxx11xx	Never use IOS error routine.
xxxxxx11	Reserved.
xxxxxx01	Reserved.
xxxxxx00	Reserved.

DCBMACR 2 Macro instruction reference. Specifies the major macros and various options associated with them. Used by OPEN to determine the access method.

For line group:

Byte 1	
INPUT	xx1xxxxx
Byte 2	
OUTPUT	xx1xxxxx

For message queue:

Byte 1	
GET	x1xxxxxx
Byte 2	
PUT	x1xxxxxx

FOUNDATION AFTER OPEN

DCBTIOT 2 Points to the DD entry in the task I/O table for this DCB. It is the offset of the DD entry relative to the beginning of the task I/O table.

DCBMACRF 2 Same as DCBMACR in foundation before OPEN.

DCBIFLGS 1 Same as DCBIFLG in foundation before OPEN.

DCBOFLGS 1 Same as DCBOFLGS in foundation before OPEN.

DCBDEBAD 4 Address of the associated DEB.

DCBRead 4 Address of the READ module.

DCBWrite 4 Address of the WRITE module.

DCBGet 4 Address of the Get module.

DCBPut 4 Address of the Put module.

QTAM Polling List Origin

DCBKSTAT 4 Threshold values for error counts:
+0 Threshold value for number of transmissions.
+1 Threshold value for number of data checks.
+2 Threshold value for

number of intervention required.
+3 Threshold value for number of time-outs.

DCBCPOLL	4	Byte 1: bits 0-3=Adapter type. bit 4=If on, World Trade Telegraph Adapter. bits 5-6=Reserved. bit 7=Internal use (Checkpoint/Restart routine). Bytes 2-4: Address of the polling list for the first line in the line group. Each line in the line group requires 4 bytes for its polling list address.

QTAM PROCESSING PROGRAM MESSAGE QUEUE

DCBRECRD	4	Not used by QTAM.
DCBSYNAD	4	Address of the user provided routine to be entered if a work unit is longer than the work area provided for input. Used only in process queue DCB.
DCBEOBLK	4	Not used in QTAM.

DATA EXTENT BLOCK

Typical Dsect:

APPENDAGE TABLE

-36	DEBEOEA
-32	DEBSIOA
-28	DEBPCIA
-24	DEBCEA
-20	DEBXCEA

PREFIX

-16	DEBWKARA	DEBDSCBA
-12		
- 8	DEBDCBMK	
- 4	DEBLNGTH	

BASIC

0	DEBNMSUB	DEBTCBAD
+ 4	DEBAMLNG	DEBDEBAD
+ 8	DEBOFLGS	DEBIRBAD
+12	DEBOPATB	DEBSYSPG
+16	DEBNMEXT	DEBUSRPG
+20	DEBPRIOR	DEBECBAD
+24	DEBPROTG DEBDEBID	DEBDCBAD
+28	DEBEXSCL	DEBAPPAD
+32	DEBUCBAD (4-byte address per device)	

Contents:

Field	Bytes	Description
		APPENDAGE TABLE
DEBEOEA	4	Address of the End of Extent Appendage branched to by IOS.
DEBSIOA	4	Address of the Start I/O Appendage branched to by IOS.
DEBPCIA	4	Address of Program Controlled Interrupt Appendage branched to by IOS.
DEBCEA	4	Address of the Channel End Appendage branched to by IOS.
DEBXCEA	4	Address of the Abnormal End Appendage branched to by IOS.
		PREFIX
DEBWKARA	1	I/O support work area.
DEBDSCBA	7	DSCB address (BBCCHHR) used by I/O support.
DEBDCBMK	4	DCB modification mask used by I/O support.
DEBLNGTH	4	Length of DEB in doublewords.
		BASIC
DEBNMSUB	1	Number of subroutines loaded by Open module.
DEBTCBAD	3	Address of the TCB for this DEB.

DEBAMLNG	1	Number of bytes in access method section.
DEBDEBAD	3	Address of the next DEB in the same task.
DEBOFLGS	1	Data set status flags.
DEBIRBAD	3	IRB address for error exit.
DEBOPATB	1	Indicates file type.
DEBSYSPG	3	Address of first IOB in system purge chain.
DEBNMEXT	1	Number of extents (number of lines in the line group).
DEBUSRPG	3	Address of first IOB in the user purge chain.
DEBPRIOR	1	Dispatching priority field from TCB, used by IOS for channel queuing of IOBs.
DEBECBAD	3	IOS internal ECB address.
DEBPROTG	1/2	Protection tag assigned to this task.
DEBDEBID	1/2	Hex '0F' identifies this block as a DEB.
DEBDCBAD	3	Address of DCB associated with this DEB.
DEBEXSCL	1	Extent scale: four for direct access device and two for nondirect access device.
DEBAPPAD	3	Address of I/O appendage table ahead of DEB.
DEBUCBAD	4(n)	Pointer to UCB. n = number of devices.

DEB DSECT for a Processing Program:

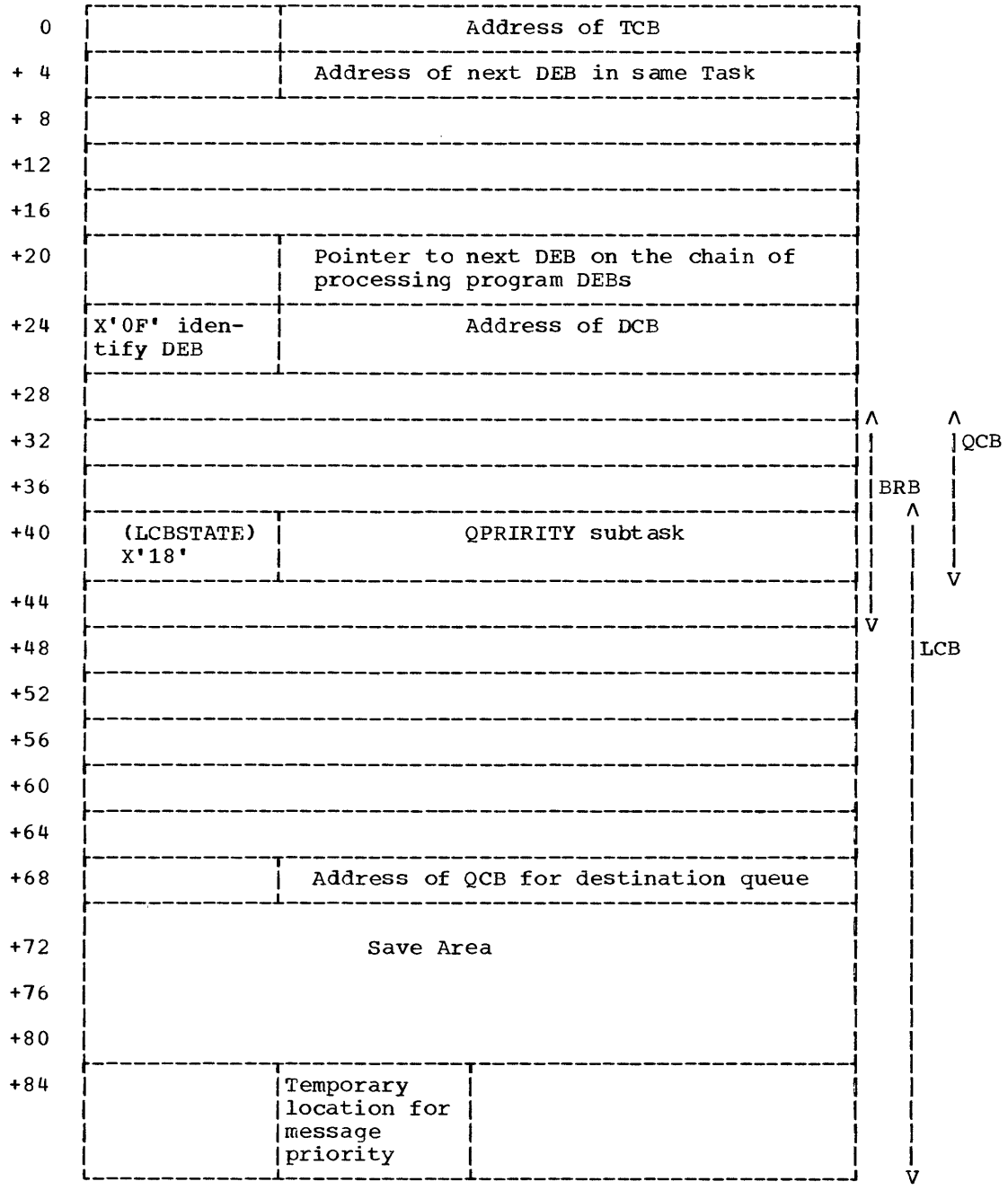
PREFIX Process and Destination Queue

-16	Work area for I/O support	DSCB address (BBCCHHR) used by I/O support
-12		
- 8	DCB modification mask used by I/O support	
- 4	Length of DEB	

BASIC for MS Process Queue

0		Address of TCB	
+ 4		Address of next DEB in the same task	
+ 8			
+12			
+16		Address of the next available record on DASD from queue	
+20		Pointer to next DEB on the chain of processing program DEBs	
+24	X'0F' identify DEB	Address of DCB	
+28		Address of DEB + 48 (BRB)	
+32		First word of a dummy LCB	
+36		Dummy last entry in queue (IECKSTOP)	^
+40			|QCB
+44		QPRIRITY subtask	|
+48			V ^
+52	X'FE'= priority		|BRB
+56	X'08' TIC command	Address of QCB for DASD Process Queue	|
+60	X'07' indicate dummy BRB	Address of DEB + 32 (LCB)	|
+64	Size of work area for GET		V
+68			
+72			
+76		Reserved	
+80			
+84			

BASIC for MS Destination Queue



DATA EVENT CONTROL BLOCK

The main storage for the DECB is reserved in the Activate routine. This routine also initializes the DECB for use by the BTAM Read/Write routine.

General Form:

0	LINEDECB Set to zero		
+ 4		BTAM opcode for current segment	Length of input area for initial read
+ 8	Pointer to DCB		
+12	Starting address for data in buffer		
+16			
+20	Count of messages received	Pointer to currently active entry in polling list	
+24		Index to UCB in DEB	
+28	Reserved		
+32	Address of addressing characters in terminal entry		
+36	Reserved	Address of the polling list	

UNIT CONTROL BLOCK

A unit control block (UCB) is built for each line at system generation time and is used by IOS during execution to determine physical locations. The only field that QTAM uses is the device-type word (UCBTYP), which gives details of the terminals on the line: control unit, adapter, model, and optional features. This word is explained in detail following the UCB figure.

0	Internal Job Number		Allocation Channel Mask	UCB ID	Status "A"
4	Flags	Channel Address	Unit Address for SIO	Flags	Device Table Index
8	Error Routine Table Index		Statistical Table Index	Logical Channel Table Index	Attention Table Index
12	Weight		Channel Mask	Unit Name	
16	Device Type				
20	Last 12* Pointer			Sense	

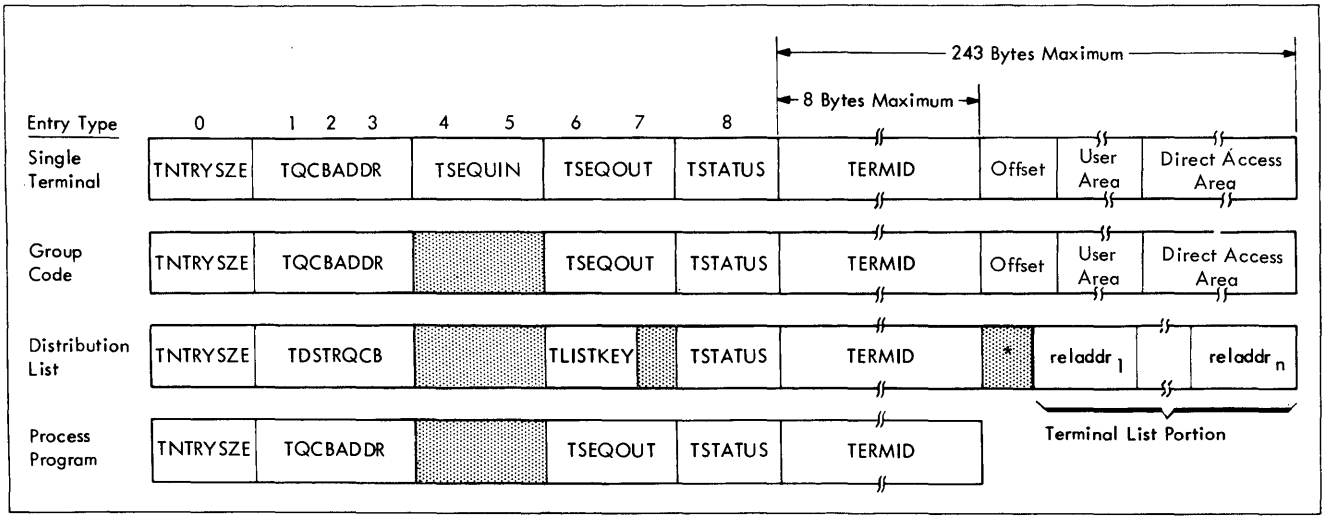
Field	Byte	Description
UCBTYP	4	Device type broken down as follows:

Field	Bit	Description
IOS Flags	0	Unassigned
	1	Data chaining (1=yes)
	2	Burst/byte (1=burst)
	3	May overrun (1=yes)
Model	4-7	If Adapter Type (Bits 24-27) Equals / Model Code Equals / Then Model Is
Optional Features	8	Automatic Calling
	9	Automatic Polling
	10	Checking
	11	Automatic Answering
	12	SCONTROL
	13	XCONTROL
	14-15	SADZER (hex value 0) SADONE (hex value 1) SADTWO (hex value 2) SADTHREE (hex value 3)
Device Class	16	Tape
	17	Communication equipment
	18	Direct access
	19	Display
	20	Unit record
	21	Character reader
	22	Spare
	23	Spare
Adapter Type	24-27	Type of Adapter IBM Terminal Adapter, Type I (hex value 1) IBM Terminal Adapter, Type II (hex value 2) IBM Telegraph Adapter (hex value 3) Telegraph Adapter, Type I (hex value 4) Telegraph Adapter, Type II (hex value 5) World Trade Telegraph Adapter (hex value 6) Synchronous Adapter, Type I (hex value 7) IBM Terminal Adapter, Type III (hex value 8)
Control Unit	28-31	Type of Control Unit 2702 (hex value 1) 2701 (hex value 2) 2703 (hex value 3)

Model Code Equals	Then Model Is
1	1050
2	1060
4	2740
1	1030
1	1050
1	83B3
2	115A
1	TWX
1	2260

TERMINAL TABLE

The terminal consists of a table of information about each terminal as specified through the TERMTBL, LIST, PROCESS, TERM, and OPTION macros.

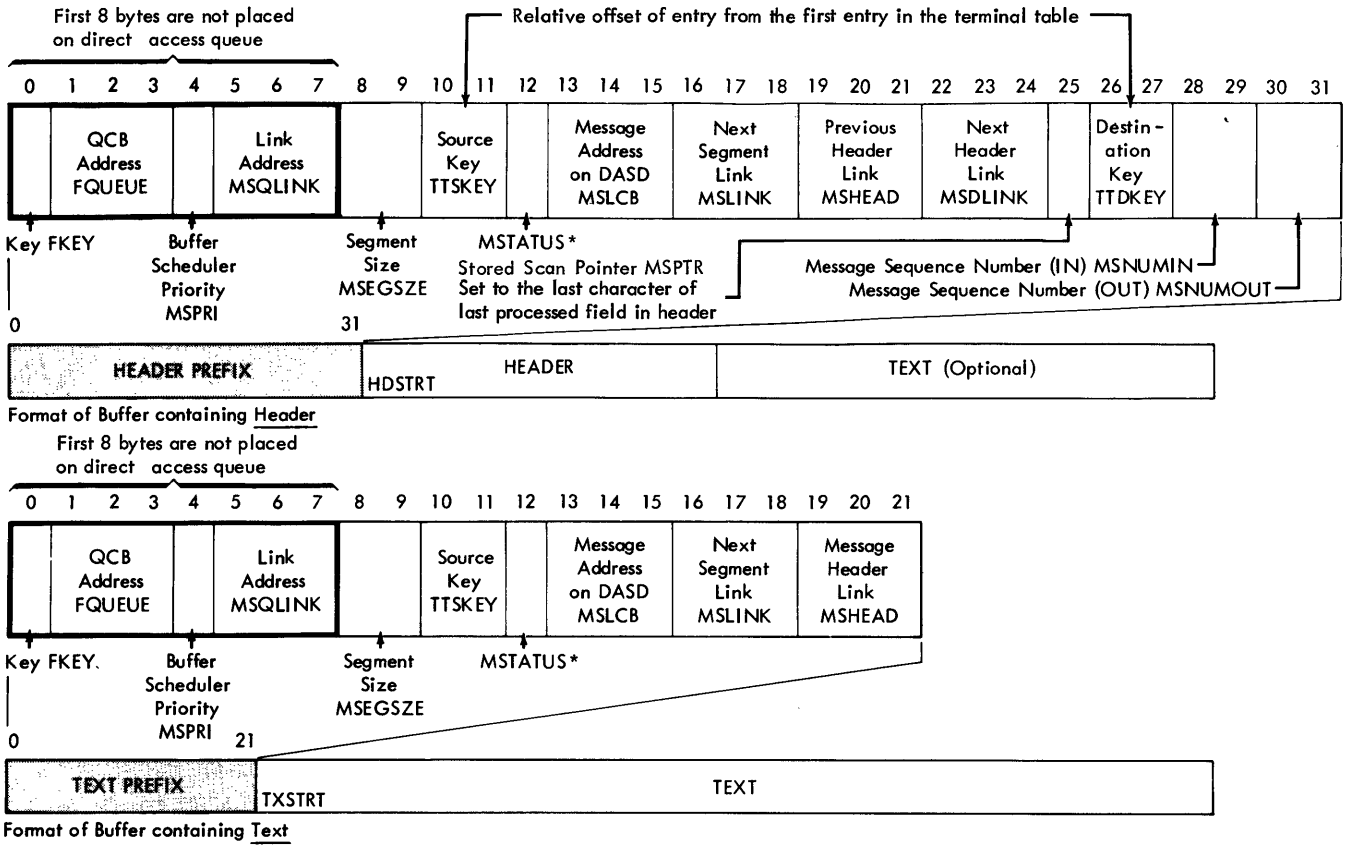


* Unused Field of One Byte

Contents:

Field	Bytes	Description
TNTRYSZE	1	Entry size.
TQCBADDR (TDSTRQCB for dist. list)	3	Address of the destination QCB for the queue of outgoing messages.
TSEQUIN	2	Sequence number for incoming messages. Used only in single-terminal entries.
TSEQOUT	2	Sequence number for outgoing messages, except for distribution list.
TLISTKEY (distribution list only)	1	Starting address of the terminal list portion of this entry.
TSTATUS	1	bit 0-3 not used 4 interval stop bit 5 "intercept" bit 6 "send" bit 7 "receive" bit (single terminal only)
TERMID	1 to 8	Name of the terminal that this entry represents.
Offset	1	Dial terminal: offset from beginning of entry to code for number of dial digits. Nondial terminal: offset from beginning of entry to direct access field (single-terminal and group code only).
User Area	variable	Subfields as defined by the user in OPTION and TERM macro instructions (Single-terminal and group code only).
Device Access Area	variable	Nonswitched: polling and addressing characters. Switched: number of dial digits, dial digits, and addressing characters. TWX: number of dial digits, dial digits, number of ID characters, ID characters, and same number of reserved bytes (single and group code only). WTTA: number of ID characters, and same number of reserved bytes.
Reladdr	variable	Address of a single terminal entry relative to the address of the terminal table (distribution list only).

BUFFER PREFIX



Contents

Field	Description	Initialized by:
FKEY	The ECB key when the buffer appears on the ready queue; always zero.	Assembler
FQUEUE	A pointer to the QCB for the queue to which the buffer has been posted. This field is meaningful only when the buffer is on the ready queue.	Post or Put
MSPRI	The priority of the buffer. This field determines the relative position of the buffer when it is linked into the ready queue or the element chain of a QCB.	Cleanup, Free BRB, Interim LPS, or Disk End Appendage
MSQLINK	A pointer to the next item in the chain in which the buffer appears.	Numerous routines
MSEGSZE	Segment size (includes buffer prefix minus 8).	Buffer-BRB, Line Appendage, Put
TTSKEY	Relative address in terminal table of entry for source terminal.	Source, Interim LPS, Line Appendage
MSTATUS	Used to indicate the status of the message segment contained in the buffer. The significance of the bits in this field is as follows:	
	Bit 0: If 1, do not send or process message.	Cancel Message
	Bit 1: If 1, duplicate copy of header.	Recall
	Bit 2: If 1, an EOB character is present in some position in the buffer other than the last.	Line Appendage
	Bit 3: If 1, the message was previously serviced or sent.	Disk End Appendage
	Bit 4: Not used.	
	Bit 5: If 1, this message was sent with priority.	Disk End Appendage
	Bit 6-7	
	00 = header segment (not last segment)	Activate, Line
	01 = text segment (not last segment)	Appendage, LPS
	10 = header segment (last segment)	Control, Put
	11 = text segment (last segment)	
MSLCB	When in main storage, the address of the LCB to which the buffer is assigned.	Buffer-BRB
	When on the disk, the relative record number of the segment.	DASD Destination

(Continued)

Contents (Continued)

Field	Description	Initialized by:
MSLINK	Relative record number of the next segment in this message.	DASD Destination
MSHEAD	For a header segment, the relative record number of the previous header segment in this queue. For a text segment, the relative record number of the header segment of this message.	DASD Destination
TXSTRT	Start of message data for a text segment. The remaining fields in the prefix apply only to a header segment.	
MSDLINK	The relative record number of the next header segment in this queue.	DASD Destination
MSPTR	Stored scan pointer; indicates the relative position within the buffer where scanning is to begin resume.	Activate, Put, Cleanup EOB, or EOBLC
TTDKEY	Relative address in terminal table of entry for destination terminal.	Lookup or Put
MSNUMIN	Sequence-in number assigned to message.	Sequence Number In
MSNUMOUT	Sequence-out number assigned to message.	Disk End Appendage
HDSTRT	Start of message data for header segment.	

Buffer Prefix formats

SPECIAL CONTROL BLOCK FORMS

QUEUE CONTROL BLOCK

The pattern of unused bytes in the QCB and the truncated STCB is such that they are capable of being combined to conserve storage, as may be seen from the following general forms:

Queue Control Block

key	element chain pointer
priority	link address
	STCB chain pointer

Subtask Control Block

return code	
priority	link address

The general form of these control blocks when combined is:

key	element chain pointer
QCB priority	QCB (ready queue) link addr
return code	*STCB chain pointer
STCB priority	STCB (STCB chain) link addr

*Address of this field minus 1

BUFFER REQUEST BLOCK

The buffer request block (BRB) is basically a resource element control block, with the element control block characteristics outlined in the preceding section. The BRB, however, takes several different forms during its processing cycle. These forms are illustrated in Figures 8 through 12 and are described below.

Figure 26 illustrates the DSECT typically used to refer to fields of the BRB. The first two fullwords (FKEY through FLINK) are typical for all element control blocks. Of the alternate labels available for the third fullword, MSTIC is most commonly used when the DSECT applies to a BRB. The fourth fullword (MSTATUS and MSLCB) is standard for this DSECT.

Figure 27 illustrates the significant fields of the BRB when it is in the element control block chain of the inactive BRB queue control block. This is also the form in which the BRB is generated on expansion of the BUFFER macro instruction.

The BRB-Ring routine removes BRBs from the inactive BRB queue and forms the BRB ring. Figure 28 is a representation of a BRB after it has been processed by the BRB Ring routine. The contents of each field are as follows:

key: always zero. The BRB is still an element, and the key of all elements is zero.

QCB address: variable. For the first BRB in a ring, this field contains the address of the active buffer request QCB if the BRB is to be used for a Receive operation, or the address of the disk I/O QCB if the BRB is to be used for a Send operation. For the remaining BRBs in the ring, the contents of this field are not significant.

priority: The first BRB in a ring to be used for a receive operation has a priority of 12. The contents of this field is normally zero for all other BRBs.

MSTIC: This fullword contains a transfer-in-channel (TIC) command. The first byte contains the TIC command code (08), and the remaining three bytes contain the address of the next BRB in the ring. For the first BRB in the ring only, the TIC address points to the actual address of the next BRB, which begins at a doubleword boundary. For all other BRBs in the ring, the last two bits of the TIC address are set to one. This configuration represents the BRB address (always on a doubleword boundary) plus the BRB idle flag (see BRB Status Codes).

MSTATUS: zero. A zero value for this field indicates that no next segment address has been assigned to this BRB.

LCB address: the address of the line control block for the line over which the Send or Receive operation is to be conducted.

The Disk I/O Appendage further initializes the BRB when the BRB is to be used for a Read from direct access storage. The appendage replaces the LCB address in the fourth fullword of the BRB with the relative record number for the message segment with which this BRB is now associated. The assigning of the next segment address is indicated by changing the value of MSTATUS to 9. Figure 29 illustrates this configuration.

Figure 30 represents a buffer request block that has been converted to a CCW (BRB/CCW). A BRB/CCW is fully initialized for a write to or read from direct access storage. The first two fullwords have been converted to a standard channel command word, and are followed (in the third fullword) by the previously initialized transfer-in-channel command. (Note that a complete BRB/CCW cannot be enqueued by the standard QTAM conventions because the queuing information fields are occupied by the channel command word. The BRB ring is formed by the TIC addresses of the BRB/CCWs.)

In the BRB/CCW, as in any other form of the BRB except that appropriate to the inactive BRB queue, the fourth fullword may contain either MSTATUS=0 and an LCB address, or MSTATUS=9 and a next segment relative record number. The next segment address is inserted in the BRB when the Disk I/O Appendage is processing another BRB in the ring. The BRB in which the next segment address is placed is selected according to its position in the ring, without reference to the queue (if any) upon which it appears.

FKEY	FQUEUE
FPRI	FLINK
MSTIC	
MSTATUS	MSLCB

Figure 26. Typical DSECT for BRB

key	QCB address
priority	link address

Figure 27. BRB on Inactive-BRB Queue

key	QCB address
priority	link address
TIC comm code	address of next BRB in ring
MSTATUS (0)	LCB address

Figure 28. BRB Assignment of Next Segment Address

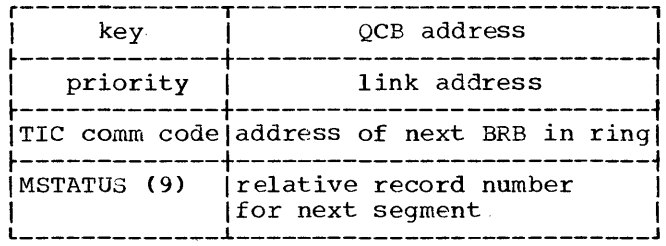


Figure 29. BRB After Assignment of Next Segment Address

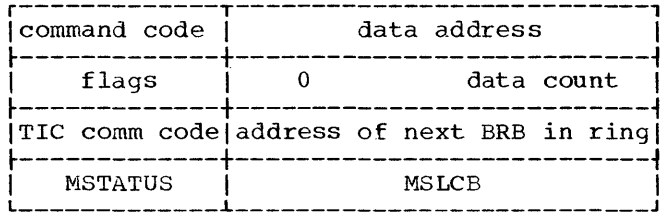


Figure 30. BRB/CCW Initialized for Direct Access Read or Write

BRB Status Codes

The status of a BRB at any point in time is indicated by a code in the two low-order bits of MSTIC+3 (the fourth byte of the third fullword of the BRB.) The codes used are:

00 - Buffer is allocated. This code, which never appears in the BRB used to send or to receive the last segment of a message, makes valid the address portion of a CCW containing a TIC command. (Refer to the discussion of the Line End Appendage for additional information on the invalid TIC address.)

01 - Buffer is in LPS queue (if receiving), or BRB is in disk I/O queue (if sending). This code appears in the BRB used for the last segment of a message.

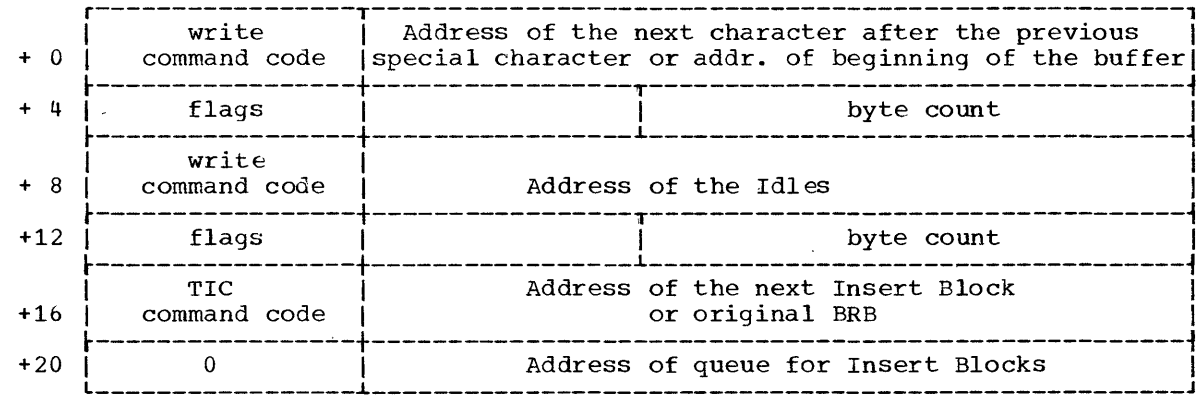
10 - BRB is in active buffer request queue.

11 - BRB is idle. This code is set, typically, when a buffer has been allocated to the BRB but could not be used because the preceding segment had not yet been read when this BRB was selected.

INSERT BLOCK

The insert block is inserted into the chain of BRBs. The chain of insert blocks contains a special purpose BRB used to schedule and to contain channel commands for the transmission of idle characters.

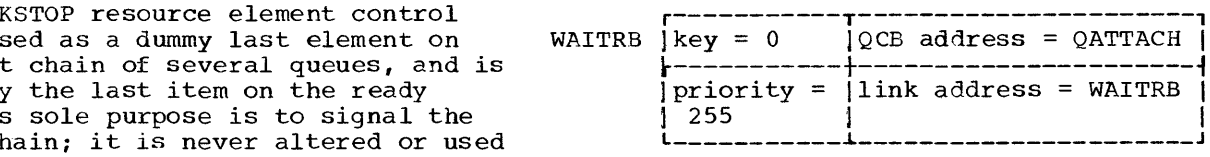
Form:



RESOURCE ELEMENT CONTROL BLOCK (IECKSTOP)

The IECKSTOP resource element control block is used as a dummy last element on the element chain of several queues, and is permanently the last item on the ready queue. Its sole purpose is to signal the end of a chain; it is never altered or used as an available element.

Form:



APPENDIX C: QTAM LINKAGES

Figure 31 depicts the linkages between the macro expansions and the modules they call, for each of the LPS, system status modifying, GET, and PUT macro instructions, with the exception of five of the LPS delimiter macro instructions. These five branch directly to the QTAM Implementation module, rather than to macro-called modules.

The entry point of each module is the same as the module name except where it is shown in parentheses below the module name. Types of linkages are as follows:

is a branch.

SVCxx

is an SVC.

is a branch and link.

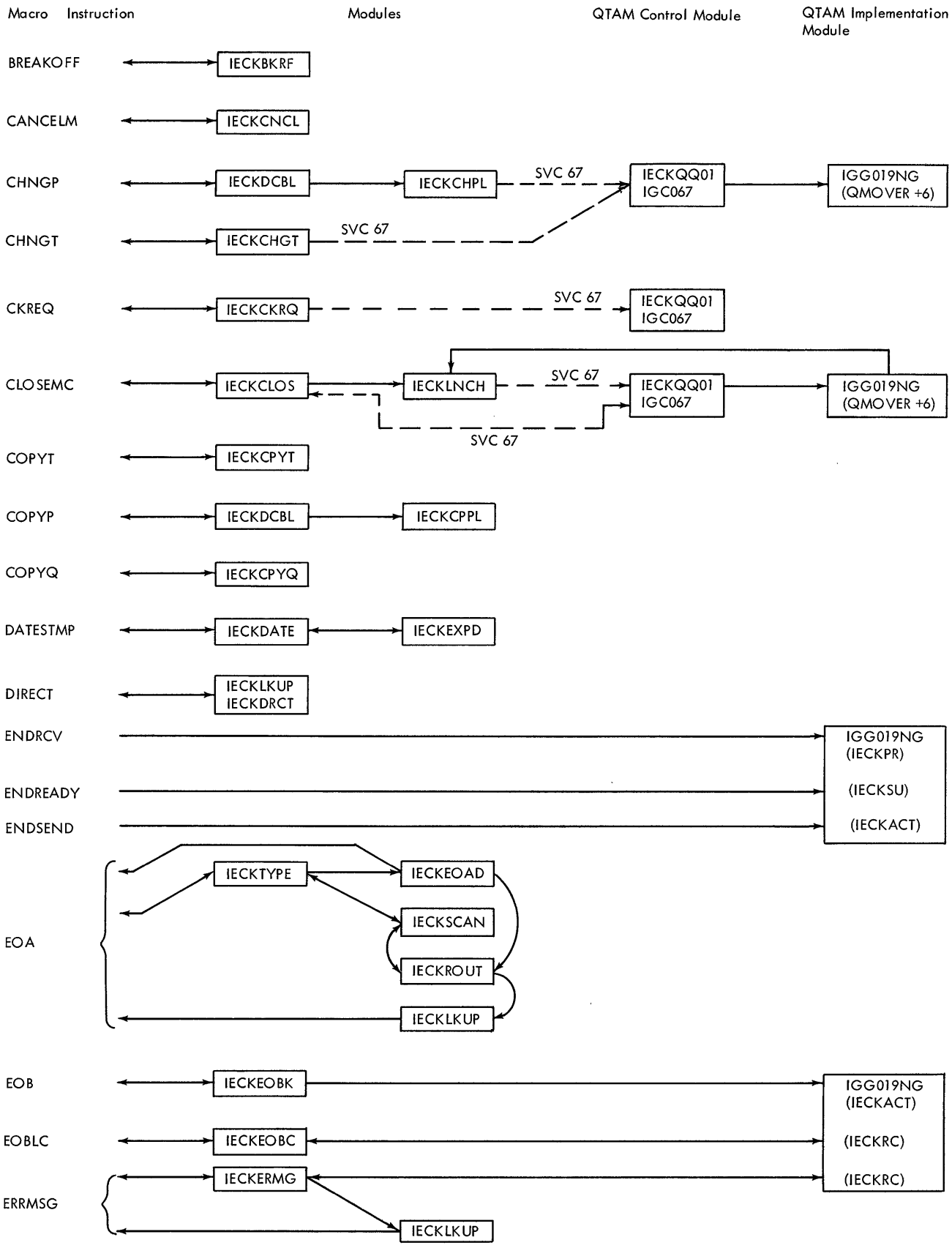


Figure 31. QTAM Linkages (Part 1 of 4)

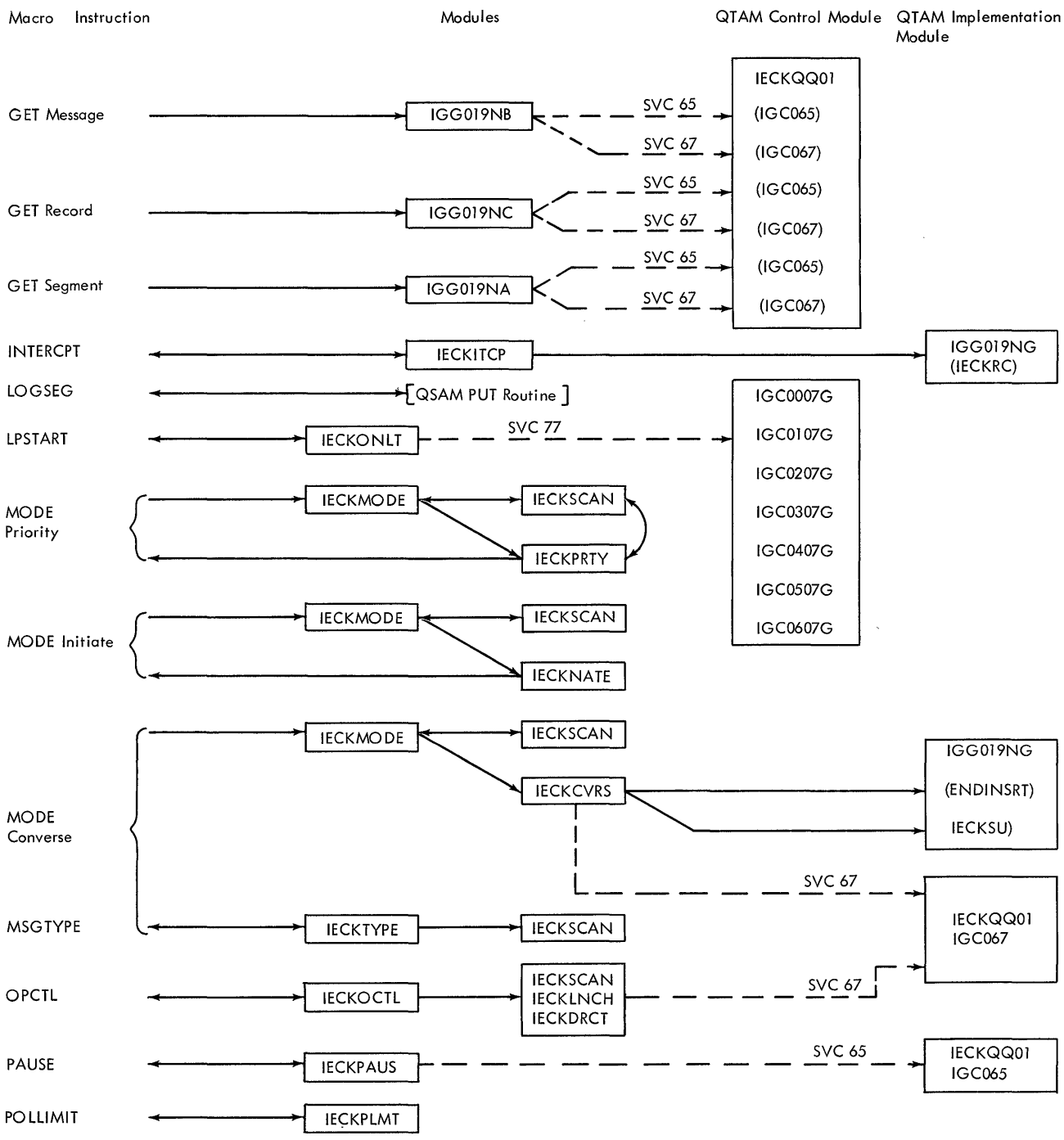
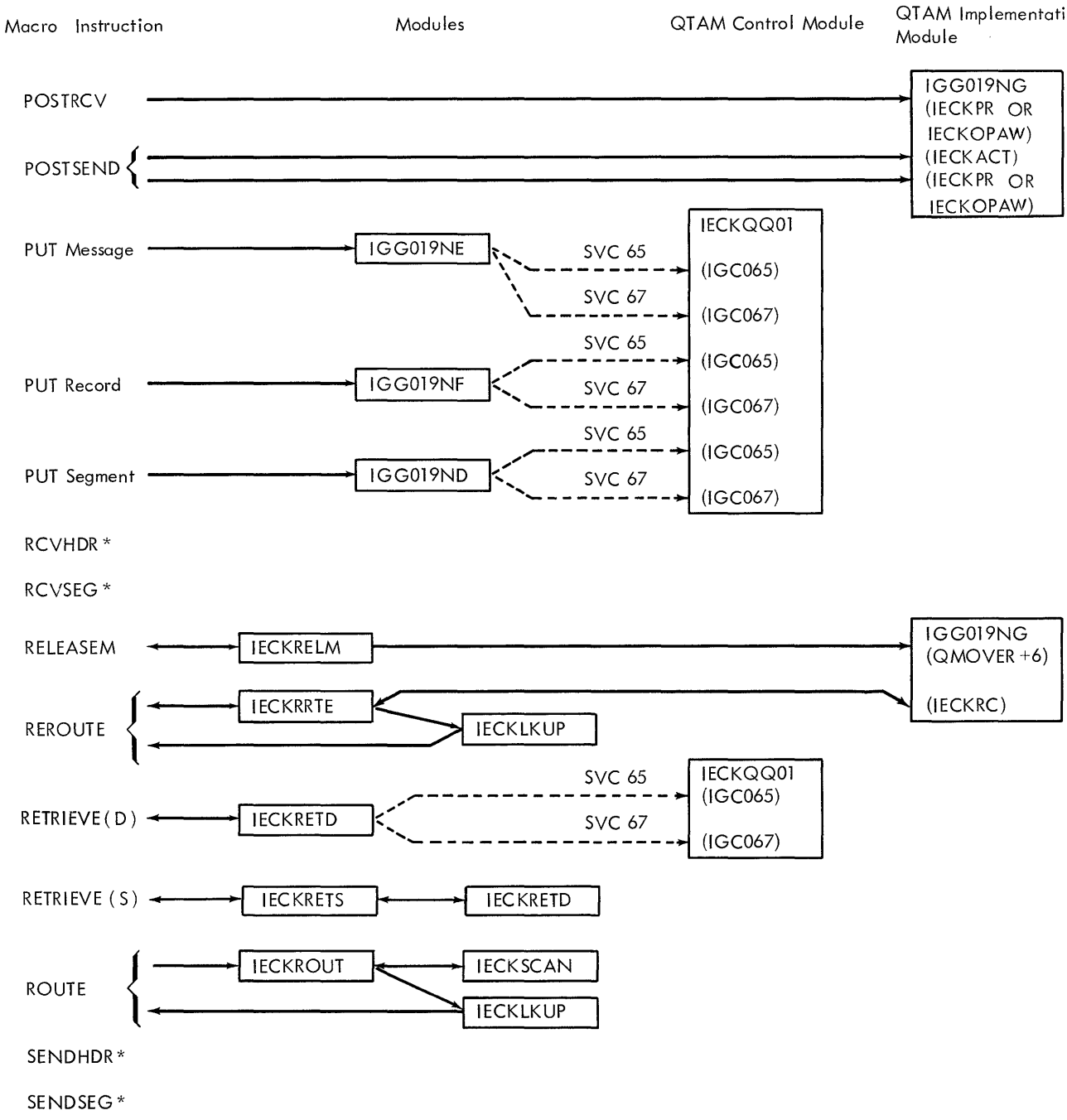
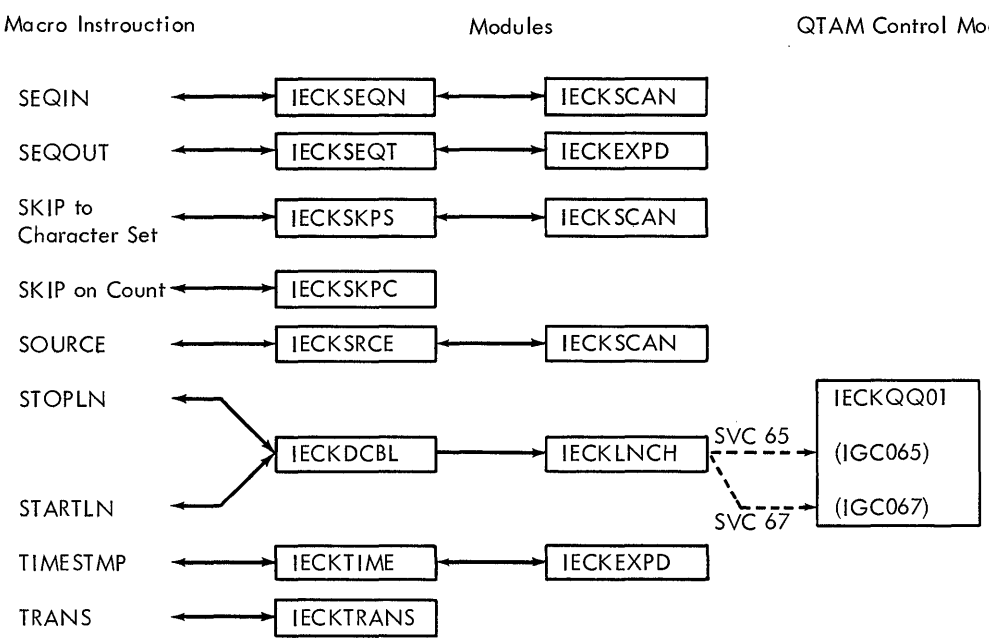


Figure 31. QTAM Linkages (Part 2 of 4)



*This macro instruction genterates no code or generates only in-line code; there is no module linkage.

Figure 31. QTAM Linkages (Part 3 of 4)



* This macro instruction generates no code or generates only in - line code; there is no linkage.

Figure 31. QTAM Linkages (Part 4 of 4)

This appendix identifies the modules that comprise QTAM. Two lists are provided. The first list presents each module name included in QTAM, with a brief description of the nature of the module. For those modules that represent macro instruction implementing routines, the mnemonic operation code for the macro instruction is included in the description (e.g., DATESTMP).

The second list provides a more convenient cross-reference for identifying the routines (as modules) that implement a given macro instruction. This list includes all macro instructions, but does not include modules that are not specifically related to one or more macro instructions. The notation (-none-) in the Module column means that the macro instruction's function is fulfilled at assembly time, usually through the macro expansion. Macro instructions are grouped alphabetically within categories.

All Open, Close, Get, and Put modules are part of the supervisor call library (SVCLIB); the QTAM Implementation module, IGG019NG, is also part of SVCLIB. The QTAM control module, IECKQQ01, is resident in the supervisor nucleus. (Entry points for the two QTAM SVCs are IGC065 and IGC067.) All other QTAM modules are contained in the library identified as SYS1.TELCMLIB.

ALPHABETICAL LIST OF QTAM MODULES

SYS1.TELCMLIB

IECKBRKF	Halt Receive (BREAKOFF) routine.
IECKCHGT	Change Terminal Table (CHNGT) routine.
IECKCHPL	Change Polling List (CHNGP) routine.
IECKCKRQ	Check Request (CKREQ) routine
IECKCLOS	Close Telecommunications System (CLOSEMC) routine.
IECKCNCL	Cancel Message (CANCELM) routine.
IECKCPPL	Copy Polling List (COPYP) routine.
IECKCPYQ	Copy Queue Status (COPYQ) routine.
IECKCPYT	Copy Terminal Table (COPYT) routine.
IECKCVRS	Conversational Mode (MODE) routine; used with IECKMODE and IECKSCAN.
IECKDATE	Datestamp (DATESTMP) routine; requires IECKEXPD.
IECKDCBL	Data Control Block Locate routine.
IECKDLQT	Distribution List (DLIST) routine.
IECKEOAD	End of Address (EOA) routine.
IECKEOBC	End of Block and Line Correction (EOBLC) routine.
IECKEOBK	End of Block (EOB) routine.
IECKERMG	Error Message (ERRMSG) routine, requires IECKLKUP.
IECKEXPD	Expand routine; a second-level routine that provides the number of spaces and the message header required for insertion of timestamp or datestamp characters, or for sequence-out numbers.
IECKITCP	Intercept (INTERCPT) routine.
IECKLKUP	Lookup routine; a second-level routine that locates the terminal table entry for a specified destination.
IECKLNCH	Line Change Routine (STARTLN, STOPLN).
IECKMODE	Message Mode (MODE) routine; requires IECKSCAN and one of the following: IECKCVRS, IECKNATE, IECKPRTY, or a user-written subroutine.
IECKNATE	Initiate mode (MODE) routine; used with IECKMODE and IECKSCAN.
IECKOCTL	Operator Control (OPCTL) routine, requires IECKSCAN, IECKLNCH, and IECKDRCT.
IECKONTL	On-Line Terminal Test routine.
IECKOPAW	Operator Awareness routine.
IECKPAUS	Pause (PAUSE) routine.
IECKPLMT	Polling Limit (POLLIMIT) routine.
IECKPRTY	Priority Mode (MODE) routine; used with IECKMODE and IECKSCAN.
IECKRELM	Release Intercept (RELEASEM) routine.
IECKRETD	Retrieve DASD (RETRIEVE) routine (by relative track number).
IECKRETS	Retrieve (RETRIEVE) routine (by sequence number).
IECKRF40	Translate Table RCAP2740; 2740 to monocase EBCDIC.
IECKRF50	Translate Table RCAP1050; 1050 to monocase EBCDIC.
IECKROUT	Routing (ROUTE) routine; requires IECKSCAN and IECKLKUP.
IECKRRTE	Reroute (REROUTE) routine.
IECKRV30	Translate Table RCVE1030; 1030 to EBCDIC.
IECKRV40	Translate Table RCVE2740; 2740 to EBCDIC.

IECKRV50	Translate Table RCVE1050; 1050 to EBCDIC.
IECKRV60	Translate Table RCVE1060; 1060 to EBCDIC.
IECKR260	Translate Table RCVE2260; 2260 to EBCDIC.
IECKRVT1	Translate Table RCVET1; 5-level Baudot to EBCDIC.
IECKRVT2	Translate Table RCVET2; TWX code to EBCDIC.
IECKRVTW	WTTA Translate table RCVEITA2; ITA2 code to EBCDIC.
IECKRVTZ	WTTA Translate table RCVEZSC3; ZSC3 code to EBCDIC.
IECKSCAN	Scan Header routine; a second level routine that steps through a header segment, maintaining a pointer to the portion of the segment to be operated upon by the next LPS routine.
IECKSD30	Translate Table SEND1030; EBCDIC to 1030.
IECKSD40	Translate Table SEND2740; EBCDIC to 2740.
IECKSD50	Translate Table SEND1050; EBCDIC to 1050.
IECKSD60	Translate Table SEND1060; EBCDIC to 1060.
IECKS260	Translate Table SEND2260; EBCDIC to 2260.
IECKSDT1	Translate Table SENDT1; EBCDIC to 5-level Baudot code.
IECKSDT2	Translate Table SENDT2; EBCDIC to TWX code.
IECKSDT3	Translate Table SENDT3; EBCDIC to TWX code with parity bit on.
IECKSDTW	WTTA Translate Table SENDITA2; EBCDIC CODE TO ITA2 code.
IECKSDTZ	WTTA Translate Table SENDZSC3; EBCDIC CODE TO ZSC3 code.
IECKSEQN	Sequence-in (SEQIN) routine; requires IECKSCAN.
IECKSEQT	Sequence-out (SEQOUT) routine; requires IECKEXPD.
IECKSKPC	Skip on Count (SKIP) routine.
IECKSKPS	Skip to Character Set (SKIP) routine; requires IECKSCAN.
IECKSRCE	Source (SOURCE) routine; requires IECKSCAN.
IECKTIME	Timestamp (TIMESTMP) routine; requires IECKEXPD.
IECKTRNS	Translate (TRANS) routine; used in conjunction with a QTAM or user-provided translations.
IECKTYPE	Message Type (MSGTYPE) routine; requires IECKSCAN.

SYS1.SVC Library

IGC0007G	Terminal Test Header Analysis routine.
IGC0107G	1030 Terminal Test routine.
IGC0207G	1050 Terminal Test routine.
IGC0307G	1060 Terminal Test routine.
IGC0407G	2740 Terminal Test routine.
IGC0507G	2848/2260 Terminal Test routine.
IGC0607G	2848/2260 Terminal Test routine.
IGE0004E	Time-Out and Data Check for Auto Poll routine.
IGE0104E	Data Check routine.
IGE0204E	Time-Out routine.
IGE0304R	Intervention Required routine.
IGE0404E	Lost Data routine.
IGE0504E	Error Post routine.
IGE0604E	Bus-Out and Overrun routine.
IGE0704E	Link routine.
IGE0804E	Status Check routine.
IGE0904E	Command Reject, Equipment Check, SNO Error, SIO CC 1 routine.
IGE0004F	Read Skip, Break Return routine.
IGE0104F	Diagnostic Write/Read routine.
IGE0204F	Line Error Recording routine.
IGE0304F	Operator Control and LER Addition routine.
IGE0404F	Special OPEN and Checkpoint Restart routine.
IGE0504F	Not Operational SIO routine.
IGE0604F	Bus-Out and Overrun for Auto Poll routine.
IGE0704F	Overrun routine.
IGG0193N	Open Communications Line Group (load 1) (OPEN).
IGG0193O	Open Direct Access Message Queue (OPEN).
IGG0193P	Open Process Queue (OPEN).
IGG0193R	Open Communications Line Group (load 2) (OPEN).
IGG0193T	Open Communications Line Group (load 3) (OPEN).
IGG0193U	Open Direct Access Message Queue (load 2) (OPEN).
IGG0193V	Open Checkpoint/Restart.
IGG0194A	Open Communications Line Group (load 4) (OPEN).
IGG019NA	Get Message Segment (GET) routine.
IGG019NB	Get Message (GET) routine.
IGG019NC	Get Record (GET) routine.
IGG019ND	Put Message Segment (PUT) routine.
IGG019NE	Put Message (PUT) routine.
IGG019NF	Put Record (PUT) routine.
IGG019NG	QTAM Implementation Module.
IGG019NH	Checkpoint/Restart routine.
IGG019NJ	IBM 2740 (Basic) Device I/O Module.
IGG019NK	IBM 2740 with Dial Device I/O Module.
IGG019NL	IBM 2740 with Transmit Control and Checking Device I/O Module.
IGG019NM	IBM 2740 with Dial and Transmit Control Device I/O Module.
IGG019NN	IBM 2740 with Dial and Checking Device I/O Module.
IGG019NO	IBM 2740 with Station Control and Checking Device I/O Module.
IGG019NP	IBM 2740 with Station Control

	Device I/O Module.
IGG019NQ	IBM 2740 with Checking Device I/O Module.
IGG019NR	IBM 2260 Device I/O Module.
IGG0203N	Close Communications Line Group (CLOSE).
IGG0203O	Close Direct Access Message Queue (CLOSE).
IGG0203P	Close Process Queue (CLOSE).
IGG019NS	TWX Device I/O Module.
IGG019NT	WU 115A Device I/O Module.
IGG019NU	AT&T 83B3 Device I/O Module.
IGG019NV	IBM 1030 Device I/O Module.
IGG019NW	IBM 1060 Device I/O Module.
IGG019NX	IBM 1050 (Switched) Device I/O Module.
IGG019NY	IBM 1050 (Nonswitched) Device I/O Module.
IGG019NZ	Read/Write Routines.
IGG019N1	IBM 1050 (nonswitched) for Auto Poll.
IGG019N2	IBM 1060 for Auto Poll.
IGG019N3	IBM 1030 for Auto Poll.
IGG019N8	IBM 2740 with Station Control and Checking for Auto Poll.
IGG019N9	IBM 2740 with Station Control for Auto Poll.
IGG019QA	WTTA Device I/O Module.
IGG019QB	WTTA Line Appendage Module.

QTAM DSECTs in SYS1.MACLIB

CTLPROGD	DSECT for QTAM Control Module IECKQQ01.
DCBD	DSECT for Data Control Blocks.
IECDSECT	DSECT for System OPEN Work Area.
IECTDEBX	DSECT for Data Extent Block.
IECTDECB	DSECT for Data Event Control Block.
IECTIOBX	DSECT for Input/Output Block.
IECKQIOB	DSECT for Input/Output Block.
LCBD	DSECT for Line Control Block.
PREFIXD	DSECT for Header & Text Prefixes.
QCBD	DSECT for Queue Control Block.
STCBD	DSECT for Full Subtask Control Block.
TCBD	DSECT for Task Control Block.
TERMTBLD	DSECT for Terminal Table.

LIST OF MODULES BY MACRO INSTRUCTION CATEGORY

SUPPORT MACRO INSTRUCTIONS

Macro instruction	Module
CLOSEMC Telecommunications System	IECKCLOS, IECKLNCH
CLOSE Communications Line Group	IGG0203N
CLOSE Direct Access Message queue	IGG0203O
CLOSE Process queue (input)	IGG0203P
CLOSE Process queue (output)	IGG0203P
GET message	IGG019NB
GET record	IGG019NC
GET message segment	IGG019NA
OPEN communications line group	IGG0193N (load1) IGG0193R (load2) IGG0193T (load3) IGG0194A (load4)
OPEN direct access message queue	IGG0193O IGG019NU (load2)
OPEN Checkpoint/Restart	IGG019NV
OPEN process queue (input and output)	IGG0193P
PUT message	IGG019NE
PUT record	IGG019NF
PUT message segment	IGG019ND

MESSAGE CONTROL MACRO INSTRUCTIONS

Macro Instruction	Module
Initialization	
ENDREADY	
Control Information	
BUFFER	-none-
DLIST	IECKDLQT
OPTION	-none-
POLL	-none-
PROCESS	-none-
TERM	-none-
TERMTBL	-none-
Line Procedure Specification	
BREAKOFF	IECKBRKF
CANCELM	IECKCNCL
COUNTER	-none-
DATESTMP	IECKDATE, IECKEXPD
DIRECT	IECKDRCT, IECKLKUP
ENDRCV	IECKEOBK(WTTA only)
ENDSEND	-none-
EOA	IECKEOAD
EOB	IECKEOBK
EOBLC	IECKEOBC
ERRMSG	IECKERMG, IECKLKUP
INTERCPT	IECKITCP
LOGSEG	-none-
LPSTART	-none-
MODE	IECKMODE, IECKSCAN
-CONVERSE	IECKCVRS
-INITIATE	IECKNATE
-PRIORITY	IECKPRTY, IECKSCAN
MSGTYPE	IECKTYPE, IECKSCAN
OPCTL	IECKOCTL, IECKSCAN, IECKLNCH, IECKDRCT
PAUSE	IECKPAUS
POLLIMIT	IECKPLMT
POSTRCV	IECKOPAW
POSTSEND	IECKOPAW
RCVHDR	-none-

RCVSEG	-none-
REROUTE	IECKRRTE, IECKLKUP
ROUTE	IECKROUT, IECKSCAN IECKLKUP
SENDHDR	-none-
SENDSEG	-none-
SEQIN	IECKSEQN, IECKSCAN
SEQOUT	IECKSEQT, IECKEXPD
SKIP on count	IECKSKPC
SKIP to character set	IECKSKPS, IECKSCAN
SOURCE	IECKSRCE, IECKSCAN
TIMESTMP	IECKTIME, IECKEXPD
TRANS	IECKTRNS
WRU	-none-

WTTA Translation Tables

RCVEITA2	-none-
RCVEZSC3	-none-
SENDITA2	-none-
SENDZSC3	-none-

MESSAGE PROCESSING MACRO INSTRUCTIONS

Macro Instruction	Module
CKREQ	IECKCKRQ
CHNGP	IECKCHPL, IECKDCBL
CHNGT	IECKCHGT
COPYT	IECKCPYT
COPYP	IECKCPPL, IECKDCBL
COPYQ	IECKCPYQ
RELEASEM	IECKRELM
RETRIEVE	IECKRETD, IECKRETS
STARTLN	IECKLNCH, IECKDCBL
STOPLN	IECKLNCH, IECKDCBL

Figure 32 is a grid showing the QTAM routines that effect the queues. The grid specifies whether the action was through a Qpost or Qwait and what was posted. The subtask associated with the queue is activated through the QTAM nucleus each time the queue is acted upon.

ROUTINE \ QUEUE	Active BRB	Additional CCW	Available Buffer	Change	Checkpoint	Check Request	Copy	DASD Destination	DASD Process	Disk I/O	Inactive BRB	LCB	LPS	Move Data	MS Destination	MS Process	Queue	Ready	Return Buffer	Stop	Terminal Test	Test	Time
BRB Ring	Post BRB									Post BRB	Wait BRB												
Buffer BRB										Post BRB	Post BRB		Post Buffer										
Buffer Recall/Cleanup		Post BRB	Post Buffer					Post Buffer	Post Buffer	Post BRB			Wait										
Change Polling List														Post Move Data									
Change Terminal														Post Move Data									
Checkpoint /Restart					Post Dummy LCB	Post Ck.pt. elem.							Post Dummy LCB										
Check Request						Post ECB																	
Close Message Control														Post Move Data									
Converse	Post BRB		Post Buffer							Post BRB													
DASD Destination										Post Buffer													
Disk End Appendage			Post Buffer			Post Dummy LCB				Post BRB			Post Buffer						Post Buffer				
Disk I/O	Post BRB												Post Start I/O elem.										
End of Poll Time Delay												Post LCB	Post Time elem.										Post Time
Free BRB											Post BRB	Post LCB											
Get																Wait		Wait	Post Buffer				
Get Scheduler										Post Buffer													
Interim LPS													Post Buffer										
Line Change												Wait Line	Post Dummy LCB	Post Move Data			Post LCB						
Line End Appendage												Post LCB	Post Buffer										
Line PCI Appendage	Post BRB		Post Buffer										Post Buffer										
LPS Control					Post Dummy LCB							Post LCB	Wait										
Pause		Wait BRB																					
PUT	Post BRB							Post Buffer							Post Wait Buffer			Wait					
Operator Control			Post Buffer	Post Change			Post Copy					Post Line	Post QCB							Post Stop			
Release Intercepted														Post Move Data									
Retrieve DASD			Post Buffer							Post BRB								Wait					
Return Buffer			Post Buffer										Post Buffer		Post Buffer								
Send Scheduler								Post Line				Wait Line											
Terminal Test Recognition			Post Buffer									Post LCB	Post Buffer								Post Buffer	Post Buffer	

Figure 32. Queues Affected by QTAM Routines

The System/360 Operating System provides interfaces among program by means of control blocks and tables. These blocks have standardized formats. They contain numerous fields of information and references by the program. Some of these fields are pointers to other blocks. Figure 33 shows the various blocks and the linkages pertinent to QTAM.

Figure 33. Control Block Linkages

APPENDIX G: HEADER AND TEXT RELATIONSHIPS ON A DASD QUEUE

Header and Text Relationships on a DASD Queue

Figure 34 illustrates how chains of message segments for destination and process queues are formed on a direct access storage device.

Each chain consists of a series of areas on the direct access device. Each area either: (1) contains a message segment and the segment's associated header or text prefix; or (2) is reserved for the next segment to be placed on the chain. The areas, and thus the segments, are linked into the chain by means of information, called relative record numbers, contained in the link fields of the prefixes. Each chain is formed as follows. At the time the direct access queues data set is opened, one area is reserved for each chain to be formed. The header segment of the first message to be put on the chain is placed in the reserved area for that chain. At the same time, the next two available areas are reserved: the first is reserved for the header of the next message to be put on the chain, and the second is reserved for the first text segment of the same (tnat is, the first) message. This process is repeated for each succeeding message segment placed on the chain. Each time a header segment is placed in its reserved area, two more areas are reserved; each time a text segment is placed on the chain, one more area is reserved.

If tne current segment is the last segment of the message, no area is reserved for a next text segment. Specifically, when a message consisting of only a header segment is placed on the chain, only one area is reserved (that is, for the header of the next message); when the last of a series of text segments is placed on the chain, no area is reserved.

At the time an area is reserved, link information is placed in the link fields of the prefixes of the associated segments. Each header prefix contains the relative record numbers of the areas occupied by: (1) the first text segment of the same message; (2) the previous header segment; and (3) the next header segment. Each text prefix contains the relative record numbers of the areas occupied by: (1) the next text segment of the same message; and (2) the header of the same message. If the header is the only segment in the message, the relative record number of the area occupied by that header is placed in its "next segment" link (MSLINK) field. If the text segment is the last segment in the message, the relative record number of the header of the same message is placed in the MSLINK field.

The figure illustrates the progressive development of two chains, one for queue A and one for queue B. The time span covered begins with the initialization of the queues (when the direct access queues data set is opened) and ends when there are three complete messages on the chain for queue A, and two complete messages on the chain for queue B.

The five messages contain a total of fourteen segments, which are placed on the chains in the following sequence:

1. Header of message 1, queue B (B-1)
2. Text segment of B-1
3. Header of A-1
4. Header of A-2
5. Header of B-2
6. Text segment of A-1
7. Text segment of A-2 (last segment)
8. Header of A-3
9. Text segment of A-1
10. Text segment of B-2 (last segment)
11. Text segment of B-1 (last segment)
12. Text segment of A-3
13. Text segment of A-1 (last segment)
14. Text segment of A-3 (last segment)

Each step in the development of the chains is shown in Figure 34. Each step shows the currently filled areas of the direct access space allotted to the chains, the areas reserved for succeeding segments, and the location of the next available area (that is, the area that will be reserved in a succeeding step).

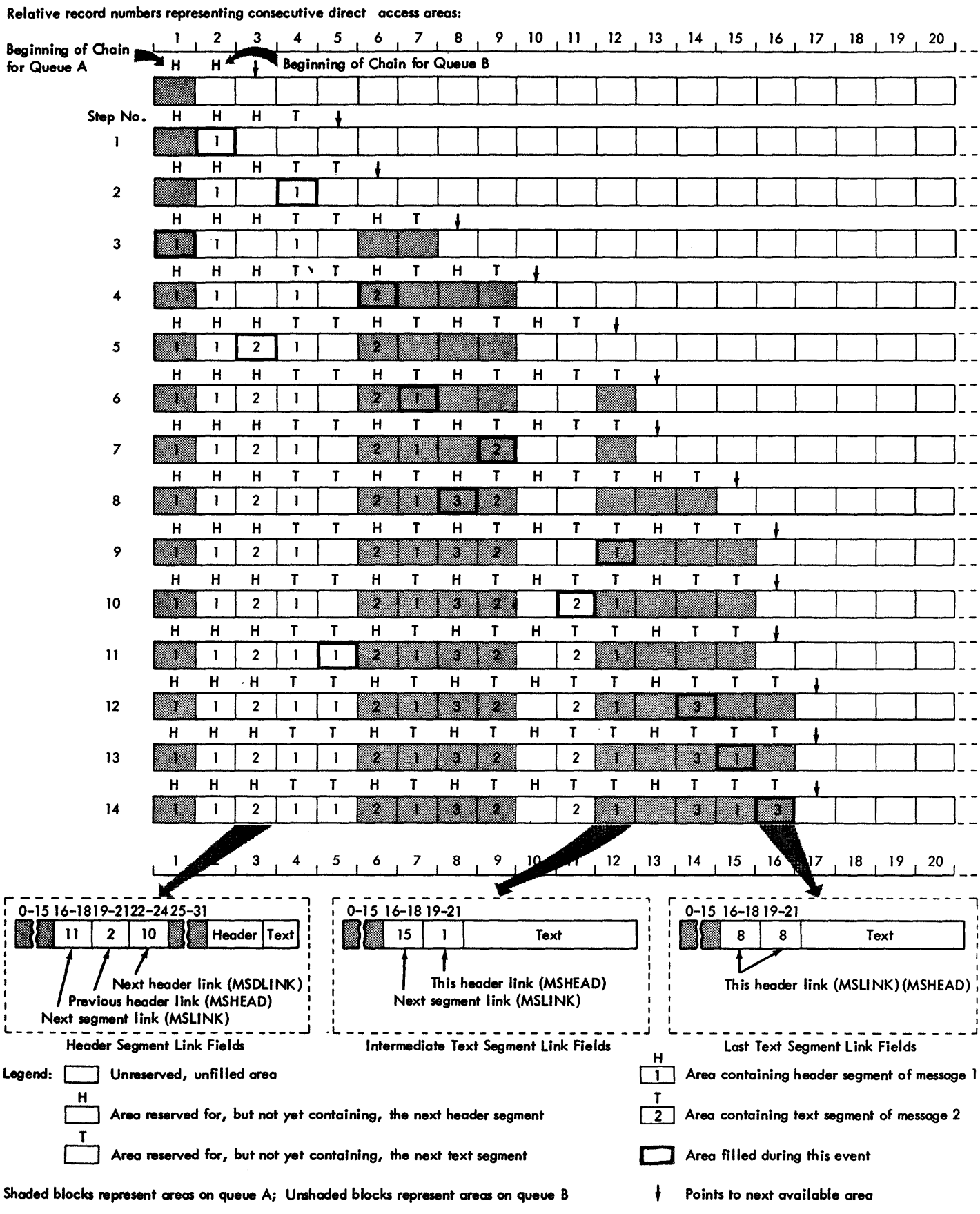


Figure 34. Example of Message Header and Text Relationships in Direct Access Destination and Process Queues

IBM S/360 OS VTAM FLM Printed in U.S.A. Y30-2002-2

IBM®

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

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]

READER'S COMMENT FORM

IBM System/360 Operating System
Queued Telecommunications Access Method
Program Logic Manual

Y30-2002-2

- 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.

Staple

Y30-2002-2

YOUR COMMENTS, PLEASE . . .

This publication is one of a series that serves as a reference source for systems analysts, programmers, and operators of IBM systems. Your answers to the questions on the back of this form, together with your comments, help us produce better publications for your use. Each reply is carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

Please note: Requests for copies of publications and for assistance in using your IBM system should be directed to your IBM representative or to the IBM sales office serving your locality.

Fold Fold

FIRST CLASS
PERMIT NO. 569
RESEARCH TRIANGLE PARK
NORTH CAROLINA

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

POSTAGE WILL BE PAID BY . . .

IBM Corporation
P.O. Box 12275
Research Triangle Park
North Carolina 27709

Attention: Programming Documentation, Dept. 844

Cut Along Line

Fold Fold

IBM®

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

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]

Additional Comments: