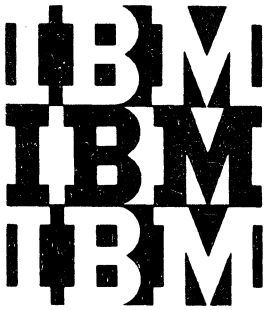


File Number S360-30
Order No. GY30-5002-4

Program Logic

IBM System/360
Disk Operating System
Queued Telecommunications Access Method
Program Logic Manual

Program Number 360N-CQ-470

This reference publication describes the internal logic of the Queued Telecommunications Access Method (QTAM) under the IBM System/360 Disk Operating System.

It is intended for persons involved in program maintenance and by systems programmers who are altering the program design. Program logic information is not necessary for the operation of the program; therefore, the distribution of this publication is limited to those with maintenance and alteration requirements.

For titles and abstracts of other associated publications, see IBM System/360 Bibliography, GA22-6822.

PREFACE

Purpose of this Publication

This Program Logic Manual (PLM) is a detailed guide to the internal structure of the Queued Telecommunications Access Method (QTAM). It supplements the program listings by providing descriptive text and flowcharts; program structure at the machine instruction level is not discussed.

Design of this Publication

This PLM presents the logic of QTAM as a series of logical levels. Each succeeding level describes the program in more detail.

- The Program Level: The first two sections describe the physical and logical organization of QTAM. QTAM is discussed from two points of view: first, as a program within the DOS control program structure; and second, as a separate control program.
- The Major Component Level: The third section gives an outline of QTAM operations at the component level. The interaction among the major QTAM components is described in some detail. Appendix E is a foldout chart showing the general flow of QTAM operations and is designed to complement this section.
- The Routine Level: The next six sections contain detailed descriptions of the QTAM routines followed by flowcharts of all the routines. The flowcharts are arranged alphabetically by chart ID for easy reference.

Detailed illustrations and descriptions concerning QTAM queues and subtasks, control block formats, and linkages are presented in appendixes.

Prerequisite and Related Literature

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, GA22-6821.

IBM System/360 Disk Operating System, QTAM Message Control Program, GC30-5002.

IBM System/360 Disk Operating System, QTAM Message Processing Program Services, GC30-5003.

IBM System/360 Disk Operating System (DOS), System Control, GY24-5017.

Fifth Edition January 1971

This edition, GY30-5002-4, corresponds to DOS release 24. It is a major revision of, and renders obsolete, GY30-5002-3. Incorporated are changes issued in Technical Newsletters Y30-5529, dated April 15, 1969, and Y30-5537 dated July 14, 1969.

Changes are indicated by a vertical line to the left of the affected text and to the left of affected parts of figures.

Specifications contained herein are subject to change from time to time. Any such change will be reported in subsequent revisions or Technical Newsletters.

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, Publications Center Dept. E01, P.O. Box 12275, Research Triangle Park, North Carolina, 27709.

FIGURES

TABLES

CHARTS

In IBM System/360 Disk Operating System, an access method is a procedure for transferring data between main storage and an input/output device. A variety of access methods is available to the user of the Disk Operating System (DOS). One of these, the Queued Telecommunications Access Method (QTAM), can control data transfer between main storage and remote terminals.

QTAM is a generalized input/output control system that extends the techniques of Logical IOCS to the telecommunications environment. Files accessed by the problem programmer are queues of messages incoming from, or outgoing to, remote terminals via communication lines. Even though the time and order of the arrival and departure of messages to and from the central processing unit (CPU) are unpredictable, the support provided by QTAM enables the programmer to handle them as if they were organized sequentially.

A telecommunications system operating under DOS/QTAM is logically divided into two types of programs each of which executes in a separate partition. These types of programs are:

1. a message control program that always operates in the Foreground-one partition, and

2. one or more message processing programs that execute in the Foreground-two and/or Background partitions.

Note: With multitasking support, one or more message processing programs can be executed in the same partition as the message control program. The message control program must have the highest priority in the partition. For a complete discussion of multitasking, see the DOS System Control publication, Form Y29-5017.

A message control program systematically and efficiently controls the flow of message traffic from one remote terminal to another, and between remote terminals and any message processing programs. A message processing program consists of the programming required to process the contents of the messages.

QTAM provides logic in support of both types of programs. Extensive support is supplied for a message control program and includes, via the linkages generated by QTAM macro instructions, the QTAM logic required to:

1. identify the configuration of the telecommunication system to the Disk Operating System,

2. establish and perform the line control procedures required for the various types of terminals and modes of connection, and

3. control the routing of messages in accordance with the requirements of the user.

GET/PUT logic is provided to a message processing program for obtaining messages to be processed and for placing response messages.

The paramount reason for dividing telecommunications support into two types of programs is that message flow to and from the computer is random and proceeds at relatively slow speeds (due to the operating speeds of the terminals supported), while the messages, once delivered to the computer, can be processed at computer speeds. To fully utilize the computing system capabilities, message traffic must proceed asynchronously with message processing. This asynchronous method of execution is based on the completion of awaited events (for example, I/O termination), and on the established priorities of Foreground-one, Foreground-two, and Background.

PHYSICAL ORGANIZATION OF QTAM

This section describes the various parts of the total package called QTAM. Little is said of the function of the various pieces and even less of the logic of their operation. However, it is important to understand what the pieces are, where they come from, how they get into the system, and their relationship to the rest of the package. The following topics are discussed in this section:

1. System Generation.
2. Assembling and Linkage-Editing a Message Control Program.
3. Assembling and Linkage-Editing a Message Processing Program.

SYSTEM GENERATION

When QTAM is called for during a system generation procedure (TP=QTAM operand in SUPVR SYSGEN macro instruction), a number of subroutines collectively called the QTAM SVC/Subtask Control routine are included as a part of the DOS Supervisor Nucleus. These subroutines are then always present in the system, regardless of whether a telecommunications application is being executed.

The QTAM nucleus routine consists of 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

Note: When TP=QTAM is specified, SVC support for BTAM is also included in the nucleus; however, QTAM has no requirement for this BTAM logic.

Other system generation considerations for including QTAM in the system are:

1. Multiprogramming support (MPS=YES operand in SUPVR macro) must be included because QTAM requires at least two partitions in DOS. Because multiprogramming support is required, storage protection (SP=YES operand in CONFG macro) must be specified.
2. The interval timer is required (TIMER=YES operand in CONFG macro) if the polling interval or time stamping options provided by QTAM are desired. Additionally, the timer must be assigned to the Foreground-1 partition (IT=F1 operand of FOPT macro) for the polling interval option.

Note: When multitasking support is included in the DOS Supervisor (AP=YES operand in SUPVR macro instruction), a number of message processing programs can operate in the system. The maximum number of message processing programs to be supported is defined at system generation (TP=QTAMn where n is equal to the number of message processing programs).

After a system generation has been performed, the various QTAM components are in the system libraries on the SYSRES volume. The distribution of the QTAM components is described as follows.

SOURCE STATEMENT LIBRARY: All QTAM macro definitions called for during the assembly of message control and message processing programs reside in this library. Appendix C lists all QTAM macro instructions and indicates what the expansion is for each. QTAM DSECTS are also included in the Source Statement Library and may be included in an assembly via a COPY statement.

CORE IMAGE LIBRARY: The QTAM SVC/Subtask Control routine is a part of the Supervisor nucleus that resides in this library. QTAM transient routines also reside in this library; these routines (QTAM Open and Close executors) are dynamically fetched into the logical transient area as needed for execution.

RELOCATABLE LIBRARY: The remaining components of QTAM are the logic modules required to implement the functions provided by QTAM. These modules reside in the relocatable library and include:

1. Macro-Introduced Modules. These modules correspond to QTAM macro instructions. Each module performs the function represented by its associated macro instruction, whose expansion generates linkage to the module. Also included in this category are certain second-level modules that perform generalized functions required by two or more modules directly associated with a macro instruction. In this publication these modules, whether first or second-level, are defined as External Routines.

2. QTAM Implementation module (IJLQIP).

3. Physical Input/Output module (IJLQRW).

4. Line Appendage module for PCI and Program Check (IJLQLC).

5. Normal Line End Appendage and ERP module (1EDQLA).

6. Disk Input/Output module (IJLQDA).

7. Audio Line Appendage module (IJLQAA).

8. 7772 Disk Appendage module (IJLQAD).

9. GET/PUT logic modules. Seven GET modules and four PUT modules are provided. The PUT and four of the GET modules correspond to the message/segment/record/audio message options. The other GET modules correspond to the three combined options of audio message with message, segment, or record.

10. WTTA Line Appendage module (IJLQTA).

Appendix D contains an alphabetical listing of all the QTAM logic modules.

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 transient 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 will normally not be extensive.

The assembled object module is then link-edited to include the necessary external routines from the relocatable library. 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 remainder of the required QTAM logic modules are also incorporated (via V-type address constants) into the message control program during this linkage-editing procedure.

A large module called the QTAM Implementation module (IJLQIP) is included from the relocatable library, because of a V-type address constant generated in the expansion of the DTFQT macro for the DASD queues, the 7772 DCV vocabulary, and the 7770 line group files. The Implementation module contains two distinct types of routines - distinct as far as their logical relationship to the rest of the system. The two types are:

1. Problem program routines, executed enabled to all interruptions as part of the message control program. These routines receive control through branches from the external routines that are link-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 SVC Subtask Control routine in the nucleus.

The Line Appendage, Audio Line Appendage, WTTA Line Appendage, and Disk I/O modules are link-edited from the relocatable library because of V-type address constants generated by the DTFQT macros for the Line Group, WTTA line group, Audio Line Group, and DASD Queue files, respectively. The 7772 Disk Appendage is link-edited from the relocatable library because of a V-type address constant generated by the BUFARU macro instruction. All these modules contain a third distinct type of routine: the I/O Appendage. These I/O appendages are executed disabled to all interruptions, again logically as part of the QTAM nucleus "task." The appendages receive control from the I/O Interruption Handler routine in the DOS Supervisor.

Note: The logical relationship of the two types of routines in the Implementation module with each other and with the I/O appendages is discussed more fully in the next section. From the point of view of physical organization, this collection of routines represents no more than a convenient and efficient packaging technique. The Implementation module,

together with the I/O appendages, can in no way be thought of as a "program."

The last module linkage-edited into the message control program is the Physical I/O module. It is included because of a V-type address constant generated, again, by the expansion of the DTFQT macro for the DASD Queues file. This module contains routines that generate and initiate execution of channel programs on the communication lines. These routines run in problem program state as part of the user's message control program.

The resulting output of the Linkage Editor run is then cataloged as a single phase into the core image library, ready to be loaded for execution. This message control program phase must be executed in foreground-1, the highest priority partition in the system.

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 file definition macro instructions. When this is the case, no QTAM external routines are required to be link-edited with the assembled object module. The only QTAM modules linkage-edited with the assembled object module are the selected GET and PUT modules. The modules selected depend on the unit or combination of units of data processed by the program: record, segment, message, or audio message.

The appropriate GET module(s) are included from the relocatable library because of a V-type address constant generated by the expansion of the DTFQT macro for the Main Storage (MS) Process Queue(s). The appropriate PUT module is included because of a V-type address constant generated by the expansion of the DTFQT macro for the MS Destination or Audio Output Queues.

An installation also writes at least one message processing program that uses the following macro instructions to examine and modify the status of the message control program and/or to initiate termination of the message control program:

```
CHNGP
CHNGT
CKREQ
CLOSEMC
COPYC
COPYP
COPYQ
COPYT
RELEASEM
RETRIEVE
STARTARU
STARTLN
STOPARU
STOPLN
```

When any of these macros are used, the linkage editor includes the corresponding external routines in the load module.

The resulting output of the Linkage Editor run is then cataloged into the core image library, ready to be loaded for execution. The message processing program phase may be linkage-edited to execute in either the foreground-2 or the background partition. With multitasking, however, it is possible to execute several processing programs in the foreground-one partition as tasks with lower priorities than the message control task.

The preceding section explained how the physical pieces of QTAM are positioned in main storage. This section explains 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 DISK OPERATING SYSTEM CONTROL PROGRAM STRUCTURE

The various pieces discussed under Physical Organization of QTAM can be grouped into three logical categories:

1. The Message Control Program
2. Message Processing Program(s)
3. The QTAM Supervisory Routines

The message control and message processing programs are both run under control of the DOS task management routines. From the point of view of DOS task management, they are in no way different from any other problem programs. They are scheduled and dispatched according to the priorities indicated in the Program Information Blocks (PIBs) for the partitions in which they are being run.

The third category, QTAM Supervisory Routines, is all that is left over after distinguishing and separating the two processing programs. These routines are executed as SVC handling routines or as asynchronously scheduled I/O interruption handling routines. Strictly speaking, they are executed as part of the message control and message processing programs. Practically speaking, however, it is more meaningful to think of these as a separate category, outside of the task framework established by DOS task management. The discussion in this section is primarily an attempt to explain the nature of this third category as it is to be understood in relation to the other two. The discussion continues under QTAM Supervisory Routines.

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 link-edited with the assembly output.

   Note: If the LIST macro instruction is used, a single supervisory routine called the Distribution List routine, in a module named IJLQDL, is link-edited into the message control load module. This routine is one of the supervisory routines, and is not part of the problem program.
3. Three of the routines in the Implementation module:
   - LPS Control
   - Buffer Recall/Cleanup
   - Free BRB
4. The QTAM Physical I/O module.
5. Two routines link-edited with the Audio Line Appendage module:
   - ARU Receive
   - ARU Internal

Figure 1 shows a simplified flowchart of this problem program. The flowchart is included here to show how the three problem program routines in the Implementation module, the QTAM Physical I/O module , and the two audio routines are related to the rest of the message control program.

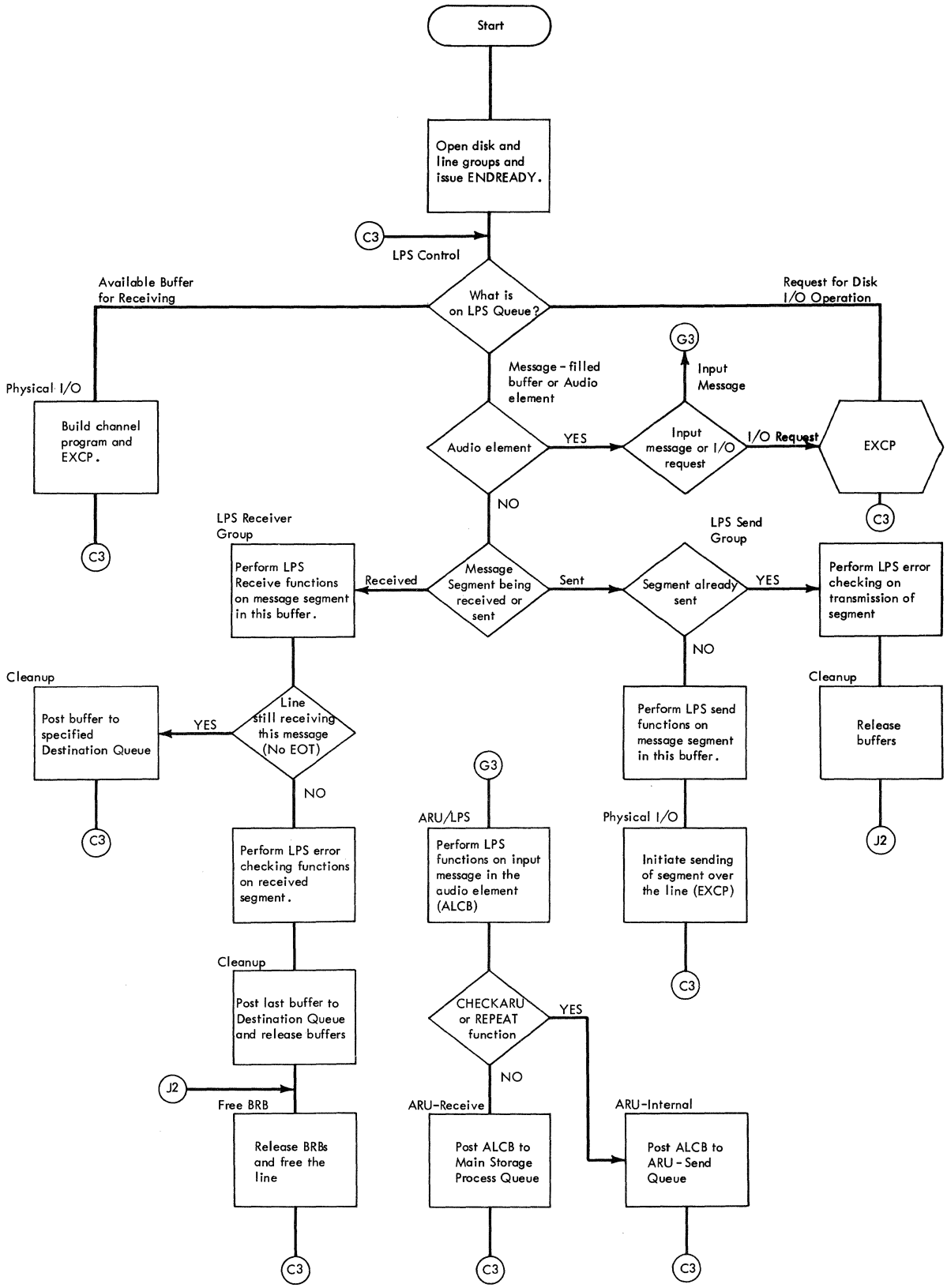


Figure 1. Message Control Problem Program

MESSAGE PROCESSING PROBLEM PROGRAM

A message processing problem program includes the assembled user code and external routines link-edited with it, including the GET, PUT routines. The only difference between a QTAM message processing program and any other processing program is the requirement for and the implementation of cross-partition communication. The various macro instructions that can be used in a message processing program are handled as follows:

1. COPYP, COPYT, and COPYQ present no problem. The corresponding external routine simply reads the requested information from the foreground-one partition using address pointers stored in a QTAM vector table and in the terminal table.

2. All other 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.

From the point of view of logical organization, unusual operations are noticed in the following cases: PUT, GET (for audio processing), CLOSEMC, STARTARU, and STOPARU macro instructions. To avoid including a large amount of code in supervisory routines for each of the corresponding modules, certain code that must be executed in supervisor state is packaged within these modules. The SVC routine entered as a result of one of these macro instructions branches back to these sections of code in the corresponding problem program modules to execute them in supervisor state.

Note: Operations under extended multiprogramming capabilities (multitasking) that allow one-partition processing do not differ from the above description.

QTAM SUPERVISORY ROUTINES

Within the framework of the DOS control program is the third category, the QTAM supervisory routines. From the point of view of physical organization, the QTAM supervisory routines consist of:

1. The QTAM subroutines within the Supervisor nucleus.

2. Those routines within the Implementation module (in partition F1) that are executed in supervisor state. This includes all but the three identified as part of the message control problem program.

3. The Disk I/O, Line Appendage, IBM 2260 Appendage, and the optional WTTA Line Appendage modules.

4. The Distribution List routine link-edited with the message control program.

5. The optional IBM 7772 Disk Appendage and Audio Line Appendage modules, except the two audio routines link-edited with the Audio Line Appendage and part of the message control program.

6. Part of the GET (for audio processing), PUT, CLOSEMC, and STARTARU/STOPARU modules in the message processing problem program partition(s).

7. The Checkpoint routine link-edited with the message control program when the checkpoint/restart option is specified.

From the point of view of the interruption handling facilities of the Disk Operating System, the QTAM supervisory routines consist of:

1. SVC routines, entered by SVCs 30 and 31 from problem program partitions.

2. Asynchronously scheduled I/O interruption handling routines, entered from the DOS I/O Interrupt Handler.

While both of these points of view are correct, neither is very helpful in understanding the logical organization of QTAM. For example, a routine within a QTAM 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 DOS 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, and it happens to be within a second control program. Later, it will be explained how the two overlap. First, however, let us

look at 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 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

To perform this allocation function efficiently, QTAM breaks the system resources into the smallest practical number of pieces, as follows:

1. The work to be done is broken into many separate units called QTAM subtasks. 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. 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 communication lines and the disk queue. Only that I/O path absolutely required at a given time need be tied up for a given function.

Allocation of the time resource is called scheduling, while "allocation" usually refers to physical resource allocation only. However, one of the most important design attributes of the QTAM Control Program, distinguishing it from other possible designs (including the Disk Operating System itself) is that the entire allocation function is performed by a single mechanism. This allows a complete interdependence of time allocation (scheduling) and physical resource allocation. Scheduling becomes a function of allocation, and allocation becomes a function of scheduling.

The following sections describe the resource allocation mechanism of QTAM. The key to the mechanism is the QTAM Ready Queue. This Ready Queue is 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 SVC Subtask Control routine performs a queue-management function that includes dispatching the subtask at the top of the Ready Queue.

QUEUE MANAGEMENT

Both physical resources and subtasks are represented to the queue management routines by control blocks. The resources are broken into elements, with each element represented by an Element Control Block (ECB). Subtasks are represented by Subtask Control Blocks (STCBs). Queues of elements are allowed to build up as a chain of ECB's starting with an address pointer in a Queue Control Block (QCB). Queues of requests for elements also may build up as a chain of STCBs starting with another address pointer in a QCB. A detailed description of all control blocks used by QTAM is contained in Appendix B.

Element Control Blocks

Five main types of permanent Element Control Blocks (ECBs) are:

1. Buffer ECBs.
2. Communications line ECBs.
3. Audio communications line ECBs.
4. Buffer request ECBs.
5. 7772 DCV buffer ECBs.

Buffers are areas of main storage that contain message data and/or control information. The first 8 bytes of each buffer comprise an ECB. As with all QTAM elements, the "identity" of the buffer at a particular time depends solely upon the queue its representative ECB is chained into at that time. The buffer proper is always physically identifiable as a fixed number of bytes of main storage.

If the ECB representing the buffer is chained into a QCB for a Destination queue, the buffer is full; that is, it contains a message segment to be transmitted to a destination. When the same ECB is subsequently chained into the Available Buffer QCB, the element involved is now an available buffer, even though there has been no change in the physical storage area.

7772 DCV buffers are main storage areas that contain the digitally coded voice (DCV) words dynamically retrieved from the 7772 DCV vocabulary file. The first eight bytes of each 7772 DCV buffer comprise an ECB. An ECB representing an available 7772 DCV buffer is chained into the QCB for the corresponding 7772 DCV buffer queue. An ECB representing a nonavailable 7772 DCV buffer is associated with an operative audio line.

Audio and nonaudio communications lines are represented to QTAM through the Audio Line Control Block (ALCB) and the Line Control Block (LCB), respectively. There is an LCB or an ALCB for each line. Therefore, the LCB and the ALCB themselves are treated as the resource element. The ECB is contained in the first eight bytes of the LCB or ALCB.

To avoid preassigning buffers before they are actually needed, QTAM uses Buffer Request Blocks (BRBs) to queue buffer requests, which is explained later (see Outline of QTAM Operation). These BRBs are elements. The ECB is contained within the BRB. The number of BRBs in the system is determined by the number of buffers in the buffer pool; there is one BRB per buffer. Thus, this pool of BRBs is itself a pool of resources to be allocated to the various subtasks that use them.

Subtask Control Blocks

The two types of Subtask Control Blocks (STCBs) are:

1. Truncated STCBs
2. Full STCBs

Truncated STCBs represent subtasks that are executed in supervisor state. These subtasks are performed by routines that are packaged within the following modules: QTAM Implementation, Disk I/O, Audio Line Appendage, 7772 Disk Appendage, GET (for audio processing), PUT (for audio processing), CLOSEMC and STARTARU/STOPARU (but also including 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 and message processing problem programs. At this point there is an overlap of the Disk Operating System control program structure with the QTAM control program structure. A QTAM problem program subtask is created when an SVC 30 or 31 is issued within a Disk Operating System task. More specifically, a control block called a full STCB is initialized to represent the problem program subtask and is used as a QTAM STCB. As a subtask then, the problem program is placed under the subtask management of QTAM and must contend for control in that multitask environment before it is released to contend with other Disk Operating System tasks in the system. This is explained more fully in the following sections.

Note at this point, however, that every problem program request that results in a QTAM SVC 30 or 31 causes a subtask to be created. These problem program subtasks are always lower in priority than any QTAM implementation subtask; thus, they are never considered for dispatching until all of the internal implementation subtasks have done all of the work that can be done with the resources available. There is one full STCB per problem program task preassembled into the QTAM nucleus.

Queue Control Blocks

The QTAM Ready Queue can be thought of as a queue of queues, each queue being associated with a QCB. The following is a list of various types of queues that may appear at any given time on the Ready Queue. A more complete and detailed list is given in Appendix A.

AVAILABLE BUFFER QUEUE: This queue keeps track of unassigned buffers. The element chain is the chain of all buffers that are currently 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 that is used whenever a buffer is made available.

LPS QUEUE: This queue passes elements from the QTAM control program to the message control problem program. As shown in Figure 1, 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 section.
3. A request for a disk I/O operation to be started.

4. An ALCB with a full audio input buffer to be passed through an ARU/LPS section.

5. An ALCB requesting an I/O operation on an audio line.

6. A 7772 DCV buffer requesting a disk read operation.

This is the QCB that the LPS Control routine in the message control problem program waits on.

MAIN STORAGE PROCESS QUEUE: This queue passes 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 on.

DASD DESTINATION AND DASD PROCESS QUEUES: There is a QCB for every destination queue and every process queue defined by the TERM and PROCESS macro instructions in the message control program. When a buffer is posted to one of these queues, it is never physically chained to the QCB. Instead, the buffer is posted directly to the Ready queue and, when it reaches the top, it is removed and the indicated QCB is put in its place. The STCB chain from one of these QCBs always ends with the STCB for the DASD Destination subtask. It may be preceded by the STCB for the line's Send Scheduling subtask (if it is a destination queue) or the GET-Scheduling subtask (if it is a process queue).

INACTIVE BRB QUEUE: This queue keeps track of inactive buffer request blocks. The element chain is the chain of all BRBs that are not currently assigned. The STCB chain may contain the STCB for a line's Receive-Scheduling subtask and/or one or more Send-Scheduling subtasks.

ACTIVE BRB QUEUE: This queue passes 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 special purpose BRBs containing the CCWs used to transmit idle or other specified characters when certain line control characters are encountered in an outgoing message. When one of these line control characters is encountered by the Pause routine in the Send portion of the LPS, the problem program waits on this queue to obtain one of these BRBs. All Pause BRBs initially appear in the element chain of this queue as a result of the BUFFER macro expansion.

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 out 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 communication line. The QCB is created in the LCB itself when the LCB is encountered on the Ready Queue, 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 created for the line unless there is already a Send-Scheduling subtask waiting for the line.

ARU SEND QUEUE: There is only one QCB for all communications lines. This QCB is part of the audio implementation programs link-edited with the Audio Line Appendage module. ALCBs containing a full address-chain buffer are posted to this queue to send the audio output message on line. The STCB chain is limited to the STCB for the ARU Send subtask.

7772 DCV BUFFER QUEUE: There is a QCB for each 7772 Audio Response Unit with at least one line using DCV words dynamically retrieved from the 7772 vocabulary file. A QCB created in the message control program through the BUFARU macro expansion keeps track of the unassigned DCV buffers of one or more 7772 line groups. The element chain is the chain of all DCV buffers that are currently not assigned to the corresponding 7772 line groups. As soon as a DCV buffer is no longer needed, it is posted to its queue. The STCB chain for this QCB generally contains the STCB for the Queue Insert subtask, but it may contain the 7772 DCV buffer subtask when no DCV buffer is available and one or more audio lines are waiting for an available DCV buffer.

QWAIT AND QPOST

A subtask requests a resource from a queue by issuing a QWAIT on the associated QCB. A subtask passes a resource with which it is finished to another subtask or a problem program 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 Ready Queue by issuing an SVC 30.

Note: All QTAM SVC's are macro-generated. The problem programmer should never have to issue one directly.

The Supervisor SVC Interrupt Handler initializes a PIB and passes it to the Entry Interface subroutine in the QTAM SVC/Subtask Control routine. The PIB information is used to initialize a full STCB to represent re-entry to the QTAM problem program, and the requesting program is placed in the wait state.

The full STCB contains the address of a special QCB labeled QSVCQCB. The address of the QCB for the element queue being waited on is passed in register 0. If the element is available, its address is placed in register 1; the requesting program is removed from the wait state, and control is passed to the problem program via the DOS Supervisor task selection mechanism.

If the element is not available, the full STCB is chained to the QCB of the element chain being waited on. The requesting problem program is left in the wait state, and control is returned to the System Supervisor. The DOS task selection routine then dispatches some other task if there is one waiting (for example, a message processing program in a lower priority partition). Otherwise, it places the entire system in the wait state.

When some other subtask subsequently posts an element to the queue that the problem program waited on, the problem program will be dispatched by QTAM by turning off the wait flag in the PIB for the program. The problem program will then be dispatched in its proper task priority by Disk 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 31. As with the QWAIT, the full STCB contains the address of, and is chained to, the QSVCQCB QCB. The QCB of the queue that the element is being posted to is passed in register 0 and the address of the ECB for the element being passed is in register 1. The ECB is placed on the Ready Queue. If a subtask is waiting for the element, it is dispatched. If no subtask is waiting for the element, the ECB is chained to the proper QCB. When all items on the Ready Queue are dispatched, the problem program regains control.

QWAIT from Internal Implementation Subtask

When one of the implementation subtasks requires an element, it looks directly at the QCB for the element queue being waited on. If the element is available, the subtask removes it from the chain and continues (relinking the element chain, if necessary). No SVC is issued.

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

QPOST from Internal Implementation Subtask

When one of the QTAM implementation subtasks has an element to pass to the Ready Queue, it branches directly to the Post subroutine in the QTAM SVC/Subtask Control routine (no SVC is issued). The ECB is placed on the Ready Queue and contains the address of the QCB to which it was posted. The STCB for the subtask that posted the element is left chained to the QCB that it was already on and that subtask resumes operation.

Summary: A QWAIT chains the STCB of the requesting subtask to a QCB. The QCB may or may not be on the Ready Queue. A QPOST places an element's ECB directly on the Ready Queue. The ECB contains a pointer to the QCB to which it is posted. When the ECB reaches the top of the Ready queue, it is replaced at the top by the QCB and the first subtask chained to the QCB is dispatched.

QDISPATCH

The QTAM SVC/Subtask Control routine in the Supervisor nucleus provides the overall queue management facilities which include:

1. Interfacing with the Disk Operating System Supervisor.
2. Placing problem programs in wait state; then posting them complete.
3. Chaining ECBs to the Ready Queue and STCBs to QCBs in the proper priority sequence.
4. Dispatching the highest priority subtask.

The QTAM nucleus routine is comprised of several subroutines, and each is discussed in the section QTAM SVC/Subtask Control Routine later in this manual. At this point, however, we can look at the queue management facility as a whole.

Figure 2 shows a very generalized flowchart of the QTAM nucleus. Part 1 illustrates the preceding discussion of the QPOST and QWAIT. The test made at E2 is the Qdispatch function. The Qdispatch subroutine 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 seven commonly used priorities, indicated by a hexadecimal code in the ECB.

HIGHEST CODE (FE): This highest priority code is given to special audio elements enabling truncated subtasks to be dispatched immediately. These subtasks, executed in supervisor mode, are located in the four GET modules working on audio messages, the audio PUT, CLOSEMC and STARTARU/STOPARU modules.

SECOND-HIGHEST CODE (FD): This priority is given to all audio elements (Audio Line Control Blocks, 7772 DCV buffers) being processed by the message control program.

THIRD-HIGHEST CODE (FC): This priority is given to all Audio Line Control Blocks being passed to message processing programs.

FOURTH-HIGHEST CODE (EC): The only element ever given a code of EC is a BRB. This is done in two instances:

1. When the buffer request is for a disk operation. This is done to optimize scanning the disk queue area.
2. When the buffer request is made by the Line PCI routine following the first PCI on a line-receive operation. The high priority causes additional buffers needed for a line-read operation to be assigned as rapidly as possible.

FIFTH-HIGHEST CODE (F4): This priority is given to all nonaudio elements being passed to implementation subtasks that run disabled to interruption.

SIXTH-HIGHEST CODE (E0): This priority is given to all nonaudio elements being passed to the message control program.

LOWEST CODE (DC): This lowest priority code is given to all nonaudio elements being passed to message processing programs.

Qdispatch follows the address pointer in location QSVCRDYQ to the item at the top of the Ready Queue. The item will be either an ECB or a QCB. Qdispatch examines the key field in the first byte to determine what the item is:

Key = 0: All ECB's have a key of zero.

Note: There is one special case where a full STCB appears directly on the Ready Queue instead of chained to a QCB. This is the full STCB created during initialization, after the ENDREADY macro instruction is executed and the first QTAM SVC is issued in the LPS Control routine. This full STCB appears to Qdispatch as an ECB pointing to a location labeled QSVCQCB at QSVCRDYQ-8. Therefore, the address at location QSVCRDYQ appears as the head of a STCB chain in a pseudo QCB labeled QSVCQCB and the program represented by the full STCB is given control.

There is always a subtask waiting on the element, though in some cases it may be only an interim subtask that removes the element from the Ready Queue and chains it to a QCB element chain (i.e., the Queue Insert subtask.)

Key = 2: A key of two indicates a QCB with a subtask at the top of its STCB chain that is "ready", or "not waiting" for an element. The STCB was chained to the QCB as the result of a QPOST. It will now regain control.

Key = 3: A key of three indicates a QCB with a subtask at the top of its STCB chain that is "not ready", or "waiting" for an

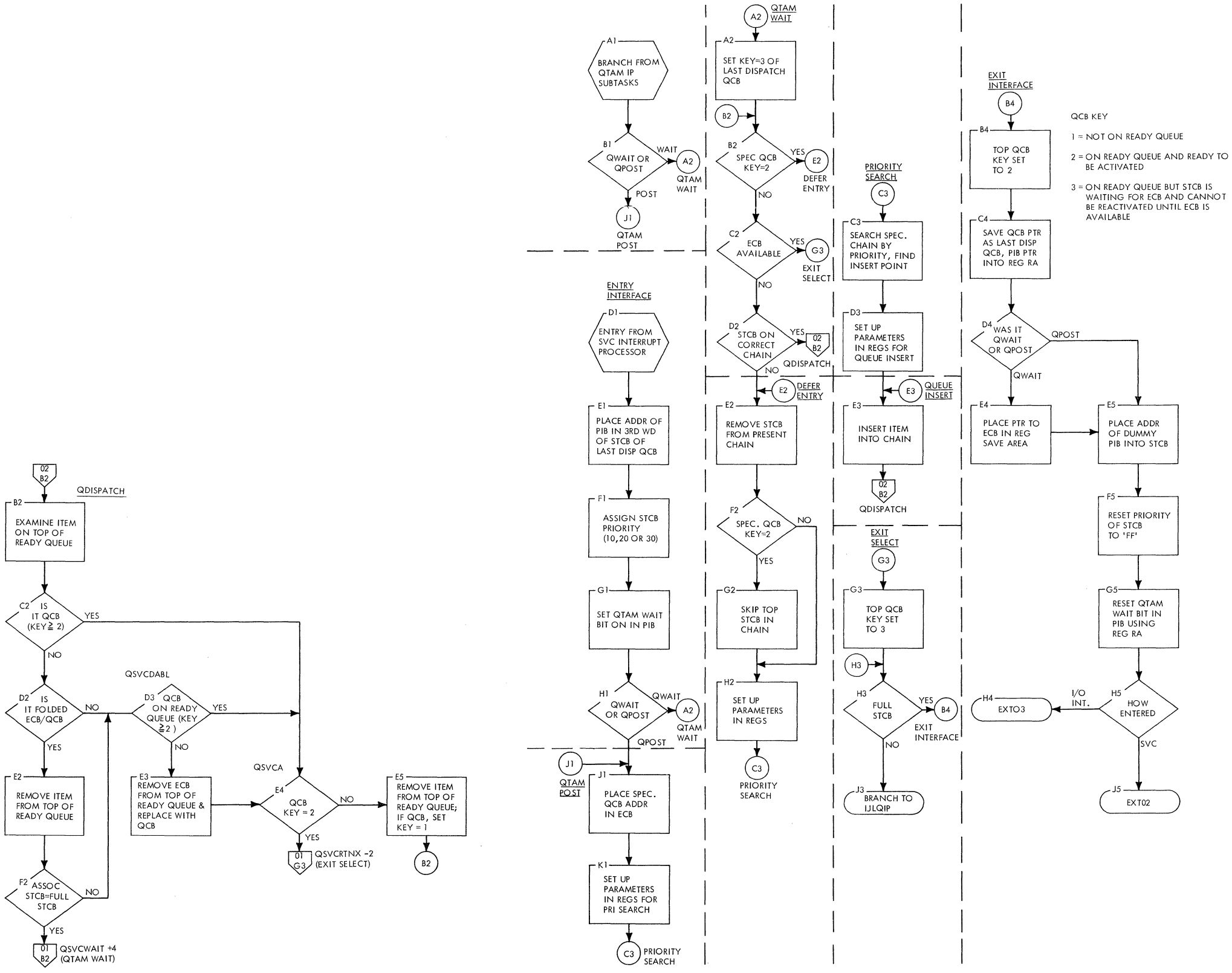


Figure 2. QTAM Nucleus

element. The STCB was chained to the QCB as the result of a QWAIT. Note that if an element had been available, the subtask that issued the QWAIT would have regained control without the Qdispatch routine being entered. Because the element is not available, the key of the QCB is set to 1, and the link address is changed to remove the QCB from the Ready Queue.

The flowchart in Figure 2 further shows how control is passed to the dispatched subtask. If the subtask is represented by a truncated STCB, the Exit Select subroutine simply branches to the entry point of the implementation subtask. If it is a problem program (full STCB), the Exit Interface routine posts completion of the SVC in the PIB for the program and returns control to the Supervisor. If the SVC request from the problem program could not be satisfied, completion is posted in a dummy PIB before return to the Supervisor, and the problem program remains in the wait state.

In the latter case, QTAM is placing one problem program in wait state and enabling another problem program that was previously placed in wait state, or that was ready but in a lower priority partition, to be dispatched again by the Disk Operating System task supervisor.

One dummy element control block indicates the end of all element chains and is permanently the last item on the Ready Queue. When this ECB reaches the top of the Ready Queue, the last QTAM problem program is placed in wait state.

INITIAL STATUS OF THE QTAM CONTROL PROGRAM

This section describes briefly the initial status of QTAM, that is, the status at the time the message control program phase is loaded from the core image library for execution. The initial status of the QTAM queues and the initial location of the primary resources are represented by the assembly listings of the various QTAM components and are summarized here. Detailed information concerning the initial status of QTAM can be obtained from the assembly listings.

Ready Queue: The initial item at the top of the Ready Queue is a full STCB to be used for handling the first QTAM SVC issued in a problem program.

Available Buffer Queue: All buffers initially appear in the element chain of this queue as a result of the BUFFER macro expansion.

Additional-CCW Queue: All Pause BRB/CCWs initially appear in the element chain of this queue as a result of the BUFFER macro expansion.

Inactive-BRB Queue: All BRBs initially appear in the element chain of this queue as a result of the BUFFER macro expansion. The STCB chain initially contains the STCB for the Queue Insert subtask.

Communication Line Queues: There is one such queue for each line in the system. The QCB is contained in the LCB generated by the expansion of the DTFQT macro for the line group file. If receiving (CPRI=R) or equal (CPRI=E) priority is specified, the STCB for the Receive Scheduler subtask is initially first in the line's STCB chain. If sending (CPRI=S) priority is specified, the STCB for the Qdispatch subtask is initially first in the STCB chain.

DASD-Destination Queue: The element chain of this queue initially contains the dummy last element labeled IJLQIP5F which indicates that no buffers are in the chain. Initially, the STCB for the line's Send Scheduler subtask is first in the STCB chain.

DASD-Process Queue: The element chain initially contains only the dummy last element labeled IJLQIP5F. The first STCB is initially the one for the Get Scheduler subtask.

ARU Send Queue: The element chain initially contains only the dummy last element labeled IJLQIP5F, which indicates that no ALCBs are in the chain.

7772 DCV Buffer Queues: All DCV buffers associated with one or more 7772 line groups initially appear in the element chain of the corresponding queue as a result of the BUFARU macro expansion. The STCB chain initially contains the STCB for the Queue Insert subtask.

All Other Queues: The element chain of all other queues initially contains either:

1. The dummy last element (IJLQIP5F in module IJLQIP) which indicates that no elements are in the chain initially, but that elements may appear in the chain when execution begins; or
2. The address of the queue itself which indicates that no elements ever appear on the element chain. The STCB chain for all queues not mentioned is limited to the STCB for the subtask having the same name as the queue. For example, the STCB chain of the Disk I/O queue initially contains, and is limited to, the STCB for the Disk I/O subtask.

This section describes the functional flow of QTAM operations at the component level. QTAM is composed of five major functional components:

1. Message control problem program-- The portion of the message control program phase that executes in problem program state. Hereafter, in this discussion, this component is referred to simply as the message control program.

2. Message processing programs-- One or more message processing programs may be executed in the foreground-one (only with multitasking), foreground-two, and/or background partitions.

3. QTAM nucleus-- The SVC/Subtask Control routine is the supervisor-resident component of QTAM that handles QTAM SVCs and controls the dispatching of QTAM subtasks.

4. QTAM subtasks-- A portion of the message control program phase that executes in supervisor mode as a logical extension of the QTAM nucleus. Each subtask is represented to the QTAM nucleus by a STCB and is selected for execution when the unit of work it performs is required.

5. QTAM appendages-- A portion of the message control program phase that services interruptions caused by QTAM line or disk operations. This component runs in supervisor mode and is entered from the DOS Supervisor each time an interruption caused by a QTAM input/output operation occurs.

   Note: QTAM transient routines might be considered as a sixth major component. However, for the sake of continuity, the QTAM Open routine is discussed as a part of the message control or message processing program depending on which type of QTAM file is being opened.

This section discusses the following three major subjects, since all QTAM operations (audio and nonaudio) use the same message control and message processing programs:

1. The message control program. This describes step by step the total process initiated for receiving and sending a message over a communication line.

2. The message processing program. This describes the message transfer to and from a message processing program.

3. IBM 2260-2848 Local operations. This describes the operations for the IBM 2260-2848 Local Display Complex, with emphasis on the areas differing from remote operations.

The interaction of the QTAM subtasks and appendage is extensive for both operations. Therefore, descriptions of these components are included as required in both subjects.

Because the functional flow of audio and nonaudio operations is very different, each major subject is covered in two sections: nonaudio and audio applications.

Processing in all the QTAM components is initiated as a result of interrupts (SVC, disk, and line) that occur during the receiving and sending of a message or the transferring of a message to or from a message processing program. These interrupts result in the processing of one or more asynchronously operating subtasks and/or an appendage. The QTAM subtasks communicate with one another and with the message control or processing program by means of the QPOST and QWAIT functions (see section on QPOST and QWAIT). When a subtask (or program) has an element to be processed by another subtask, the QPOST function is used. When a subtask (or program) is ready to receive an element, the QWAIT function is used. Subtask selection is performed by the QTAM nucleus and is controlled by the ordering of items on the Ready queue as discussed previously under Queue Management.

To avoid excessive repetition, the operation of the QTAM nucleus which is entered to perform a QPOST or QWAIT function is not included in this description. Instead, the switch from one subtask to another is indicated by "*** ENTER subtask name SUBTASK***". The return to a problem program (message control or processing) that issued a QPOST or QWAIT is indicated in a similar manner.

Note: The following discussion assumes that multitasking support is not included. This affects terminology rather than QTAM operations.

The interference of one line with another is handled by the queueing provided within the QPOST/QWAIT functions. For this reason and for the sake of simplicity, the operation of more than one line is not discussed. In this manner, the logical sequence of events for an operation can be described without regard to other items on the Ready queue or unrelated interrupts that may occur. For example, when an element is posted to a queue, it is assumed that no unrelated subtask is contending for control; therefore, the subtask associated with that queue is activated immediately.

Appendix E is a foldout chart showing the functional flow of the four QTAM components discussed in this chapter. The four major components are separated by solid lines. The subtasks, appendages, or modules within a component are divided by broken lines. The labels on the chart are the names of routines or LPS subgroups and define the boundaries of a particular routine or subgroup. Detailed discussions and flowcharts of all the routines described in this chapter are contained elsewhere in this publication (refer to the table of contents for page numbers).

MESSAGE CONTROL PROGRAM FOR A NONAUDIO APPLICATION

After the message control program phase has been loaded from the core image library into the foreground one partition, it is entered for execution by the DOS Supervisor.

ENTER MESSAGE CONTROL PROGRAM

INITIALIZATION

1. QTAM Open Monitor/Open DASD Message Queues File Routine, Phase 1: The address of the QTAM Vector Table is placed into the DOS Communication Region so that QTAM routines requiring information from the Vector Table can access it. The extent data for these QTAM files is read, and the File Protect subroutine is called into the transient area. It is passed a parameter directing it to call into the transient area the open module that must be executed next.

2. Open DASD Message Queues File Routine, Phases 2 and 3: If the DASD Message Queues file is a multivolume file, this routine prepares the proper initialization functions. Extent data for the next extent (if any) is read, and Phase 3 is called into the transient area. Phase 3 tests if this is the last extent for the file. If yes, return is made through the File Protect subroutine to the system Open Monitor. If no, return is made to Phase 2.

3. Open Checkpoint/Restart Routine, Phase 1: If a checkpoint records file is to be opened, this routine tests to determine if a restart is to be performed. If so, Phase 2 of the Open Checkpoint/Restart routine is called into the transient area. (This discussion assumes that a restart is not to be performed.) The routine checks the size of the checkpoint work area and formats the disk extents with dummy records. Return is made to the QTAM Open Monitor.

4. Open Line Group/MS Queues Routine: A Set Address (SAD) command is prepared to set the telecommunications control unit (except for an IBM 2701) to the proper transmission speed for the terminal devices in the line group. If the line is a nonswitched line, an Enable command is prepared. If neither command is required, a No Op command is prepared. An SVC 31 (QPOST) is issued to post the LCB to itself. This causes the line to be scheduled for a receiving operation.

ENTER RECEIVE SCHEDULER SUBTASK

5. Receive Scheduler Routine: If there is an active entry in the polling list for the line, exit is made to the BRB Ring routine to continue initialization procedures for receiving. If the end of the polling list is detected, exit is made to the End of Poll Time Delay routine to observe a polling interval, if any.

6. BRB Ring Routine: A ring of buffer request blocks (BRBs) is built for dynamic buffer allocation (BRBs are obtained from the Inactive BRB queue). The number of BRBs in the ring is equal to the value specified in the BUFNO operand. The address of the first BRB in the ring is stored in the LCB so the Activate routine can access it later, and the read-initial operation code is placed in the LOPC field of the LCB. The first BRB is then posted to the Active BRB queue with a high priority (X'EC') to cause immediate servicing of the request for a buffer.

ENTER ACTIVE BRB SUBTASK

7. Active BRB Routine: Recognition of the high priority BRB causes an exit to the Buffer-BRB routine when a buffer is available (an available buffer is assumed for this description).

8. Buffer-BRB Routine: An empty buffer is obtained from the Available Buffer queue, and it is assigned to the line by placing the address of the LCB in the buffer prefix. The buffer is then posted to the LPS queue with a priority of X'E0'.

RETURN TO MESSAGE CONTROL PROGRAM

9. QTAM Open Routine (continued): If there are other lines in the line group, an SVC 31 (QPOST) is issued and Steps 3 through 6 are repeated for each line. Thus, when the line group has been completely opened, the following conditions exist:

   a. there is a BRB ring for each line in the group.

   b. there is one empty buffer for each line in the group chained into the LPS queue.

10. ENDREADY Macro Instruction: The user's registers are saved in the user save area. If the Checkpoint Records file has been opened, a QPOST (SVC 31) is issued to post the checkpoint element to itself. This action causes the interval timer to be set for the first checkpoint interval. (This discussion assumes that the checkpoint interval method of checkpointing is used.)

ENTER CHECKPOINT SUBTASK

11. Checkpoint Routine: The user-specified interval is obtained and is passed to the Time Delay routine.

12. Time Delay Routine: A special element representing a request to set the timer for the specified interval is posted to the LPS queue. Control returns to the Message Control Program.

RETURN TO MESSAGE CONTROL PROGRAM

13. ENDREADY Macro Instruction: Exit is made to the LPS Control routine.

14. LPS Control Routine: A QWAIT (SVC 30) is issued to wait on the LPS queue. The timer request element posted to this queue in Step 12 is returned, and a SETIME macro instruction is issued to set the timer for the specified checkpoint interval.

RECEIVING INITIATION

1. LPS Control Routine: An SVC 30 is issued to QWAIT for the next item on the LPS queue. The QTAM nucleus removes an empty buffer from the LPS queue and returns it to the LPS Control routine. (The buffer previously was placed on the LPS queue by the Buffer-BRB routine.) The registers are initialized for the Activate routine, and the buffer is passed to that routine.

2. Activate Routine: The address of the first BRB in the ring formed for this receiving operation is obtained from the LCB for the line. A CCW for reading data into the entire buffer is prepared in the first two words of this BRB (the third word contains the TIC address to the next BRB in the ring). Idle characters are inserted into the buffer per the value specified by the LPSTART macro instruction. The BRB is passed to the Channel Program Generator routine.

3. Channel Program Generator Routine: The CCWs for terminal selection (polling) and reading the first segment are prepared (read-initial channel program). In preparing the channel program, the CCW for the buffer is transferred from the passed BRB to the channel program area, and the TIC command is moved from the BRB to the end of the CCWs in the channel program area.

   Note that the TIC address is invalid at this point because a buffer has not yet been assigned to the next BRB in the ring. The PCI flag is set in the read data CCW to cause a program-controlled interrupt (PCI) at the start of this first buffer. An EXCP (SVC 0) is then issued to start the I/O operation. After the I/O operation has been started, exit is made to the LPS Control routine.

4. LPS Control Routine: QWAIT for the next item on the LPS queue. If there are other lines in the system, steps 2 through 4 are repeated until input operations have been started on each line. When an I/O operation has been started for each line, the LPS Control routine finds no further buffers on the LPS queue, and the message control program enters a wait state. At this point a message processing program in a lower priority partition is given control through the DOS task selection mechanism. Subsequent PCIs and I/O interruptions cause buffers to be posted to the LPS queue, thereby allowing the message control program to proceed.

FIRST PCI (RECEIVING)

It was emphasized under QTAM Opens that initially only one buffer is assigned to a line for receiving. QTAM causes a PCI to occur at the start of the first buffer so that an empty buffer can be assigned to each of the remaining BRBs (after the first) in the ring formed for receiving the message. The DOS Supervisor passes control to the QTAM Line Appendage which recognizes the PCI and enters the Line-PCI routine.

1. Line-PCI Routine: The remaining BRBs (after the first BRB) in the BRB ring are posted to the Active BRB queue with high priority (X'EC').

ENTER ACTIVE BRB SUBTASK

2. Active BRB Routine: Recognition of the high-priority BRB causes an exit to the Buffer-BRB routine if a buffer is available (an available buffer is assumed for this description).

3. Buffer-BRB Routine: Another empty buffer is assigned for receiving on the line and is posted to the LPS queue with a priority of X'E0'.

   Steps 2 and 3 are repeated for each BRB posted to the Active BRB queue by the Line-PCI routine. The buffers are obtained from the Available Buffer queue.

4. Return to DOS Supervisor: Servicing of the PCI is now complete; therefore, control is returned to the DOS Supervisor. Before returning to the Supervisor, however, the message control program is removed from the wait state by turning off the wait flag in the PIB for the foreground-1 partition. This procedure is performed because the buffer(s) posted to the LPS queue satisfies the QWAIT SVC issued by the LPS Control routine in step 5 of Receiving Initiation.

5. DOS Supervisor: Because the message control program is the highest-priority program ready to proceed, that program is selected for activation by the DOS Supervisor. Control is returned to the LPS Control routine at the instruction following the QWAIT.

   Note: The procedures described in steps 4 and 5 apply in most cases where a PCI or I/O interrupt is processed by QTAM routines. For the sake of brevity, any discussion of returns to the DOS Supervisor and the action taken by the Supervisor is omitted from the remainder of this description except when a deviation from these procedures occurs.

RETURN TO MESSAGE CONTROL PROGRAM

6. LPS Control Routine: The empty buffer returned by the QWAIT is passed to the Activate routine.

7. Activate Routine: A CCW (with the PCI flag set) is prepared for the entire buffer in the second (next) BRB in the ring. The BRB is then added to the channel program already in process for the line. This is accomplished by making the TIC address valid in the CCW at the end of the channel program previously built (see Step 3 of Receiving Initiation). Exit is then made to the LPS Control routine. In this example, the second BRB now contains a TIC to the third BRB in the ring, but the TIC address is invalid because a buffer has not yet been assigned to the third BRB.

8. LPS Control Routine: QWAIT for the next item on LPS queue. Steps 6 and 7 are repeated until an empty buffer has been assigned to each BRB in the ring. The message control program then enters the wait state pending the arrival of another item on the LPS queue.

PCI--FULL BUFFER (RECEIVING)

During line receiving operations, a PCI occurs at the start of each buffer. The

action required for the PCI at the start of the first buffer has already been described. The PCI for each buffer subsequent to the first indicates that the previous buffer has been filled with message data and can be processed by the user's LPS section. The handling of each such PCI is described here.

1. Line-PCI Routine: In this example, the PCI at the start of the second buffer indicates that the first buffer has been filled with incoming data. The first BRB in the ring (the BRB to which the first buffer was assigned) is posted to the Active-BRB queue with a low priority (X'E4'), and the message-filled buffer is posted to the Interim LPS queue. More generally, for all PCIs subsequent to the first:

   a. the preceding BRB is posted to the Active-BRB queue, and

   b. the message filled buffer is posted to the Interim LPS queue.

   The former allows the same BRB ring to be reused for the case where the incoming message requires more buffers than specified in the BUFNO operand. The latter causes each message-filled buffer to eventually be routed to the user's LPS section.

ENTER ACTIVE BRB SUBTASK

2. Active BRB Routine: The BRB posted by the Line-PCI routine in the previous step is chained into the element chain of the Active BRB queue.

ENTER INTERIM LPS SUBTASK

3. Interim LPS Routine: The message-filled buffer posted to the Interim LPS queue in step 1 is now posted to the LPS queue with low priority (X'E0').

   Note: The Interim LPS subtask provides the means of delaying the processing of all buffers until all BRBs are processed when a PCI is missed due to extended CPU disable time.

RETURN TO MESSAGE CONTROL PROGRAM

4. LPS Control Routine: The SVC 30 (QWAIT) issued by this routine in step 8 of First PCI (Receiving) is satisfied by the appearance of the message-filled buffer on the LPS queue. The registers are initialized, and the buffer containing the header segment of the incoming message is passed to the receive group of the user's LPS.

5. RCVSEG Subgroup of LPS: Executed for all segments.

6. RCVHDR Subgroup of LPS: This portion of the LPS is executed only for the first buffer of the message (after the second PCI). The Lookup Terminal Table Entry routine (linked to by either the Route Message routine or the DIRECT macro expansion) sets the address of the QCB for the destination in the LQDT field of the source line LCB.

7. ENDRCV Subgroup of LPS: The ENDRCV macro expansion tests the BSTA field of the buffer prefix for EOB, EOT or duplicate header. If any of these are indicated, the functional macros within the ENDRCV subgroup are executed; otherwise, exit is made to the Cleanup routine. In this example, it is assumed that no EOB or EOT appears in the first buffer. (The duplicate header condition applies when multiple routing of the message is being performed via the EOA macro and is not considered in this discussion.)

8. Cleanup Routine: An SVC 31 (QPOST) is issued to post the buffer containing the header to the DASD-destination queue with a priority of X'E4'. (The address of the QCB for the destination was saved in the LCB in step 6.) For the first buffer, this causes the Send Scheduler subtask to be activated. For subsequent buffers containing segments of this message, it causes the DASD-Destination subtask to be activated.

ENTER SEND SCHEDULER SUBTASK

9. Send Scheduler Routine: This routine detects a message-filled buffer to be written on the disk and exits to the DASD-Destination routine.

10. DASD-Destination Routine: A disk location (relative record number) is assigned for this buffer. Disk locations are reserved and recorded for the next segment of this message (if any) and the header segment of the next message for this destination queue. The buffer is then posted to

the Disk I/O queue; meanwhile, the Send Scheduler waits for the destination line to become free.

ENTER DISK I/O SUBTASK

11. Disk I/O Routine: The relative record number assigned to the buffer in the previous step is converted to an actual DASD address. A channel program is set up for writing the buffer (less the first eight bytes) on the disk, and a special control block is posted to the LPS queue to request that the I/O be started.

RETURN TO MESSAGE CONTROL PROGRAM

12. Cleanup Routine: The QPOST (SVC 31) issued by this routine in step 8 is completed. This routine then exits to the LPS Control routine.

13. LPS Control Routine: QWAITs on the LPS queue. The request to start a Disk I/O operation posted to the LPS queue in step 11 is returned. This routine issues an EXCP (SVC 0) to start the I/O and QWAITs for the next item on the LPS queue. At this point, no item is on that queue so the message control program enters a wait state.

Summary of Operations to this Point

Operations completed:

- Filling of first buffer from line.
- Processing of first buffer by receive group of user's LPS.

Operations in process:

- Filling of second buffer from line
- Writing of first buffer on the disk

Possible Interruptions

At this point, any one of four possible interruptions may occur due to the operations in process.

- A PCI indicates that another buffer has been filled from the line. Steps 1 through 13 of PCI--Full Buffer (Receiving) are repeated with the following differences:

  1. The buffer is not processed by the RCVHDR subgroup of the LPS.
  2. The DASD-Destination routine initializes the buffer for a text segment and reserves only one disk location (for the next text segment).

- Channel-End (CE), Device End (DE) from the disk indicates that writing of the first buffer on the disk is complete.
- CE, DE from the line indicates that an EOB was received from the terminal.
- CE, DE, and unit exception (UE) from the line indicates that an EOT was received from the terminal.

This discussion assumes that the disk interruption occurs first followed by a line interruption for an EOB.

DISK INTERRUPTION (RECEIVING)

CE, DE from the disk indicates that the disk write operation has been completed. The DOS Supervisor passes control to the Disk-End Appendage.

1. Disk-End Appendage: The message segment contained in the buffer has been recorded in the appropriate DASD-Destination (or process) queue; therefore, the buffer is now released. This is accomplished by posting the buffer to the Available Buffer queue.

ENTER AVAILABLE BUFFER SUBTASK

2. Available Buffer Routine: The BRB placed on the element chain of the Active BRB queue by the Line-PCI routine (step 1 of PCI--Full Buffer (Receiving)) is found, removed, and passed to the Buffer-BRB routine.

3. Buffer-BRB Routine: An empty buffer is removed from the Available Buffer queue and assigned for receiving on the line. The buffer is posted to the LPS queue with a priority of X'E0'.

RETURN TO MESSAGE CONTROL PROGRAM

4. LPS Control Routine: The appearance of an empty buffer on the LPS queue causes the QWAIT earlier issued by this routine to be satisfied. The buffer is passed to the Activate routine.

5. Activate Routine: A CCW is prepared in the BRB for the entire buffer and is added to the channel program already in process.

6. LPS Control Routine: QWAIT on the LPS queue. The message control program enters the wait state.

LINE END INTERRUPTION (RECEIVE EOB)

CE, DE on a line I/O interruption indicates that an EOB was received from the terminal. Control passes to the Line-End routine in the Line Appendage.

1. Line-End Routine: A check is made for errors in the transmission (data check, intervention required, or time-out). If any is detected, it is recorded in the error halfword of the LCB. The offset to the next available location (following the EOB) in the buffer is recorded in the BSSZ field of the buffer prefix, and the buffer is passed to the Interim LPS routine.

2. Interim LPS Routine: The offset in the terminal table of the source terminal is obtained from the LCB and placed in the buffer prefix (BSTO field). The buffer containing the EOB is posted to the LPS queue with X'E0' priority.

RETURN TO MESSAGE CONTROL PROGRAM

3. LPS Control Routine: The buffer posted to the LPS queue in the preceding step is obtained and passed to the user's LPS section.

4. RCVSEG Subgroup of LPS: Executed for all segments.

5. RCVHDR Subgroup of LPS: This subgroup is bypassed because the current buffer is not the first buffer.

6. ENDRCV Subgroup of LPS: Because the current buffer contains an EOB, this subgroup is executed up to and including the EOB (or EOBLC) macro expansion.

7. EOB (or EOBLC) Routine: The Read Continue indicator is set in the LCB, and the buffer is passed to the Activate routine.

8. Activate Routine: A CCW is prepared for the entire buffer in the BRB associated with the buffer.

9. Channel Program Generator Routine: A Read Continue channel program is generated to respond to the EOB, and to read the next block of data into the portion of the buffer following the EOB. An EXCP is issued to start the I/O operation, and exit is made to the LPS Control routine.

10. LPS Control Routine: QWAIT for the next item on the LPS queue. No item is on the LPS queue, so the message control program enters the wait state.

LINE END INTERRUPTION (RECEIVE EOT)

CE, DE and UE from the line indicate that an EOT was received from the terminal. Control passes to the Line-End routine in the Line Appendage.

1. Line-End Routine: Any errors detected are recorded in the error halfword of the LCB. The buffer is passed to the Interim LPS routine.

2. Interim LPS Routine: The buffer containing the EOT (and the previous EOB) is posted to the LPS queue with a priority of X'E0'.

RETURN TO MESSAGE CONTROL PROGRAM

3. LPS Control Routine: The buffer posted to the LPS queue in the preceding step is obtained and passed to the user's LPS section.

4. RCVSEG Subgroup of LPS: Executed for that portion of the segment that follows a previous EOB.

5. RCVHDR Subgroup of LPS: This subgroup is bypassed because this buffer is not the first buffer. (A previous EOB also causes this subgroup to be bypassed.)

6. ENDRCV Subgroup of LPS: This entire subgroup is executed because an EOT has been received.

7. POSTRCV Macro Instruction: The expansion of this macro is a branch to the Cleanup routine.

8. Cleanup Routine: An SVC 31 (QPOST) is issued to post the buffer to the DASD-Destination queue with a priority of X'E4'.

ENTER DASD-DESTINATION SUBTASK

9. DASD-Destination Routine: The disk location previously reserved for the next segment in this message is assigned to this buffer. Because this buffer contains an EOT and is not the first segment of the message, disk locations are not assigned for the next segment or the header segment of the next message. The buffer is then posted to the Disk I/O queue.

ENTER DISK I/O SUBTASK

10. Disk I/O Routine: The relative record number assigned to the buffer is converted to an actual DASD address. A special control block requesting that the disk I/O operation be started is posted to the LPS queue. (As previously described, this control block is passed to the LPS Control routine which issues the EXCP to start the I/O.)

RETURN TO MESSAGE CONTROL PROGRAM

11. Cleanup Routine (continued): The SVC 31 issued in step 8 has been completed. Any buffers assigned for this operation but not used are posted to the Available Buffer queue with X'E4' priority. The following step is repeated for each such buffer posted.

ENTER AVAILABLE BUFFER SUBTASK

12. Available Buffer Routine: The buffer being released is linked into the element chain of the Available Buffer queue.

RETURN TO MESSAGE CONTROL PROGRAM

13. Cleanup Routine (continued): At this point, the Distribution List, EOA, and Conversational Mode routines would be executed if any of the functions performed by these routines are specified for this message (this example assumes none were specified). Exit is made to the Free BRB routine.

14. Free BRB Routine: Each BRB in the ring constructed for the receiving operation just completed is posted to the Inactive BRB queue.
Exception: If a BRB is in the Active BRB queue, it is not posted. A flag is set in this BRB so that when a buffer is available, it will not be assigned to the BRB and the BRB is then posted to the Inactive BRB queue.

The LCB for the line is then posted (SVC 31) to itself to free the line for its next use.

SUMMARY OF RECEIVING

Operations Completed

- Received complete message including EOT.
- Message written on DASD-destination queue.
- Indicated to the destination line that a message is queued for transmission (this was accomplished by chaining the STCB for the Send Scheduler into the LCB for the destination line).
- Released buffers and BRBs associated with source line.
- Freed the source line to poll or send.

Next Use of Source Line

The next use of the nonswitched line over which the message was received depends on the relative priority of receiving versus sending as specified in the CPRI operand for the line group.

- Receiving has priority over sending (CPRI=R). Messages are sent on the line only while a polling interval is being observed. In this example, if CPRI=R were specified, the Receive Scheduler would now be dispatched to initialize for repolling the same terminal.
- Receiving and sending have equal priority (CPRI=E). Messages are sent on the line at the end of a polling pass of the terminals on the line, regardless of any specified polling interval. In this example, polling would continue until the end of the polling list is detected.

- Sending has priority over receiving (CPRI=S). The Send Scheduler gains controls for sending on the line each time an EOT is received, the end of the polling list is detected, or a negative response to polling is received.

Because this discussion assumes a one-line system, the message just received is to be sent to a terminal on the same line. Also, for the purpose of this example, it is assumed that sending has priority over receiving. For either receiving or equal priority, the procedures already described would be repeated at this time. If the message just received was destined for a terminal on another line, the time at which it would be sent would depend on the status of the system and the priority assigned to that line. But at some point in time, the destination line would become free, and the Send Scheduler would gain control of the line.

SENDING INITIATION

Continuing with the example, because sending priority is assumed for the line, the line's Send Scheduler is activated for sending the message just received.

ENTER SEND SCHEDULER SUBTASK

1. Send Scheduler Routine: This routine exits to the BRB Ring routine when the line is free for sending and a complete message is on the DASD-destination queue for the line.

2. BRB Ring Routine: A ring of BRBs to be used for dynamic buffer allocation is built. (BRBs are obtained from the Inactive BRB queue.) The address of the first BRB in the ring is stored in the LCB for later access by the Activate routine. Because this is a sending operation, the disk relative record number of the first segment in the message is placed in the first BRB, and the BRB is assigned (RSTA=9) for a disk read. The first BRB is then posted to the Disk I/O queue with a priority of X'E0'.

RETURN TO MESSAGE CONTROL PROGRAM

At this point, the QPOST of the LCB issued by the Free BRB routine (step 14 of Line End Interruption (Receive EOT)) is completed and control returned to that routine which exits immediately to the LPS Control routine.

3. LPS Control Routine: QWAIT for item on the LPS queue.

ENTER DISK I/O SUBTASK

4. Disk I/O Routine: A buffer from the Available Buffer queue is assigned for the disk read operation. (If no buffer is available, the BRB is posted to the Active BRB queue to wait for an available buffer.) The relative record number of the first segment is converted to an actual disk address, and a request to start the disk read operation is posted to the LPS queue.

RETURN TO MESSAGE CONTROL PROGRAM

5. LPS Control Routine: Issues an EXCP to start the disk read operation, and then QWAITs for an item on the LPS queue. The message control program enters a wait state pending completion of the disk read.

DISK INTERRUPTION--SENDING, FIRST BUFFER

CE, DE from the disk indicates that reading of the first buffer is complete. The DOS Interrupt Handler passes control to the Disk End Appendage.

1. Disk End Appendage: A sequence-out number is assigned to the message and recorded in the header prefix. The message-sent (or serviced) flag is set in the prefix (bit 3 of BSTA) and a new start I/O is requested to rewrite the same buffer on the disk. (Rewriting the same buffer on the disk with the message-sent flag turned on indicates to QTAM that the message has been handled even though it has not actually been transmitted yet.) Control returns to the DOS Supervisor which starts the I/O to rewrite the buffer and then returns to the interrupted program.

DISK INTERRUPTION--SENDING ALL BUFFERS, SECOND TIME FOR FIRST BUFFER

1. Disk End Appendage: The buffer filled by the disk read operation is posted

to the LPS queue. If this is the second interrupt for the first buffer, the message-sent flag is turned off in the buffer prefix prior to posting the buffer to the LPS queue. The disk relative record number of the next segment is placed into the next BRB in the ring, and the BRB is posted to the Disk I/O queue.

ENTER DISK I/O SUBTASK

2. Disk I/O Routine: A buffer from the Available Buffer queue is assigned for reading the next segment of the message from the disk. The relative record number of the next segment is converted to an actual DASD address. A request to start the disk-read operation is posted to the LPS queue with a high priority, so it will be processed before the message-filled buffer placed on the LPS queue in step 1.

RETURN TO MESSAGE CONTROL PROGRAM

3. LPS Control Routine: This routine issues an EXCP to begin reading the next segment from the disk queue. A QWAIT is then issued to obtain the message-filled buffer from the LPS queue. This buffer is passed to the user's LPS section.

4. SENDHDR Subgroup of LPS: Executed for the first segment only.

5. SENDSEG Subgroup of LPS: Executed for all segments.

6. ENDSEND Macro Instruction: The expansion of this macro generates a branch to the Activate routine and an entry point to the End Send subgroup. At this point, the branch to the Activate routine is taken.

7. Activate Routine: The message-sent flag is set in the buffer prefix of all buffers. For the first buffer, the address of the first BRB in the ring is obtained from the LCB for the destination line. A CCW is prepared in the BRB for writing the entire buffer. The BRB also contains an invalid (two low order bits are nonzero) TIC address to the next BRB. The BRB is passed to the Channel Program Generator routine with an indication that a Write Initial channel program is to be generated.

   For all buffers except the first buffer, a CCW is prepared in the BRB for the entire buffer. This BRB is added to the channel program already in process by making the TIC address in the previous BRB valid (two low-order bits are cleared to zero). The PCI flag is set in the CCW, and exit is made to the LPS Control routine. Step 8 is executed only for the first buffer of the message.

8. Channel Program Generator Routine: A write-initial channel program is generated in the channel program area for the destination line. It consists of the CCWs for terminal selection (addressing) and writing the first segment. In preparing the channel program, the CCWs for writing the segment and the TIC to the next BRB are transferred from the passed BRB to the end of the channel program area. An EXCP is issued to start the I/O operation, and exit is made to the LPS Control routine.

9. LPS Control Routine: A QWAIT is issued for the next item on the LPS queue. The message control program enters the wait state.

   To avoid repetition in this discussion, the following operations are assumed to be completed or in progress at this point:

Operations completed:

1. Writing of first segment of message on the line.

2. Reading of second segment from the disk.

3. Processing of the second segment by the Send group of the LPS.

Operation in progress:
Writing of second segment on the line.

PCI--SENDING

A PCI occurs during sending operations at the start of every buffer except the first. The PCI flag was set in the CCW prepared in step 7 of the preceding section.

1. Line-PCI Routine: The preceding BRB is posted to the Active BRB queue (priority X'E4') to request a buffer for the next disk read. The preceding buffer is posted to the Available Buffer queue because the data in the buffer has been sent over the line. This procedure allows the BRB ring to

be reused as many times as necessary to send the entire message.

ENTER ACTIVE BRB SUBTASK

2. Active BRB Routine: The BRB posted to the Active BRB queue in the preceding step is linked into the element chain of that queue.

ENTER AVAILABLE BUFFER SUBTASK

3. Available Buffer Routine: Posting of a buffer to the Available Buffer queue in step 1 causes this routine to be entered. The BRB chained into the Active BRB queue in step 2 is removed from the element chain and passed to the Buffer-BRB routine.

4. Buffer-BRB Routine: Because the BRB is a buffer request for a disk read, a buffer (from the Available Buffer queue) is reserved for the disk read, and the BRB is posted to the Disk I/O queue.

ENTER DISK I/O SUBTASK

5. Disk I/O Routine: The relative record number of the next segment to be read from the disk is converted to an actual DASD address. A request to start the disk read operation is posted to the LPS queue.

RETURN TO MESSAGE CONTROL PROGRAM

6. LPS Control Routine: An EXCP is issued to start the disk read operation. (The interruption that occurs when this disk operation is completed was described previously.) The routine then QWAITs for the next item on the LPS queue.

LINE END INTERRUPTION (SEND EOB)

This example assumes that the second segment of the message contains an EOB. When this EOB is sent over the line, an interruption occurs, and control is passed to the Line Appendage.

1. Line-End Routine: This routine sets up a restart address to read the response to the EOB. An exit is then made to the DOS Supervisor which issues SIO to read the response.

LINE END INTERRUPTION (RESPONSE TO EOB)

1. Line-End Routine: Error checking is performed for the block just sent. Any error detected is recorded in the error halfword for the line. The buffer containing the EOB is posted to the LPS queue with X'E0' priority. This causes the buffer to be routed through the Send LPS again for further error checking of the block that was sent over the line.

RETURN TO MESSAGE CONTROL PROGRAM

2. LPS Control Routine: The buffer posted to the LPS queue in the preceding step is routed to the user's LPS section.

   Note: Just before sending the segment over the line, the message-sent flag was turned on in the buffer prefix (bit 3 of BSTA). When the segment is routed to the Send LPS for the second time, the functions in the ENDSEND subgroup are performed. The SENDHDR and SENDSEG subgroups are bypassed because the segment was processed by these subgroups before it was sent over the line.

3. ENDSEND Subgroup of LPS: For each EOB detected in an outgoing message, this subgroup is executed up to and including the EOB (or EOBLC) routine.

4. EOB (or EOBLC) Routine: The write-continue code is set in the LCB, and the buffer containing the EOB is passed to the Activate routine.

5. Activate Routine: A CCW is prepared for the entire buffer in the BRB associated with the buffer.

6. Channel Program Generator Routine: The CCW prepared in the preceding step is modified to write that portion of the buffer that follows the EOB character. An EXCP is issued to start the I/O, and exit is made to the LPS Control routine.

7. LPS Control Routine: QWAIT for the next item on the LPS queue.

LINE END INTERRUPTION (SEND EOT)

1. Line End Routine: The buffer containing the EOT is posted to the LPS queue with X'EO' priority.

RETURN TO MESSAGE CONTROL PROGRAM

2. LPS Control Routine: The buffer posted to the LPS queue in the preceding step is routed to the Send group of the user's LPS. (Because the message-sent flag is on, the buffer is routed directly to the ENDSEND subgroup.)

3. ENDSEND Subgroup of LPS: The entire subgroup is executed because a complete message has been sent.

4. POSTSEND Macro Instruction: The expansion of this macro generates a branch to the Cleanup routine.

5. Cleanup Routine: QPOST is issued to release the buffer to the Available Buffer queue.

6. Free BRB Routine: Each BRB in the ring formed for sending the message is posted to the Inactive BRB queue.

   Exception: If a BRB is in the Active BRB queue, it is not immediately released. A flag is set in the BRB so that when a buffer is available, it will not be assigned to the BRB. The BRB is then posted to the Inactive BRB queue.

   The LCB for the line is posted to itself to free the line for its next use.

   Note: Once sending commences on a line, the line's Send Scheduler continually gains control until all messages on the destination queue for the line have been sent. The procedures beginning at Sending Initiation are repeated for each complete message on the destination queue.

ENTER SEND SCHEDULER SUBTASK

7. Send Scheduler Routine: A test is made to determine if another complete message is on the DASD destination queue. In this example, no other message is on the queue at this point. Therefore, the Send Scheduler removes itself from the line's STCB chain and waits for other messages to arrive on the destination queue.

ENTER RECEIVE SCHEDULER SUBTASK

The cycle of receiving a message and then sending it over the same line is now complete. Because this discussion assumes queueing by line, the Receive Scheduler subtask gains control to resume polling the terminals on the line. If queuing were by terminal, another Send Scheduler subtask would be contending with the Receive Scheduler for control of the line.

MESSAGE CONTROL PROGRAM FOR AN AUDIO APPLICATION

The message control program is loaded and entered as described in the section Message Control Program (for a Nonaudio Application).

ENTER MESSAGE CONTROL PROGRAM

INITIALIZATION

1. QTAM Open 7772 DCV Vocabulary File Routine: The address of the QTAM Vector Table is placed, if necessary, in the DOS Communication Region so that QTAM routines requiring information from the vector table can have access to it. The initialization of the 7772 DCV buffers (BABs) is completed and all the user-specified DCV words are loaded into main storage. The extent data for this QTAM file is read, and the File Protect Subroutine is called into the transient area. A parameter is passed directing it to call into the transient area the open module that must be executed next.

2. QTAM Open Audio Line Group/Output Queue File Routine: The initialization of all ALCBs is completed and enabling is initiated on each line according to initial conditions.

END-OF-RECEIVING INITIATION

1. ENDREADY Macro Instruction: The user's registers are saved in the user save area. A branch is made to the LPS Control routine

2. LPS Control Routine: A QWAIT (SVC 30) is issued to wait for the next ALCB on the LPS queue. The LPS Control routine finds no ALCB on the LPS queue, and the message control program enters a wait state. At this point, control is given to a message processing program in a lower priority partition through the DOS task selection mechanism. Subsequent I/O interruptions cause ALCBs to be posted to the LPS queue. This allows the message control program to proceed.

   Possible interruptions: At this point, any of the following line interruptions may occur as a result of audio input operations:

   - Channel-end (CE) and Device-end (DE) on an enable operation indicate that a 7772 invitational message may be sent.
   - Channel-end and Device-end on a read operation indicate that an input message has been received in an ALCB. However, if the input length is null or if only one EOT character has been received, the line is disabled and reinitiated according to initial conditions.
   - Channel-end, Device-end, and Unit Exception (UE) indicate that the caller hung up during a read operation. In this case, the line is disabled and reinitiated according to initial conditions.

AUDIO LINE END INTERRUPTION (RECEIVING)

Channel-end and Device-end on an audio line I/O interruption indicate that an input message was received from a remote terminal. Control passes to the Audio Line-End routine in the Audio Line Appendage.

1. Audio Line-End Routine: A check is made for errors in the transmission (data check, or input message overlength). If any is detected, it is recorded in the error byte of the ALCB. The length of the input message is computed and saved in the ALCB. But if this length is null or if the received message consists of one EOT character only, the line is disabled and reinitiated according to initial conditions. At this point, the ALCB is posted to the LPS queue.

RETURN TO MESSAGE CONTROL PROGRAM

2. LPS Control Routine: The ALCB posted to the LPS queue in the preceding step is obtained and passed to the user's ARU/LPS section.

3. User's ARU/LPS: The user can make a check for input message overlength, invalid repetition request as first message in a transaction, or data check during a read operation. As soon as an error is detected, the ALCB is passed to the ARU Internal routine. If no error is detected, the ALCB is passed to the ARU Receive routine (step 4 is bypassed).

4. ARU Internal Routine: The user's error message, in the form of an address chain, is placed in the address-chain buffer located in the ALCB. The length of the address chain is stored in the ALCB. The ALCB is posted to the ARU Send queue, and exit is made to the LPS Control routine.

5. ARU Receive Routine: The passed ALCB is given a priority code of X'FC'. To post this ALCB to the message processing program in which it must be processed, the corresponding MS-process queue is searched. If found, the ALCB is posted to it, and control is given to the LPS Control routine. If not found (the message processing program has not been opened), the ALCB is queued in a waiting chain located in the corresponding PROCESS macro expansion, and control is given to the LPS Control routine. This waiting chain is directly transferred into the MS-process queue as soon as the message processing program is opened.

6. LPS Control Routine: Waits for the next item on the LPS queue. If none, the message control program enters the wait state.

SUMMARY OF RECEIVING

An input message has been received in the ALCB that must be processed by the message processing program, and the caller is waiting for an audio answer on the line.

SENDING INITIATION

Assume the message processing program provided a 7772 output message expressed in the form of an address chain. The audio PUT routine places this address chain in the address chain buffer of the ALCB, gives the ALCB a priority code of X'FD', and posts the ALCB to the ARU Send queue to activate the ARU Send scheduler.

ENTER ARU SEND SUBTASK

1. ARU Send Routine: Checks the type of unit related to the passed ALCB. Assuming it is a 7772 ALCB, the initial conditions for a 7772 write operation are set, and a test is made to determine whether or not a DCV buffer pool has been declared. If not, all DCV words to be used are already in main storage (word table), and control is directly passed to the 7772 Disk Read routine. If yes, an available DCV buffer is extracted from the element chain of the DCV buffer queue and attached to the requesting ALCB. Then, control is given to the 7772 Disk Read routine.

   Note: When the element chain of the DCV buffer queue contains no available DCV buffer, the ALCB is queued in a temporary waiting chain located in the DCV buffer queue. In this queue, the STCB field is updated with the address of the 7772 DCV buffer STCB, and control returns to the Qdispatch routine. At this point, a DCV buffer is expected to become free by another line. When this occurs, the available DCV buffer is posted to the DCV buffer queue, and the DCV buffer subtask is activated.

ENTER 7772 DCV BUFFER SUBTASK

2. 7772 DCV Buffer Routine: Dequeues the ALCB from the waiting chain and associates it with the passed DCV buffer. When the last ALCB in the waiting chain has been thus dequeued, the address of the STCB field of the DCV buffer queue is updated with the address of the Queue Insert STCB. No update is performed if one or more ALCBs remain in the waiting chain. In either case, control is given to the 7772 Disk Read routine.

3. 7772 Disk Read Routine: Analyzes the next address in the address chain buffer of the ALCB.

   - If this address is that of a DCV word in main storage, it is computed and stored into the ALCB. The 7772 Disk Read routine then makes a test for a user's pause following a DCV word address. If this pause is present, it is stored into the ALCB. In either case, control passes to the 7772 Line Write routine.

   - If this address is that of a DCV word in the DCV vocabulary disk file, a part of the DCV buffer allocated to the ALCB is assigned to the DCV word to be read. The disk address is computed, and the disk channel program is updated in the DCV buffer header. The 7772 Disk Read routine then makes a test for a user's pause following a DCV word address. If this pause is present, it is stored into the ALCB. In either case, the DCV buffer is posted to the LPS queue.

4. LPS Control Routine: On recognition of a DCV buffer, this routine issues an EXCP to start a disk read operation, and then waits for an item on the LPS queue. The message control program enters the wait state until the next interruption.

IBM 7772 DISK INTERRUPTION

Channel-end and Device-end from the vocabulary disk indicate that reading of the DCV word is completed. The DOS Interrupt Handler passes control to the 7772 Disk End Appendage.

1. Disk End Appendage: The address and length of the DCV word being read are stored into the ALCB associated with the DCV buffer, and control passes to the 7772 Line Write Routine.

IBM 7772 AUDIO SENDING

1. 7772 Line Write Routine: Updates the channel program in the ALCB for a write operation. The PCI flag is set in the Write CCW used for the current DCV word, except when this DCV word corresponds to the last address in the address chain. If a user's pause has been stored into the ALCB, a write

CCWfor pause is updated in the channel program. In any case, the channel program keeps looping on a Write CCW for elementary pause until the next DCV word is ready to be sent from the DCV buffer. Then, the following operations are executed, depending on whether the DCV word to be sent is the first word, an intermediate word, or the last word.

- For the first DCV word, the ALCB is posted to the LPS queue.
- For the intermediate DCV words, the waiting loop created in the channel program for the preceding DCV word is broken, and the next DCV word to be sent (except the last) is chained to the Write CCW. At this point, the section PCI Sending is entered.
- For the last DCV word, there is no PCI flag for the corresponding Write CCW nor waiting loop in the channel program. But, the last Write CCW is chained to a Read or Disable CCW, depending on the mode of transaction. At this point, the section Audio Line End Interruption is entered.

RETURN TO MESSAGE CONTROL PROGRAM

2. LPS Control Routine: Issues an EXCP to start a write line operation, and waits for the next element on the LPS queue.

PCI SENDING

A PCI occurs during the sending operation of each DCV word except the last. The DOS Interrupt Handler passes control to the Audio Line Appendage.

1. Audio Line Appendage: On recognition of a PCI on a 7772 write operation, the ALCB is passed to the Disk Read routine, and the previous steps (from step 3 in section Sending Initiation) are repeated until complete emission of the last DCV word.

AUDIO LINE END INTERRUPTION

Assuming a 7772 audio line is working in inquiry mode, the Write CCW (in the channel program) for the last DCV word has been chained to the Disable CCW. An interruption occurs, and control passes to the Audio Line Appendage.

1. Audio Line Appendage: On recognition of a Channel-end or Device-end on a 7772 Disable operation, the channel program in the ALCB is updated on the Enable CCW according to initial conditions. The exit from the Exit interface routine is changed to a return to the Audio Line Appendage after the DCV buffer has become free. The DCV buffer is posted to the DCV buffer queue, and control is given to the Qdispatch routine. If no ALCB is waiting for a DCV buffer, the Queue Insert subtask is activated.

ENTER QUEUE INSERT SUBTASK

2. Queue Insert Routine: The DCV buffer is placed last-in-first-out in the element chain of the DCV buffer queue, the normal exit from the Exit Interface routine is bypassed, and the special return to the Audio Line Appendage is taken.
3. Audio Line Appendage: The normal exit from the Exit Interface routine is reestablished, and the ALCB is posted to the LPS queue.

RETURN TO MESSAGE CONTROL PROGRAM

4. LPS Control Routine: Issues an EXCP to start an enable operation on line, and waits for the next element on the LPS queue.

At this point, the 7772 cycle of receiving a message and sending an audio answer on the same line is complete.

Note: The sending part of a corresponding 7770 cycle is very simple and short, as shown below:

- The ARU Send routine updates the channel program, and posts the ALCB to the LPS Control routine.
- The LPS Control routine issues an EXCP to start the complete write operation.
- Channel-end and Device-end passed to the Audio Line Appendage indicate the completion of the transaction.
- The channel program is updated on the Enable CCW according to initial conditions.

- The ALCB is posted to the LPS queue which issues an EXCP to start the enable operation.

At this point, too, the 7770 cycle is complete.

MESSAGE PROCESSING PROGRAM FOR A NONAUDIO APPLICATION

After the message processing program phase has been loaded from the core image library into either the foreground-2 or background partition, it is entered for execution by the DOS Supervisor at some point when the message control program (in foreground one) enters the wait state. This discussion assumes that the message processing program is executing in foreground-2.

ENTER MESSAGE PROCESSING PROGRAM

INITIALIZATION

1. QTAM Open Routine, Main Storage (MS) Process and Destination Queues: When an MS-process queue is being opened, the address of the QCB for the corresponding DASD-process queue (generated by the expansion of a PROCESS macro in the message control program) is placed into the DTF table. The address of the LCB for the MS-process queue is placed into the QSTL field of the QCB for the DASD-process queue. This action opens the "door" between the message control and message processing programs and allows messages to be transferred to a processing program in response to a GET.

   To open a MS-destination queue, the open bit is turned on in the DTF table.

FIRST GET

The transfer of message-filled buffers to the MS-process queue commences when the first GET macro instruction is issued in the message processing program. Prior to the first GET, any incoming messages having this processing program as a destination are accumulated on the corresponding DASD-process queue.

Note: The procedures for handling an incoming message destined for a processing program prior to the first GET are basically the same as those described in the section Message Control Program, except that the message is written on a DASD-process queue rather than on a DASD-destination queue.

This discussion assumes that the work unit requested by a GET is a complete message (WU=M) terminated by an EOT character, and that the user-defined work area is large enough to contain an entire message. It is also assumed that the Expedite option was not specified in the PROCESS macro.

1. Get Message Routine: This routine normally releases the previous buffer to the Return Buffer queue through a QPOST (SVC 31). The previous buffer is that buffer in the MS-process queue from which data was transferred to the user work area on the previous GET. Because this is the first GET, there is no previous buffer; therefore, a dummy buffer (contained in the DTF table for the MS-process queue) is posted to the Return Buffer queue.

ENTER RETURN BUFFER SUBTASK

2. Return Buffer Routine: The QCB for the MS-process queue is contained in the DTF table for that queue and serves a dual purpose because it is also used as a BRB. The BRB is initialized for requesting a buffer into which a segment can be read from the DASD-process queue. Because the dummy buffer is being returned, exit is made to the Get Scheduler routine. Subsequent to the first GET, the previous buffer being returned from the MS-process queue is posted to the Available Buffer queue before exiting to the Get Scheduler.

3. Get Scheduler Routine: A test is made to determine if there is a message on the DASD-process queue (a message is assumed in this description). The disk relative record number of the header segment of the message is obtained from the QCB for the DASD-process queue and placed into the BRB. The BRB is then posted to the disk I/O queue.

ENTER DISK I/O SUBTASK

4. Disk I/O Routine: A buffer from the Available Buffer queue is assigned

forthe disk-read operation. (If no buffer is available, the BRB is posted to the Active BRB queue to wait for a buffer to become available.) The relative record number of the segment is converted to an actual DASD address, and a request to start the disk operation is posted to the LPS queue.

RETURN TO MESSAGE CONTROL PROGRAM

5. LPS Control Routine: The message control program has been in the wait state because a message processing program gains control only when the message control program is waiting for an item to appear on the LPS queue. The start I/O request posted in the preceding step causes this wait to be satisfied, and the LPS Control routine issues an EXCP to start the disk read operation. It then QWAITs for the next item on the LPS queue, and the message control program enters the wait state again.

RETURN TO MESSAGE PROCESSING PROGRAM

6. Get Message Routine (continued): The QPOST issued by this routine in step 1 has been completed. A QWAIT (SVC 30) is issued to wait for a buffer in the MS-process queue. At this point, the message processing program also enters the wait state pending completion of reading the header segment from the disk. Because both the message control and message processing programs are in the wait state, the DOS Supervisor gives control to a non-QTAM program in the background partition.

Disk Interrupt, Header Segment

CE, DE from the disk indicates that reading of the header segment into the buffer is complete. Control passes to the Disk End Appendage via the DOS Supervisor.

7. Disk End Appendage: A sequence-out number is assigned to the message and recorded in the header prefix. The message-sent flag is set in the prefix to indicate to QTAM that the message has been serviced. Return is then made to the Supervisor with a request to rewrite the header segment on the disk. The Supervisor starts the I/O and returns to the interrupted program.

Disk Interrupt, Rewrite of Header Segment

The disk-end appendage is entered when a disk interruption indicates that rewriting of the header segment has been completed.

8. Disk End Appendage: The BRB and the buffer containing the header segment are removed from the Disk I/O queue. The message-sent flag is turned off in the header prefix, and the buffer containing the header segment is chained into the MS-process queue.

   A test is then made to determine if the MS-process is full. The number of buffers allocated to the MS-process queue is specified by the BUFNO operand of the DTFQT macro used to define the MS-process queue. BUFNO=4 is assumed in this discussion; therefore, the MS-process queue is not full at this point.

   The relative record number of the next segment in the message (last segment in this example) is placed into the same BRB, and the BRB is posted to the Disk I/O queue again.

   At this point, Steps 4 and 5 are repeated to start reading the next segment into a buffer.

RETURN TO MESSAGE PROCESSING PROGRAM

9. Get Message routine (continued): The message-filled buffer placed on the MS-process queue in step 8 causes the QWAIT issued by this routine in step 6 to be satisfied. Control returns to the instruction after the QWAIT with the address of the buffer containing the header segment in register 1.

   Note: Completion of the disk read of the second segment would cause the message processing task to be interrupted; however, discussion of this interrupt is deferred for the moment.

   The Get Message routine transfers the header segment from the buffer to the user's work area and moves the name of the source terminal into the TRMAD field. A test is then made for end-of-message. This discussion assumes that two buffers are required to contain the complete message; therefore, a QWAIT is issued to obtain another buffer from the MS-process queue.

Prior to the QWAIT, a QPOST is issued to return the previous buffer (buffer just processed) to the Return Buffer Queue. This QPOST causes steps 2 through 5 to be repeated to start reading the header segment of the next message from the disk when the current disk operation is completed.

Disk Interrupt, Text Segments

CE, DE from the disk indicates that reading of the text segment has been completed.

10. Disk End Appendage: The buffer containing the text segment is chained into the MS-process queue. This discussion assumes that this text segment ends the message. Because the MS-process queue is still not full, the BRB is initialized with the relative record number of the next segment and is posted to the disk I/O queue. This causes reading of the next segment from the disk queue to commence as previously described.

RETURN TO MESSAGE PROCESSING PROGRAM

11. Get Message Routine (continued): The appearance of the buffer containing the last segment of this message on the MS-process queue causes the QWAIT issued in step 9 to be satisfied. The data in the buffer (up to and including the EOT) is transferred to the work area starting at the position after the end of the header segment. Because a complete message has now been placed in the work area, the length of the message and the end-of-message code are set in the GET/PUT prefix. Control then returns to the instruction after the GET for processing of the message according to the application.

REPLENISHING THE MS-PROCESS QUEUE

After the first GET has been issued, QTAM automatically replenishes the MS-process queue with incoming messages having the MS-process queue as their destination. As long as messages are arriving faster than they are being processed, QTAM can constantly fill the MS-process queue in anticipation of subsequent GETs. This action reduces the time required to provide a message in response to a GET by ensuring that a message-filled buffer is in the MS-process when a QWAIT is issued by the GET routine.

QTAM uses two methods to constantly replenish the MS-process queue subsequent to the first GET. The first method is initiated by the GET routine and has already been described. In summary, when a GET is issued, the GET routine issues a QPOST to release the buffer used for the previous GET. This QPOST causes reading of the next segment from the DASD-process queue to be started.

When reading of the next segment is completed, the disk-end appendage chains the filled buffer into the MS-process queue. Thus, the buffer that was released from the MS-process queue is replaced by another buffer containing data to be provided in response to a subsequent GET. Additionally, the Disk-End appendage continues to initiate disk reads until the MS-process queue is full or until there are no more unserviced messages in the DASD-process queue.

The second method of replenishing the MS-process queue is performed by the message control program at the time the message segment is received over the line. The procedures for receiving a message segment destined for a processing program are identical to those for a segment destined for a terminal up to the point when the segment is posted to the DASD queue by the Cleanup routine. The buffer containing the segment for a processing program is posted to a DASD-process queue rather than a DASD-destination queue, and the following occurs.

ENTER GET SCHEDULER SUBTASK

1. DASD Destination Routine: A relative record number is assigned to the buffer. If the buffer contains a header segment (assumed for this discussion), disk locations are reserved for the next segment of this message (if any) and for the header segment of the next message. The buffer is then chained into the Disk I/O queue to be written on the DASD-process queue. Exit is made to the Get Scheduler routine.

2. Get Scheduler Routine: A test is made to determine if the MS-process queue is full. If the MS-process queue is full, any attempt to place the buffer in that queue is deferred until a previous buffer is returned by a GET. However, this discussion assumes that the MS-process queue is not full, and that a BRB requesting a read of the

current buffer is already in the Active BRB queue. This BRB is posted to the Disk I/O queue behind the buffer posted in step 1.

ENTER DISK I/O SUBTASK

3. Disk I/O Routine: At this point, the element chain of the disk I/O queue contains two elements. The first element is a buffer containing a header segment to be written on the DASD-process queue. The second element is a BRB requesting that the same header segment (first element) be read from the DASD-process queue. The relative record number assigned to the buffer is converted to an actual DASD address, and a request to start the disk-write operation is posted to the LPS queue. The LPS Control routine issues an EXCP to start the I/O operation, and QWAITs for the next item on the LPS queue. The message control program then enters the wait state.

Disk Interrupt (Write)

An interrupt from the disk indicates that the disk write has been completed.

4. Disk-End Appendage: The buffer and the BRB are removed from the disk I/O queue. Because the BRB is a request to read the buffer that was just written, the buffer is simply chained in the MS-process queue without actually doing the disk read. This technique has two advantages:

   a. the disk-read operation required for the first GET or when the MS-process queue is full, is eliminated, and

   b. the MS-process queue is replenished more rapidly in anticipation of subsequent GETs.

   The Disk-End appendage then turns on the message-sent bit in the header prefix and assigns a sequence out number to the message. Control returns to the DOS Supervisor with a request to rewrite the same buffer on the disk. Rewriting of the header segment with the message-sent bit on is a standard QTAM technique indicating that the message has been serviced.

TRANSFERRING RESPONSE MESSAGES

When a response message is to be sent to a terminal, a PUT macro is issued in the message processing program. This discussion assumes that a complete message (WU=M) is being PUT.

1. Put Message Routine: The message data in the work area must be transferred into a buffer. A buffer is requested for this purpose by posting (QPOST) a BRB to the Active BRB queue with a priority of X'EC'. (The BRB is contained in the first four words of the DTF table for the MS-destination queue.)

ENTER ACTIVE BRB SUBTASK

2. Active BRB Routine: The BRB with high priority is passed to the Buffer-BRB routine if a buffer is available. This discussion assumes that a buffer is available.

3. Buffer-BRB Routine: A buffer is removed from the Available Buffer queue, and exit is made to a special entry point in the PUT Message routine.

4. PUT Message Routine (Special Entry): This is a special section of the PUT Message routine that executes in supervisor mode to circumvent a violation of the storage protection feature. Upon entry, the address of the buffer requested is in register 1.

   The terminal table is searched to locate the entry for the destination terminal specified in the TRMAD field. The address of the QCB for the DASD-destination queue is obtained from the terminal table entry and placed in the buffer prefix and in the LCB. The message data is moved from the work area to the buffer using the length specified in the GET/PUT prefix. The end-of-message code is set in the BSTA field of the buffer prefix, and the buffer is posted to the DASD-destination queue.

   At this point, the procedures for writing the buffer on the disk are performed. The action caused by posting the buffer to the DASD-destination queue is identical to that described in the section, Message Control Program. When the destination line becomes free for sending, the message is sent to the terminal as described under the Message Control Program.

Note: If the entire message cannot be contained in a single buffer, another buffer is requested and steps 2 through 4 are repeated. This continues until the entire message has been transferred to buffers, and the buffers posted to the DASD-destination queue. The end-of-message code is set in the prefix of the last buffer required for the message.

RETURN TO MESSAGE PROCESSING PROGRAM

5. Put Message Routine (continued): The QPOST issued by this routine in step 1 has been completed. Control is returned to the instruction after the PUT so that the GET/process/PUT cycle may continue.

IBM 2260-2848 LOCAL OPERATIONS

The IBM 2260 Local differs in operation from other QTAM-supported devices. Because it is locally attached, the IBM 2260 Local is neither polled nor addressed. Instead, when the operator at the IBM 2260 desires to send a message to the CPU, he keys in the START symbol followed by the text, and depresses the ENTER key. Depressing the ENTER key results in an I/O interrupt with the attention bit set in the CSW. This I/O interrrupt is referred to as an Attention interrupt or read request.

When an Attention interrupt occurs at the CPU, a command control block (CCB) for the IBM 2260 initiating the read request must be in the DOS channel queue. If it is not, the interrupt is ignored by the DOS Interrupt Handler. The CCB is a part of the terminal table entry for the IBM 2260.

When an IBM 2260 Local line group is opened, the QTAM Open routine causes the CCB for each IBM 2260 from which messages can be received to be placed in the channel queue. A NOP channel program is initiated for the IBM 2260. When the CE, DE interrupt ending the NOP occurs, the DOS Interrupt Handler passes control to the IBM 2260 Local Appendage. This appendage sets the channel end (CE) flag in the CSW status (all other bits are cleared) and returns to the DOS Interrupt Handler. The appearance of the CE flag along with the device-end posting bit (assembled as a one in transmission byte zero of the CCB) causes the DOS Interrupt Handler to leave the CCB on the channel queue so that an Attention Interrupt from that IBM 2260 can be recognized and serviced. After the line group has been opened, read requests from the IBM 2260s are serviced on a 'first-come, first-served' basis.

When an Attention Interrupt occurs, the IBM 2260 Local Appendage again gains control. The CCB for the IBM 2260 is posted to an Attention queue maintained in the LCB for the line group. Channel end, device end is set in the CSW to cause the DOS Interrupt Handler to remove the CCB from the channel queue. If the line control block (LCB) is available, the procedures necessary to perform the read operation are initiated immediately. These procedures include forming a ring of buffer request blocks (BRBs) and assigning the buffers required for the read operation. Due to the high data rate of the IBM 2260 Local, all buffers required to contain an incoming message must be assigned before one read operation is started. This differs from the buffer management technique used for receiving from remote terminals in which only one buffer is assigned initially and additional buffers are obtained as needed through program-controlled interrupts.

When all buffers have been assigned, an EXCP is issued to read the message. A channel end, device end interrupt occurs when reading is completed. The IBM 2260 Local Appendage routes the message-filled buffers to the Receive Group of the user's LPS section. The CE flag is set in the CSW to cause the DOS Interrupt Handler to leave the CCB on the channel queue so that any subsequent read request from the 2260 can be serviced. Processing and further routing of the message are the same as described for a message received from a remote terminal under the Message Control Program.

The general techniques for sending a message to an IBM 2260 Local (or to the IBM 1053 printer attached to the IBM 2848) are the same as for sending a message to a remote terminal, except that addressing is not performed. Additionally, an SVC 25 is issued to initiate the write channel program. When the write operation completes, the IBM 2260 Local Appendage causes the CCB to remain on the channel queue as described previously.

Data transfer can occur between the CPU and only one IBM 2260 Local at a time. For this reason, only one LCB is generated for an IBM 2260-2848 Local line group. This LCB contains all control information required for I/O operations in the line group and for LPS processing of messages. If a read request (Attention Interrupt) from an IBM 2260 occurs while the LCB is

not available due to LPS processing of a previous message or because QTAM is preparing to send a message to a terminal in the line group, the CCB is removed from the channel queue and posted to the Attention queue. The read request is serviced when the LCB becomes available for receiving.

For further information, refer to the section IBM 2260 Local Appendage.

MESSAGE PROCESSING PROGRAM FOR AN AUDIO APPLICATION

After the message processing program has been loaded from the core image library into either the foreground 2 or background partition, it is entered for execution by the DOS Supervisor at some point when the message control program (always in foreground 1) enters the wait state. Assume that the message processing program is executed in foreground 2 partition and handles only audio messages.

ENTER MESSAGE PROCESSING PROGRAM

INITIALIZATION

1. QTAM Open Main Storage (MS) Process and Destination Queues File Routine: The opening procedure for the MS-process queue file is practically the same as that used in a nonaudio application, because an MS-process queue may consist of audio as well as nonaudio messages. But, at assembly time in the message control program, a PROCESS macro instruction does not indicate if it corresponds to a message processing program working on all types of message or on audio messages only. On the other hand, the PROCESS macro expansion always generates a DASD-process queue, and the audio programming uses only the QSTL field of the DASD-process QCB to obtain the address of the MS-Process QCB. However, the ALCBs waiting for opening of the message processing program and for processing are located in the waiting chain of the corresponding PROCESS macro expansion, and are transferred into the element chain of the MS-process queue. Then, the open bit is turned on in the MS-process queue DTF table.

2. QTAM Open Audio Line Group/Output Queue File Routine: The ARU-Send queue address is placed in the Audio Output queue DTF table and the open bit is turned on.

OBTAINING A MESSAGE

The ALCBs containing the input messages to be processed may have been transferred at open time of the MS-process queue file. Assume that the user-defined work area is large enough to contain any input message, and that the work unit requested by the GET macro instruction is an audio message (WU=A).

1. Get Audio Message Routine: A QWAIT (SVC 30) is issued to wait for an ALCB in the MS-Process Queue. When no ALCB is available, the message processing program enters the wait state. Then, because both the message control and message processing programs are in the wait state, the DOS Supervisor gives control to a non-QTAM program in the background partition. When an ALCB is waiting for processing, it is passed to the Get Audio Message routine. This routine transfers the input message from the input buffer (in the ALCB) into the user's work area, places the message length in the GET/PUT prefix, and moves the name of the source line into the LINAD field. At this point, a high-priority audio element is posted to a combined QCB/STCB (in the Get Audio Message module) to avoid a violation of the storage protection feature.

ENTER GET AUDIO SUBTASK

2. Get Audio Message Routine (in Supervisor Mode): The GET bit in the ALCB is turned on, and control is passed to the Qdispatch routine. This GET bit will be used to check the destination line, which must be the same as the source line (no switching is allowed) when the user's answer is sent via the Put Audio Message routine.

3. Get Audio Message Routine (continued): Control is given to the instruction following the GET macro instruction, to process the message according to the user's application.

TRANSFERRING RESPONSE MESSAGES

After the message has been processed, the user must answer to the calling terminal on line. Therefore, he provides an output message in the form of an address chain, and issues a PUT macro instruction.

1. Put Audio Message Routine: The ALCB is searched from the line name specified by the user in the LINAD field. A check is made for the presence of the GET bit in the ALCB. If this bit is found, this indicates that an audio answer is awaited on the line. At this point, a high-priority audio element is posted to a combined QCB/STCB (in the Put Audio Message module) to avoid a violation of the storage protection feature.

ENTER PUT AUDIO SUBTASK

2. Put Audio Message Routine (in Supervisor Mode): The message data is transferred from the work area into the address chain buffer of the ALCB. In the ALCB, the address chain length is updated, the priority code is set to X'FD', and the GET bit is turned off. Then, control is given to the Qdispatch routine.

3. Put Audio Message Routine (continued): The ALCB is posted to the ARU Send queue, and control is given to the instruction following the PUT macro instruction so that the GET/process/PUT cycle can continue.

   Note: The action caused by posting the ALCB to the ARU Send queue is identical to that described in the section Message Control Program for an Audio Application.

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 collectively form 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 phase 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 generalized routines; each of these is linkage-edited into the message control program phase because of a linkage generated in any of several other LPS routines.

Each of the following LPS routine descriptions provides the name of the routine, the name of the module that contains it, the function of the routine, entry point and linkage information, and names of external routines used.

BREAKOFF ROUTINE

Module Name: IJLQBO (Chart BO)

Entry Point: Expansion of the BREAKOFF macro instruction generates a BALR to the routine at IJLQBO, 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.

Function: Causes transmission of an incoming message to be terminated and an error bit set if the incoming message exceeds the maximum length specified, or if the characters in the buffer are identical (usually an indication of terminal or line malfunction). If the characters are not identical, the count of characters is added to the previous count in the LECT field of the LCB.

The specified maximum length of a message is passed to the routine via register 14. If the specified length is greater than zero, the accumulated length is compared with the length specified. If the accumulated length is greater than the maximum length, the receive bit in the LSTA field of the LCB is turned off. This stops further assignment of buffers for this operation and causes a program check interrupt to occur.

If the accumulated length is less than or equal to the maximum length, a test is made for end of message. If it is not the end of a message, return is made to the next LPS instruction; otherwise, a test for program check is made.

If no program check has occurred, return is made to the next LPS instruction. If a program check has occurred, as a result of no buffer assignment, the read operation code is cleared and the breakoff bit is set in the error-halfword (LEHW). The address of a CCW with the BREAK command code is moved into the CCB area of the LCB, and EXCP is issued to write the breakoff characters. The routine branches to the LPS control routine (IJLQIP20 in module IJLQIP) to wait for the breakoff.

External Routines Used: EXCP (SVC 0)

CANCEL MESSAGE ROUTINE

Module Name: IJLQCM (Chart CM)

Entry Point: Expansion of the CANCELM macro instruction generates a BALR to the routine at IJLQCM, 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.

Function: Causes the message to be canceled 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 are indicated in the error-halfword, return is made to the next LPS instruction. When any of the error conditions specified by the error mask is indicated in the error-halfword, or when the error mask is

zero, linkage is made to the Recall routine to obtain the message header.

After the header segment is returned (address in register 6) by the Recall routine, the cancel bit is set in the buffer status byte (BSTA) of the header prefix. This effects cancellation of the message. A zero is moved into the LMPL byte of the LCB to cancel the multiple route option. This effects cancellation of the message for any further destination that may be specified in the message header. Also, the converse bit is turned off so that the line is not left connected unnecessarily. Return is then made to the next LPS instruction.

Note: Linkage to the Recall routine is not performed if an invalid destination specification was previously detected for this message. The header need not be recalled because QTAM previously routed it to a special error queue on the DASD.

External Routines Used: Recall (IJLQIP22 in module IJLQIP).

DISTRIBUTION LIST ROUTINE

Module Name: IJLQDL (Chart DL)

Entry Points:

1. This routine is entered at IJLQDL + 14 from the QTAM SVC/Subtask Control Routine. Entry at this point occurs when the Distribution List subtask is dispatched because the Distribution List QCB appears at the top of the Ready Queue. The Distribution List QCB appears on the Ready Queue when an incoming message has a distribution list as its destination.

2. A second entry point into the Distribution List routine is made from the Cleanup routine (IJLQIP23 in module IJLQIP). This entry is to an unlabeled instruction.

Function: Implements multiple routing of a single message by causing a copy of the message to be placed on the DASD queue for each of the several destinations contained in a distribution list as defined by a LIST macro instruction.

This routine is composed of two separate, but related, sections of code. The first section (explained in item 1, which follows) of code is entered from the QTAM SVC/Subtask Control routine and performs an initialization procedure necessary for multiple routing via a distribution list. The second section of code (item 2, which follows) is entered from the Cleanup routine each time that routine is performing its function of routing the incoming message to its appropriate DASD queue.

The second section performs the function of recalling the header segment from the DASD and causes the current segment to be placed on the DASD queue for the next destination specified in the distribution list.

1. First section of Distribution List routine: This section of code is entered when a distribution list is specified as the destination for a message. It obtains the address of the LCB and the destination key (relative address) of the distribution list entry from the buffer prefix, and places the destination key into the LDLD field of the LCB.

   The terminal table entry for the first entry in the list is accessed, and the QCB address for this destination is placed into the LCB (LQDT field) and into the buffer prefix (BQAD field). The relative address of the terminal table entry is also placed into the prefix (BTDO field). Exit is then made (via a direct branch) to the Priority Search (QSVCPRI) subroutine in the SVC/Subtask Control routine. This exit causes the filled buffer to be posted to the QCB for the first destination in the list.

   If this is the first time this routine has been entered, linkage is made to the End Insert Routine before exiting to the Priority Search subroutine. The End Insert routine causes the address of the second section of the Distribution List routine to be placed permanently into the chain of routines to be entered from the Cleanup routine in its handling of each incoming message segment. After the first time, this entry into the End Insert routine is not made (use of the End Insert routine is described in detail in the description of the EOA routine).

2. Second section of the Distribution List routine: This section of code is entered from the Cleanup routine (after its address is placed in the Cleanup address chain as described above) each time the Cleanup routine has completed the routing of a message segment to the appropriate DASD queue. If multiple routing via a distributionlist is not in progress

orhas been completed (indicated by zeros in the LDLD field of the LCB or by the dummy last entry in the list), return is made to the Cleanup routine via the return address in register 14.

If multiple routing is in progress, i.e., there is another valid entry in the distribution list, the terminal table entry for that destination is accessed, and its QCB address is placed into the LCB (LQDT field). The LDLD field is updated for accessing the next entry in the list. Linkage is then made to the Recall routine to obtain a copy of the header.

Upon return, the relative address of the terminal table entry for this destination is placed into the BTDO field of the buffer prefix. Exit is then made to the beginning of the Cleanup routine (IJLQIP23 in module IJLQIP). This causes the current segment to be posted to the queue for this destination. After posting this segment, Cleanup returns to this section of the Distribution List routine for handling of the next entry in the list. This continues until all entries in the distribution list have been processed.

External Routines Used:

Recall (IJLQIP22 in module IJLQIP)

Cleanup (IJLQIP23 in module IJLQIP)

End Insert (IJLQNDRT in module IJLQIP)

Priority Search (QTAM nucleus)

DATE STAMP ROUTINE

Module Name: IJLQDT (Chart DT)

Entry Point: Expansion of the DATESTMP macro instruction generates a BALR to the routine at IJLQDT, using register 15 as the branch address register and register 14 as the return register. Register 14 serves also as the parameter list register. The parameter list passed to the routine consists of one item: a halfword containing, in binary, the length (9) of the date field to be inserted in the message header.

Function: Obtains the current date from the DCS communication region, and inserts it in the message header in either the format bmm/dd/yy or bdd/mm/yy, where b is a blank, dd is the day, mm the month, and yy the year. Prior to inserting the date, the routine links to the Expand Header routine, which "expands" the header by shifting, nine places to the left, all message characters preceding the place in the header where the date is to be inserted. The date is inserted in the field thus created. Return is made to the next LPS instruction.

External Routines Used: Expand Header (module IJLQEX)

END OF ADDRESS (EOA) ROUTINE

Module Name: IJLQEA (Chart EA)

Entry Points:

1. Expansion of the EOA macro instruction generates a BALR to the Message Type routine at IJLQMT, using register 15 as the entry register and register 14 as the return register. Register 14 also serves as a parameter list register. The parameter list passed to the Message Type routine consists of the size of the EOA character sequence (one to eight) and the actual EOA character sequence specified in the macro. Parameter register 1 contains the address of the End of Address (EOA) routine. If the Message Type routine does not find the specified EOA sequence in the next nonblank field of the header, it effects linkage to the EOA routine at IJLQEA.

2. A second entry point into the EOA routine is effected from the Cleanup routine (IJLQIP23 in module IJLQIP). This entry is to an unlabeled instruction that begins a separate section of code executed only when entry is from the Cleanup routine. The address of the LCB for the line over which the current message was received is passed in register 4.

Function: Causes multiple routing of an incoming message when more than one destination is specified in the header.

The EOA routine is divided into two separate, but related, sections of code. The first section, item 1 which follows, is entered from the LPS (via the Message Type routine) and performs the function of locating each successive destination code specified in the header of the message. It also performs an initialization procedure that causes the second section of code to be executed at the appropriate time.

The second section of code, item 2 which follows, is entered from the Cleanup routine each time that routine is performing its function of routing the incoming message to its appropriate destination queue on the direct-access storage device (DASD). This second section performs the function of recalling the header segment from the DASD and causing the current segment to be placed on the DASD destination queue for each destination (after the first) specified in the message header.

1. First Section of EOA routine: This section of code is entered from the EOA macro expansion (via the Message Type routine) for handling of each successive destination code in the header. When the Message Type routine finds the specified EOA sequence in the header, it does not enter the EOA routine because this is a signal that all destination codes have already been handled. Instead, it returns to the next instruction in the EOA macro-expansion.

   When entered, this section of the EOA routine computes and saves (in the LMPL field of the LCB) the offset from the start of the header to the next destination code in the header. (The previous destination code was located by the Route Message routine.)

   If this is the first time the EOA routine has been entered (from any LPS), it links (BAL) to the End Insert routine in module IJLQIP. The End Insert routine places the address of the second section of the EOA routine into a chain of addresses (by priority); the addresses in this chain are the addresses of routines to be entered from the Cleanup routine in its handling of each incoming message.

   The End Insert routine is never entered again from EOA while QTAM is active. This is accomplished by an instruction modification technique as follows: Prior to returning to the EOA routine, the End Insert routine overlays the second operand of the BAL instruction (in the EOA routine) with the address of the Skip Character Set routine (module IJLQSK). End Insert then returns, via the return register minus four, to the same BAL instruction. Thereafter, the BAL instruction is always a branch to the Skip Character Set routine.

   After computing the offset to the next destination code (or after return from End Insert on the first entry as just described), the EOA routine exits to the Skip Character Set routine. This routine advances the scan point past the specified EOA character sequence (bypassing any intervening destination codes) and returns to the next instruction in the EOA macro expansion.

   The instruction in the EOA macro expansion to which control is returned makes a test to determine if a duplicate message header is being handled. If the header is not a duplicate, normal LPS header processing continues. If it is a duplicate header, a branch is made to the ENDRCV macro expansion because LPS header processing has already been performed for this header.

2. Second Section of EOA routine: This section of code is entered from the Cleanup routine (after its address is placed in the Cleanup address chain as just described) each time the Cleanup routine has completed the routing of a message to the appropriate DASD destination or process queue. If multiple routing is not in progress or has been completed (indicated by a zero in the LMPL field of the LCB), return is made to the Cleanup routine.

   If multiple routing is in progress, i.e., if there is another destination code to be handled in the header, linkage is made to the Recall routine (IJLQIP22 in module IJLQIP, charts 16, 17) to obtain a duplicate copy of the header. The scan pointer (register 5) is reset to point to the next destination code in the header by using its offset previously computed and saved in the LMPL field of the LCB. The LMPL field is then set to zero. Exit is then made to the Route Message routine which handles the next destination code and exits to the first instruction of the EOA macro expansion in the appropriate LPS (actually the return address established by the ROUTE macro expansion and saved by the Route routine).

   Execution of both sections of the EOA routine is repeated until all destination codes in the header have been handled.

External Routines Used:

Skip Character Set (module IJLQSK)

End Insert (IJLQNDRT in module IJLQIP)

Recall (IJLQIP22 in module IJLQIP)

Route Message (module IJLQRG)

END OF BLOCK ROUTINE

Module Name: IJLQEB (Chart EB)

Entry Point: Expansion of the EOB macro instruction or of the ENDRCV macro instruction (if this macro is present in a WTTA LPS) generates a BALR to the routine at IJLQEB, using register 15 as the entry register and register 14 as the return register. No parameters are passed.

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

1. This routine is entered from the EOB macro expansion whenever a message block (delimited by an EOB or EOT character) has been received or sent. The routine performs error checking on the transmission and causes a read or write continue operation to be initiated if the message block was received or sent without error.

   If the message has been cancelled, an error message sent, or the message rerouted, return is made to the next LPS instruction. Similarly, if the status byte of the CSW indicates an end of transmission, a unit exception, or that the residual count in the CSW is zero, return is to the next LPS instruction. If none of the preceding conditions are detected, the End-of-Block bit in the buffer prefix is set because there was a positive indication that this message was sent or received correctly.

   In setting up for sending the next block, the scan pointer is adjusted to segment size and stored in the header prefix. The LEOB halfword in the LCB is updated by storing the segment size in this field. Otherwise, a write continue indication is set in the LOPC field of the LCB. This causes the Physical I/O routine, when entered, to generate a write continue channel program to write the next block of the message.

   If the transmission was a read operation, the read continue indicator is set in the LOPC field of the LCB. This causes the Physical I/O routine, when entered, to generate a read continue channel program to read the next block of the message. For both operations the buffer is set to be reused. The routine exits to the Physical I/O routine which generates and initiates execution of the appropriate channel program.

2. This routine is entered from the ENDRCV macro expansion to test the WRU flag in the LCB. If this flag is off, return is made to the next LPS instruction. If the WRU flag is on, this indicates that the last received character is WRU. In this case, an exchange of identification sequences 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 LOPC 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 exchange of identification sequences and reads the rest of the input message, provided EOM is different from WRU.

External Routines Used: QTAM Physical I/O (module IJLQRW)

END OF BLOCK AND LINE CORRECTION ROUTINE

Module Name: IJLQEC (Chart EC)

Entry Point: Expansion of the EOBLC macro instruction generates a BALR to the routine at IJLQEC, using register 15 as the entry register and register 14 as the return register. No parameters are passed.

Function: This routine is entered from the EOBLC macro expansion whenever a message block (delimited by an EOB or EOT character) has been received or sent. In general, the routine performs error checking on the transmission and causes a read or write continue operation to be initiated if the message block was received or sent without error.

If the message has been canceled, an error message sent, or the message rerouted, return is made to the next LPS instruction. If the error halfword in the LCB indicates a transmission error, the retry counter is incremented by one. If retransmission has already failed three

times, the cancel bit is set in the LDFG field of the LCB for the line so that succeeding segments of the message in error will be cancelled, and return is made to the next LPS instruction. If not, the routine branches and links to the Recall routine to obtain the header. If the line is sending to a remote terminal, the write continue indicator is set in the LOPC field of the LCB, and the routine exits to the start of the LPS for another try at transmission. Prior to the exit the error halfword is cleared for the retry.

EOBLC is also used to initiate a retry operation if a bus-out check occurred while sending to an IBM 2260 or IBM 1053 locally attached. The IBM 2260 Local Appendage sets an indicator in the LOPC field of the LCB to indicate the operation to be retried.

Failing Operation	Retry Operation
Write Initial to IBM 2260 Local	Erase, Write Initial
Write Initial to IBM 1053 Local	Write Initial
Write-at-line-address to IBM 2260 Local	Write-at-line-address

After recalling the Header of the message in error (via linkage to the Recall routine), exit is made to the start of the LPS for retry. If the error occurs a second time, EOBLC returns to the next in-line LPS instruction for user error processing.

If the line is receiving, action is taken to retry receiving the message. If there is no EOB present in any buffer position other than the last position, the cancel bit of the BSTA field in the prefix is set. The sequence-in number in the TSIN field of the terminal table entry is adjusted to account for the retry. Linkage is made to the Recall routine to obtain the header. The scan pointer and the LEOB field of the LCB are updated. Any space in the header area that is reserved for time, date, and sequence-out number insertions are filled with idle characters. The distribution list and multiple routing indicators are cleared, and the error halfword is reset to zero. The retry indicator is set in the LOPC field of the LCB.

The buffer size is stored in the BSSZ field of the prefix to indicate the message size, and the buffer is set for reuse. Exit is to the Physical I/O routine which initiates the retry. If there were no transmission errors, tests are made in the CSW. If the status byte of the CSW indicates an end of transmission, a unit exception, or that the residual count in the CSW is zero, return is to the next LPS instruction. The End of Block bit in the prefix is set, because there was a positive indication that this message was sent or received correctly.

In setting up for sending the next message, the scan pointer is adjusted to segment size and stored in the header prefix. The retry counter is set to zero in the LEHW field of the LCB. The LEOB halfword is updated by storing the segment size in this field. If the transmission was a write operation, a test is made to see if the next character to be written is an End of Transmission. If the next character is an EOT, there is no further execution of the routine, and a return is made to the next LPS instruction. Otherwise, the write continue indication is set in the LOPC field of the LCB and exit is made to the Physical I/O routine.

If the transmission was a read, the read-continue indicator is set before exit to the Physical I/O routine. The Physical I/O routine generates and initiates execution of the appropriate channel program to continue reading or writing the next block of the message.

External Routines Used:

QTAM Physical I/O (module IJLQRW)

Recall (IJLQIP22 in module IJLQIP)

ERROR MESSAGE ROUTINE

Module Name: IJLQER (Chart ER)

Entry Point: Expansion of the ERRMSG macro instruction generates a BALR to the routine at IJLQER, using register 15 as the entry 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 the error mask in hexadecimal notation. Register 0 contains the length of the error message (0 if an address is specified). The address of an area that contains the destination code is passed in register 2, and the address of the error message text is passed in register 1.

Function: 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 are indicated in the error-halfword, return is to the next LPS instruction. When an error condition is encountered, linkage is made to the Recall routine (in module IJLQIP) 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 text area of 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 BSSZ field of the prefix, and a single segment message is indicated in the BSTA field of the prefix.

If the error message is to be sent because of an invalid incoming sequence number, the error message will be scanned. If the special character $ is encountered, the correct input sequence number is moved into the four bytes following the $, and the $ is overlaid with a blank. If a second $ is found before the end of the error message, the invalid sequence number is moved into the four bytes following the $, and this second $ is also overlaid with a blank.

Linkage is made to the Lookup Terminal Table Entry routine, which looks up the destination code in the terminal table and places the relative address of its terminal table entry in the BDTO field of the header prefix. Return to the next LPS instruction is made by the Lookup routine.

External Routines Used:

Recall (IJLQIP22 in module IJLQIP)

Lookup Terminal Table Entry (Module IJLQLK)

EXPAND HEADER ROUTINE

Module Name: IJLQEX (Chart EX)

Entry Point: The routine is entered via a BALR from SEQOUT, TIMESTMP, and 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.

Function: Creates in the message header the space needed to insert a new field in the header.

The number of characters to be shifted is computed by subtracting from the value in the scan pointer register the sum of the address of the buffer plus 31 (so as not to include the prefix) plus the number of characters to be expanded. If the result is negative, return is made to the next LPS instruction, because there is no space for the shift. If there is space available, all message characters that have already been processed are shifted to the left of the specified amount.

After the characters of the header have been shifted, a blank is inserted as a left delimiter at the start of the created field. The scan pointer offset for the next header destination code (saved in LMPL field of the LCB) is decremented by the length of the new field in case multiple routing is in progress. Return is to the calling routine.

External Routines Used: None

INTERCEPT MESSAGE ROUTINE

Module Name: IJLQIT (Chart IT)

Entry Point: Expansion of the INTERCPT macro instruction generates a BALR to the routine at IJLQIT, using register 15 as the entry 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 the specified error mask in hexadecimal notation. The parameter register 1 contains the address of an optional subfield in the terminal table defined by the user to contain the relative record number of the message being intercepted.

Function: Causes suppression of all message transmission to a terminal when any of the error conditions specified by the 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 mask are indicated in the error-halfword, return is to the next LPS instruction. If the intercept function is to be performed, the routine makes linkage to the Recall routine to recall the header.

Upon return from Recall, the routine turns off the serviced bit and turns on the priority bit in the header prefix so that a new sequence number is not assigned. The send bit in the TSTA byte of the terminal table entry for the destination terminal is turned off to indicate that messages on the queue for the destination are to be withheld from transmission. If the intercept bit in the TSTA byte is on, indicating that a previous header disk address is in the Intercpt subfield, and if the header disk address is greater than the disk address already in the Intercpt subfield, return is to the next LPS instruction.

If the header disk address is less than the address in the Intercpt subfield or if the intercept bit in the TSTA byte is not on, the intercept bit is set to one, to indicate that a message on the queue was not transmitted, and the header disk address is put into the Intercpt subfield in the user area of the terminal table entry. The offset of the intercept subfield from the beginning of the terminal table entry is computed and saved (for use by the Release Message routine) in a byte reserved by the expansion of the LPSTART macro. Return is to the next LPS instruction.

External Routines Used: Recall (IJLQIP22 in module IJLQIP)

AUDIO INPUT MESSAGE LOGGING ROUTINE

Module Name: IJLQLG (Chart LG)

Entry Point: The Logging routine is entered at IJLQLG via linkage from the LOGSEG macro expansion when the ARU operand has been specified. Upon entry, register 4 contains the address of the ALCB of the line requiring the input message logging.

Function: Prepares in the ALCB the preformatted prefix to be logged with the input message. This prefix immediately precedes the input buffer and includes, in succession:

1. Logical unit number of the symbolic line assignment ("nnn" of SYSnnn)
2. Length of the input data to be logged
3. Date stamped information
4. Time stamped information when the timer is available and the keyword operand LOGTIME=YES has been specified in the DTF line group of the line. The unused part of the buffer is reset to binary zeros.

Register 2 contains the address of the first byte of the prefix, and control returns to the next instruction in the LOGSEG macro expansion.

External Routines Used: None.

LOOKUP TERMINAL TABLE ENTRY ROUTINE

Module Name: IJLQLK (Chart LK)

Entry Point: The Lookup routine is entered at IJLQLK via linkage from the DIRECT macro expansion or via a direct branch from the Error Message, Route Message, or Reroute Message routine. Upon entry, register 2 contains the address of the work area containing the name to be looked up.

Function: Obtains, in succession, the name contained in each terminal table entry and compares it with the name provided in a work area. Each time a nonequal compare results, the process is repeated with the name from the next terminal table entry. When an equal compare results, the routine obtains, from the terminal table entry, the address of the queue control block for the destination (or process) queue, and places this QCB address in the LQDT field of the LCB. The offset of the entry relative to the beginning of the terminal table is placed in the BDTO field of the buffer prefix.

If the terminal name in the work area does not compare with any entry 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 error (dead-letter) queue in the LQDT field of the LCB.

External Routines Used: None

CONVERSATIONAL MODE ROUTINE

Module Name: IJLQMC (Chart MC)

Entry Points:

1. If there is no character specified in the second operand of the macro, the expansion of the MODE macro instruction generates a BALR to the routine at IJLQMC, using register 15 as the entry 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 passed in register 1 to the Message Mode

Interface routine (module IJLQMM) which enters this routine at IJLQMC if the specified character is found in the header.

2. A second entry into the Conversational Mode routine is made from the Cleanup routine to an unlabeled entry point. This is a separate section of code that is executed only when entry is from the Cleanup routine. The address of the LCB for the line that is to operate in conversational mode is passed in register 4.

Function: Causes the line over which the current message is being received to be placed in a conversational mode; that is, the line connection is maintained until a reply to the inquiry is sent.

The Conversational Mode routine is composed of two separate, but related, sections of code. The first section, item 1 that follows, performs the initialization procedures necessary for the conversational mode function to be implemented. The second section of code, item 2 that follows, implements the function.

1. First section of Conversational Mode routine: This section of code is entered from the MODE macro expansion (unconditional CONVERSE) or from the Message Mode Interface routine (conditional CONVERSE). Entry is at IJLQMC. The "converse" bit is set in the LSTA (line status) byte of the LCB.

   If this is the first time this routine has been entered, linkage is made to the End Insert routine (IJLQNDRT in module IJLQIP). End Insert performs an initialization function that ensures that the second section of the Conversational Mode routine is executed (via an entry from the Cleanup routine) at the proper time. (The function performed by the End Insert routine is identical to that described under the ECA routine.) Return is to the next LPS instruction.

2. Second Section of the Conversational Mode routine: This section of code is entered (at an unlabeled entry point) from the Cleanup routine during its handling of each message. If the converse bit is not on (in LSTA field of LCB), if there has been a polling or addressing error is on the line, orthe destination is not a processing program, the request for conversational mode is ignored and return is made (via register 14) to the Cleanup routine.

   If the line is still receiving, exit is made to the LPS Control routine to wait for completion of the incoming inquiry message. If the line is sending, the line is turned around to receive by setting the converse and receive bits in the LSTA field of the LCB.

   The QCB for the process queue (to which the inquiry is being directed) is assigned a high priority to ensure most rapid handling of the inquiry message and its reply. If the end of the polling list for the line (if nonswitched) has been reached, the pointer to the current polling list entry (LPPT field of LCB) is reset to the beginning of the polling list. Exit is made to the LPS Control routine to continue processing.

   The buffer containing the inquiry message was flagged to indicate conversational mode. When this buffer is released after completion of a GET by the processing program, the Return Buffer routine (in module IJLQIP) posts this buffer to the LPS Queue. When this buffer is obtained from the LPS Queue by the LPS Control routine, control returns to the Conversational Mode routine (via the Cleanup routine). The Conversational Mode routine then begins a scan to determine if a message (the reply) is on the DASD queue for the source terminal (that is, the terminal that initiated the inquiry).

   If no message is on the queue (or if the source terminal was not identified), exit is made to the Cleanup routine. If there is a message, initialization procedures are performed for sending the reply: the send bit is set in the LSTA field of the LCB, and the write initial indicator is set in the LOPC field. The buffer containing the original inquiry message is released, via a QPOST, to the Available Buffer queue. The disk address of the reply message header is placed into the LCB. A request for sending the reply is posted (via a QPOST) to the Disk I/O queue (this causes the Disk I/O subtask to be activated for reading the reply from disk and most rapid sending of the reply message). Exit is made to the LPS Control routine to continue normal processing.

External Routines Used:

LPS Control (IJLQIP20 in module IJLQIP)

End Insert (IJLQNDRT in module IJLQIP)

INITIATE MODE ROUTINE

Module Name: IJLQMI (Chart MM)

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 IJLQMI, using register 15 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 the parameter register 1, and the routine is entered by a branch from the Message Mode Interface routine.

Function: The routine sets the Initiate bit in the LSTA field of the LCB. Return is to the next LPS instruction.

External Routines Used: None

MESSAGE MODE INTERFACE ROUTINE

Module Name: IJLQMM (Chart MM)

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

Function: This routine is entered when there is a specific character specified in the second operand of the MODE macro. Linkage is made to the Scan Header 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 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.

External Routines Used: Scan Header (Module IJLQSH)

PRIORITY MODE ROUTINE

Module Name: IJLQMP (Chart MM)

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 IJLQMP, using register 15 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 Mode routine is placed in parameter register 1, and the routine is entered by a branch from the Message Mode Interface routine.

Function: Linkage is made to the Scan Header routine which obtains and provides the address of the first nonblank character in the header. The first nonblank character is moved into the LTPR field of the LCB and becomes the priority of the message. Return is made to the next LPS instruction.

External Routines Used: Scan Header (Module IJLQSH)

MESSAGE TYPE ROUTINE

Module Name: IJLQMT (Chart MT)

Entry Point: Expansion of the MSGTYPE macro instruction generates a BALR to this routine at IJLQMT, 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 field size and a constant containing the character sequence with which the scanned character sequence is to be compared.

Function: Saves the scan pointer, then links to the Scan Header routine, which obtains and provides to the Message Type routine the next character sequence in the message header. The Message Type routine compares the character sequence provided with the character sequence specified in the MSGTYPE macro statement.

If the character sequences 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 MSGTYPE or delimiter macro in the LPS, thus bypassing any instructions preceding the next MSGTYPE or delimiter macro. Because the scan pointer is restored when the two character sequences are not the same, a

series of Message Type routines may be executed, each examining the same message type character sequence in the header.

External Routines Used: Scan Header (Module IJLQSH)

OPERATOR CONTROL ROUTINE

Module Name: IJLQOC (Charts OC, OD, OE, OF, OG, and OH)

Entry Point: Expansion of the OPCTL macro instruction generates a BALR to the Operator Control routine at IJLQOC. Register 15 is the branch register, register 14 is the return register, and register 1 contains the address of a parameter list generated by the macro instruction.

Function: Interpreting, selecting, and performing a control operation requested by the user by entering a message from a user-specified control terminal.

The routine is branched to if the code generated by the macro instruction determines that the message is being sent from an operator control terminal. The routine compares the next field in the buffer with the specified control message identifier. If it is not equal, return is made to the next LPS instruction. The routine tests if the scan pointer has run out of the header. If yes, the scan pointer is reset to where it was when the OPCTL macro instruction gained control, and return is made to the next LPS instruction. The routine tests if the message is to be cancelled. If yes, the message is routed back to the terminal that sent the message by the Routing subroutine.

The routine then branches to the Scan Header routine to access the next field in the header. Upon regaining control, it tests to determine if the field is a valid operator control operation. If it is not, the message is routed back to the terminal that sent the message by the Routing subroutine. If it is valid, the routine selects the correct subroutine for the operation and branches to it. The operation subroutines are discussed in detail. Also discussed are four auxiliary subroutines used by more than one operation subroutine, SUB1, SUB2, UNPAK, and ROUTE (the Routing subroutine).

CHNGT Operation Subroutine: The subroutine branches to SUB1, which returns to the CHNGT Operation subroutine the address of the terminal table entry for the terminal specified in the operator control message and the number of bytes in the buffer following the termname operand.

The information in the buffer, from the beginning of the data field to the end of the buffer, is translated according to a special translate table labeled TABLETRN. Use of this table translates IBM 1050 code into hexadecimal.

A translate-and-test instruction is then executed on the buffer from the beginning of the data field to the end of the buffer. The table used with this instruction (TABLETST) causes the instruction to halt whenever a byte is tested that is not equal to one of the bytes X'F0' through X'FF'.

If the instruction exhausts the buffer count without a halt, the message is returned to the source terminal via the Routing subroutine. If the instruction halts, the byte at which the instruction halted is tested to determine if it is one of the valid delimiting characters for the data field (blank, EOB or EOT). If it is not, the message is returned to the source terminal via the Routing subroutine.

The subroutine tests to determine if the number of characters in the data field (excluding the delimiting character) is even. If it is not an even number, the message is returned to the source terminal via the Routing subroutine.

The subroutine tests to determine if the number of characters in the data field (excluding the delimiting character) can fit into a single terminal table entry. If not, the message is returned to the source terminal via the Routing subroutine.

The subroutine then posts the Change Terminal Table Entry QCB to itself via a QPOST (SVC 31). The Change Terminal Table Entry subtask is entered in supervisor state.

The subtask tests the QCB for the terminal table entry to make sure that the entry is for a terminal. If it is not, the message is returned to the source terminal via the Routing subroutine.

The subtask tests to determine if the line for this terminal is active. If it is not active, the terminal table entry is changed starting with the sequence-in (IJLQTSIN) field.

If the line is active, the sequence-in and sequence-out fields in the entry are not changed. The entry is changed starting with the status field (IJLQTSTA) and the data field of the operator control message is truncated accordingly.

The information from the data field in the buffer is then packed into a work area, and the work area is moved into the terminal table entry. The subtask exits to Qdispatch.

Upon reentry, the subroutine performs checking procedures, then returns to the macro generated code to branch to the ENDRCV delimiter macro instruction.

COPYC Operation Subroutine: An audio switch is set on to allow SUB1 to scan the audio line table. Then a branch is made to SUB1, which returns the address of the terminal or audio line entry specified. For a terminal entry, a test is made to make sure that the entry is for a terminal. If it is not, exit is made to the Routing subroutine.

The address of the LCB or ALCB is obtained and the buffer is posted to a special QCB (COPYCLR) to enter the COPYCLR subtask at COPYCLR+6 in disabled state.

The threshold line error counters are added to the cumulative line error counters. The cumulative counters are converted and formatted into the work area. A branch is made to UNPAK to unpack and translate the work area into the buffer. The threshold counters are cleared and exit is made to Qdispatch.

Upon return, the subroutine branches to the Routing subroutine to return the buffer to the caller.

COPYT Operation Subroutine: The subroutine branches to SUB1, which returns to the COPYT Operation subroutine the address of the terminal table entry for the terminal specified in the operator control message and the number of bytes in the buffer following the termname operand.

The COPYT Operation subroutine tests to determine if the length remaining in the buffer is enough to contain the information to be placed in it from the terminal table. If there is enough length, the subroutine branches to UNPAK to insert the terminal table entry into the buffer starting with the IJLQTSIN field. If there is not enough length, the subroutine sets flags to indicate to UNPAK to truncate the entry at the low-order end and to insert the amount of the entry that will fit.

On return, the subroutine branches to the Routing subroutine to send the message back to the source terminal.

INTERCPT Operation Subroutine: The subroutine tests the parameter list to determine if the INTRCPT=YES operand was specified in the OPCTL macro instruction. If it was not, the message is returned to the source terminal via the Routing subroutine.

The subroutine then branches to SUB1 to obtain the address of the terminal table entry specified in the operator control message. It tests to determine if the entry is a terminal entry. If not, the message is returned to the source terminal via the Routing subroutine.

If it is a terminal entry, the send bit in the IJLQTSTA field is turned off, and the subroutine returns to the macro generated code to branch to the ENDRCV delimiter macro instruction.

INTREL Operation Subroutine: The subroutine turns on a switch (SW1) to indicate that the entry is from INTREL, and branches directly to STOPLN. Processing proceeds as described in STOPLN Operation Subroutine up to entry into SUBTASK1.

On entry into SUBTASK1, the LCB is deactivated by insertion of X'00' into the first byte of the LCB. A test is then made to determine if entry is from INTREL. If not, processing continues as described under STOPLN. Otherwise, the subtask branches to SUBTASK2.

At SUBTASK2, the LCB is removed from the Ready Queue, and the switch representing entry from INTREL (SW1) is cleared.

The buffer is modified to cause it to appear to the system as an LCB. The subtask prepares this pseudo-LCB for insertion into the time queue with a two-minute delay. The status field of the LCB (LSTA) is changed to prevent the LCB from being started while the pseudo-LCB is in the Time Delay routine.

In the Time Delay routine, the pseudo-LCB is recognized as coming from INTREL. A branch is made to the QTAM Post routine to post the pseudo-LCB to itself to activate the next subtask on the STCB chain of the pseudo-LCB, causing entry into STCB2.

At STCB2, a branch is made to the QTAM Post routine to post the real LCB to itself. At this time, the system also dispatches the pseudo-LCB, which is now treated as a buffer again, to the Available Buffer queue.

After the line has been stopped and entry has been made into TRYEXIT, another

test is made to determine if entry is from INTREL. If yes, the switch indicating entry from INTREL (SW1) is cleared and the INTREL switch in the LCB (in field LDFG) is set. A test is made to determine if the LCB has been deactivated. If not, exit is made to Qdispatch. Otherwise, a branch is taken back into SUBTASK2 at the point where the buffer is modified to appear to the system as an LCB (BUILD2).

Entry into the INTREL subroutine may be from the Operator Awareness routine when it posts the buffer to a special QCB (IJLQOC70) associated with a special STCB (IJLQOC80). (Both are located in the Operator Control module.) The subtask activated tests to determine if the LCB is active. If it is not, a branch is taken into SUBTASK2 at the point where the buffer is modified to appear to the system as an LCB (BUILD2). If the LCB is active, the switch indicating entry from INTREL (SW1) is set, and a branch is taken to the STOP2 subtask at the point where the combined QCB-STCB is generated in the buffer (GENSTCB).

RELEASEM Operation Subroutine: The subroutine tests to determine if INTRCPT=YES was specified in the OPCTL macro instruction. If it was not, the message is returned to the source terminal via the Routing subroutine.

If the operand was so specified, the subroutine branches to SUB1 to get the address of the terminal table entry specified in the operator control message. Upon return, it tests to determine if the terminal is in intercept mode. If it is not, the subroutine returns to the macro generated code to branch to the ENDRCV delimiter macro instruction.

If the terminal is in intercept mode, the subroutine obtains from the parameter list the offset to the intercept optional field in the terminal table entry, and compares the current disk address in the QCB with the disk address saved there. If they are not equal, the disk address in the optional field replaces the current disk address.

The status of the terminal is then returned to active mode and the subroutine returns to the macro generated code to branch to the ENDRCV delimiter macro instruction.

STARTARU Operation Subroutine: An audio switch is set on to allow SUB1 to scan the audio line table. Then, a branch is made to SUB1, which returns the address of the audio line table entry specified in the operator control message. The address of the DTF line group is obtained and a test is made to determine if the DTF table has been opened. If not, the message is returned to the source terminal via the Routing subroutine.

If ALL is specified in the operator control message, the address of the first ALCB in the line group is accessed. If the relative line number is specified, a test is made to check if this relative line number is valid. If not, the message is returned to the source terminal via the Routing subroutine. If yes, the address of the corresponding ALCB is accessed.

When the ALCB stop flag is off, the line is already active and the line processing is completed.

When the ALCB stop flag is on, but the line is not effectively stopped, the stop flag is reset and the line processing is completed. When the ALCB stop flag is on and the line is effectively stopped, an EXCP (SVC 0) is issued to restart the audio line. If ALL has been specified, this restart procedure is made for each line. In any case, return is made to the macro generated to branch to the ENDRCV delimiter macro instruction.

STARTLN Operation Subroutine: The subroutine branches to SUB1, which returns to the STARTLN Operation subroutine the address of the terminal table entry specified in the operator control message and the number of bytes in the buffer following the termname operand. The subroutine then branches to SUB2, which translates the entry into EBCDIC in the low-order bytes of the buffer.

The subroutine tests the QCB for the terminal table entry to make sure that the entry is for a terminal. If it is not, the message is returned to the source terminal via the Routing subroutine.

The subroutine tests to determine if ALL is specified in the operator control message. If it is, the subroutine exits to the Change Line routine at the "start all" entry point. If it is not, entry is at the "start single line" entry point.

The Change Line routine returns an error code to the STARTLN Operation subroutine in register 15. The subroutine tests this register for a code of 0, which indicates that no errors occurred. If the register does not contain 0, the message is returned to the source terminal via the Routing subroutine. If the code is 0, no errors occurred, and return is made to the macro

generated code to branch to the ENDRCV delimiter macro instruction.

STOPARU Operation Subroutine: An audio switch is set on to allow SUB1 to scan the audio line table. Then, a branch is made to SUB1, which returns the address of the audio line table entry specified in the operator control message. The address of the DTF line group is obtained and a test is made to determine if the DTF table has been opened. If not, the message is returned to the source terminal via the Routing subroutine.

If ALL is specified in the operator control message, the address of the first ALCB in the line group is accessed. If the relative line number is specified, a test is made to check if this relative line number is valid. If not, the message is returned to the source terminal via the Routing subroutine. If yes, the address of the corresponding ALCB is accessed.

The ALCB stop flag is set on for the specified line or for each line in the line group if ALL has been specified, and return is made to the macro generated code to branch to the ENDRCV delimiter macro instruction.

STOPLN Operation Subroutine: The subroutine branches to SUB1, which returns to the STOPLN Operation subroutine the address of the terminal table entry for the terminal specified in the operator control message and the number of bytes in the buffer following the termname operand. The subroutine then branches to SUB2, which translates the entry into EBCDIC in the low-order bytes of the buffer.

A test is made to determine if at least 22 bytes past the buffer prefix are available for use in the buffer. If not, the message is returned to the source terminal via the Routing subroutine.

A test is made to determine if all lines are to be stopped in the line group that includes the line on which the operator control terminal is located. If so, the message is returned to the source terminal via the Routing subroutine.

A test is made to determine if the DTF table containing the line to be stopped has been opened. If not, the message is returned to the source terminal via the Routing subroutine.

The buffer size is extended, into the buffer prefix, to 56 bytes.

A test is made to determine if the subroutine was entered via an INTREL operator control message. If so, a branch is made to SUBTASK2 (described in INTREL Operation Subroutine).

A test is made to determine if ALL is specified in the operator control message. If so, parameters are set up for the number of LCBs in the line group.

The subroutine then issues a QPOST to post a special QCB (STOP1QCB) to itself. Entry into the STOP1 subtask is in supervisor state.

The STOP1 subtask changes a branching address in the Implementation module at IJLQIP25. This change causes the Post Send or the Post Receive routine (whichever is to gain control) to the exit to the Operator Awareness routine rather than to the Buffer Recall/Cleanup routine.

The STOP1 subtask tests the status field in the LCB (IJLQLSTA) for the line that is to be stopped. If the line is not active, SUBTASK1 branches to TRYEXIT. (Action taken at TRYEXIT is discussed later in this section.)

The STOP1 subtask tests to determine if an operator control STOPLN procedure is pending by examining the first STCB in the STCB chain of the LCB. If it is the STOPLN STCB (STOP2), a branch is made to TRYEXIT.

If the line is active, tests are made to determine if it is a switched line. If so, and if it is not actively transmitting, the subtask issues a HALT I/O to stop the Enable command for the line.

If a Read Initial channel program is initiated on a WTTA line, a test is made to determine whether data is being received over that line. If not, the subtask issues a HALT I/O to stop the Prepare command for the WTTA line.

A test is made to determine whether this is the first time this subtask has been reentered to stop other lines in a line group. If this is a reentry, the subtask branches to STOP2.

The STOP1 subtask loads the address of another special QCB (STOP2QCB) into the destination field of the buffer, then branches to Qdispatch. Control returns to the macro generated code.

The STOP2 subtask is activated when the Cleanup routine posts the buffer to the destination queue. Because the STOP2 QCB was inserted in the destination field of the buffer, the buffer is passed to STOP2 instead.

The STOP2 subtask generates in the buffer a combined QCB-STCB and coding to branch to SUBTASK1. This combined QCB-STCB is inserted as the first STCB in the STCB chain of the LCB for the line that is to be stopped.

A test is made to determine if entry was from INTREL. If not, exit is made to Qdispatch. If entry was from INTREL, the switch representing entry from INTREL (SW1) is cleared, the INTREL switch in the LCB (in field LDFG) is set, and exit is made to Qdispatch.

When activity ceases on the line, entry is made to the next subtask on the LCB STCB chain, which is now the combined QCB-STCB set up in the buffer. The coding in the buffer branches to SUBTASK1.

SUBTASK1 deactivates the LCB by inserting X'00' in the first byte of the LCB.

A test is made to determine if entry is from INTREL. If so, the subtask branches to SUBTASK2.

The next STCB from the STCB chain of the LCB is examined to determine if it is a full STCB. If it is not, the LCB is removed from the Ready Queue, thus stopping the line.

If there are other lines to be stopped, the subtask branches to TRYEXIT. Otherwise, exit is made to the QTAM Post routine to post the buffer to the Available Buffer queue.

At TRYEXIT, a test is made to determine if there are more lines to be stopped. If so, the next LCB in the DTF table for the line group is accessed, and return is made to the STOP1 subtask at the point where the line is examined to determine if it is active.

If there are no more lines to be stopped, a test is made to determine if SUBTASK1 has been executed. If it has, then the buffer has already been posted to the Available Buffer queue, and exit is made to Qdispatch. If SUBTASK1 has never been executed, this subtask exits to the QTAM Post routine to post the buffer to the Available Buffer queue.

SWITCH Operation Subroutine: The parameter list generated by the OPCTL macro instruction includes a pointer to the terminal table entry for the operator control terminal, and (if specified in the OPCTL macro instruction) a pointer to the terminal table entry for the alternate operator control terminal. The SWITCH Operation subroutine reverses these two addresses in the parameter list, then returns to the macro generated code to branch to the ENDRCV delimiter macro instruction.

If an alternate operator control terminal has not been specified, the message is returned to the source terminal via the Routing subroutine.

Common Operator Control Subroutines:

SUB1. The subroutine branches to the Scan Header routine to get the next field in the message (termname operand). Upon return, it branches to the Lookup Terminal Table Entry routine to determine if the termname operand is valid. If not and if the audio switch is off, the message is returned to the source terminal via the Routing subroutine. If not and if the audio switch is on, the audio line table is scanned to search a line table entry corresponding to the linename operand. If such a line entry is not found, the message is returned to the source terminal via the Routing subroutine. If the line entry is found, return is made to the calling routine.

For a terminal entry, the subroutine then calculates the number of bytes in the buffer following the termname operand. Passing these parameters, return is made to the calling subroutine.

SUB2. The subroutine gets the QCB for the terminal table entry specified, and tests to make sure that it is an entry for a terminal. If it is not, the message is returned to the source terminal via the Routing subroutine.

Otherwise, the relative line number and the address of the DTF table are saved. The scan pointer is moved past any delimiting blanks in the buffer, and return is made to the calling routine.

UNPAK. The subroutine unpacks and translates into valid EBCDIC the terminal table entry starting at the IJLQTSIN field into the remaining bytes of the buffer. If the remaining bytes cannot contain the terminal table entry, its low-order end is truncated and the data that can fit is inserted. Return is made to the calling subroutine.

Routing Subroutine. The subroutine moves the source identifier field (BSTO) to the destination identifier field (BDTO) in the header prefix. The address of the QCB for the source terminal is placed in the LCB and return is made to the macro generated code to branch to the ENDRCV delimiter macro instruction.

External Routines Used:

Lookup Terminal Table Entry (Module IJLQLK)

Operator Awareness (Module IJLQOA)

Scan Header (Module IJLQSH)

Change Line (Module IJLQCL)

Time Delay (in Implementation module)

POLLING LIMIT CONTROL ROUTINE

Module Name: IJLQPL (Chart PL)

Entry Point: Expansion of the POLLIMIT macro instruction generates a BALR to this routine at IJLQPL using register 15 as the branch address register and register 14 as the return register. Bits 24-31 of parameter register 1 contain the specified polling limit.

Function: Limits the number of messages to be accepted from a nonswitched 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 to the next LPS instruction. Otherwise, the current poll count (contained in the LMCT field of the LCB) is compared to the limit specified. If the count, incremented by one, is less than the limit, then return is to the next LPS instruction. If the count, incremented by one, exceeds or is equal to the limit, the polling pointer is set to the next terminal (in the polling list for the line) before returning to the next LPS instruction.

External Routines Used: None

PAUSE ROUTINE

Module Name: IJLQPZ (Chart PZ)

Entry Point: Expansion of the PAUSE macro instruction generates linkage to this routine at IJLQPZ, using register 15 as the entry register and register 14 as the return register. Register 14 also serves as a parameter list register. The parameter list consists of three items:

1. The control character which causes insertion of idle (or other) characters.
2. The number of idle (or other) characters to be inserted.
3. The hexadecimal configuration of the idle (or other) character(s) to be inserted.

Function: Causes the specified number of idle (or other) characters to be inserted in an outgoing message segment each time the specified control character is encountered in the segment.

The address of the first byte to be scanned for the special control character is computed for a header or text segment. The address of the previous BRB/CCW in the BRB-ring formed for sending this message is obtained from the LCCW field of the LCB. If the previous BRB/CCW is a Pause BRB/CCW, the number of Pause BRB/CCWs remaining in the Additional-CCW queue is obtained from the location labeled COUNT. Otherwise, the number of Pause BRB/CCWs is obtained from expansion of the BUFFER macro (location IJLQBFRZ+2). The buffer is then scanned for the specified control character.

If the character does not appear in the buffer, return is made to the next LPS instruction. If the character is found, the number of remaining Pause BRB/CCWs is decremented by one and stored in location COUNT. (If the count is zero, return is made to the next LPS instruction without further action. Insufficient Pause BRB/CCWs were specified in the BUFFER macro instruction, and no idle characters are transmitted.) The routine then loads the address of the Additional- CCW QCB into register 0 and links to the LPS Control routine at IJLQIP21.

The LPS Control routine issues a QWAIT (SVC 30) to request a Pause BRB/CCW from the element chain of the Additional- CCW queue. When the request is satisfied, control is returned to the Pause routine with the address of the Pause BRB/CCW in register 1.

The Pause BRB/CCW is a block of 24 bytes (for 3 CCWs) and is initialized and inserted into the BRB ring in the following manner (Figures 3 and 4 illustrate the BRB ring and buffer before and after insertion of the Pause BRB/CCW): The TIC address in the previous BRB/CCW is inserted into the third CCW of the Pause BRB/CCW and is overlaid by the address of the Pause BRB/CCW. The data address from the previous next BRB/CCW is placed into the first Pause CCW so that this CCW writes all message data preceding the control character and the control character itself.

The address of the data character following the specified control character is placed into the previous "next BRB/CCW." The counts for both data fields are computed (count 1 and count 2 in Figure 4)

and placed in the appropriate CCW's. The second Pause CCW is then initialized with the address and count of the idle (or other) characters to be transmitted. The final step is to place the address of the Additional CCW queue into the third Pause CCW; this is used for releasing the Pause BRB/CCW to the Additional-CCW queue after the segment has been transmitted.

The routine then branches back to resume scanning the remainder of the buffer for the specified control character. If it is encountered again, the preceding procedure is repeated. This process continues until the entire buffer has been scanned or until no further Pause BRB/CCWs are available. Return is made to the next LPS instruction.

External Routines Used: LPS Control (IJLQIP20 in Module IJLQIP)

ROUTE MESSAGE ROUTINE

Module Name: IJLQRG (Chart RG)

Entry Point: Expansion of the Route macro instruction generates a BALR to the routine at IJLQRG, 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 one item: a halfword containing in binary, 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).

Function: Links to the Scan Header routine to obtain the destination code in the incoming message header; then branches to the Lookup Terminal Table Entry routine, which looks up the destination code in the terminal table and places the relative address in the BDTO field of the incoming header prefix. Return to the next LPS instruction is made by the Lookup routine, rather than the Route Message routine.

External Routines Used:

Scan Header (Module IJLQSH)

Lookup Terminal Table Entry (Module IJLQLK)

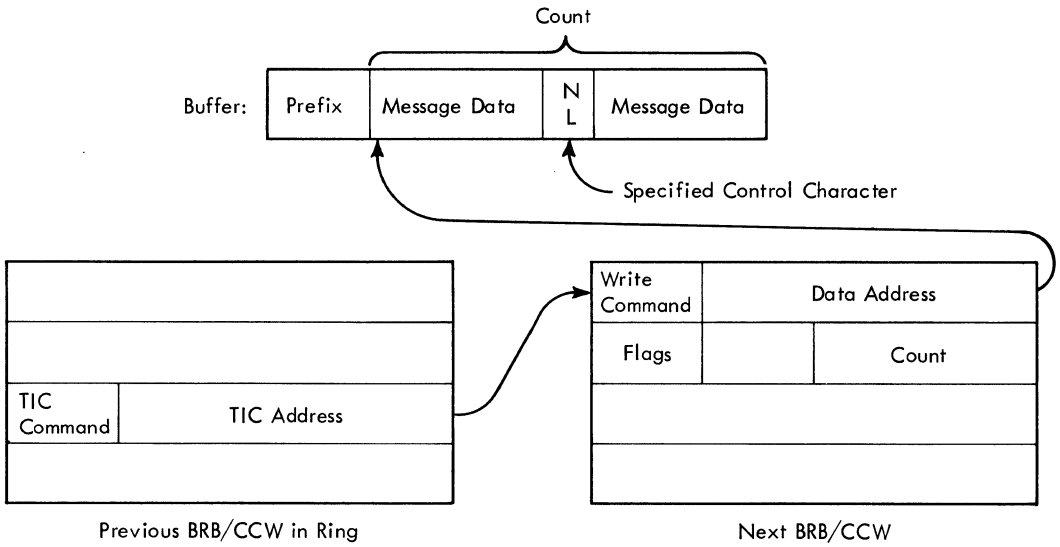


Figure 3. BRB Ring Before Insertion of Pause BRB/CCW

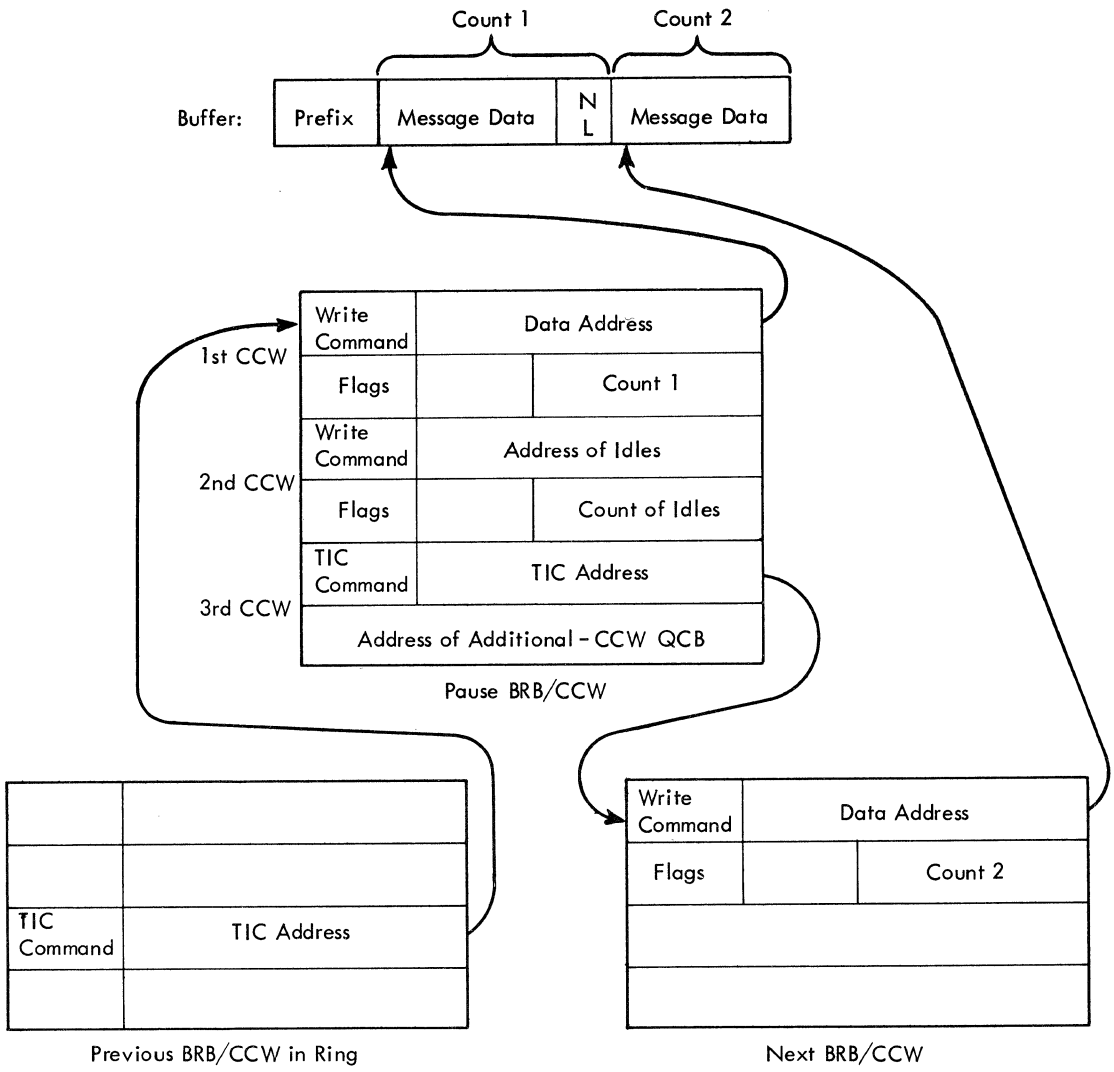


Figure 4. BRB Ring After Insertion of Pause BRB/CCW

REROUTE MESSAGE ROUTINE

Module Name: IJLQRR (Chart RR)

Entry Point: Expansion of the REROUTE macro instruction generates a BALR to the routine at IJLQRR, 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. In parameter register 2 is the address of a field containing the code of the alternate destination.

Function: Causes a message to be sent to an alternate destination 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 are indicated in the error-halfword, return is made to the next LPS instruction. Otherwise, linkage is made to the Recall routine (in module IJLQIP), which obtains the header. Upon return, the routine branches to the Lookup Terminal Table Entry routine, which looks up the terminal table entry for the alternate destination and places its relative address in the BDTO field of the incoming header prefix. Return to the next LPS instruction is made by the Lookup routine.

External Routines Used:

Recall (IJLQIP22 in Module IJLQIP)

Lookup Terminal Table Entry (Module IJLQLK)

SCAN HEADER ROUTINE

Module Name: IJLQSH (Chart SH)

Entry Point: This routine is entered via a BALR from the IJLQMP, IJLQSI, IJLQSK, IJLQMM, IJLQRG, IJLQSR, and IJLQMT 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 the variable-field-length indicator. Upon entry, the scan pointer register (register 5) contains the address of either the last character of the last header field scanned, or the first blank character following the field.

Function: Obtains one or more nonblank characters from a fixed or variable-length header field and places them in a work area, the address of which is returned in register 2 to the calling routine.

The Scan Header routine moves the scan pointer register one position at a time, and places any nonblank header character found into the work area. This operation is repeated until 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 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 variable-length, an indicator (2X'FF') is passed to the routine in the parameter list. The Scan Header routine scans to the first nonblank character and then places it and all succeeding nonblank characters into the work area. The scan is delimited when either:

1. the first blank character is encountered, or
2. the eighth nonblank character is moved into the work area.

During any scan operation if the end of the segment is reached before the scan is completed, the scan is terminated and the "incomplete header" bit (bit 5) is set in the error-halfword.

External Routines Used: None

SEQUENCE NUMBER IN ROUTINE

Module Name: IJLQSI (Chart SI)

Entry Point: Expansion of the SEQIN macro instruction generates a BALR to the routine at IJLQSI, 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 one item: the number of character positions for the input-message sequence number field. If this operand is omitted, a hexadecimal "FF" indicates a variable-length field.

Function: The routine links to the Scan routine to obtain the sequence number from the header. If the number is in proper sequence, it is converted to binary notation and put into the BNIN (sequence number in) field of the header prefix. If the number is not in sequence according to the TSIN 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 (bit 3) in the LEHW (error-halfword) field of the LCB and return is to the next LPS instruction. If the sequence number is too high, the "too high" bit (bit 2) is set in the LEHW field of the LCB, and return is to the next LPS instruction. When either error condition occurs, the sequence-in field in the terminal table entry remains unchanged. If the number is in correct sequence, the expected sequence number from the terminal table is stored in the LBCT field of the LCB. The number in the sequence number field in the terminal table entry is incremented by one for the next message before returning to the next LPS instruction.

External Routines Used: Scan Header (Module IJLQSH)

SKIP CHARACTER SET ROUTINE

Module Name: IJLQSK (Chart SK)

Entry Point: Expansion of the SKIP macro instruction specifying a particular sequence of characters to be skipped generates a BALR to the routine at IJLQSK, 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:

1. a halfword containing the length of the character sequence to be found, and

2. a character constant containing the characters to be skipped.

Function: Advances the scan pointer (contained in register 5) from its current position past all header characters up to and including a specified character sequence. At the completion of the operation, the scan pointer points to the last character in the specified sequence. Incrementing the scan pointer causes all characters bypassed to be ignored during header processing. If the end of the segment is reached before the specified sequence is found, the operation is terminated and the "incomplete header" bit (bit 5) is set in the error halfword for the line.

External Routines Used: Scan Header (Module IJLQSH)

SEQUENCE OUT ROUTINE

Module Name: IJLQSO (Chart SO)

Entry Point: Expansion of the SEQOUT macro instruction generates a BALR to the routine at IJLQSO, 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 one item: the number of character positions for the output sequence number.

Function: The routine links to the Expand Header 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 sequence number out (BNOT) field in the header prefix and converted to decimal form. The sequence number is unpacked and inserted into the new header field. Return is to the next LPS instruction.

External Routine Used: Expand Header (Module IJLQEX)

SOURCE TERMINAL VERIFICATION ROUTINE

Module Name: IJLQSR (Chart SR)

Entry Point: Expansion of the SOURCE macro instruction generates a BALR to the routine at IJLQSR, 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 one item: a halfword containing a source code field-length indicator.

Function: Determines the validity of the source terminal code field of an incoming message header.

Linkage is made to the Scan Header routine which returns in register 2 the address of an area containing the source terminal code found in the header. Verification of the source code is made as follows:

1. If the message originated from a nonswitched 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, the source code is considered valid. If they do not match, the "invalid source code" bit (bit 6) is set in the error-halfword for the line.

2. If the message originated from a switched or autopolled terminal the contents of the source code field are compared with each terminal entry name in the terminal table, in succession, until a match is found. If the source code matches any of the entry names in the terminal table, it is considered valid. If no match is found among any of the entry names in the table, the code is considered invalid and the "invalid source code" bit is set to one. If the source code is valid, the routine places its relative address (within the terminal table) in the source key (BSTO) field of the header prefix in the buffer whose message segment is being operated upon. The relative address is also placed in the LTTD field of the LCB.

In all cases, control returns to the next instruction in the LPS.

External Routines Used: Scan Header (Module IJLQSH)

SKIP-ON-COUNT ROUTINE

Module Name: IJLQST (Chart ST)

Entry Point: The expansion of a SKIP macro instruction specifying a number of characters to be skipped generates a BALR to the routine at IJLQST, 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 one item: a halfword containing the number of characters to be skipped.

Function: Advances the scan pointer (contained in register 5) 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. Incrementing of the scan pointer causes all characters bypassed to be ignored during header processing. If the end of the segment is reached before the SKIP operation is completed, the operation is terminated and the "incomplete header" bit (bit 5) is set in the error-halfword for the line.

External Routines Used: None

TRANSLATE ROUTINE

Module Name: IJLQTR (Chart TR)

Entry Point: Expansion of the TRANS macro instruction generates a BALR to the routine at IJLQTR, 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.

Function: 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 non-negative, the message data is translated using the table specified in the macro. Return is to the next LPS instruction.

External Routines Used: None

TIME STAMP ROUTINE

Module Name: IJLQTS (Chart TS)

Entry Point: Expansion of the TIMESTMP macro instruction generates a BALR to the routine at IJLQTS, 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 one item: a halfword containing, in binary, the length of the time information field to be inserted in the message header.

Function: Obtains the current time in packed decimal format (via a GETIME macro), unpacks it, and inserts a specified portion of the time information in the message header. Prior to inserting the time, the routine links to the Expand Header routine, which "expands" the header by shifting to the left all message characters preceding the location in the header where the time information is to be inserted. The time is inserted in the field thus created. The maximum field size is nine characters in the format bhh.mm.ss, where b = blank, hh = hours, mm = minutes, and ss = seconds. Lesser field sizes have a similar format, truncated from the right.

External Routines Used: Expand Header (Module IJLQEX)

This section contains a detailed description of all the QTAM routines that support a message processing program. These routines include the GET and PUT routines and the routines associated with the macro instructions that examine and modify the status of the telecommunications system. The routines discussed in this section have several characteristics in common:

1. Each routine is linkage-edited into the message processing program due to a V-type address constant generated by the expansion of a QTAM macro instruction.

2. Each routine is entered for execution by linkage generated by the expansion of its associated macro instruction.

3. Each routine saves and restores registers using standard register saving techniques.

All of the routines discussed, with the exception of the Close Message Control routine (CLOSEMC), provide error returns when an invalid condition is detected. Error codes are returned in register 15, right-adjusted. The codes for specific error conditions are contained in the discussions of the individual routines. Zeros are returned in register 15 when no error is detected.

CHANGE LINE ROUTINE

Module Name: IJLQCL (Chart CL)

Entry Points:

1. IJLQCL--from STOPLN macro expansion for a specified line.

2. IJLQCL + 2--from STARTLN macro expansion for a specified line.

3. IJLQCL + 4--from STOPLN macro expansion if ALL is specified.

4. IJLQCL + 6--from STARTLN macro expansion if ALL is specified.

Upon entry, register 0 contains the address of the specified DTF table with the relative line number in the high-order byte. It should be noted that entry to this routine is made only when the DTF Locater routine was able to locate the specified DTF table.

Function: Causes a specified line (or all lines) in the specified line group to be activated or deactivated, depending on whether a STARTLN or STOPLN macro has been issued.

If either of the following errors is detected, no action is taken and return is made to the calling routine with an error code in register 15, right-adjusted:

1. DTF table has not been opened (error code = X'01'); or

2. Invalid relative line number (error code = X'08').

If no errors are detected, the LCB for the specified line (or the first LCB in the group if ALL) is accessed.

If entry is for a STARTLN, and if the line is not already active, the line is activated as follows: The address of the LCB is loaded into the SVC parameter registers 0 and 1, and the line (LCB) is posted to itself via an SVC QPOST. Posting a line to itself is the standard technique used for activating (freeing) the line.

If entry is for a STOPLN on a nonswitched and non-WTTA line, and if the line is not already inactive, the line is stopped as follows: the address of the LCB for the specified line is loaded into SVC parameter register 0 and an SVC QWAIT is issued. This causes the line to be placed in an inactive state. Note that for autopolled lines in read initial the wraparound polling list is broken before issuing the QWAIT.

If the STOPLN is for a switched line or for a WTTA line, a test is made to determine if the line is involved in active transmission (that is, the line has been assigned for a dial operation or is currently connected to a terminal). If it is, a special ECB representing a request for a Halt I/O operation is posted by a QPOST to the LPS queue. LPS control recognizes the element and issues a Halt I/O (SVC 27). If the line is not currently involved in active transmission, the SVC 27 is not issued.

If ALL has been specified in the STOPLN or STARTLN macro, the applicable procedure

is repeated for each line in the line group. Return is to the calling routine.

External Routines Used:

QPOST (SVC 31)

QWAIT (SVC 30)

HALT I/O (SVC 27)

CHANGE POLLING LIST ROUTINE

Module Name: IJLQCP (Chart CP)

Entry Point: Expansion of the CHNGP macro instruction generates linkage to this routine at IJLQCP, using register 15 as the entry register and 14 as the return register. Upon entry, register 0 contains the address of the named DTF table with the specified relative line number in its high-order byte; register 1 contains either:

1. the address of the work area if the old list is to be replaced, or
2. the address of a field containing the "change status" indicator if the list is to be activated or deactivated.

It should be noted that entry is made to this routine only when the DTF Locater routine (Module IJLQFL) was able to locate the specified DTF table.

Function: Replaces the current polling list with a new polling list, or changes the status of the polling list.

If any of the following errors are detected, no movement of data occurs. Return is to the calling routine with an error indicator in register 15, right-adjusted:

1. DTF table has not been opened (error code = X'01')
2. Invalid relative line number (error code = X'08')
3. Length of new polling list is not equal to that of the current list (error code = X'10').

The specified polling list is located as follows: The appropriate LCB is found by multiplying the relative line number by the size of the LCBs in this line group and adding the result to the beginning address of the first LCB in the group. The LPOL field in this LCB contains the address of the desired polling list.

If the polling list is to be replaced, the QMOVE QCB (IJLQIP70 in module IJLQIP) is posted to itself (via an SVC QPOST). This causes the QMOVER routine to be activated for the cross-partition movement of the new polling list. Parameters passed to the QMOVER routine are:

1. register 3--length of the area to be moved;
2. register 4--address of the current polling list being replaced; and
3. register 5--address of the work area containing the new list.

If just the status byte in the current polling list is to be changed, the following parameters are passed to the QMOVER routine:

1. register 3--contains zeros to cause the QMOVER routine to change just the status byte of the list;
2. register 4--address of the current polling list;
3. register 5--address of the area containing the change status indicator.

After the SVC QPOST is satisfied, i.e., after the change has been completed, return is made to the calling routine.

External Routine Used:

QMOVER (IJLQIP80 in Module IJLQIP)

QPOST (SVC 31)

CHANGE TERMINAL TABLE ENTRY ROUTINE

Module Name: IJLQCT (Chart CT)

Entry Point: Expansion of the CHNGT macro instruction generates a BALR to the routine at IJLQCT, using register 15 as the entry register and register 14 as the return register. Upon entry, parameter register 0 contains the address of the work area; the address of an area containing the terminal name is passed in register 1.

Function: Causes a current terminal table entry to be replaced with a new entry contained in the specified work area.

The address of the QTAM Vector Table is obtained from the DOS Communication Region; the address of the terminal table (in the F1 partition) is obtained from the Vector Table. A search of the terminal table is

performed in an attempt to find the entry to be changed. The name specified in the macro is compared with the name in the TTID field of each entry until a match is found or the end of the terminal table is reached.

If the name specified is not found, return is made with an error code of X'20' in register 15. If the name is found, the length of the new entry is compared with that of the old. If the lengths are unequal, an error indicator of X'10' is set in register 15 and return is made to the calling program.

The LCB for the line to which the specified terminal is attached is accessed. If LSTA in the LCB is zero, the line is inactive so the entire old entry may be replaced. The QMOVE QCB (IJLQIP70) is posted to itself (via a QPOST). This causes the QMOVER routine (in module IJLQIP) to be entered for the cross-partition movement of the new entry. Parameters passed to the QMOVER routine are:

1. register 3--length of the area to be moved;
2. register 4--address of the terminal table entry being replaced; and
3. register 5--address of the work area containing the new entry.

If LSTA is not zero (the line is active), the sequence-number-in (TSIN) and sequence-number-out (TSOT) fields in the old entry must not be disturbed. Thus, the change is effected by two separate movements of data (via 2 QPOSTs) so as to skip over these fields.

After the SVC QPOST is completed, return is made to the calling program.

External Routines Used: QMOVER (IJLQIP80 in module IJLQIP)

CHECKPOINT REQUEST ROUTINE

Module Name: IJLQCR (Chart CR)

Entry Point: The expansion of the CKREQ macro instruction generates a BALR to the Checkpoint Request module at IJLQCR. No parameters are passed.

Function: The Checkpoint Request routine issues a QPOST to post a checkpoint request element to the checkpoint queue (located in the Checkpoint module). The routine then issues a QWAIT on the checkpoint request queue (also located in the Checkpoint module). This QWAIT is satisfied when the Checkpoint module, after it has taken the checkpoint, issues a QPOST to post a checkpoint request element to the checkpoint request queue. Upon normal return, the Checkpoint Request routine passes to the message processing program, in register 15, a code of X'00' to indicate that no errors have occurred.

If the user issues a CKREQ macro instruction when checkpointing has not been specified in the message control program, the routine returns an error code of X'04' in register 15. If the user issues a CKREQ macro instruction when the checkpoint interval method of checkpointing has been specified, the routine returns an error code of X'08' in register 15.

External Routines Used:

QPOST (SVC 31)

QWAIT (SVC 30)

COPY LINE ERROR COUNTERS ROUTINE

Module Name: IJLQDC (Chart CC)

Entry Point: Expansion of the COPYC macro instruction generates a BALR to the Copy Line Error Counters routine at IJLQDC. Register 0 contains the address of the user's work area. Register 1 contains the address of the terminal or line name specified. Register 15 is the branch register, and register 14 is the return register. Register 14 also contains the address of an area containing the relative line number if a relative line number is specified by the user. If it is not specified, the area contains zeros.

Function: Copies the line error counters for a line into a user-specified work area.

The routine gets the address of the vector table and locates the QTAM portion of the vector table. From this, the routine finds the location of the terminal table and searches the terminal table for the entry for the terminal specified by the user. If a terminal entry is not found and if an audio line table is specified, its address is obtained from the vector table, and the routine searches for a line table entry corresponding to the line name specified. If no entry is found, the routine returns to the user's program, passing an error code of X'20', right-adjusted, in register 15. If a terminal entry is found, the routine locates the QCB for the entry, and finds

inthe QCB the address of the DTF table for the line group in which the terminal is located. If the user has specified the relative line number, the routine uses it to locate the LCB for the particular line whose counters are to be copied. Otherwise, the routine uses the relative line number specified in the QCB.

If a line entry is found, the routine locates the ALCB for the particular audio line whose counters are to be copied.

The routine copies the threshold counters and the cumulative counters from the LCB or ALCB, adds them together, and places the total in the user-specified work area. The Qmover routine is entered via a QPOST (SVC 31) two times, first to set the cumulative counters to the new total, then to set the threshold counters to 0. The routine then returns to the user's program with a code of X'00', right-adjusted, in register 15.

External Routines Used:

Qmover routine (IJLQIP80 in Module IJLQIP)

QPOST (SVC 31)

COPY TERMINAL TABLE ENTRY ROUTINE

Module Name: IJLQDE (Chart DE)

Entry Point: Expansion of the COPYT macro instruction generates a BALR to the routine at IJLQDE, using register 15 as the entry register and register 14 as the return register. Upon entry, register 0 contains the address of the work area into which the entry is to be copied and register 1 contains the address of a location containing the entry name.

Function: Moves a specified terminal table entry into a specified work area.

The terminal table address (in the F1 partition) is accessed through the DOS communication region and the QTAM Vector Table. A search of the terminal table is made to find the specified entry. If the specified entry is not found, return is made to the calling program with an error code of X'20' in register 15. If the entry is found, the entire entry is moved to the specified work area. The length of the entry to be moved is found in the first byte (TSZE field) of the entry. Return is to the calling program.

External Routines Used: None.

COPY POLLING LIST ROUTINE

Module Name: IJLQDP (Chart DP)

Entry Point: Expansion of the COPYP macro instruction generates linkage to the routine at IJLQDP, using register 15 as the entry register and register 14 as the return register. Register 1 contains the address of a work area into which the polling list is to be copied. The address of the DTF table for the line group is contained in the low-order 3 bytes of register 0. The relative line number of the line associated with the polling list is passed in the high-order byte of register 0. It should be noted that entry is made to this routine only when the DTF Locator routine (module IJLQFL) has been able to locate the specified DTF table.

Function: Moves a copy of the specified polling list into a specified work area.

If either of the following errors in specification is detected, return is made to the calling program with an error indicator in register 15, right-adjusted:

1. DTF table not opened (error = '01'); or
2. invalid relative line number (error code = '08').

The specified polling list is located as follows: The appropriate LCB is found by multiplying the relative line number by the size of the LCBs in this line group and adding the result to the beginning address of the first LCB. The LPOL field in this LCB contains the address of the desired polling list. The polling list is then moved into the specified work area and return is made to the calling routine.

External Routines Used: None.

COPY QUEUE CONTROL BLOCK (QCB) ROUTINE

Module Name: IJLQDQ (Chart DQ)

Entry Point: Expansion of the COPYQ macro instruction generates linkage to this routine at IJLQDQ, using register 15 as the entry register and register 14 as the return register. Upon entry, register 0 contains the address of the specified work area, and register 1 points to a field containing the name (character string) of a DASD terminal or process queue associated with the requested QCB.

Function: Moves a copy of the requested QCB into the specified work area.

The address of the terminal table is accessed through the DOS communication region and the QTAM Vector Table. The specified terminal name is compared, successively, with the name in the TTID field of each terminal table entry until an equal compare or the end of the terminal table is reached. If the specified name is not found in the terminal table, return is made to the calling routine with an error indicator of X'20' in register 15, right-adjusted.

The address of the requested QCB is obtained from the TQAD field in the specified terminal table entry. The QCB is then moved into the specified work area (the QCB for all DASD process or destination queues is 32 bytes in length). Return is made to the calling routine.

External Routines Used: None.

DTF LOCATOR ROUTINE

Module Name: IJLQFL (Chart FL)

Entry Point: Expansion of the STOPARU, STARTARU, STOPLN, STARTLN, COPYP, and CHNGP macro instructions generate linkage to this routine at IJLQFL, using register 15 as the entry register and 14 as the return address register. Upon entry, the address of the area containing the specified DTF name is supplied in register 1.

Function: Finds the address of the DTF table corresponding to the DTF name (character string) specified in the macro. The logic module associated with the macro requires the DTF table address for proper execution.

In operation, the routine obtains the address of the last QTAM DTF opened from the VECLDTF field of the QTAM Vector Table. Using this as a beginning address, it searches all QTAM DTF tables in an attempt to locate the one corresponding to the name specified. The name contained in the DNME field of the last DTF table is compared with the specified name. If they are unequal, the address of the previous DTF in the chain as created by the QTAM Open routine, is obtained from the DFLK field in the DTF and the search continues. When an equal compare is made, the address of the DTF table located is placed into register 1 and return is made to the next instruction in the macro expansion.

If the specified DTF table is not found, or if it is not a DTF table for a line group file, return is made with an error code of X'20' in register 15. The next instruction in the macro expansion tests for an error return in register 15. If no error is indicated, linkage is made to the appropriate logic module. If an error is indicated, return is made to the calling routine.

External Routines Used: None

GET AUDIO MESSAGE ROUTINE

Module Name: IJLQGA (Chart GA)

Entry Points: This routine is entered at the following points:

1. IJLQGA+8 via linkage generated by the GET macro expansion. Upon entry, register 0 contains the address of the work area, and register 1 contains the address of the DTF table for an MS-process queue.

2. IJLQGA20+14 from the QTAM Control Program. This entry is a special section of the GET Audio Message routine which runs in supervisor state to avoid cross-partition storage protection in turning on the GET bit in an ALCB (in the foreground-1 partition). Upon entry into the GET audio subtask, the address of the high-priority GET audio element, which contains the address of the passed ALCB, is in register 1.

Function: Provides the next sequential audio message in a specified work area. The routine first removes the MS-process queue from the Ready Queue through a QWAIT (SVC 30) on the Ready Queue itself and then makes a test to determine whether an ALCB is in the process queue:

1. If not, and if a user-written routine has been specified to handle this situation (SYNCAD keyword operand), linkage is made to this routine.

   Exception: If CLOSEMC has been issued (master receive switch off), control transfers to the routine specified in the FLUSHAD keyword operand.

2. If yes (or if no user-written routine was specified), the next ALCB is requested by issuing a QWAIT (SVC 30) on the process queue. (If there is no ALCB in the queue, control does not

return until another ALCB arrives.) The address of the ALCB is returned to register 1. From the ALCB, the line table entry enables the address of the source line name to be obtained. Then, this line name is moved into the area specified in the LINAD keyword operand.

If the work area is not large enough to contain the message received in the input buffer of the ALCB, and if a user-written routine has been specified in the OVAD keyword operand, linkage is made to this routine. On return from this routine, or if it has not been specified, the length of the input messages is reduced to the size of the work area. The remainder of the message, which cannot be placed in the work area, is ignored.

The length of the message is stored into the GET/PUT prefix (first two bytes) of the work area and, according to this length, the message data is moved from the ALCB into the work area.

The address of the ALCB is stored into the high priority GET audio element which is posted (SVC 31) to the special GET audio queue. On activation of the GET audio subtask, the GET bit is set on in the ALCB, and the control returns to the calling section of the GET Audio Message routine.

Finally, return is made to the next instruction in the message processing program.

External Routines Used:

QPOST (SVC 31)

QWAIT (SVC 30)

GET NONAUDIO OR AUDIO MESSAGE ROUTINE

Module Name: IJLQGB (Charts GB and GC)

Entry Points: This routine is entered at the following points:

1. IJLQGB+8 via linkage generated by the GET macro expansion. Upon entry, register 0 contains the address of the work area, and register 1 contains the address of the DTF table for an MS process queue.

2. IJLQGB20+14 from the QTAM Control Program. This entry is an audio section that runs in supervisor state to avoid cross-partition storage protection in turning on the GET bit in an ALCB (in the foreground-one partition). Upon entry into the GET audio subtask, the address of the high-priority GET audio element, which contains the address of the passed ALCB, is in register 1.

Function: Provides the next sequential message in a specified work area. This work area is made up of two consecutive parts: the first part used to receive nonaudio messages, the second to receive audio messages. The lengths of these parts are specified in the SOWA and SOWARU keyword operands, respectively.

The routine first removes the MS process queue from the Ready Queue through a QWAIT (SVC 30) on the Ready Queue itself and then tests to determine whether an element (buffer or ALCB) is in the process queue:

1. If not, and if a user-written routine has been specified to handle this situation (SYNCAD keyword operand), linkage is made to this routine.

   Exception: If CLOSEMC has been issued (master receive switch off), control is transferred to the routine specified via the FLUSHAD keyword operand.

2. If yes (or if no user-written routine was specified), a test is made to determine whether any message data remains in a buffer obtained from a previous GET.

   a. If not, and if the last processed element is not an ALCB, the previous buffer is released to the Return Buffer queue through an SVC 31 (QPOST), and the next element is requested through an SVC 30 (QWAIT) on the process queue. (If no element is in the queue, control does not return until another element arrives.) The address of the buffer is returned to register 1. At this point, the header disk pointer is saved for use by the next checkpoint.

   b. If yes, the QPOST/QWAIT sequence is bypassed, because the message data remaining in the buffer must be handled before releasing the buffer. The count and starting address of the remaining data are computed before continuing.

The routine then tests to determine whether the next element to be processed is a buffer or an ALCB.

1. If it is a buffer, if this buffer contains the header, and if the source terminal displacement (BSTC in buffer prefix) is not zero, the source terminal name is moved into the area specified in the TRMAD keyword operand. (The source displacement is zero if the message is from another message processing program, or if the message arrived over a switched or autopolled line for which the SOURCE macro instruction was not included in the incoming LPS.) The user's switch specified in the SWITCH keyword operand is turned off.

   The message data is moved from the buffer into the first part of the work area using the size of the data, or the size of the remainder of the work area (first part) if smaller. After this has been executed, a test for end-of-message is made. If end-of-message is not found, a waiting loop is entered to release (QPOST) the buffer and to wait (QWAIT) for the next. If found, the type and length of the message data moved are stored into the GET/PUT prefix.

   If the first part of the work area is not large enough to contain the entire message, and if a user-written routine has not been specified to handle this situation (OVAD keyword operand), the count of the message data remaining in the buffer is computed and stored before control returns to the calling routine. If a user-written routine has been specified, linkage is made to this routine. In either case, the remainder of the message is moved into the first part of the work area when the next GET is issued.

2. If it is an ALCB, the address of the source line name is computed, and the line name is moved into the area specified in the LINAD keyword operand. The user's switch specified in the SWITCH keyword operand is turned on.

   If the second part of the work area is not large enough to contain the message received in the input buffer of the ALCB, and if a user-written routine has been specified in the OVAD keyword operand, linkage is made to this routine. On return from this routine, or if it has not been specified, the length of the input message is reduced to the size of the second part of the work area. The remainder of the message, which cannot be placed in the work area, is ignored.

   The length of the message is stored in the GET/PUT prefix (first two bytes) of the work area (second part) and, according to this length, the message data is moved from the ALCB into the work area (second part).

   The address of the ALCB is stored in the high-priority GET audio element, which is posted (SVC 31) to the special GET audio queue. On activation of the GET audio subtask, the GET bit is set on in the ALCB, and control returns to the calling section of this module.

When a complete message is in the work area return is made to the next instruction in the message processing program.

External Routines Used:

QPOST (SVC 31)

QWAIT (SVC 30)

GET RECORD OR AUDIO MESSAGE ROUTINE

Module Name: IJLQGC (Charts GD and GE)

Entry Points: This routine is entered at the following points:

1. IJLQGC+8 via linkage generated by the GET macro expansion. Upon entry, register 0 contains the address of the work area, and register 1 contains the address of the DTF table for an MS process queue.

2. IJLQGC20+14 from the QTAM Control Program. This entry is an audio section that runs in supervisor state to avoid cross-partition storage protection in turning on the GET bit in an ALCB (in the foreground-1 partition). Upon entry into the GET audio subtask, the address of the high-priority GET audio element, which contains the address of the passed ALCB, is in register 1.

Function: Provides the next sequential message record or audio message in a specified work area. This work area is made up of two consecutive parts: the first part used to receive message records, the second to receive audio messages. The length of these parts is specified in the SOWA and SOWARU keyword operands, respectively.

The routine first removes the MS-process queue from the Ready Queue through a QWAIT (SVC 30) on the Ready Queue itself and then tests to determine whether an element (buffer or ALCB) is in the process queue:

1. If not, and if a user-written routine has been specified to handle this situation (SYNCAD keyword operand), linkage is made to this routine.

   Exception: If CLOSEMC has been issued to initiate a closedown procedure (master receive switch off), control is transferred to the routine specified via the FLUSHAD keyword operand.

2. If yes (or if no user-written routine specified), a test is made to determine whether part of a record remains in a buffer obtained from a previous GET.

   a. If not, and if the last processed element is not an ALCB, the previous buffer is released to the Return Buffer queue through an SVC 31 (QPOST), and the next element is requested through an SVC 30 (QWAIT) on the process queue. (If no element is in the queue, control does not return until another element arrives.) The address of the buffer is returned to register 1. At this point, the header disk pointer is saved for use by the next checkpoint.

   b. If yes, the QPOST/QWAIT sequence is bypassed, because the remaining data in the buffer must be handled before releasing the buffer. The count and starting address of the remaining data are computed before continuing.

The routine then tests to determine whether the next element to be processed is a buffer or an ALCB.

1. If it is a buffer, if this buffer contains the header, and if the source terminal displacement (BSTC in buffer prefix) is not zero, the source terminal name is moved into the area specified in the TRMAD keyword operand. (The source displacement is zero if the message is from another message processing program, or if the message arrived over a switched or autopolled line for which the SOURCE macro instruction was not included in the incoming LPS.) The user's switch specified in the SWITCH keyword operand is turned off.

   The data in the buffer is moved into the first part of the work area, one byte at a time, until either:

   a. A new-line (NL) or EOB character is encountered in the buffer, or

   b. The first part of the work area is filled. If a NL or EOB character is found, it is moved into the work area along with any consecutive NLs or EOBs that immediately follow it. The type of message unit and the record length are placed in the GET/PUT prefix.

   If the first part of the work area is not large enough to contain the entire record, and if a user-written routine has not been specified to handle this situation (OVAD keyword operand), the count of the data not transferred is computed and stored for the next GET. If a user-written routine has been specified, linkage is made to this routine. In either case, the remaining data will be supplied on the next GET.

   If neither of the preceding conditions specified is encountered in the current buffer, an SVC 30 (QWAIT) is issued to request another buffer, and the above procedure is repeated.

2. If it is an ALCB, the address of the source line name is computed, and the line name is moved into the area specified in the LINAD keyword operand. The user's switch specified in the SWITCH keyword operand is turned on.

   If the second part of the work area is not large enough to contain the message received in the input buffer of the ALCB, and if a user-written routine has been specified in the OVAD keyword operand, linkage is made to this routine. On return from this routine, or if it has not been specified, the length of the input message is reduced to the size of the second part of the work area. The remainder of the message, which cannot be placed in the work area, is ignored.

   The length of the message is stored into the GET/PUT prefix (first two bytes) of the work area (second part) and, according to this length, the message data is moved from the ALCB into the work area (second part).

   The address of the ALCB is stored into the high-priority GET audio

element which is posted (SVC 31) to the special GET audio queue. On activation of the GET audio subtask, the GET bit is set on the ALCB, and control returns to the calling section of this module.

When a record or an audio message is in the work area, return is made to the next instruction in the message processing program.

External Routines Used:

QPOST (SVC 31)

QWAIT (SVC 30)

GET SEGMENT OR AUDIO MESSAGE ROUTINE

Module Name: IJLQGD (Charts GF and GG)

Entry Points: This routine is entered at the following points:

1. IJLQGD+8 via linkage generated by the GET macro expansion. Upon entry, register 0 contains the address of the work area, and register 1 contains the address of the DTF table for an MSprocess queue.

2. IJLQGD20+14 from the QTAM Control Program. This entry is an audio section that runs in supervisor state to avoid cross-partition storage protection in turning on the GET bit in an ALCB (in the foreground-1 partition). Upon entry into the GET audio subtask, the address of the high-priority GET audio element, which contains the address of the passed ALCB, is in register 1.

Function: Provides the next sequential message segment or audio message in a specified work area. This work area is made up of two consecutive parts: the first part used to receive message segments, the second to receive audio messages. The lengths of these parts are specified in the SOWA and SOWARU keyword operands, respectively.

The routine first removes the MS process queue from the Ready Queue through a QWAIT (SVC 30) on the Ready Queue itself and then tests to determine whether an element (buffer or ALCB) is in the process queue.

1. If not, and if a user-written routine has been specified to handle this situation (SYNCAD keyword operand), linkage is made to this routine.

   Exception: If CLOSEMC has been issued to initiate a closedown procedure (master receive switch off), control is transferred to the routine specified via the FLUSHAD keyword operand.

2. If yes (or if no user-written routine specified), a test is made to determine whether part of a segment remains in a buffer obtained from a previous GET.

   a. If not, and if the last processed element is not an ALCB, the previous buffer is released to the Return Buffer queue through an SVC 31 (QPOST), and the next element is requested through an SVC 30 (QWAIT) on the process queue. (If no element in the queue, control does not return until another element arrives.) The address of the buffer is returned to register 1. At this point the header disk pointer is saved for use by the next checkpoint.

   b. If yes, the QPOST/QWAIT sequence is bypassed, because the remaining data in the buffer must be handled before releasing the buffer. The count and starting address of the remaining data are computed before continuing.

The routine makes a test to determine whether the next element to be processed is a buffer or an ALCB.

1. If it is a buffer, if this buffer contains the header, and if the source terminal displacement (BSTO in buffer prefix) is not zero, the source terminal name is moved into the area specified in the TRMAD keyword operand. (The source displacement is zero if the message is from another message processing program, or if the message arrived over a switched or autopolled line for which the SOURCE macro instruction was not included in the incoming LPS.) The user's switch specified in the SWITCH keyword operand is turned off.

   The message segment is moved from the buffer into the first part of the work area using the segment size, or the size of the work area (first part) if smaller. If the entire data has been moved into the work area, the type and length of the data moved is stored into the GET/PUT prefix.

   If the first part of the work area is not large enough to contain the

entire segment, and if a user-written routine has not been specified to handle this situation (OVAD keyword operand), the count of the data not transferred is computed and stored for the next GET. If a user-written routine has been specified, linkage is made to this routine. In either case, the remaining data will be supplied on the next GET.

2. If it is an ALCB, the address of the source line name is computed, and the line name is moved into the area specified in the LINAD keyword operand. The user's switch specified in the SWITCH keyword operand is turned on.

   If the second part of the work area is not large enough to contain the message received in the input buffer of the ALCB, and if a user-written routine has been specified in the OVAD keyword operand, linkage is made to this routine. On return from this routine, or if it has not been specified, the length of the input message is reduced to the size of the second part of the work area. The remainder of the message, which cannot be placed in the work area, is ignored.

   The length of the message is stored into the GET/PUT prefix (first two bytes) of the work area (second part) and, according to this length, the message data is moved from the ALCB into the work area (second part).

   The address of the ALCB is stored into the high-priority GET audio element which is posted (SVC 31) to the special GET audio queue. On activation of the GET audio subtask, the GET bit is set on in the ALCB, and control is returned to the calling section of this module.

When a segment or an audio message is in the work area, return is made to the next instruction in the message processing program.

External Routines Used:

QPOST (SVC 31)

QWAIT (SVC 30)

GET MESSAGE ROUTINE

Module Name: IJLQGM (Charts GM and GN)

Entry Point: This routine is entered at IJLQGM+8 via linkage generated by expansion of the GET macro. Upon entry, register 0 contains the address of the work area, and register 1 contains the address of the DTF table for an MS process queue.

Function: Provides the next sequential message in a specified work area.

The routine first makes a test to determine if there is a message in the process queue. If there is none and if a user-written routine has been specified to handle this situation (SYNCAD keyword operand), linkage is made to the specified routine. Exception: If no message is in the queue and CLOSEMC has been issued (the master receive switch is off), exit is made to the routine specified via the FLUSHAD keyword operand.

If there is a message in the queue, or if no message is in the queue but no user-written routine is specified, a test is made to determine if there is message data remaining in the buffer from a previous GET. If there is no message data, the previous buffer is released to the Return Buffer queue through an SVC 31 (QPOST), and a request for the next buffer is made by a QWAIT (SCV 30) on the process queue. (If there is no message in the queue, control does not return until another message arrives.) The address of the buffer is returned in register 1.

At this point, the header disk pointer is saved for use by the next checkpoint.

If there was data remaining in the previous buffer, the QPOST/QWAIT sequence is bypassed because the remaining data in that buffer must be handled before releasing it; the count and starting address of the remaining data are computed before continuing.

If the current buffer contains the header and if the source terminal displacement (BSTO in buffer prefix) is not zero, the source terminal name is moved into the area specified in the TRMAD keyword operand. (The source displacement is zero if the message is from another processing program or if the message arrived over a switched or autopolled line for which the SOURCE macro was not included in the incoming LPS.)

The message data is moved from the buffer into the work area using the size of the data or size of remainder of work

area,if smaller. If all data in the buffer has been moved into the work area, a test is made for end of message. If it is not end of message, a loop is made to release this buffer and to QWAIT for the next buffer. If it is the end of message, the type and length of data moved are stored in the GET/PUT prefix and return is made to the next instruction in the processing program.

If the work area was not large enough to contain the entire message and if a user-written routine is not specified to handle this case (OVAD keyword operand), a count of the data remaining in the buffer is computed and stored before returning to the calling routine. If a user-written routine is specified, linkage is made to the specified routine. In either case, the remainder of the message is moved to the work area when the next GET is issued.

External Routines Used:

QPOST (SVC 31)

QWAIT (SVC 30)

GET RECORD ROUTINE

Module Name: IJLQGR (Charts GQ and GR)

Entry Point: This routine is entered at IJLQGR+8 via linkage generated by expansion of the GET macro. Upon entry, register 0 contains the address of the work area, and register 1 contains the address of the DTF table for an MS process queue.

Function: Provides the next sequential message record in a specified work area.

The routine first makes a test to determine if there is a message in the process queue. If there is none and if a user-written routine has been specified to handle the situation (SYNCAD keyword operand), linkage is made to the specified routine. Exception: If no message is in the queue and CLOSEMC has been issued to initiate a closedown procedure (the master switch is off), exit is made to the routine specified via the FLUSHAD keyword operand.

If there is a message in the queue or if no message is in the queue but no user-written routine is specified, a test is made to determine if part of a record remains in the buffer from a previous GET. If not, the previous buffer is released to the Return Buffer queue through an SVC 31 (QPOST), and a request for the next buffer is made by a QWAIT (SVC 30) on the process queue. (If there is no message in the queue, control does not return until another message arrives.) The address of the buffer is returned in register 1.

At this point, the header disk pointer is saved for use by the next checkpoint.

If there was data remaining in the previous buffer, the QPOST/QWAIT sequence is bypassed because the remaining data in that buffer must be handled before releasing it. The count and starting address of the remaining data are computed before continuing.

If the current buffer contains the header and if the source terminal displacement (BSTO in buffer prefix) is not zero, the source terminal name is moved into the area specified in the TRMAD keyword operand. (The source displacement is zero if the message is from another processing program or if the message arrived over a switched or autopolled line for which the SOURCE macro was not included in the incoming LPS.)

The data in the buffer is moved into the work area one byte at a time until either:

1. a new line (NL) or EOB character is encountered in the buffer; or

2. the work area has been filled.

If a NL or EOB character is found, it is moved to the work area along with any consecutive NLs or EOBs that immediately follow it. The type of message unit and record length are placed into the GET/PUT prefix, and return is made to the calling routine.

If the work area is not large enough for the entire record, the count of the data not transferred is computed and stored for the next GET. If a user-written routine has been specified to handle this situation, linkage is made to that routine. If no such routine is specified, return is to the calling routine. In either case, the remaining data will be supplied on the next GET.

If neither of the preceding conditions specified is encountered in the current buffer, a QWAIT is issued requesting another buffer and the procedure is repeated.

External Routines Used:

QWAIT (SVC 30)

QPOST (SVC 31)

GET SEGMENT ROUTINE

Module Name: IJLQGS (Chart GS)

Entry Point: This routine is entered at IJLQGS+8 via linkage generated by expansion of the GET macro. Upon entry, register 0 contains the address of the work area, and register 1 contains the address of the DTF table for an MS-Process Queue.

Function: Provides the next sequential message segment in a specified work area.

The routine first makes a test to determine if there is a message in the process queue. If there is none and if a user-written routine has been specified to handle this situation (SYNCAD keyword operand), linkage is made to the specified routine. Exception: If no message is in the queue and CLOSEMC has been issued to start a closedown procedure (master receive switch is off) exit is made to the routine specified by the FLUSHAD keyword operand.

If there is a message in the queue or if no message is in the queue but no user-written routine is specified, a test is made to determine if part of the segment remains in the buffer from a previous GET. If not, the previous buffer is released to the Return Buffer queue through an SVC 31 (QPOST), and a request for the next buffer is made by issuing a QWAIT (SVC 30) on the process queue. (If there is no message in the queue, control does not return until another message arrives.) The address of the buffer is returned in register 1.

At this point, the header disk pointer is saved for use by the next checkpoint.

If there was data remaining in the previous buffer, the QPOST/QWAIT sequence is bypassed because the remaining data in that buffer must be handled before releasing it. The count and starting address of the remaining data are computed before continuing.

If the current buffer contains the header and if the source terminal displacement (BSTO in buffer prefix) is not zero, the source terminal name is moved into the area specified in the TRMAD keyword operand. (The source displacement is zero if the message is from another processing program or if the message arrived over a switched or autopolled line for which the SOURCE macro was not included in the incoming LPS.) The message segment is moved from the buffer into the work area using the size of the segment or the size of the work area, if smaller. If all data has been moved into the work area, the type and length of data moved is stored in the GET/PUT prefix and return is made to the next instruction in the processing program.

If the work area was not large enough to contain the entire segment and if a user-written routine is not specified to handle this case (OVAD keyword operand), a count of the data remaining in the buffer is computed and stored before returning to the calling routine. If a user-written routine is specified, linkage is made to the specified routine before returning. In either case, the remainder of the segment is moved to the work area when the next GET is issued.

External Routines Used:

QPOST (SVC 31)

QWAIT (SVC 30)

PUT AUDIO MESSAGE ROUTINE

Module Name: IJLQPA (Chart PA)

Entry points: This routine is entered at the following points:

1. IJLQPA+12 via linkage generated by the PUT macro expansion. Upon entry, register 0 contains the address of the work area, and register 1 contains the address of the DTF table for an Audio Output queue.

2. IJLQPA20+14 from the QTAM Control Program. This entry is a special section running in the supervisor state to avoid cross-partition storage protection in moving the audio output message from the work area (in a message processing program) into the ALCB (in the foreground-1 partition). Upon entry in the PUT audio subtask, the address of the high-priority PUT audio element, which contains the address of the passed ALCB, is in register 1.

Function: Moves an audio output message from a specified work area into an ALCB, and causes this ALCB to be posted on the ARU-Send queue for emission of the audio answer.

The audio line table is searched to locate the line entry for the destination specified in the LINAD keyword operand. If the audio line name specified by the user is invalid, an error code of X'04' is returned to the calling routine in register 15, right-adjusted.

The ALCB is searched from the line entry, and the GET bit is tested to determine if this ALCB corresponds to a line waiting for an audio answer. If the GET bit is off, the audio destination is considered invalid because no switching is allowed on audio lines, and an error code of X'02' is returned to the calling routine in register 15, right-adjusted.

The length of the output message is checked to make sure it is greater than the size of the GET/PUT prefix (2 bytes), and the address of the ALCB is stored into the high-priority PUT audio element which is posted (SVC 31) to the special PUT audio queue.

On activation of the PUT audio subtask, the following operations are performed in the passed ALCB: the audio output message is moved from the work area into the address chain buffer, the GET bit is set off, and the priority code is set to X'FD' before return is made to the calling section of the PUT Audio Message routine.

The ALCB is posted (SVC 31) to the ARU-Send queue, and control is transferred to the next instruction in the message processing program. When a 7772 line is waiting for a DCV buffer a warning code of X'08' is returned in register 15 but the audio answer will be sent as soon as a DCV buffer is available.

External Routines Used:

QPOST (SVC 31)

PUT MESSAGE ROUTINE

Module Name: IJLQPM (Chart PM)

Entry Points:

1. IJLQPM+12 from the PUT macro expansion. Upon entry, register 0 contains the address of a work area, and register 1 contains the address of the DTF table.

2. IJLQPM+60 from the Buffer-BRB routine. This entry is to a special section of the PUT message routine which runs in supervisor state to circumvent cross-partition storage protection in moving the message from the work area (in a message processing program) to a buffer (in the F1 partition). Upon entry, register 1 contains the address of the available buffer.

Function: Moves a complete message from a specified work area to a buffer (more than one buffer is used, if necessary) and causes the buffer(s) to be routed to the specified destination (or process) queue on the direct-access storage device.

A request for a buffer into which to move the message data is initiated by posting (via an SVC 31) a BRB (contained in the first 4 words of the DTF table) to the Active BRB queue, A QWAIT (SVC 30) is then issued to wait for a buffer to become available. Normal return from this QWAIT does not occur until the following procedure is completed:

When the buffer assigned for the PUT operation is passed to the Buffer-BRB routine (in module IJLQIP), that routine links to a special section of the PUT Message routine (entry point 2 above) that executes under the storage protection key of the Supervisor. If the buffer is to contain the message header, the following procedures are performed:

1. The address of a special LCB contained in the DTF table is placed into the BLCB field of the buffer prefix.

2. The number of idle characters (X'17'), if any, at the beginning of the work area is computed and placed in the scan pointer field (BSPT) of the prefix for later use in the Send Group of the LPS.

3. The sequence-in and sequence-out fields (BNIN and BNOT) in the prefix are set to zero.

4. The terminal table is searched to locate the entry for the destination specified via the TRMAD keyword operand. The relative address of the located entry is placed into the destination key (BDTO field) in the prefix. The address of the QCB for the destination is obtained from the entry and placed into the LCB.

5. If the user has specified a priority for the message (4th byte of GET/PUT prefix is nonzero), the priority code is placed in the LCB.

The message data is then moved into the buffer using the length specified in the GET/PUT prefix, or the size of the buffer (minus the prefix) if the buffer cannot contain the entire message. If the latter occurs, the residual count of the message data in the work area is computed and saved, and the work area pointer is updated and saved. The count of the data moved into the buffer is placed into the BSSZ field of the prefix, and the source key

(BSTO field) in the prefix is set to zero to indicate a processing program as the source.

If the entire message (or the last portion of the message) is in this buffer, the end-of-message bit is set in the BSTA field of the prefix. Otherwise, the appropriate indicator is set in BSTA depending on whether the buffer contains a header or text segment. Exit is then made to the Interim LPS routine, which causes the QWAIT to be satisfied. Control returns to the PUT message routine in problem program state to the instruction following the QWAIT.

The buffer returned by the QWAIT has already been filled with the message data as described. This buffer is now posted (SVC 31) to the specified destination queue. A QWAIT (SVC 30) is then issued to remove the BRB from the top of the Ready Queue. If the entire message has been handled, return is made to the calling routine. If further message data remains in the work area, the routine branches to request another buffer and the procedure described above is repeated. This continues until the entire message has been transferred and posted.

Note: If an invalid destination is specified, no transfer of message data occurs. An error code of X'20' is returned to the calling routine in register 15, right-adjusted.

External Routines Used:

QPOST (SVC 31)

QWAIT (SVC 30)

PUT RECORD ROUTINE

Module Name: IJQPR (Charts PQ and PR)

Entry Points:

1. IJLQPR+12 from the PUT Macro expansion. Upon entry, register 0 contains the address of the work area and register 1 contains the address of the DTF table.

2. IJLQPR+60 from the Buffer BRB routine (see PUT Message routine description).

Function: Moves the message record from the specified work area into a buffer (or more than one buffer if necessary). When a buffer is filled, it is posted to the specified destination queue.

This routine issues a QPOST/QWAIT sequence to request a new buffer in the same manner described in the PUT Message routine. The QWAIT is not satisfied until the following procedure is completed:

When the buffer assigned for the PUT operation is passed to the Buffer-BRB routine (in module IJLQIP), that routine branches to a special section of the PUT Record routine (entry point 2 above) that executes under the protection key of the Supervisor.

Preparation for later releasing of the buffer obtained (called the "new" buffer in this discussion) by the QWAIT is performed by placing the address of the Available Buffer queue into the BQCB field of the buffer prefix. The BRB contained in the DTF table for the MS Destination queue is linked into the top of the Ready Queue in preparation for requesting the next buffer. A test is made to determine if there is a buffer (called the "previous" buffer) that was not completely filled with message data by the previous PUT.

Note: This routine does not post a buffer to the specified destination queue until the buffer is filled or until an end of message is indicated.

The example of a previous buffer is discussed later; however, it should be noted here that there will always be a previous buffer except when the preceding PUT was for the last record in the message (end of message).

The procedures performed when there is no previous buffer are now described. Because the last record PUT was the end of that message, the current record must be a header segment; therefore, the following initialization procedures are performed for the "new" buffer:

1. The address of the LCB is placed into the buffer prefix.

2. The number of idle characters (X'17'), if any, at the beginning of the specified work area is computed and placed in the scan pointer field (BSPT) of the prefix for later use in the Send Group of the LPS.

3. The sequence-in and sequence-out fields in the prefix are set to zero.

4. The source key (BSTO field) in the prefix is set to zero to indicate a processing program as the source.

5. The terminal table is searched to locate the entry for the specified destination. The relative address of

the located entry is placed into the destination key (BDTO field) of the prefix. The address of the QCB for the destination is obtained from the entry and placed into the LCB.

6. If the user has specified a priority for the message (4th byte of GET/PUT prefix is nonzero), the priority code is placed in the LCB.

The message data is then moved into the new buffer using the record length specified in the GET/PUT prefix, or the size of the buffer (minus the prefix) if the buffer cannot contain the entire record.

If the record size is used for the move, the third byte in the GET/PUT prefix is tested to determine if this is the end of the message (EOM). If EOM is detected, the EOM bit is set in the buffer prefix; the new buffer is linked into the top of the Ready Queue; and exit is made to the Qdispatch subroutine.

Qdispatch causes the new buffer to be routed to the specified destination and effects return to the QWAIT issued in the problem program portion of the PUT Record routine, which then returns to the calling routine. If this was not the end of the message, the new buffer address is saved (it is not posted) and return is effected to the calling routine. On the next PUT issued, the current new buffer then becomes the previous buffer.

If the buffer was not large enough to contain the entire record, the residual count of the message data in the work area is saved, and the work area pointer is updated and saved. The message-filled buffer is then linked into the top of the Ready Queue pushing the BRB previously placed at the top of the Ready Queue to second in line. Exit is made to Qdispatch, which routes the buffer to its destination. The BRB now becomes the first item on the Ready Queue. Therefore, another buffer is assigned for this PUT operation, and the supervisor section of the PUT Record routine is again entered.

The procedure is repeated, but only those functions required for a buffer containing a text segment are performed. This process continues until the entire record has been transferred. When this occurs, control is effected to the calling routine.

The procedures performed upon entry into this routine when there is a previous, unfilled buffer from the preceding PUT are now described. The new buffer (obtained by the QWAIT) is released by linking it in to the top of the Ready Queue (the address of the Available Buffer QCB was previously inserted in the ECB of the buffer). This pushes the BRB down to second position on the Ready Queue.

A test is then made to determine if the record currently in the user's work area contains a header. It should not because the previous record did not have the end of message indicator. However, if the current record does begin a new message, end-of-message must be forced for the record in the previous buffer. The EOM bit is set in the buffer prefix, and the field containing the address of the previous buffer is cleared to zero. The previous buffer is linked into the top of the Ready Queue, and exit is made to the Qdispatch subroutine. Upon exit, the order of items on the Ready Queue is:

1. the previous buffer,
2. the new buffer, and
3. the BRB.

This causes the following action: The previous buffer containing the last record of the preceding message is posted to the destination queue; the new buffer is released to the Available Buffer queue; and another new buffer is assigned for the current record which is the first record of a message. Control then returns to the supervisory section of the PUT Record routine at IJLQPR+60. The current record is now handled in the same manner described for the case where there was no previous buffer.

If the current record in the work area does not contain a header, the routine branches to move the record into the remaining area of the previous buffer. The current record is handled in the manner already described for handling a record that contains text only.

External Routines Used:

QPOST (SVC 31)

QWAIT (SVC 30)

PUT SEGMENT ROUTINE

Module Name: IJLQPS (Chart PS)

Entry Points:

1. IJLQPS+12 from the PUT macro expansion. Upon entry, register 0 contains the address of a work area, and register 1 contains the address of the DTF table.

2. IJLQPS+60 from the Buffer-BRB routine. This entry is to a special section of the PUT Segment routine which runs in supervisor state to circumvent cross-partition storage protection in moving the message from the work area (in a message processing program) to a buffer (in the F1 partition). Upon entry, register 1 contains the address of the available buffer.

Function: Moves a message segment from a specified work area to a buffer (more than one buffer is used, if necessary) and causes the buffer(s) to be routed to the specified destination (or process) queue on the direct-access storage device.

A request for a buffer into which to move the segment is initiated by posting (via an SVC 31) a BRB (contained in the first 4 words of the DTF table) to the Active BRB queue. A QWAIT (SVC 30) is then issued to wait for a buffer to become available. Normal return from this QWAIT does not occur until the following procedure is completed:

When the buffer assigned for the PUT operation is passed to the Buffer-BRB routine (in module IJLQIP), that routine links to a special section of the PUT Segment routine (entry point 2 above) that executes under the storage protection key of the Supervisor. If the buffer is to contain the message header, the following procedures are performed:

1. The address of a special LCB contained in the DTF table is placed into the BLCB field of the buffer prefix.

2. The number of idle characters (X'17'), if any, at the beginning of the work area is computed and placed in the scan pointer field (BSPT) of the prefix for later use in the Send Group of the LPS.

3. The sequence-in and sequence-out fields (BNIN and BNOT) in the prefix are set to zero.

4. The terminal table is searched to locate the entry for the destination specified via the TRMAD keyword operand. The relative address of the located entry is placed into the destination key (BDTO field) in the prefix. The address of the QCB for the destination is obtained from the entry and placed into the LCB.

5. If the user has specified a priority for the message (4th byte of GET/PUT prefix is nonzero), the priority code is placed in the LCB.

The message data is then moved into the buffer using the length specified in the GET/PUT prefix, or the size of the buffer (minus the prefix) if the buffer cannot contain the entire segment. If the latter occurs, the residual count of the message data in the work area is computed and saved, and the work area pointer is updated and saved. The count of the data moved into the buffer is placed into the BSSZ field of the prefix, and the source key (BSTO field) in the prefix is set to zero to indicate a processing program as the source.

The appropriate indicator is set in the BSTA field of the prefix depending on whether the buffer contains a header or text segment. Exit is then made to the Interim LPS routine which causes the QWAIT to be satisfied. Control returns to the PUT Segment routine in problem program state to the instruction following the QWAIT.

The buffer returned by the QWAIT has already been filled with the message data as described. This buffer is now posted (SVC 31) to the specified destination queue. A QWAIT (SVC 30) is then issued to remove the BRB from the top of the Ready Queue.

If the entire segment has been handled, return is made to the calling routine. If further message data remains in the work area, the routine branches to request another buffer and the procedure described above is repeated. This continues until the entire segment has been transferred and posted.

Note: If an invalid destination is specified, no transfer of message data occurs. An error code of X'20' is returned to the calling routine in register 15, right-adjusted.

External Routine Used:

QPOST (SVC 31)

QWAIT (SVC 30)

CLOSE MESSAGE CONTROL ROUTINE

Module Name: IJLQQT (Chart QT)

Entry Point: Expansion of the CLOSEMC macro instruction generates linkage to this routine at IJLQQT, using register 15 as the entry register and 14 as the return register. No parameters are required.

Function: This routine stops all active lines in the system, sets the "master-receive-switch" so that no more incoming traffic is accepted, and then resumes outgoing message traffic.

The routine turns off the master-receive switch by posting (via an SVC QPOST) the QMOVE QCB to itself. This causes the QMOVER subtask to be activated to perform the actual manipulation of the switch (in the F1 partition). The address of the master-receive-switch, as obtained from the VECMRSW field in the QTAM Vector Table, is passed to the QMOVER routine in register 4. The master-receive-switch is located at IJLQMRSW in the Implementation module (IJLQIP).

At open time, the QTAM Open routines (Modules IJLQ01, IJLQ02, IJLQ07, and IJLQ08) chained together all of the DTF tables for QTAM files opened in the message control program and placed the address of the last such DTF into the VECLDTF field of the QTAM Vector Table. When this address is used as a starting point, the Close Message Control routine is able to access each DTF table for a line group file; and through each such DTF table, all of the LCBs (or ALCBs) for the lines in the line group.

Each LCB generated in the system is accessed, in turn, and a test is made to determine if the line associated with the LCB is currently active (LSTA = X'00' indicates inactive). If the line is inactive, it is bypassed. If it is active, the following action is taken:

1. A STOPLN macro instruction is issued for the line; then

2. A STARTLN macro instruction is issued for the line.

The significance of the preceding procedure is: The STOPLN causes all operations on the designated line to cease after the current transmission is completed (see Change Line routine description). The following STARTLN causes the same line to be reactivated for output only operations. Each time the Receive Scheduler routine (in module IJLQIP) is entered, it examines the master-receive-switch. If it is off (as is the case at this point), input operations are not initiated on the line.

Each ALCB generated in the system is accessed, in turn, and a test is made to determine if the audio line associated with the ALCB is currently enabled (LSTS=X'01' indicates the enable state). If the line is not enabled, the next ALCB is accessed. If the line is enabled, the ALCB address is stored in a high-priority audio element. This element posted (via an SVC QPOST) to the LPS queue requests a HALT I/O operation (via an SVC 27) on the line associated with the passed ALCB. Each time the Audio Line Appendage routine is entered (in module IJLQAA), it examines the master-receive switch. If it is off (as is the case at this point), no enabling is initiated on the line.

The preceding procedure is repeated for each active line in the system. The net effect, upon return to the calling routine, is that all line input into the system has been stopped, while line output operations continue as normal. In addition, the QTAM Ready Queue is scanned to search for ALCB elements to be processed in a message processing program. Any such ALCBs found are removed from the Ready Queue and queued by priority in the element chain of their corresponding MS-process queues.

External Routines Used:

QPOST (SVC 31)

Change Line (module IJLQCL)

DTF File Locator (module IJLQFL)

RETRIEVE DASD ROUTINE

Module Name: IJLQRD (Chart RD)

Entry Point: Expansion of the RETRIEVE macro instruction generates linkage to this routine at IJLQRD. Upon entry, register 1 contains the DASD relative record number of the message segment to be retrieved, and register 0 contains the address of the specified work area.

Function: The Retrieve-DASD routine transfers a message segment previously placed in a DASD destination or DASD process queue to a specified work area.

If an invalid relative record number is specified, no action is taken, and return is made to the calling routine with an error code of X'02' in register 15, right-adjusted. (QTAM maintains the last relative record number used on the disk in the VECDRNN field of the QTAM Vector Table).

If the operation is to proceed, the routine builds a BRB-QCB (in the work area specified by the user) as follows: The passed relative record number is placed in the BRB (RNSA field) as the address of the next segment to be read from the disk, and the status byte (RSTA field) in the BRB is set to 9 to indicate a read-from-disk

operation. The address of the STCB for the Queue Insert by Priority subroutine is placed into the QSTC field of the QCB. The BRB/QCB is then posted (via an SVC QPOST) to the Disk I/O QCB. Return from the SVC indicates that the read-from-disk operation has been initiated. An SVC QWAIT is then issued to wait for completion of the disk operation. This QWAIT SVC returns the main storage address of the retrieved segment in register 1. Yet another SVC QWAIT is required to ensure that the BRB/QCB is taken off the Ready Queue before it is overlaid in the work area.

The size of the segment retrieved is obtained from the BSSZ field in its buffer prefix and is used to move the segment into the specified work area. Return is to the calling routine. Note that the buffer prefix, minus the first 8 bytes, is included in the data moved to the work area.

External Routines Used:

QPOST (SVC 31)

QWAIT (SVC 30)

RETRIEVE BY SEQUENCE NUMBER ROUTINE

Module Name: IJLQRS (Chart RS)

Entry Point: Expansion of the RETRIEVE macro generates linkage to this routine at:

1. IJLQRS if input sequence number is specified, or
2. IJLQRS+2 if output sequence number is specified.

On entry, register 1 contains the sequence number of the message header to be retrieved and register 0 contains the address of the work area into which the header segment is to be placed. The first 8 bytes of the work area contain the name of the terminal (or processing program) from whose DASD queue the message header is to be retrieved.

Function: This routine causes the header segment of the requested message to be retrieved from the specified DASD queue and placed into a user-provided work area.

If an invalid terminal name or an invalid sequence number is specified, no action is taken. An error code (X'20' for invalid terminal name or X'40' for invalid sequence number) is set in register 15, right-adjusted, and return is made to the calling routine.

This routine searches the terminal table to locate the terminal table entry for the specified terminal. The address of the QCB for the DASD queue is obtained from the TQAD field of the terminal table entry located. In the QBAK field of the QCB is the relative record number of the header segment of the last message placed on the DASD queue. Using this as a starting point, this header segment and each preceding header segment in the DASD queue are retrieved, in order, until the header segment having the desired sequence number is found.

Implementation of the preceding general procedure is as follows: This routine generates linkage to the Retrieve DASD routine (Module IJLQRD) which actually causes the message segment to be read from the DASD queue and placed into the work area. Parameters passed to the Retrieve DASD routine are: register 0 contains the address of the work area and register 1 contains the relative record number of the segment to be retrieved.

Upon return from the Retrieve DASD routine, the specified sequence number is compared with the sequence number in the header prefix (BNIN field if input sequence number was specified or BNOT field if output sequence number was specified). If the sequence numbers are equal, return is made to the calling routine with a X'00' in register 15. If they are unequal, the relative record number of the previous header segment in the DASD queue is obtained from the BMHD field of the buffer prefix, and the preceding procedure is repeated.

External Routines Used: Retrieve DASD (Module IJLQRD)

RELEASE MESSAGE ROUTINE

Module Name: IJLQRM (Chart RM)

Entry Point: Expansion of the RELEASEM macro instruction generates linkage to this routine at IJLQRM, using register 15 as the entry register and 14 as the return register. Upon entry, parameter register 1 points to an area containing the name (character string) of the terminal to which intercepted messages are to be released.

Function: Turns off the "intercept" bit in the terminal table entry for the specified terminal and restores, if necessary, the address of the next message to be read from the disk.

A search is made of the terminal table to locate the terminal table entry for the specified terminal. If the entry is not in the table, return is made to the calling routine with an error code of X'20' in register 15. If the specified entry is found, the intercept bit (in the TSTA field) is tested. If it is not on, return is made to the calling routine with an error code of X'04' in register 15, because messages to the terminal have not been previously intercepted (via an INTERCPT macro in the LPS).

The displacement (from the beginning of the entry) to the intercept subfield in the terminal table entry is obtained from the location where the Intercept Message routine placed it. If the disk address in the QNRA field of the QCB for the DASD queue is greater than the disk address in the intercept subfield, a priority message has arrived on the DASD queue. Therefore, the address in the intercept subfield must replace the address in the QNRA field or the intercepted messages will not be sent. This is accomplished by the QMOVER routine via an SVC QPOST.

After the header address of the first intercepted message has been restored in the QCB, if necessary, this routine issues another SVC QPOST which causes the QMOVER routine to turn on the "send" bit and turn off the intercept bit in the terminal table entry. Return is made to the calling routine.

External Routines Used:

QMOVER (IJLQIP80 in Module IJLQIP)

QPOST (SVC 31)

START/STOP AUDIO LINE

Module Name: IJLQSS (Chart SS)

Entry Points:

1. IJLQSS, from STOPARU macro expansion for a specified line.
2. IJLQSS+2, from STARTARU macro expansion for a specified line.
3. IJLQSS+4, from STOPARU macro expansion if ALL is specified.
4. IJLQSS+6, from STARTARU macro expansion if ALL is specified.

Upon entry, register 0 contains the address of the specified DTF table with the relative line number in the high-order byte, unless ALL has been specified. Noted that entry to this routine is made only when the DTF Locator routine is able to locate the specified DTF table. Therefore, this routine is also entered from the QTAM Control Program in the supervisor mode at IJLQSS20+14 to prepare a stop or start line operation.

Function: Causes a specified line (or all lines) in the specified line group to be activated or deactivated depending on whether a STARTARU or STOPARU macro instruction has been issued.

If either of the following errors is detected, no action is taken, and return is made to the calling routine with an error code in register 15, right-adjusted:

1. DTF table not opened (error code X'01'); or
2. Invalid relative line number (error code X'08').

If no errors are detected, the ALCB for the specified line (or the first ALCB in the line group if ALL has been specified) is accessed.

The address of the ALCB is stored in a high-priority audio element, which is posted (via an SVC QPOST) to a start/stop audio queue. The dispatching of this queue by the QTAM Control Program activates a truncated subtask included in the start/stop audio module. If the entry is from a STARTARU macro expansion, and if the line is not already active, the start request flag is set on in the ALCB (LSSF=X'40' indicates start request), the stop flag is set off, and the EXCP flag is set on (LSSF=X'01' indicates EXCP request). If the entry is from a STOPARU macro expansion, the stop flag is set on in the ALCB (LSTS=X'20' indicates stop request), and if the line group is a 7770 line group operating in information mode, in the channel program the command chaining flag is suppressed from the DISABLE CCW. On return to the problem program, if the EXCP flag is on in the ALCB, this ALCB is posted (via an SVC QPOST) to the LPS queue requesting a START I/O operation.

If ALL has been specified in the STARTARU or STOPARU macro instruction, the applicable procedure is repeated for each line in the line group, before return is made to the calling routine.

External Routines Used:

QPOST (SVC 31)

Service facilities provided by QTAM include Checkpoint/Restart, On-line Terminal Testing, Operator Control, Error Recovery Procedures, and Operator Awareness.

The Operator Control facility is logically associated with the OPCTL macro instruction in the Line Procedure Specification, and is discussed in the section Message Control Program (LPS) Routines.

The Error Recovery Procedures are physically a part of the Line Appendage module, and are discussed in the section Line Input and Output.

CHECKPOINT/RESTART

The checkpoint facility causes records to be written on a checkpoint records file on a disk. The records contain information on the status of the queues and the telecommunications network. The checkpoint records may be written either:

1. At fixed intervals of time, specified by the user, or
2. Whenever each message processing partition in the system has issued a CKREQ macro instruction.

The information from the checkpoint records may be used to perform a Restart operation after a system failure. This is done by performing an initial program load of the system, and loading the message control program into the same location it occupied when the failure occurred. If a restart is to be performed, this is done by Phase 2 of the Open Checkpoint Records File routine. The information saved at the most recent checkpoint is moved to the proper areas, overlaying the initial values. Thus the system is reinitialized to the status it had at the time that checkpoint was taken.

If CKREQ macro instructions are issued in the message processing programs to perform checkpointing, the Checkpoint Request routine is entered. This routine links to the Checkpoint routine to take the checkpoint. The Checkpoint Request routine is discussed in the section Message Processing Program Routines.

CHECKPOINT ROUTINE

Module Name: IJLQCK (Chart CK)

Entry Points: The routine is entered either at CHKPTRTN or at CKSTCB+6.

Entry is at CHKPTRTN when:

- The ENDREADY macro instruction is executed.
- A timer interruption occurs, or a CKREQ macro instruction is issued in a message processing program.
- The checkpoint element reaches the top of the Disk I/O queue.
- A disk write operation to write a checkpoint record on the disk has been completed.

Entry is at CKSTCB+6 when:

- Any but the last checkpoint request element is posted to the checkpoint request queue.

The action taken for each type of entry is discussed under separate headings.

Function: This routine causes checkpoint records to be written on the checkpoint records file either at specified intervals or when a CKREQ macro instruction has been issued in every message processing partition.

Entry from ENDREADY: The expansion of the ENDREADY macro instruction generates a QPOST (SVC 31) to post the checkpoint element to the checkpoint queue. The subtask activated links to the Time Delay routine in the Implementation module, passing the checkpoint interval in register 6 and the address of the checkpoint element in register 4.

Entry via a Timer Interruption or a CKREQ Macro Instruction: The subtask activated tests to determine if entry was made because of a CKREQ macro instruction. If so, it tests to determine if all message processing partitions have issued CKREQ. If not, the subtask branches to Qdispatch.

If all message processing partitions have issued CKREQ or if the entry was made because of a timer interruption, the checkpoint data is stored into the

checkpoint save area. The format of the checkpoint save area is shown in Figure 5. Data saved for the checkpoint record is:

1. From the terminal table entries, all fields except the first word. DSECT names for fields saved are: IJLQTSIN, IJLQTSOT, IJLQTSTA and IJLQTTID, plus all user-specified optional fields.

2. From the destination QCBs (DSECT names: IJLQQNRA, IJLQQMCT, IJLQQNWA, and IJLQQBK). From the process QCBs (DSECT names: IJLQQNRA, IJLQQMCT, IJLQQNWA, IJLQQBAK, and IJLQQFST).

3. From the LCBs (DSECT names: IJLQLSTA, IJLQLHDR, IJLQLNAS, and IJLQLTTD).

4. From the polling lists, all fields except the first byte. (There are no DSECT names for fields in a polling list.)

The element chain of the Disk I/O queue is then examined. If other elements appear on the Disk I/O queue, the routine places the checkpoint element on the Disk I/O queue below the other elements and branches to Qdispatch. When the checkpoint element reaches the top of the Disk I/O queue element chain, this routine is reentered to prepare to write the checkpoint record.

If no element is on the Disk I/O queue element chain, the routine proceeds immediately to prepare to write the checkpoint element.

Entry from the Checkpoint Element Reaching the Top of the Disk I/O Queue: The subtask activated sets up the channel program to write the checkpoint record on the DASD. It then branches to the QTAM Post subroutine to post a dummy ECB to the LPS queue. The LPS Control routine recognizes the ECB, and issues an EXCP to write the checkpoint record.

Entry from the Completion of a Disk Write Operation: The routine tests to determine if all the checkpoint data has been written. If it has not, the disk addresses are set to the next record, and the routine again branches to the QTAM Post subroutine to post the dummy ECB to the LPS queue. This process continues until all the checkpoint data is written on the disk. The routine then prepares the checkpoint control record and causes it to be written on the disk in the same way.

When return is made from writing the control record, the routine tests to determine what method of checkpointing is being used. If the checkpoint interval method is being used, the routine sets up the necessary parameters, and links to the Time Delay routine to set the next checkpoint interval.

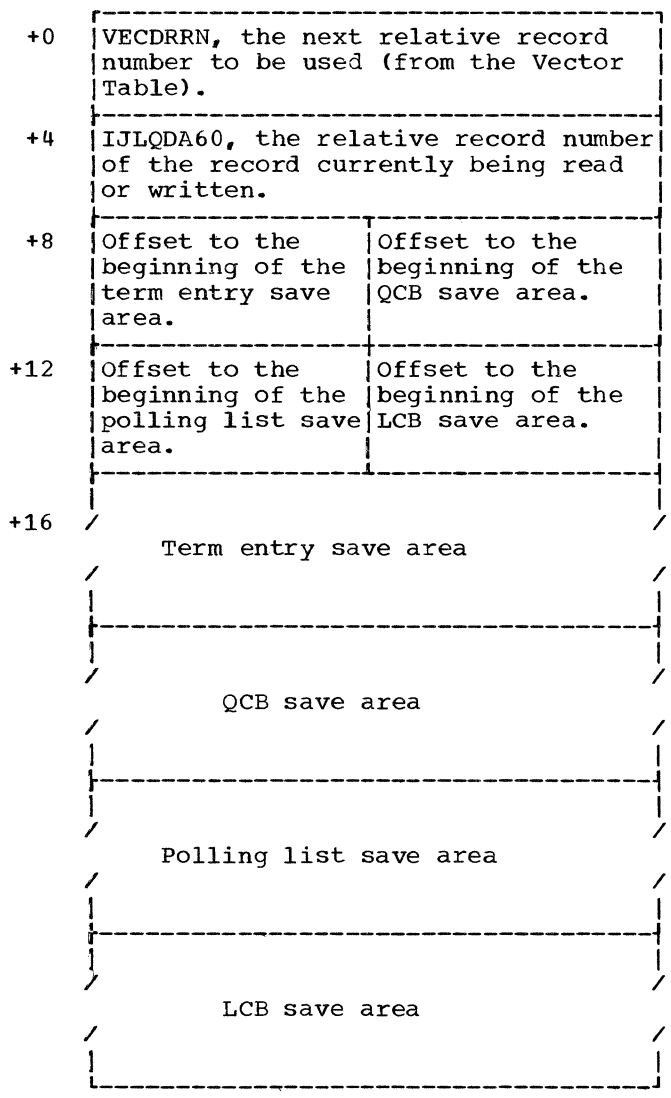


Figure 5. Format of the Checkpoint Record

If the checkpoint request method is being used, the routine inserts the checkpoint element at the top of the Ready Queue, and links to the QTAM Post subroutine to post a check request element to the check request queue. The routine is reentered at CKSTCB+6 to post subsequent check request elements.

Entry from Posting a Check Request Element: The routine tests to determine if there is more than one check request element yet to be posted. If so, the routine again inserts the checkpoint element at the top of the Ready Queue before linking to the QTAM Post subroutine to post another check request element to the check request queue. This is done until only one check request element remains to be posted. At this

time, the routine does not insert the checkpoint element at the top of the Ready Queue, but effectively surrenders control in posting the final check request element to the check request queue.

External Routines Used:

Time Delay routine in Implementation module

Disk I/O routine in Disk I/O module (IJLQDA)

LPS Control (IJLQIP20 in Module IJLQIP)

ON-LINE TERMINAL TESTING

The On-Line Terminal Test facility provides tests that can be used by the terminal operator as a startup procedure, and by the IBM customer engineer for terminal checkout and diagnosis of terminal failure. A test is requested by entering a test request message at a terminal.

The Terminal Test Recognition routine, which is entered via the LPSTART macro instruction, is the only resident terminal test routine. The Terminal Test Header Analysis routine and the five terminal test modules for tests on particular terminal types are transient routines. They are called into the logical transient area from the Core Image Library when needed. These routines are discussed in QTAM Transient Routines.

TERMINAL TEST RECOGNITION ROUTINE

Module Name: IJLQTT (Chart TT)

Entry Point: Expansion of the LPSTART macro instruction generates a BALR to the Terminal Test Recognition routine. Entry is at IJLQTT. Register 1 contains the address of a parameter list generated by the LPSTART macro. Register 6 contains the address of the buffer to be processed. Register 14 is the return register.

Function: Recognizes terminal test activity, calls terminal test transient routines, performs cleanup, and stops and restarts line operation.

The routine checks the incoming messages for the test activation code (99999). If the code is not present, return is made to the LPSTART macro expansion, and normal LPS processing resumes.

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) sets test identification flags in the buffer prefix containing the test request and posts it to the LPS queue. Subsequent buffers are posted to the available buffer queue. (Terminal Tests 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 are 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 the buffer is posted to the LPS queue. The 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 are then freed, and QTAM line operation is 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.

External Routines Used:

EXCP (SVC 2)

Recall (IJLQIP22 in module IJLQIP)

Cleanup (IJLQIP23 in module IJLQIP)

Defer Entry (QTAM nucleus)

Terminal Test Header Analysis (module IJLQTM)

OPERATOR AWARENESS ROUTINE

Module Name: IJLQOA (Chart OA)

Entry Points: If operator awareness is specified in the TERMTBL macro instruction (OPCTL=chars) or if the OPCTL macro instruction is specified, the LPS macro instructions POSTSEND and POSTRCV enter the Operator Awareness routine at IJLQOA instead of the Buffer Recall/Cleanup routine (IJLQIP22).

Function: Sends I/O error messages or threshold line error counter messages to the operator control terminal.

Upon entry from POSTSEND or POSTRCV, a test is made to determine if ERP has specified that an operator control error message is to be sent. If not, a branch is made to RETURN.

The routine tests to determine if a threshold message is to be sent. If yes, the routine goes to THRESH to set up the threshold message. If not, the routine tests to determine if an I/O error message is to be sent. If yes, the routine goes to IOERR to set up the I/O error message. If neither message is indicated, the routine branches to RETURN.

At THRESH, the routine first branches to a subroutine within the IJLQOA module called RECALL, which returns a buffer to be used to contain the error message. The threshold error message is constructed in the buffer. The flag that indicated that a threshold error message was to be written is turned off.

The routine then enters SENDMSG, where construction of the message is completed. The buffer prefix is set to indicate that the message is to be routed to the primary operator control terminal. A test is made to determine if an I/O error message is also required. If it is, the routine goes to IOERR to set up the I/O error message. Otherwise, the routine branches to RETURN.

At IOERR, the routine first branches to RECALL to obtain a buffer to contain the error message. The routine gets the index into the I/O error message table, turns off the flag that indicated that an I/O error message was to be sent, and constructs the I/O error message in the buffer. The routine then branches to SENDMSG. Note that the routine then must branch to RETURN from SENDMSG because the I/O error message flag has been turned off.

At RETURN, a test is made to determine if a permanent error has occurred on the line and if INTREL has been indicated for this line. If so, the routine branches to the Buffer Recall/Cleanup routine at IJLQIP22, which writes the error message on the proper DASD destination queue, gets another buffer, and returns it to this routine. The buffer prefix of this buffer is modified to indicate that it is to be posted to the INTREL queue.

The routine exits to the Buffer Recall/Cleanup routine at IJLQIP23. If INTREL was not indicated, the error message is written on the disk at this time. If INTREL was indicated, the new buffer accessed is posted to the INTREL queue.

RECALL Subroutine: A test is made to determine if the buffer just processed by LPS is to be placed on disk. If not, then it can be used to contain the error message and return is made to the caller.

If the buffer is to be written on disk, the subroutine links to the Buffer Recall/Cleanup routine at IJLQIP22. Here the buffer is written on the disk, another buffer is acquired, a header is read into it and it is returned to the RECALL subroutine. RECALL returns it to the caller.

External Routines Used: Buffer Recall/Cleanup routine in the Implementation module (IJLQIP22, IJLQIP23).

OBR/SDR ROUTINE

Module Name: IJLQOB (Chart OB)

This module is included in the message control program by means of a V-type address constant generated when the OBRSDR operand is specified in the TERMTBL macro instruction. The module is a collection of four distinct routines necessary for the optional OBR/SDR function (see the DOS QTAM Message Control Program, Form C30-5004, section on OBR/SDR Error Recording).

1. SDR Step-Counters Routine.

   This routine is entered by a BAL from IJLQEP, IJLQLO, or IJLQTA. The routine assigns sets of in-core SDR counters to lines or terminals and increments these counters according to the following rules:

   - If the SDR first-time switch (X'08' bit in the third-from-last byte of the LCB) is off, control is returned immediately to the calling routine. Otherwise, this switch is turned off and the routine continues.

- If the IJLQLTTD field in the LCB is nonzero (indicating that the terminal ID is known) and the X'10' bit is on in the third-from-last byte of the LCB (indicating that counts should be by terminal, if possible), the routine prepares to count by terminal. Otherwise the routine prepares to count by line.

- The routine checks the SDR counter offset field (IJLQTSDR in the TERM entry if counting by terminal; the last halfword of the LCB if counting by line). If the field is 0, no SDR counter set has been assigned yet; the next available SDR counter set is then assigned to the line or terminal by storing its offset in the SDR counter offset field.

- The "total transmissions" counter in the assigned SDR counter set is incremented by 1.

- If the halfword following the BAL instruction contains a nonzero value, this indicates that an error should also be counted. The appropriate error counter is incremented by 1.

If the maximum possible value has been reached for either counter (255 for total transmissions, 15 for error counters), the X'04' bit is turned on in the third-from-last byte of the LCB. This flag will later cause this SDR counter set to be added to the corresponding set of counters on disk.

Control is then returned to the calling routine. If not enough SDR counter sets are provided, some requests for assignment of counter sets must be ignored. Error counts for lines or terminals without assigned SDR counter sets will be lost.

2. OBR/SDR Test Routine.

This routine is entered from the LPS control routine. The routine checks the third-from-last byte in the LCB. If the X'02' bit is on, the routine builds an OBR record from data in the LCB, passes the record to the system OBR routines via an SVC 44, waits for completion, and resets the bit to zero. If the X'04' bit is on, the routine builds an SDR record containing the counters that are about to overflow, passes the record to the system SDR routines via an SVC 43, and waits for completion. The bit and the SDR counters are then reset to zero. Control is returned to the LPS control routine.

3. Check SDR First-Time Routine.

This routine is entered by a BAL from IJLQEP. The routine turns on the X'08' bit in the third-from-last byte of the LCB if the error retry count in the LCB is zero. This bit is a first-time switch to prevent the SDT step-counters routine from counting retries.

4. OBR/SDR Polltest Routine.

This routine is entered by a BAL from IJLQLA. The routine determines the source terminal when an initial interrupt occurs on an autopoll line if the index byte from the read response or the residual count from a poll command is available. The terminal table offset of the source terminal is stored in the IJLQLTTD field of the LCB.

External Routines Used:

SVC 43 DOS System SDR routine.
SVC 44 DOS System OBR routine.

There are five groups of QTAM routines that are dynamically fetched into the logical transient area for execution:

1. QTAM Open routines (six phases).
2. QTAM Close routines (three phases).
3. QTAM Message Writer routines (four phases).
4. On-Line Terminal Test routines (six phases).
5. QTAM Cancel Routine (two phases).

This section contains a detailed description of these routines.

QTAM OPEN MONITOR/OPEN DASD MESSAGE QUEUES FILE ROUTINE

Module Name: IJLQO1 (Chart 01)

Entry Point: Whenever a QTAM DTF table is opened at execution time, the system Open Monitor recognizes the file as a QTAM file, reads the extent information if it is a DASD file, and fetches into the logical transient area the QTAM Open Monitor routine, which is cataloged on the Core Image Library under the name $$BOQO01. Control is then passed to this routine at IJLQO1. Upon entry, register 2 contains the address of the DTF table being opened.

Return is made to the QTAM Open Monitor routine when the other Open modules have completed processing.

Function: Performs preliminary initialization functions, analyzes the type of QTAM file being opened, and either fetches the proper OPEN module or performs the proper opening functions

This routine tests to determine which type of QTAM file is being opened. The eight types of QTAM files are:

1. DASD Messages Queues (Message Control).
2. DASD Checkpoint Records (Message Control).
3. DASD 7772 DCV Vocabulary (Message Control).
4. Nonaudio Line Group (Message Control).
5. Audio Line Group (Message Control).
6. MS Process Queue (Message Processing).
7. MS Destination Queue (Message Processing).
8. Audio Output Queue (Message Processing).

If the Audio or Nonaudio Line Group, MS-Process Queues, MS-Destination Queues, or Audio Output Queues files are being opened, the routine issues a FETCH to bring the proper OPEN module into the transient area.

The routine tests to determine if the DLBL information for the file has been initially processed. If it has not, the routine fetches the system Open Monitor ($$BOPEN), which performs this function.

If the DASD Message Queues file is being opened, the routine initializes each partition's communication region with the address of the QTAM vector table. The address of the DTFQT is placed in the vector table and in the chain of opened DTFs in the vector table. The PIB table of the message control program partition is modified to allow exits to QTAM appendages from the supervisor. The relative record number of the first available record on the file is placed in the QTAM vector table. If operator awareness is included, the Implementation module (IJLQIP) is modified to cause the LPS macro instructions POSTSEND and POSTRCV to enter the Operator Awareness routine before entering the Buffer Recall/Cleanup routine. The number of QTAM buffers per track for the device type is computed and stored in the DTF table.

If the Checkpoint Records file is being opened, a test is made to determine if the DASD Message Queues file was opened immediately previous to this file. If it was not, an error message is sent and the job is cancelled.

The routine then fetches the Sequential Disk Open Input routine ($$BOSDI1), which validates and file protects the extent and returns control. The routine then posts the limits of the extent information to the DTF table.

If the DASD 7772 DCV Vocabulary file or the Checkpoint Records file is being opened, the routine exits to Phase 1 of the Open Checkpoint/Restart routine (IJLQ03) or to the Open IBM 7772 DCV Vocabulary File routine (IJLQ07), via a FETCH.

If the DASD Message Queues file is being opened, the extent is checked for preformatting. If it is not preformatted correctly, an error message is sent and the job is cancelled. The routine then tests to determine if there are more extents in the file. If so, the Sequential Disk Open Input routine is fetched to validate the next extent. When all extents are validated, the routine sets the open bit in the DTF and tests to determine if there are additional QTAM files to be opened by this OPEN macro instruction. If so, control returns to the beginning of the routine. If there are no more, control is passed to the system Open Monitor via a FETCH.

If this routine has passed control to other QTAM Open modules, it is reentered and it passes control back to the system Open Monitor.

External Routines Used:

FETCH (SVC 2)

OPEN NONAUDIO LINE GROUP/MS QUEUES ROUTINE

Module Name: IJLQ02 (Chart 02)

Entry Point: This routine is brought into the transient area and control passed to it through a FETCH issued by the QTAM Open Monitor routine. The QTAM Open Line Group/MS Queues routine is cataloged on the Core Image Library under the name $$BCQQ02. Register 2 contains the address of the DTF table being opened.

Function: Performs initialization functions necessary to prepare line group and main storage queues files for operation.

The routine first tests to determine which file is being opened, and branches to the proper subroutine.

Open Line Group Subroutine: As each line group is opened, the address of its DTF table is added to the chain of opened DTF tables that was started by placing the address of the DTF for the DASD Message Queues file in the Vector Table. If the DTF table is for a remote line group, a channel program is built for each line consisting of the necessary commands for making the line ready. For nonswitched lines, an Enable command is prepared. If the TCU type is an IBM 2702, the appropriate SAD command is prepared. For all other remote lines, a No Op command is prepared. After each channel program is built, a test is made to determine if a Restart is being performed. If it is not, the channel program is executed by issuing EXCP, followed by a WAIT to await completion of the channel program.

If a Restart is being performed, the subroutine adds an additional command to the channel program (Read Skip for IBM terminals, Break for non-IBM terminals). The channel program is executed by issuing EXCP, followed by a WAIT.

Upon completion of the channel program, for a remote line the subroutine issues an SVC 31 (QPOST) to post the LCB to itself. This causes the line to be scheduled for a receiving operation.

If the DTF table is for an IBM 2260 Local line group, the terminals are prepared for transmission as follows. First, the ACLOC field in the DTF table is adjusted to the next highest fullword boundary, which is the CCB offset in the device access area. The LCB for the line group is posted to itself, via a QPOST, to initialize the Receive Scheduler. If the DTF is not being opened for receive operations, no further initialization is performed, and the routine exits to open the next DTF.

For receive operations, each terminal in the polling list is prepared for receiving by turning on the 'receive allowed' flag in the terminal table entry and issuing a NOP channel program. When the channel program completes, the IBM 2260 Local Appendage causes the CCB to remain on the DOS channel queue awaiting Attention Interrupts (read requests). When all IBM 2260s have been enqueued, the routine exits to open the next DTF.

The subroutine now tests if all LCBs in this line group have been handled. If not, the next LCB address is acquired and initialized. If all LCBs have been handled, the next DTF address is acquired and tested to determine if it is a line group file. If it is not, return is made to the QTAM Open Monitor routine through a FETCH.

Open MS Queues Subroutine: The address of the DTF table for the file is linked into the chain from the VECPDQLK field in the Vector Table. When an MS Process Queue is being opened, the address of the QCB for the corresponding DASD Process Queue is placed into the DTF table. The address of the LCB for the Process Queue (formed in

the DTF table for the MS Process Queue) is placed into the QSTL field of the QCB for the DASD Process Queue. This action establishes the interface between the message control and message processing programs and allows messages to be transferred to a processing program in response to a GET. Control is returned to the QTAM Open Monitor routine through a FETCH.

External Routines Used:

EXCP (SVC0)

FETCH (SVC2)

OPEN CHECKPOINT/RESTART ROUTINE (PHASE 1)

Module Name: IJLQO3 (Chart 03)

Entry Point: Entry into this routine is from the File Protect subroutine, which is called by the QTAM Open Monitor. Phase 1 of the QTAM Open Checkpoint/Restart routine is cataloged on the Core Image Library under the name $$BOQO03. Register 2 contains the address of the DTF table being opened.

Function: Performs initialization functions necessary to open the checkpoint records file.

The routine reads the checkpoint control record and tests it for correct formatting. If it is incorrectly formatted, an error message is sent and the job is cancelled. If it is formatted correctly, the first byte of the record is tested for X'00'. If it is not X'00', a Restart must be performed. The routine sets a bit in the checkpoint flag byte in the QTAM vector table to indicate that a Restart is being performed, reads the proper checkpoint record, and issues a FETCH to call in Phase 2 of the Open Checkpoint/Restart routine.

If the first byte of the control record is equal to X'00', the routine goes through a scanning loop to compute the size of the work area needed to contain the checkpoint data. It compares the result with the size of the work area specified in the SOWA keyword operand and assembled into the DTF table. If the work area assembled is too small, an error message is sent and the job is cancelled.

The routine then formats the disk extent (as far as necessary) with dummy records. Control is returned to the QTAM Open Monitor through a FETCH.

OPEN CHECKPOINT/RESTART ROUTINE (PHASE 2)

Module Name: IJLQO4 (Chart 04)

Entry Point: Phase 2 of the Open Checkpoint/Restart routine is called by Phase 1 of the Open Checkpoint/Restart routine when it determines that a Restart is to be performed. The QTAM Restart routine is cataloged in the Core Image Library under the name $$BOQO04 and is entered for execution at IJLQO4. Register 2 contains the address of the DTF table.

Function: Reestablishes the nonaudio queues and the telecommunications network to the status it had at the time the last checkpoint was taken.

The routine goes through a scanning loop to compute the size of the work area needed to contain the checkpoint data. It compares the result with the size of the work area assembled into the DTF table. If the work area assembled is too small, an error message is sent, and the job is cancelled.

The data in the checkpoint record is then used to reinitialize the queues, polling lists, terminal table, and LCBs. In initializing the LCBs, the routine tests to determine if the line is in Send or Receive status. If in Send status, the Send Scheduler STCB is placed ahead of the Receive Scheduler STCB in the LCB. If in Receive status, the header of the message being received at the time the checkpoint was taken is read in from the disk, canceled, and written back on the disk. Return is made to the QTAM Open Monitor through a FETCH.

External Routines Used:

Message Writer, Phase 3 (IJLQW3)

EXCP (SVC 0)

WAIT (SVC 7)

OPEN IBM 7772 DCV VOCABULARY FILE ROUTINE

Module Name: IJLQO7 (Chart 07)

Entry Point: Entry into this routine is from the File Protect subroutine, which is called by the QTAM Open Monitor. The QTAM Open 7772 DCV Vocabualry File routine is cataloged on the Core Image Library under the name $$BOQO07. Register 2 contains the address of the DTF table being opened.

Function: Performs initialization functions necessary to use the 7772 DCV Vocabulary file.

When the 7772 DCV Vocabulary DTF is the first opened DTF table, the routine places the address of the QTAM Vector Table into the DOS communication region. This is necessary to allow the cross-partition communication required in QTAM. The address of this DTF table begins the chain of opened DTF tables in the Vector Table (VECLDTF field). When the 7772 DCV Vocabulary DTF is not the first-opened DTF table, the routine checks that the previously opened file is the DASD Message Queues file, and adds the address of this DTF table to the chain of opened DTF tables.

Each DCV buffer in each buffer pool associated with the 7772 line group is searched. The disk assignment unit order of the 7772 DCV Vocabulary file is stored in the CCB of each DCV buffer.

If a word table is provided, the routine searches the address of the first word-table entry, and the DCV word is read from the 7772 DCV Vocabulary file into the corresponding WORD macro expansion. This procedure is repeated for each DCV word required by the user in main storage.

Control is returned to the QTAM Open Monitor through a FETCH.

External Routines Used:

FETCH (SVC 2)

EXCP (SVC 0)

OPEN AUDIO LINE GROUP/OUTPUT QUEUE FILES ROUTINE

Module Name: IJLQO8 (Chart 08)

Entry Point: This routine is brought into the transient area and control passed to it through a FETCH issued by the QTAM Open Monitor routine. The QTAM Open Audio Line Group/Output Queues routine is cataloged on the Core Image Library under the name $$BOQO08. Register 2 contains the address of the DTF table being opened.

Function: Performs initialization functions to prepare audio line group and output queues files for operation.

The routine first tests to determine which file is being opened and branches to the proper subroutine.

Open Audio Line Group Subroutine: When a 7770 Line Group DTF is the first opened DTF table, the subroutine places the address of the QTAM Vector Table into the DOS communication region. This is necessary to allow the cross-partition communication required in QTAM. The address of this DTF table begins the chain of opened DTF tables in the Vector Table (VECLDTF field). When a 7770 Line Group DTF is not the first opened DTF table, the address of this DTF table is added to the chain of previously opened DTF tables.

When a 7772 Line Group DTF is opened, the subroutine checks that the 7772 DCV Vocabulary file has been opened, and then places the address of this DTF table in the chain of previously opened DTF tables.

In any case, the addresses of the ARU-Send queue and the QTAM SVC routines are placed in the Vector Table.

As each audio line group is opened, the address of the first ALCB is obtained and the line table address is stored into the VECLTBL field of the Vector Table. The address of the line table entry is searched and stored in the ALCB. If a 7770 line group requires an initial write operation, the ALCB channel program is updated with the informational or invitational data. In any case, the ALCB status is set to ENABLE (LSTS=X'01' indicates enable status), and the ALCB channel program is executed by issuing an EXCP.

The subroutine then tests if all ALCBs in the line group have been handled. If not, the next ALCB address is acquired and initialized. If all ALCBs have been handled, the next DTF address is acquired and tested to determine if it is that of an audio line group file. If it is not, return is made to the QTAM Open Monitor routine through a FETCH.

Open Audio Output Queues Subroutine: The subroutine moves the ARU-Send queue address from the VECARUSQ field of the Vector Table to the Audio Output queue DTF table. The address of this DTF table is linked into the chain from the VECPDQLK field in the Vector Table. Control returns to the QTAM Open Monitor routine through a FETCH.

External Routines Used:

FETCH (SVC 2)

EXCP (SVC 0)

QTAM CLOSE ROUTINE (PHASE 1)

Module Name: IJLQC1 (Chart C1)

Entry point: Whenever a QTAM file is to be closed, this routine is fetched into the logical transient area by the DOS Close Monitor and entered at IJLQC1. The QTAM Close module is cataloged in the Core Image Library under $$BCQC01. Upon entry, register 2 contains the address of the DTF table for the file being closed.

Function: Analyzes the type of QTAM file to be closed, performs a cleanup action for the files used in the message control program, and fetches the QTAM Close Phase 2 for the files used in the message processing program.

When this phase is entered because of a CLOSE macro for a type DA, AV, or LG file, all line operations have already been stopped. Therefore, only minor cleanup functions are required to close these files. The DTF table for the file being closed is removed from the chain of those opened, and the open bit is turned off. When the type CK file is being closed the first byte of the checkpoint control record is set to X'00', and it is written onto the disk to indicate a proper close. the Ready Queue in the QTAM nucleus is reinstated to its initial status (that is, the status of the Ready Queue at the time the message control program phase is loaded to begin execution). This action eliminates the necessity for an IPL procedure each time a QTAM message control program is loaded for execution.

Upon completion of processing, return is made to the DOS Close Monitor via a FETCH.

External Routines used:

FETCH (SVC 2)

EXCP (SVC 0)

Message Writer (Module IJLQMW)

QTAM CLOSE ROUTINE (PHASE 2)

Module Name: IJLQC2 (Chart C2)

Entry Point: This routine is called into the logical transient area via a FETCH issued by Phase 1 of the QTAM Close Routine. Phase 2 of the QTAM Close routine is cataloged in the Core Image Library under the name $$BCQC02. Control is passed to this routine at IJLQC2 and upon entry register 2 contains the address of the DTF table for the file to be closed.

Function: Causes a QTAM file opened in a message processing program to be closed. When the closing of the last PQ, DQ, or AQ type file is completed, this routine issues a FETCH to call in Phase 3 of the QTAM Close routine.

Because of the nature of QTAM, all PQ, DQ, and AQ files opened in any message processing programs must be closed before the QTAM files in the message control program can be closed.

For a PQ (MS-process), DQ (MS-destination), or AQ (Audio Output) file, the address of the DTF table is removed from the chain of DTF tables created at Open time. The chain begins at the address of the last PQ, DQ, or AQ table opened and is obtained from the VECPDQLK field of the QTAM Vector Table. If this is a type PQ DTF table, the open bit is turned off and the link address to the corresponding DASD-process queue is destroyed. The link address to this processing queue placed in the QSTL field of the QCB for the DASD-process queue at Open time is also cleared. The only thing required for a type DQ or AQ is to turn off the open bit.

Each time a type PQ, DQ, or AQ file is being closed, this routine tests the master-receive switch to determine if a closedown procedure is in progress. If so, return is made to the DOS Close Monitor. But, if there is no closedown procedure and if type PQ file operates on audio messages, all the audio line group files sending messages to the MS-process queue are searched from the VECLDTF field of the QTAM Vector Table. For each active audio line, a special ECB representing a request for a Halt I/O operation is posted via a QPOST to the LPS queue before issuance of the QWAIT. LPS control recognizes the element and issues a Halt I/O (SVC 27) on the line to break the connection. If necessary, the DCV buffer associated with a 7772 line is freed by posting it (via an SVC QPOST) to the DCV buffer queue from which it has been obtained, and control is then returned to the DOS Close Monitor. If a closedown procedure is indicated, a check is made to see if all PQ, DQ, or AQ files in the system have been closed (this condition exists when the VECPDQLK field contains zeros). If not, return is made to the DOS Close Monitor. The Close routine goes through the chain of DTF table for type DA, AV, and LG files (starting at the address in the VECLDTF field) to access each line in the system.

For each active nonaudio line, a QWAIT (SVC 30) is issued for the LCB to stop all operations on the line. If the line is a switched line and is currently connected to

a terminal, a special ECB representing a request for a Halt I/O operation is posted via a QPOST to the LPS queue before issuance of the QWAIT. LPS control recognizes the element and issues a Halt I/O (SVC 27) on the line to break the connection. Before issuing any QWAITs to stop line operations, a switch (actually part of the master-receive switch) is set which causes each subsequent QWAIT on an LCB to have a low priority. This action causes all outgoing messages still on the destination queue for the line to be sent before the QWAIT is staisfied.

For each active audio line, the end of the write operation is expected by testing the ALCB status flag (LSTS=X'01' indicates, at this time, no activity on the line).

In this manner, all outgoing message traffic is flushed before this routine releases control. All incoming traffic was stopped previously by the Close Message Control routine upon issuance of a CLOSEMC macro in the message processing program.

Return is made to the Close Monitor unless the file being closed is the last QTAM file closed in the message processing program(s). In this case Phase 3 of the QTAM Close routine is fetched.

External Routines Used:

HALT I/O (SVC 27)

QPOST (SVC 31)

QWAIT (SVC 30)

Message Writer (Module IJLQRW)

FETCH (SVC 2)

QTAM CLOSE ROUTINE (PHASE 3)

Module Name: IJLQC3 (Chart C3)

Entry Point: This routine is called into the logical transient area via a FETCH issued by Phase 2 of the QTAM Close routine when the last PQ, DQ, or AQ file is closed from a message processing program. Phase 3 of the QTAM Close routine is cataloged in the Core Image Library under the name $$BCQC03. Control is passed to this routine at IJLQC3.

Function: Closed, the Close routine goes through the chain of DTF table for type DA, AV, and LG files (starting at the address in the VECLDTF field) to access each line in the system.

For each active nonaudio line, a QWAIT (SVC 30) is issued for the LCB to stop all operations on the line. If the line is a switched line and is currently connected to a terminal, a special ECB representing a request for a Halt I/O operation is posted via a QPOST to the LPS queue before issuance of the QWAIT. LPS control recognizes the element and issues a Halt I/O (SVC 27) on the line to break the connection. Before issuing any QWAITs to stop line operations, a switch (actually part of the master-receive switch) is set which causes each subsequent QWAIT on an LCB to have a low priority. This action causes all outgoing messages still on the destination queue for the line to be sent before the QWAIT is staisfied.

For each active audio line, the end of the write operation is expected by testing the ALCB status flag (LSTS=X'01' indicates, at this time, no activity on the line).

In this manner, all outgoing message traffic is flushed before this routine releases control. All incoming traffic was stopped previously by the Close Message Control routine upon issuance of a CLOSEMC macro in the message processing program. It performs action to initialize for entering the closedown routine in the message control program. The address of the save area for the F1 partition is obtained from the F1 PIB. The return PSW in the save area is overlaid with the address specified in the EOJAD keyword operand of the DTFQT macro instruction for the first opened file in the message control program. The registers saved by the ENDREADY macro expansion are moved from the user's save area into the F1 save area, and the QTAM wait flag in the PIB is turned off, making the message control program eligible for activation. When activated, the message control program is entered at the EOJAD address, which begins a section of code that includes CLOSE macros that complete the closing of the type LG, AV, and DA files. being closed, the Ready Queue in the QTAM nucleus is reinstated to its initial status (that is, the status of the Ready Queue at the time the message control program phase is loaded to begin execution). This action eliminates the necessity for an IPL procedure each time a QTAM message control program is loaded for execution. Return is made to the Close Monitor.

External Routines Used:

FETCH (SVC 2)

QTAM AUDIO MESSAGE WRITER ROUTINE

Module Name: IJLQWA (Chart WA)

Entry Point: This routine is called into the logical transient area by a FETCH (SVC 2) issued by the LPS Control routine upon recognition of a special audio ECB posted to the Ready Queue by the Message Writer Initiator routine (IJLQMW). The Audio Message Writer is cataloged on the Core Image Library under the name $$BQWTRA. Entry to the routine is at IJLQWA.

Function: The routine tests to determine what error message is to be written. This message was registered by the Error Recovery Procedures in the Audio Line Appendage module (IJLQAA) or the 7772 Disk Appendage (IJLQAD). The routine writes the message to the system console by issuing an EXCP (SVC 0). Upon return, an SVC 11 is immediately issued to request return to the routine which issued the FETCH.

External Routines Used:

EXCP (SVC 0)

QTAM MESSAGE WRITER ROUTINE (PHASE 1)

Module Name: IJLQW1 (Chart W1)

Entry Points: This routine is called into the logical transient area by a FETCH (SVC 2) issued by the QTAM Open or Close routines, or by the LPS Control routine upon recognition of a special ECB posted to the Ready Queue by the Message Writer Initiator routine (IJLQMW). Phase 1 of the Message Writer routine is cataloged on the Core Image Library under the name $$BQWTR1. Entry to the routine is at IJLQW1. If entry is from the Open or Close routine, register 0 contains the message number, right-adjusted, and register 2 contains the address of the DTF table.

Function: The routine tests to determine what error message is to be written. If a message to be written was registered by the Error Recovery Procedures in the Line Appendage module (IJLQLA) or in the Audio Line Appendage module (IJLQAA), the routine writes the message to the system console (or, if the operator control feature is specified requesting error messages sent to the operator control terminal, to that terminal) by issuing an EXCP (SVC 0). Upon return, an SVC 11 is immediately issued to request return to the routine which issued the FETCH.

If the message to be written was requested by one of the QTAM Open or Close routines, the routine calls into the logical transient area, via a FETCH (SVC 2), either Phase 2 or Phase 3 of the QTAM Message Writer, depending upon what message is to be written.

External Routines Used:

EXCP (SVC 0)

FETCH (SVC 2)

QTAM MESSAGE WRITER ROUTINE (PHASE 2)

Module Name: IJLQW2 (Chart W2)

Entry Point: This routine is called into the logical transient area via a FETCH (SVC 2) issued by Phase 1 of the QTAM Message Writer routine upon recognition of a request for an error message which this routine is to write. Phase 2 of the Message Writer routine is cataloged on the Core Image Library under the name $$BQWTR2. Entry to the routine is at IJLQW2. Register 0 contains the message number, right-adjusted, and register 2 contains the address of a DTF table.

Function: The routine writes error messages requested by the QTAM Open or Close routines. The message may be notification of an invalid Open, an invalid Close, an invalid DTFQT, a TERM or LINE entry not found, an invalid DCV word address, or any of a group of messages notifying the operator of errors in the extent data. The routine writes the message to the system console by issuing an EXCP (SVC 0). Upon return, a WAIT (SVC 7) is issued to await completion of the writing of the message. Upon completion, the job is cancelled via a CANCEL (SVC 6).

External Routines Used:

EXCP (SVC 0)

CANCEL (SVC 6)

WAIT (SVC 7)

QTAM MESSAGE WRITER ROUTINE (PHASE 3)

Module Name: IJLQW3 (Chart W2)

Entry Point: This routine is called into the logical transient area via a FETCH (SVC 2) issued by Phase 1 of the QTAM Message Writer routine upon recognition of

a request for an error message that this routine is to write. Phase 3 of the Message Writer routine is cataloged on the Core Image Library under the name $$BQWTR3. Entry to the routine is at IJLQW3. Register 0 contains the message number, right-adjusted, and register 2 contains the address of a DTF table.

Function: The routine writes any of a second group of messages requested by the QTAM Open or Close routines, which notify the operator of errors in the extent data. The routine writes the message to the system console by issuing an EXCP (SVC 0). Upon return, a WAIT (SVC 7) is issued to await completion of the writing of the message. Upon completion, the job is cancelled via a CANCEL (SVC 6).

External Routines Used:

EXCP (SVC 0)

CANCEL (SVC 6)

WAIT (SVC 7)

QTAM CANCEL ROUTINE (PHASE 1)

Module Name: IJLQXL (Chart XL)

Entry Point: If a QTAM Message Control Program or Message Processing Program is cancelled, this routine is fetched into the logical transient area by the DOS Terminator Monitor ($$BEOJ) and entered at IJLQXL. Phase 1 of the QTAM Cancel routine is cataloged on the Core Image Library under the name $$BQCNCL.

Function: If a message Processing Program is cancelled, all references to that Message Processing Program are purged from the Message Control Program and the Supervisor.

Immediately upon entry, two SVC 22s are issued to disable the system for I/O interrupts. The first SVC 22 seizes the system and sets the system mask. The second SVC 22 releases the system. A test is then made to determine if the Message Control Program or a Message Processing Program is being cancelled. If the Message Control Program is being cancelled, exit is made by fetching $$BQCNCM, Phase 2 of the QTAM Cancel routine.

This routine is fetched during cancellation of a Message Processing Program only for abnormal termination of a Message Processing Program, or when cancellation is requested from the console.

The return PSW in the program save area is examined to determine if the cancelled Message Processing Program was in a QWAIT status due to a GET, PUT, or RETRIEVE operation. If it is in a QWAIT status, the STCB chain of the QCB or BRB being waited on is searched to locate the full STCB representing reentry to the Message Processing Program. When located, the full STCB is removed from that chain and reset to an inactive status. It is then linked to the bottom of the Ready Queue for its next usage (otherwise the full STCB would be lost). If the item waited on is a BRB in the element chain of the Active BRB or Disk I/O queue, the BRB is removed from the chain and the chain is relinked.

The remaining cleanup functions are performed regardless of whether a QWAIT is pending. The chain of Message Processing Program DTF tables (starting at VECPDQLK in the Vector Table) is searched for DTF tables defined and opened in the cancelled Message Processing Program. Each DTF table associated with the cancelled program is thus found and removed from the chain of DTFs. If the DTF table is for a MS-destination queue, the "in source chain" flag (LALT+2) in the DTF's LCB is tested to determine if a PUT is currently in progress. If so, the DTF's LCB is removed from the chain of LCBs currently sending to that destination.

If the DTF table is for a MS-process queue, further message transfer to that queue is stopped by clearing the QSTL field in the corresponding DASD-process QCB. Next, any buffers currently associated with the DTF table are returned to the QTAM buffer pool in the Message Control Program to avoid losing them. The last buffer passed to the Message Processing Program in response to a GET is released by posting via an SVC 31 the buffer to the Return Buffer queue. Each unprocessed buffer in the element chain of the MS-process queue is posted to the Available Buffer queue via an SVC 31. Additionally, if the EXPEDITE option has not been specified for the queue, the next disk read address (QNRA field in the DASD-process QCB) is reset with the relative record number of the first unprocessed header segment in the MS-process queue. This action causes the unprocessed messages to be read from the disk queue again if the Message Processing Program is reloaded and the DTF table reopened.

The system is enabled for interrupts, and exit is made by fetching $$BEOJ3.

External Routines Used:

SVC 22

SVC 2

QPOST (SVC 31)

QTAM CANCEL ROUTINE (PHASE 2)

Module Name: IJLQXM (Chart XM)

Entry Point: If a QTAM Message Control Program is being cancelled, this routine is fetched into the the logical transient area by Phase 1 of the QTAM Cancel routine and is entered at IJLQXM. Phase 2 of the QTAM Cancel routine is cataloged on the Core Image Library under the name $$BQCNCM.

Function: This Cancel phase performs cleanup functions for the Message Control Program and initiates the cancellation of each Message Processing Program in the system (if any exists).

Immediately upon entry, two SVC 22s are issued to disable the system for I/O interupts. The first SVC 22 seizes the system and sets the system mask. The second SVC 22 releases the system. Line error statistics for each line in the system are then printed (via an EXCP) on the operator console during this cancellation procedure.

A QTAM Message Processing Program cannot run without the Message Control Program. Therefore cancellation of each Message Processing Program (if any) is initiated. The chain of Message Processing Program DTF tables created at Open time (starting at VECPDQLK in the QTAM Vector Table) is searched for the presence of any Message Processing Programs in the system. As each Message Processing Program is found, a cancel flag of X'8163' is set in the PIB for that Message Processing Program. This action causes the Message Processing Program to be cancelled the next time the DOS Task Selection mechanism selects that program for activation. The QTAM Ready Queue in the Supervisor is then reinstated to its initial status so that a new copy of the Message Control Program can be loaded for execution without the necessity for an IPL procedure. The QTAM word in the DOS communication region is cleared to remove any indication of QTAM from the system. Two SVC 22s are issued to enable the system for interrupts, and exit is made by fetching $$BEOJ3 for completion of the cancellation procedure.

External Routines Used:

SVC 22

EXCP (SVC 0)

WAIT (SVC 7)

SVC 2

TERMINAL TEST HEADER ANALYSIS ROUTINE

Module Name: IJLQTM (Charts TM and TN)

Entry Point: This routine is called into the logical transient area via a FETCH (SVC 2) issued by the Terminal Test Recognition routine. The Terminal Test Header Analysis routine is cataloged on the Core Image Library under the name $$BQHDCK. Entry is at IJLQTM21. The address of a parameter list is passed in register 0.

Function: Performs preliminary validation of the test request, translates the input message as necessary, and sets up the terminal addressing characters.

The input message is located, and any translation necessary is performed. Translations that may be needed are for symbolic addresses of terminals and translation between ASCII and EBCDIC.

The proper terminal addressing characters, the address of the LCB, and the device type and features are placed in the buffer prefix.

The proper Terminal Test routine is then called into the logical transient area via a FETCH.

External Routines Used:

FETCH (SVC2)

Terminal Test Module for IBM 1030 (IJLQT1)

Terminal Test Module for IBM 1050 (IJLQT2)

Terminal Test Module for IBM 1060 (IJLQT3)

Terminal Test Module for IBM 2260 (IJLQT4)

Terminal Test Module for IBM 2740 (IJLQT5)

TERMINAL TEST MODULE FOR IBM 1030

Module Name: IJLQT1 (Chart T1)

Entry Point: This routine is called into the logical transient area via a FETCH (SVC 2) issued by the On-Line Terminal Test Header Analysis routine. The Terminal Test Module for IBM 1030 is cataloged on the Core Image Library under the name $$BQ1030. Entry is at IJLQT1. Register 0 contains the address of a parameter list.

Function: Generates channel program for IBM 1030 on-line tests.

The input message is located and the header is validated. The header is analyzed for type of request. CCWs for the proper data and control purposes are formed in a special area defined for this purpose. (If the buffer can be utilized, the CCWs are formed in the text area of the buffer.) If the requested data pattern is not in use, the pattern is moved into a new area. If the new area is not available, the test request is deferred for later processing. If the request is for a stored data comparison and the data is not correct, the input message is switched. Return is to the Terminal Test Recognition routine.

External Routines Used: None.

TERMINAL TEST MODULE FOR IBM 1050

Module Name: IJLQT2 (Chart T2)

Entry Point: This routine is called into the logical transient area via a FETCH (SVC 2) issued by the On-Line Terminal Test Header Analysis routine. The Terminal Test Module for IBM 1050 is cataloged on the Core Image Library under the name $$BQ1050. Entry is at IJLQT2. Register 0 contains the address of a parameter list.

Function: Generates channel program for IBM 1050 on-line tests.

The input message is located and the header is validated. The header is analyzed for type of request. CCWs for the proper data and control purposes are formed in a special area defined for this purpose. (If the buffer can be utilized, the CCWs are formed in the text area of the buffer.) If the requested data pattern is not in use, the pattern is moved into a new area. If the new area is not available, the test request is deferred for later processing. If the request is for a stored data comparison and the data is not correct, the input message is switched. Return is to the Terminal Test Recognition routine.

External Routines Used: None.

TERMINAL TEST MODULE FOR IBM 1060

Module Name: IJLQT3 (Chart T3)

Entry Point: This routine is called into the logical transient area via a FETCH (SVC 2) issued by the On-Line Terminal Test Header Analysis routine. The Terminal Test Module for IBM 1060 is cataloged on the Core Image Library under the name $$BQ1060. Entry is at IJLQT3. Register 0 contains the address of a parameter list.

Function: Generates channel program for IBM 1060 on-line tests.

The input message is located and the header is validated. The header is analyzed for type of request. CCWs for the proper data and control purposes are formed in a special area defined for this purpose. (If the buffer can be utilized, the CCWs are formed in the text area of the buffer.) If the requested data pattern is not in use, the pattern is moved into a new area. If the new area is not available, the test request is deferred for later processing. If the request is for a stored data comparison and the data is not correct, the input message is switched. Return is to the Terminal Test Recognition routine.

External Routines Used: None.

TERMINAL TEST MODULE FOR IBM 2260

Module Name: IJLQT4 (Chart T4)

Entry Point: This routine is called into the logical transient area via a FETCH (SVC 2) issued by the On-Line Terminal Test Header Anaylsis routine. The Terminal Test Module for IBM 2260 is cataloged on the Core Image Library under the name $$BQ2260. Entry is at IJLQT4. Register 0 contains the address of a parameter list.

Function: Generates channel program for IBM 2260 on-line tests.

The input message is located and the header is validated. The header is analyzed for type of request. CCWs for the proper data and control purposes are formed in a special area defined for this

purpose.(If the buffer can be utilized, the CCWs are formed in the text area of the buffer.) If the requested data pattern is not in use, the pattern is moved into a new area. If the new area is not available, the test request is deferred for later processing. If the request is for a stored data comparison and the data is not correct, the input message is switched. Return is to the Terminal Test Recognition routine.

External Routines Used: None

TERMINAL TEST MODULE FOR IBM 2740

Module Name: IJLQT5 (Chart T5)

Entry Point: This routine is called into the logical transient area via a FETCH (SVC 2) issued by the On-Line Terminal Test Header Analysis routine. The Terminal Test Module for IBM 2740 is cataloged on the Core Image Library under the name $$BQ2740. Entry is at IJLQT5. Register 0 contains the address of a parameter list.

Function: Generates channel program for IBM 2740 on-line tests.

The input message is located and the header is validated. The header is analyzed for type of request. CCWs for the proper data and control purposes are formed in a special area defined for this purpose. (If the buffer can be utilized, the CCWs are formed in the text area of the buffer.) If the requested data pattern is not in use, the pattern is moved into a new area. If the new area is not available, the test request is deferred for later processing. If the request is for a stored data comparison and the data is not correct, the input message is switched. Return is to the Terminal Test Recognition routine.

External Routines Used: None.

The QTAM SVC/Subtask Control routine is included in the DOS Supervisor at System Generation to handle interrupts caused by the issuance of a QTAM SVC (SVC 30 or SVC 31) and to control the dispatching of QTAM subtasks. This supervisory routine consists of nine subroutines which are summarized below. (See Charts QW, QX, QY.)

ENTRY INTERFACE SUBROUTINE

This subroutine is entered from the SVC Interrupt Handler of the Supervisor whenever a QTAM SVC (QWAIT or QPOST) is issued. This subroutine performs initialization functions for the QTAM SVC/Subtask Control routine.

Associated with each entry to this subroutine is a Program Information Block (PIB). One PIB per partition is pre-assembled into the DOS Supervisor, and the address of the PIB associated with the partition from which the SVC was issued is passed (in register 10) to the Entry Interface subroutine. This PIB contains information about the partition and includes:

1. a ready/waiting flag; and
2. the address of the program save area where registers and the return program status word (PSW) are stored.

The program issuing the QTAM SVC is represented to the SVC/Subtask Control routine by a full STCB. One full STCB per partition (or task when support for multitasking is included) is pre-assembled into the QTAM SVC/Subtask Control routine for this purpose. These full STCBs are initially in an inactive state and are linked into the STCB chain of the pseudo QCB labeled QSVCQCB.

When the Entry Interface subroutine is entered, it obtains an inactive full STCB and initializes it to represent re-entry to the program requesting service when the request has been satisfied. The full STCB to be used for this purpose is always the first STCB in the STCB chain of the last QCB dispatched by the QTAM SVC/Subtask Control routine. The address of this "last dispatched QCB" is obtained from the location labeled QSVCSAVE and is initially the address of the pseudo QCB (QSVCQCB).

The initialization functions performed by the subroutine are:

1. Saves registers and the old SVC PSW in the save area pointed to by the PIB (this is accomplished via a branch and link to the supervisor's generalized save routine, SVEREG).
2. Saves all registers (7 through 6) in the QTAM save area.
3. Inserts the PIB address into the full STCB (the full STCB contains the address of QTAMs dummy PIB when not initialized). Cancel exit when no STCB available.
4. Assigns a priority to the full STCB.
5. Sets a wait flag in the PIB so that the program associated with the PIB is not selected for re-activation by the task selection mechanism of the DOS Supervisor.

After these initialization functions are performed, the Entry Interface subroutine exits to the QTAM Wait or Post subroutine, depending on which SVC has been issued.

QTAM POST SUBROUTINE

The Post subroutine is entered from the Entry Interface subroutine when an SVC 31 (QPOST) has been issued. Three nonaudio implementation routines (Interim LPS, BRB Ring, and Line Appendage routines) and three audio implementation routines (7772 Disk Read, 7772 Line Write, and Audio Line Appendage routines) enter this subroutine, via a direct branch, to cause an effective post. Entry is always at QSVCPOST. Upon entry, register 1 contains the address of an element and register 2 contains the address of the QCB to which the element is to be posted.

The Post subroutine places the address of the passed QCB into the QCB address field of the specified element control block. This is the means by which an element becomes associated with a QCB. This subroutine then branches to the Priority Search subroutine to cause the element to be placed in the Ready Queue in priority order.

QTAM WAIT SUBROUTINE

The Wait subroutine is entered at QSVCWAIT from the Entry Interface subroutine when an SVC 30 (QWAIT) has been issued. Upon entry, register 2 contains the address of the QCB from which an element is requested by the SVC. Register 9 contains the address of the last QCB dispatched on the previous entry into the QTAM SVC routine. The first STCB in the subtask chain of this last-dispatched QCB is the full STCB representing re-entry (when the wait condition is satisfied) to the program that issued the SVC.

The Wait subroutine determines what disposition is necessary to schedule the waiting program (represented by the full STCB) for re-activation. Depending on current conditions, any one of four possible courses of action may be necessary.

1. If the highest-priority STCB in the subtask chain of the QCB specified by the SVC is "not waiting" (QKEY=2), this subroutine branches immediately to the Defer-Entry subroutine at QSVCUNAV. Reasons for this branch are explained in the discussion of that subroutine.

2. If the specified QCB has an element available on its ECB chain, the element is removed from the chain and its address placed into parameter register one. The Wait subroutine then branches to the Exit Select subroutine at QSVCRTNX. This branch causes the full STCB to be selected for activation. Control is thus returned (via the DOS Supervisor) to the program that issued the SVC 30 with the address of the requested element in register 1.

   The net effect is that at the time the program requesting an element is dispatched, its full STCB still appears in the subtask chain of the last-dispatched QCB; however, the element chain from which it is drawing elements is that of the QCB specified by the requesting program. This action insures immediate satisfaction of the wait condition when the requested element is available.

3. If the specified QCB has no elements available, but the last-dispatched QCB and the QCB specified by the requesting program are the same, the STCB is already chained into the correct QCB and the QCB is already waiting on the Ready Queue. The Wait subroutine branches to the Qdispatch subroutine.

4. If the specified QCB has no elements available and is not the last-dispatched QCB, the Wait subroutine branches to the Defer-Entry subroutine at QSVCUNAV. This action causes the full STCB to be removed from the subtask chain of the last-dispatched QCB, and linked (by priority) into the subtask chain of the specified QCB so that it can be dispatched when an available element is posted to the specified QCB.

The Wait subroutine is also entered at the special entry point, QSVCTSTQ, by the BRB Ring routine in the Implementation module. A detailed discussion of the reasons for this special entry appears in the description of that routine. Briefly, this determines if the last-dispatched QCB and a QCB specified by the calling routine are the same. Disposition is the same as just described in item 3 or 4. The main thing here is that the item being manipulated is an LCB rather than a full STCB.

PRIORITY SEARCH SUBROUTINE

The Priority Search subroutine is entered at QSVCPRI:

1. from the QTAM Post subroutine or the Defer-Entry subroutine;

2. when the Queue Insert by Priority subtask is dispatched; or

3. by a direct branch from the End of Poll Time Delay or Active Buffer Request routine in the Implementation module.

This subroutine performs the generalized function of determining the position within a chain that an item should assume to be in correct priority sequence. Items in the chain are arranged in descending order of priorities from the top of the chain. The subroutine acts on all chains including the Ready Queue.

Upon entry, register 9 points to the head of the chain, and register 1 contains the address of the item to be ordered within the chain. The item to be ordered may be an element, an STCB, or a QCB, depending on the source of this entry.

In operation, this 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 a priority of 255). When either condition is met, the subroutine exits to the Queue Insert subroutine which actually inserts the item into the chain.

QUEUE INSERT SUBROUTINE

The Queue Insert subroutine is entered at QSVCLIFO:

1. from the Priority Search subroutine;
2. when the Queue Insert subtask is dispatched; or
3. via a direct branch from the Available Buffer, Buffer-BRB, or Disk I/O routine.

This subroutine performs the generalized function of linking an item into a chain. It is applied to all chains including that of the Ready Queue. 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 priority of the item.

When this subroutine is entered, register 1 contains the address of the item to be inserted. Register 9 points to the link address field of the item already in the chain which the new item is to follow. The link address field of the item already in the chain is placed into the new item and is replaced by the address of the new item. This subroutine then exits to the Qdispatch subroutine for handling of the item currently at the head of the Ready Queue.

DEFER-ENTRY SUBROUTINE

The Defer-Entry subroutine is entered at QSVCUNAV either:

1. from the QTAM Wait subroutine or the Qdispatch subroutine, or
2. via a direct branch from the Send Scheduler or Receive Scheduler routine in the Implementation module.

This subroutine causes the activation of a subtask to be deferred. Which subtask is being deferred and the reason depends on the source of entry into this subroutine. If entered from the QTAM Wait subroutine, re-entry to the program that issued an SVC QWAIT is deferred because the element requested is not yet available. If entered from the Send Scheduler routine, the Send Scheduler subtask for a nonswitched line is deferred because the line is not currently available for sending.

When one of the preceding routines encounters an STCB for a subtask that cannot be activated, a branch to the Defer-Entry subroutine is taken. Upon entry, register 9 contains the address of the QCB in which the STCB was encountered; register 2 contains the address of a QCB specified by the calling routine. This subroutine removes the STCB from the STCB chain of the QCB in which it appears, and branches to the Priority Search subroutine. This causes the removed STCB to be placed (by priority) into the STCB chain of the QCB specified by the calling routine.

An exception arises if the QKEY field of the QCB specified by the calling routine is 2. This condition indicates that the highest-priority subtask on the STCB chain of that QCB is a ready subtask, that is, a 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 pre-empting the ready status that applies to the current top STCB. For maximum efficiency, the ready STCB should be honored first. Therefore, this subroutine enters the Priority Search subroutine by a path that insures that the new STCB is queued by priority order below the current top STCB.

QDISPATCH SUBROUTINE

This subroutine is entered at QSVCDISP:

1. from the Queue Insert subroutine or the QTAM Wait subroutine;
2. when the Qdispatch subtask is dispatched; or
3. via a direct branch from numerous routines in the Implementation module or in the Audio Line Appendage Module.

The Qdispatch subroutine performs the primary internal management function within QTAM. This subroutine maintains continuity by receiving control from a completed subtask and selecting the next subtask which is to receive control, except for those cases in which another subroutine is able to determine the subtask to be activated next (for example, when the Wait subroutine finds that an element is

already available for a full subtask requesting the element and that the Qdispatch subroutine can be bypassed).

In most cases, when a QTAM subtask has completed its unit of work or is unable to perform its unit of work (for example, if the Send Scheduler subtask is dispatched to initiate sending on a line that is designated as an input only line), entry is made into the Qdispatch subroutine for handling of the item currently at the head of the Ready Queue. This entry may be either direct, that is, by a direct branch, or indirect. An indirect entry occurs when a subtask branches to some other subroutine in the SVC/Subtask Control routine, and that subroutine, in turn, causes entry into the Qdispatch subroutine. No parameters are passed.

In operation, this subroutine examines the item currently at the head of the Ready Queue (location QSVCRDYQ) and takes the appropriate action, depending on the type of item encountered. Items that can appear on the Ready Queue are:

1. Queue Control Blocks (QCB) for which the highest-priority subtask is not waiting for elements (QKEY = 2) and is ready to be dispatched.

2. Queue Control Blocks for which the highest-priority subtask is waiting for elements (QKEY = 3) and cannot be dispatched.

3. Element Control Blocks (ECB), containing the address of the QCB to which the element (Buffer, BRB, or LCB) has been posted (ECB key is 0).

4. Full STCBs for which the key value (SKEY) is also zero. The first word of a full STCB always contains the address of the pseudo QCB labeled QSVCQCB.

The explanation of the effect of the appearance of each type of item at the head of the Ready Queue follows.

Queue Control Block -- Not Waiting (QKEY is 2): When the item at the head of the Ready Queue is a "not waiting" QCB, this subroutine exits to the Exit Select subroutine at QSVCBD. This causes control to be given to the first (highest priority) subtask represented in the STCB chain of that QCB and the QCB becomes a "waiting" QCB (QKEY is set to 3).

Queue Control Block -- Waiting (QKEY is 3): When a "waiting" QCB appears at the head of the Ready Queue, it is removed from the Ready Queue and its key is set to 1. The QCB is replaced at the head of the Ready Queue by the item to which it linked. A branch is made to the beginning of the Qdispatch subroutine for examination of the new item.

A QCB waiting for elements cannot contend for control; however, it will automatically be returned to the Ready Queue when an element becomes available.

Element Control Block: An element control block appears on the Ready Queue as a result of an SVC 31 (QPOST) and 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 next item on the Ready Queue. However, the QCB pointer in the element control block 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, that is, the QCB with which the active subtask control block is associated, is not necessarily at the head of the Ready Queue.

3. It explains that an element control block 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 there of a "not waiting" (key is 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 (QSVCQCB) is set to 3.

The mechanism by which this is accomplished is as follows:

1. Location QSVCRDYQ contains a pointer to the full STCB.

2. The full STCB itself has the appearance (to the Qdispatch subroutine) of an element. Its QCB address is QSVCQCB.

3. QSVCQCB is a storage location equivalent to QSVCRDYQ minus 8 bytes. It also appears to be the first word of a "not waiting" QCB.

4. Because the full STCB is apparently an element control block associated with a "not waiting" QCB, the first STCB in the chain of that QCB should be selected for activation. The address of the first STCB is found in the third fullword of the QCB.

5. The third fullword of QSVCQCB which appears as a QCB is the location QSVCRDYQ. Therefore, the full subtask, whose address is at QSVCRDYQ, is selected for control.

EXIT SELECT SUBROUTINE

This subroutine activates QTAM subtasks represented by truncated STCBs or exits to the Exit Interface subroutine if the STCB is a full STCB.

The first byte of a truncated STCB is the entry code field and 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 entry code, it computes the branch address and branches to the computed entry point in the QTAM Implementation module (IJLQIP).

If the entry code is zero, the STCB is a full STCB representing re-entry to a program that issued a QTAM SVC. In this case, this subroutine exits to the Exit Interface subroutine.

EXIT INTERFACE SUBROUTINE

This subroutine is entered from the Exit Select subroutine to process full STCBs. Before any other action is taken, the subroutine determines whether the problem program 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 parameter register 1 and the register stored in the save area of the partition associated with the full STCB being processed. The address of the save area is obtained from a PIB, the address of which has previously been inserted into the STCB.

Before control is returned to the DOS Supervisor, several functions are performed. The wait flag is removed from the PIB. This action indicates to the Supervisor that the requested service has been completed and the program issuing the SVC can be reactivated. The registers are restored from the save area. The full STCB being used is restored to an "unused, available" state by removing the PIB address from the STCB. After these functions are complete, the Exit Interface subroutine branches to the Supervisor General Exit routine for eventual re-entry to the program issuing the SVC.

The routines in this section, together with those in the Line Input/Output section, perform the primary functions necessary for the implementation of QTAM. These routines are in the QTAM Implementation module (IJLQIP), with the following exceptions:

- The Disk-End Appendage and the Disk I/O routine, which are packaged in the Disk I/O module (IJLQDA).
- The 7772 Disk-End Appendage, the Disk Read routine, the Line Write routine, and the DCV Buffer routine, which are packaged in the 7772 Disk Appendage module (IJLQAD).

Note that the greatest amount of QTAM audio implementation is performed in the routines packaged in the Audio Line Appendage module (IJLQAA) described in the Line Input and Output section.

As previously stated, all but three of these routines are supervisory routines which execute as a logical extension of the QTAM SVC/Subtask Control routine. The three routines in the Implementation module which run in problem program state are the LPS Control, Buffer Recall/Cleanup, and Free BRB routines. These routines provide the interface between the supervisory routines and the remainder of the message control problem program.

RECEIVE SCHEDULER ROUTINE (CHART 00)

This routine is entered at RCVSCH when the Receive Scheduler subtask is activated. This subtask is activated when the LCB (Communication Line QCB) appears at the top of the Ready Queue with the STCB for the Receive Scheduler at the head of its STCB chain.

Several tests are made to determine if receiving operations are to be initiated on the line represented by the LCB. If the status byte in the polling list for the line is zero, or if the master-receive-switch is off (system close is in process), receive operations are not initiated. If the LCB is for an IBM 2260 Local line group and no CCB has been posted to the Attention queue for the line group, receive operations are not initiated. In all cases, exit is made to the Exit Select subroutine at QSVCRETN to dispatch the next subtask in the STCB chain of the Communication Line QCB. The next subtask will normally be the line's Send Scheduler subtask, if any, or the Qdispatch subtask.

An exception arises if the next STCB is a full STCB representing a waiting problem program. If this is the case, the waiting program cannot be reactivated because the SVC it issued has not been satisfied. Therefore, the LCB is removed from the Ready Queue and exit is made to the Defer-Entry subroutine at QSVCUNAV to remove the full STCB from the STCB chain.

For remote terminals, this routine then examines the current polling list entry for the line. If the current entry is valid (that is, it is not the dummy entry signalling the end of the polling list), exit is made to the BRB-Ring routine at RQCONSTR to initialize for receiving. If the end of the polling list has been detected, the current entry pointer (LPPT field in LCB) is reset to the first entry in the list and exit is made to the End of Poll Time Delay routine. This causes either a polling interval to be observed or rescheduling of polling on the line.

For the LCB for an IBM 2260 Local line group in which a CCB is found in the Attention queue, the CCB is removed and the next CCB (if any) is promoted to the top of the queue. The address of the removed CCB is saved in the LCB, and the Attention Queue flag in the terminal table entry for the IBM 2260 Local is turned off. Exit is made to the BRB-Ring routine to continue initialization for a read operation.

If the line is a WTTA line, the Receive Scheduler routine tests the EOT flag in the LWTT field of the LCB. If this flag is on, exit is made to the Exit Select subroutine to enter the Send Scheduler subtask of the line, if present. If the EOT flag is off, exit is made to the BRB-Ring routine at RQCONSTR to initialize for receiving.

Note: A possible endless loop is circumvented by a test to determine that the polling list contains at least one active entry. If the list contains no active entry, the exit to the End of Poll Time Delay routine is not taken. Instead, the Receive Scheduler subtask is skipped over in the STCB chain, and control is passed to the second subtask in the chain via a branch to the Exit Select subroutine.

SEND SCHEDULER (CHART 1)

This routine is entered when an LCB appears at the top of the Ready queue with the Send Scheduler STCB at the head of its STCB chain, or when a message is to be written on the disk. If a message is to be written on the disk, the routine links to the DASD Destination routine at SCREEN to cause a post to the Disk I/O QCB.

If the DTF table has not been opened for output or the relative line number is greater than the number of extents, the routine branches to the Qdispatch subroutine. If the DTF table is open for output and this is a dial line, the routine tests for the availability of an access line. If it is not available, it looks at the next access line. If an access line is available, it disables the line to make it unavailable for incoming traffic.

If the DTF table is open for output and this is a WTTA line, the Send Scheduler routine tests whether data is being received. If so, control is returned to the Defer Entry subroutine at QSVCUNAV. If not, the line is disabled to be available for outgoing traffic.

If the line is nondial or non-WTTA, it is set to indicate the line is trying to send. If the line is not free, a branch to the Defer-Entry subroutine is made. If the line is free, the LCB is linked to the top of the Ready queue to cause the Send Scheduler subtask to be activated for line operations.

If the routine was entered because the LCB was on top of the Ready queue, the routine tests for an incoming priority message. If there are no completed nonpriority messages and if initiate mode has been specified, or if priority messages are coming in, the LCB is removed from the source chain. If neither an initiate mode nor sending is indicated in the LSTA field, the status of the LCB is cleared. If a partial message is in the queue (an invalid condition), the routine branches to the Wait subroutine. After the status code is set, the routine exits:

1. if entered via the Get Scheduler, return is to that routine, or

2. if the line is sending, the routine branches to the BRB-Ring routine at RQCONST.

END OF POLL TIME DELAY ROUTINE (CHART 2)

This routine has three entry points:

1. At DELAY, from the Receive Scheduler routine when the end of the polling list for a nonswitched line is detected. Upon entry, register 4 contains the address of the LCB representing the line for which a polling pass has been completed.

2. At TIMEEXIT, from the DOS Interrupt Handler whenever a timer interruption occurs. This entry is established by a STXIT macro issued in the LPS Control routine prior to setting of the timer.

3. At TIMEEND+6, when the Time Delay subtask is dispatched upon the appearance of the Time Delay QCB at the top of the Ready Queue.

When entry is at DELAY from the Receive Scheduler routine, a test determines if any messages were received during the polling pass just completed. If messages were received, a direct branch is made to the Priority Search subroutine in the QTAM nucleus. This causes the line to be rescheduled for polling as follows: the Receive Scheduler STCB is inserted into the STCB chain of the Communication Line QCB according to its priority relative to the priority of any Send Scheduler STCB already in the chain. Polling on the line resumes when the Communication Line QCB (the LCB) appears at the top of the Ready Queue with the Receive Scheduler STCB at the head of its STCB chain.

If no messages were received, the interrupt request time (current time plus user-specified polling interval) is calculated and placed in the LBCT field of the passed LCB. The interrupt request time is then used as the search argument for chaining the LCB into its proper position in the Time Delay queue relative to the interrupt request times of other LCBs already in the queue.

After the current LCB is inserted into the Time Delay queue (or when entry is due to activation of the Time Delay subtask), the current time is subtracted from the interrupt request time stored in the LCB at the top of the Time Delay queue. The result is tested to determine the required course of action:

1. If the result exceeds zero, that is, the interrupt request time is later than the current time, the calculated value is stored for use by the LPS Control routine. A special ECB

(labeled IP1IREB) representing a request for a polling interval is posted to the LPS queue via a direct branch to the Post subroutine. This causes entry into the LPS Control routine where a SETIME macro (SVC 10) is issued to set the timer for the calculated interval.

2. If the result is less than or equal to zero, that is, the current time is later than the interrupt request time, or an interval of zero was specified, a test is made to determine if the LCB is the dummy LCB in the checkpoint element. If it is the checkpoint element, it is removed from the top of the Time Delay queue and posted to the checkpoint queue to activate the checkpoint subtask. A test is made to determine if it is a pseudo-LCB passed from the Operator Control module. If it is, the pseudo-LCB is posted to itself to activate the INTREL subtask. If it is an LCB, it is removed from the top of the Time Delay queue, and a direct branch is made to the Priority Search subroutine. This causes polling on the line to be rescheduled as previously described for the case where messages were received during a polling pass. If this routine has been entered because of dispatching of the Time Delay subtask, a test determines if the line is free, that is, not currently being used for sending. If the line is free, the LCB is placed at the top of the Ready Queue prior to branching to the Priority Search subroutine. This causes polling on the line to resume immediately.

When a time interruption occurs, the End of Poll Time Delay routine is entered at TIMEEXIT from the DOS Interrupt Handler. The Time Delay queue is posted to itself via a QPOST (SVC 31). This action causes the Time Delay subtask (discussion follows) to be activated. Upon return from this SVC, exit is made to the DOS Interrupt Handler so that the program that was executing when the timer interrupt occurred can become eligible to regain control.

The End of Poll Time Delay routine is entered at TIMEEND+6 when the Time Delay subtask is activated. This entry occurs when the Time Delay QCB appears at the top of the Ready Queue (see QPOST after timer interruption just discussed). A test determines if there are any LCB's currently chained into the Time Delay queue. If there are, the current time is obtained via a GETIME macro and the LCB at the top of the Time Delay queue is handled as previously described. If no LCB is in the Time Delay queue, exit is made to the Qdispatch subroutine for handling of the next item on the Ready Queue.

BRB-RING ROUTINE (CHART 3)

This routine constructs the BRB ring used to send or receive a message, and begins initialization of a CCW in each BRB. The BRBs are drawn from the element chain of the Inactive BRB queue generated on expansion of the BUFFER macro instruction. The routine attempts to form a ring containing the number of BRBs specified in the BUFNO parameter of the DTFQT macro instruction.

When the routine is entered, register 9 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 QSVCTSTQ in the QTAM Wait 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.

Because 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. 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 complete, 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 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 - X'EC'

   SEND - X'E0'

2. Into the LCB (LOPC field) is inserted an operation-type code for subsequent use by the Physical I/O module:

   RECEIVE - 1 (Read Initial)

   SEND - 2 (Write Initial)

3. Register 2 is initialized for the QCB for the queue to which the first BRB/CCW is to be posted:

   RECEIVE - Active-Buffer-Request Queue

   SEND - Disk Input/Output Queue (for send operations, additional initialization consists of setting a code of X'09' in the RSTA field of the BRB and inserting the relative record number for the first segment of the message in the RNSA field).

A test is made to determine if a message is to be received from a 2260 Local terminal. If not, the routine causes the first BRB/CCW in the ring to be posted to the appropriate queue by exiting to the Post subroutine in the QTAM SVC/Subtask Control routine.

If a message is to be received from a 2260 Local terminal, additional procedures are performed. The entire ring of BRBs is linked to the top of the Ready Queue. Into each BRB is placed a high priority (X'EC') and the address of the Active Buffer Request queue. Exit is made to the Qdispatch subroutine in the QTAM nucleus to cause requests for all needed buffers to be initiated. This action is necessary because all required buffers must be assigned before the Read operation (EXCP) is initiated in the QTAM Physical I/O module.

ACTIVE-BUFFER-REQUEST ROUTINE (CHART 4)

This routine is entered at IJLQIP81+6 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 queued 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 queued on the element chain of the Active Buffer-Request queue.

AVAILABLE-BUFFER ROUTINE (CHART 5)

This routine is entered at BFRREQ 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 active BRB. If no BRB is active, 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 active, this routine branches to the Buffer-BRB routine.

BUFFER-BRB ROUTINE (CHART 6)

This routine is entered at BFRSCH+6 from either the Active Buffer Request routine or the Available Buffer routine. Upon entry, register 6 contains the address of an active BRB and register 1 contains the address of an available buffer.

This routine makes the appropriate disposition of the buffer depending upon the status of the BRB. The BRB may be associated with any one of three operations.

1. The BRB is associated with a read-from-disk operation: This is an indication that the BRB was previously in the Disk I/O Queue, but that a buffer was not available at that time. This routine effectively (but not via an SVC) posts the BRB to the Disk I/O Queue, and exits to the QTAM SVC/Subtask Control routine at QSVCLIFO to cause the buffer to be posted to the Available Buffer Queue. This action ensures that the buffer will now be available for the read-from-disk operation.

2. The BRB is associated with a receive-from-line operation: After determining that the line is still available for receiving, the buffer is assigned to the line and placed on the LPS Queue via an exit to the Interim-LPS routine. If the line is not available for receiving, this routine exits to the Available Buffer routine in an attempt to assign the buffer elsewhere.

3. The BRB is associated with a PUT operation: This case arises when a PUT module in a message processing program has requested a buffer via a QPOST or QWAIT SVC. This routine passes the buffer to a special section of the PUT module, which places the data into the buffer while executing in supervisor state (movement of the data must be effected in supervisor state so as not to violate storage protection).

INTERIM LPS ROUTINE (CHART 7)

This routine is entered when the Interim LPS QCB (IJLQIP76) appears at the top of the Ready queue. This routine provides the means of delaying the processing of all buffers until all BRBs are processed when a PCI interruption is missed because of extended CPU disable time.

When entered the routine immediately posts the buffer passed to it to the LPS queue. This causes the LPS Control routine, which is waiting for a buffer, to be entered for processing of the buffer.

QMOVER ROUTINE (CHART 9)

This routine is entered at IJLQIP80+6 upon activation of the Qmover subtask. This subtask is activated when the Qmove QCB appears at the top of the Ready Queue.

The Qmover routine moves data from the Foreground-2 or Background partition into the Foreground-1 partition, while operating under the storage protection key of the supervisor. The routines associated with the CHNGT, CHNGP, COPYC, and CLOSEMC macro instructions require this function. The function is requested by posting the Qmove QCB to itself via a QPOST (SVC 31).

The routine moves the desired data per parameters passed by the requesting routine:

1. register 3 contains the length of the data to be moved;

2. register 4 contains the address of the F1 area into which the data is to be moved; and

3. register 5 contains the address of the data to be moved.

After the data is moved, this routine exits to the Qdispatch subroutine for handling of the next item on the Ready Queue. The next item may be the full STCB representing reentry to the requesting program.

Note: This routine is used in the same way in a one-partition processing environment (i.e., with multitasking).

DASD DESTINATION ROUTINE (CHART 10)

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 to be written on the disk.

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 is 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 11)

This routine is entered when a buffer has been returned from a GET operation or when a disk read from a DASD-process queue has been completed. The routine makes three tests to determine whether the message processing program is ready to accept another segment. If

1. there is no message segment in the DASD process queue,

2. there are too many buffers in the main storage process queue for the processing program to handle, or

3. a segment is currently being read from the DASD process queue.

No further disk read can be initiated, and this routine exits to the Qdispatch routine.

If none of the three conditions exits, the routine initiates a disk read from the DASD process queue.

RETURN-BUFFER ROUTINE (CHART 12)

This routine returns a buffer from the main storage process queue and exits to the GET-Scheduler routine to allow resumption of disk reads from the DASD process queue.

END INSERT ROUTINE (CHART 13)

This routine is entered by a branch and link from the End of Address, Conversational Mode, or Distribution List routine to enter the address of these routines in a chain of routines to be executed according to the specified priority by the Buffer Cleanup Routine.

The 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 the highest priority routine, the priority of the calling routine is compared with the next priority routine in the chain until the priority is high.

When the priority is higher than the priority of the one already 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 point 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.

LPS CONTROL ROUTINE (CHART 15)

This routine is entered at IJLQIP20 from the expansion of the ENDREADY macro instruction during initialization of the telecommunications system. Thereafter, it is entered at the same entry point from numerous implementation routines. The most common entries are from:

1. the Physical I/O module after a line I/O operation has been initiated, and

2. the Free BRB routine after the buffers and BRBs for a previous operation have been released and the line freed.

3. the ARU-Internal and ARU-Receive routines (packaged in the Audio Line Appendage module) after any ARU/LPS processing has been completed for a line.

When entered, the routine issues an immediate SVC 30 to QWAIT for the next item on the LPS queue. The item returned by the SVC may be any of the following:

1. An available buffer into which a message segment is to be read.

2. A buffer containing a header or text segment, that is, a message-filled buffer, that has been read or is to be written.

3. A special ECB representing a request to start a disk I/O operation.

4. A special ECB representing a request to set the timer for a polling interval.

5. A Pause BRB/CCW

6. An audio line control block into which an input message is to be processed by the user's LPS routines.

7. An audio line control block requesting an I/O operation.

8. A 7772 DCV buffer requesting a disk read operation from the 7772 DCV Vocabulary file.

9. A special ECB representing a request to stop a specified audio line.

10. A special ECB requesting a Halt I/O on a dial line.

When an available "first" buffer is encountered, the routine exits to the Activate routine (in module IJLQRW) at IJLQRW to cause receipt of the message to be initiated. When a full buffer or the last buffer is encountered, the routine exits to the beginning (LPSTART expansion) of the user's LPS section. Prior to entering either the Activate routine or theuser's LPS, the registers are initialized as follows:

Register 4--address of the appropriate LCB.

Register 5--the scan pointer is set for header analysis or for data address in read initial operation.

Register 6--address of the buffer.

Register 7--address of the beginning instruction in the user's LPS section.

Register 8--address of the appropriate terminal table entry.

Register 9--end-of-segment address.

If the item is a request to start a disk I/O operation, the routine loads the address of the CCB for disk operations into register 1 and issues an EXCP (SVC 0). The necessary channel program was previously built by the Disk I/O routine which requested that the I/O operation be started.

For a polling interval request, the address of the QTAM timer interrupt handling routine (labeled TIMEEXIT) is first established via a STXIT macro. Then a SETIME (SVC 10) is issued to set the timer for the interval requested by the End of Poll Time Delay routine.

After either a disk I/O operation has been initiated or the timer set, control is returned to the beginning of this routine (at IJLQIP20) to QWAIT for the next item on the LPS queue.

If the item is a Pause BRB/CCW, exit is made to the Pause routine.

When an audio input message is to be processed, the routine exits to the beginning (LPSTART expansion) of the user's ARU/LPS section. Before this user's section is entered the registers are initialized as follows:

Register 4--address of the appropriate ALCB.

Register 7--address of the beginning instruction in the user's ARU/LPS section.

If an audio line control block requests an I/O operation, the channel program is ready and an EXCP (SVC 0) is immediately issued on the line. Control returns to the beginning of this routine (at IJLQIP20).

If a 7772 DCV buffer requests a disk read operation, the channel program is ready in the buffer itself, and an EXCP is immediately issued to read a DCV word. Control returns to the beginning of this routine (at IJLQIP20).

If a special ECB representing a request to stop an audio line is encountered, the CCB address is extracted from the ALCB and a Halt I/O is issued through an SVC 27. Control returns to the beginning of this routine (at IJLQIP20).

BUFFER RECALL/CLEANUP ROUTINE (CHART 16, 17)

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 IJLQIP23 through the calling sequence generated by a POSTSEND, ENDRCV, or POSTRCV macro instruction. The recall function is performed when entry is at IJLQIP22 through the calling sequence from the Cancel Message, EOBLC, Error Message, Reroute Message, EOA, or Distribution List routines. The difference between the two entry paths is that in the second case the recall flag is set on in the LCB (LSTA = 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 31. (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. Pause BRB/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) is found, the LPS Control routine branches back into the Buffer Recall/Cleanup 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 the direct-access storage, obtains the segment being recalled, and exits to the calling routine.

When the cleanup operations performed by this routine are finished, the routine exits to the Free-BRB routine. The Free-BRB routine completes the cleanup operation by releasing all BRBs in the ring to the Inactive-BRB queue and freeing the line for another operation.

FREE-BRB ROUTINE (CHART 18)

This routine is entered at LINEFREE from the Buffer Recall/Cleanup routine. Its function is to complete cleanup operations by releasing certain resources so they will be available for their next usage.

The routine examines each BRB in the ring of BRBs passed to determine its disposition. If the BRB is not currently in the Active BRB queue, the BRB is released to the Inactive BRB queue through an SVC 31. If the BRB is in the Active BRB queue (this is an indication that a request for a buffer is still pending), the BRB is not released. Instead, it is bypassed and the next BRB in the ring is examined. Any BRB bypassed in this manner will be subsequently released by the Buffer-BRB routine.

After all BRBs in the ring have been released (or bypassed), the routine posts the LCB to itself through an SVC 31. This is the standard technique for returning a line to the free condition. Exit is made to the LPS Control routine at IJLQIP20.

DISK I/O ROUTINE (CHART DC)

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) onto the element chain of the Disk Input/Output queue. The relative record number used by QTAM is converted into a relative track address. Then, the relative track address is converted into an actual DASD address. The routine then posts a special element to the LPS queue. After gaining control, the LPS Control routine issues the EXCP to write on or read from disk.

When the checkpoint element is on the Disk I/O queue, the routine links all subsequent elements into the queue below the checkpoint element. When the checkpoint element reaches the top of the queue, the routine places the Disk I/O queue on the top of the Ready Queue and posts the checkpoint element to the checkpoint queue.

DISK-END APPENDAGE (CHARTS DA AND DB)

This routine is an I/O appendage, entered from the I/O Supervisor following a DASD read or write operation. If a receive operation over a line is taking place and the interrupt is from writing a buffer just received onto the disk, the routine routes the empty buffer required for the next segment to be received to the Available Buffer queue. If a send operation over a line is taking place and the interrupt is from reading a buffer to be sent from the disk, 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 I/O queue. If the interrupt is from writing a checkpoint record on the disk, the routine gets the address of the checkpoint element and the checkpoint queue, and posts the element to the queue to activate the checkpoint subtask.

IBM 7772 DISK-END APPENDAGE (CHART D1)

This routine is an I/O appendage entered from the I/O Supervisor following a disk read operation on the 7772 DCV Vocabulary file. The routine stores the DCV word length in the ALCB associated with the 7772 DCV buffer, and branches to the 7772 Line Write routine.

IBM 7772 DISK READ ROUTINE (CHART D2)

This routine analyzes the user-provided address chain located in the address-chain buffer of the ALCB to obtain the characteristics of the next DCV word to be sent.

It is entered at IJLQAD30 from the ARU-Send routine when a DCV buffer has beenallocated to the ALCB of a 7772 line waiting for an audio answer. It is reentered at the same point from the Audio Line PCI routine each time the

transmissionof a DCV word begins and the next DCV word must be prepared. It is also reentered from the 7772 DCV Buffer routine when a DCV buffer is available for a waiting 7772 line.

When a DCV word is to be read from the Vocabulary file, the address of the buffer part in which it will be read is stored in the ALCB. The routine is reentered at REREAD to check the presence of a pause or a repeat element in the address chain. Then, the DCV buffer representing a read disk request is posted to the LPS queue via a branch to the Post subroutine.

When a DCV word is already in main storage in the word table, the length and address of this word are stored in the ALCB. The routine is reentered at REREAD to check the presence of a pause or a repeat element in the address chain. Then, a branch is made to the 7772 Line Write routine.

If a pause element is encountered after a word address, the pause count is saved in the AICB and the routine is reentered at REREAD to check the possible presence of a repeat element. If a repeat element is encountered after a word address or a pause element, the repeat count is decremented and the address chain pointer is reinitialized at the beginning of the address chain.

IBM 7772 LINE WRITE ROUTINE (CHART D3)

This routine is entered at IJLQAD60 from the 7772 Disk-End Appendage and from the 7772 Disk Read routine when a write (or disable) operation must be requested or continued on a 7772 line.

If the first DCV word is to be sent, the channel program in the 7772 ALCB is updated and this ALCB, representing a line write request, is posted to the LPS queue via a branch to the POST subroutine.

When one of the intermediate DCV words is to be sent, the channel program is updated to allow the construction of the audio transmission, and a branch is made to the QDISPATCH subroutine.

If the last DCV word is to be sent, the channel program is updated, and the last write command is chained on either a disable command (if the line is operating in information or inquiry mode) or a read command (if the line is operating in conversation mode). Then, exit is made to the Qdispatch subroutine.

IBM 7772 DCV-BUFFER ROUTINE (CHART D4)

This routine is entered at IJLQAD40+6 on activation of the 7772 DCV buffer subtask when the Audio Line Appendage posts an available DCV buffer to the 7772 DCV buffer queue. This DCV buffer is allocated to the first ALCB dequeued from the ALCB waiting chain located in the DCV buffer queue. The address of the Queue Insert subtask (IJLQIP8B) is placed in the subtask address field of the DCV buffer queue only when there is no more ALCB waiting for a DCV Buffer. In any case, the routine branches to the 7772 Disk Read routine.

GENERAL FLOW

Four routines are provided to execute the nonaudio line I/O functions: the Activate and Channel Program Generator routines in the QTAM Physical I/O module and the Line PCI and Line End routines in the Line Appendage module.

Five routines packaged in the Audio Line Appendage module are provided to prepare the execution of the audio line I/O functions: the ARU-Internal and ARU-Receive routines operating in problem program mode, and the ARU-Send, the 7772 Line PCI, and Audio Line End routines.

The Activate routine is passed a message-filled or empty buffer and a BRB. It creates a CCW in the BRB and sets up pointers for the Channel Program Generator routine for a read or write initial operation or adds the buffer to an already initiated operation.

The Line PCI routine releases a previously filled buffer to the LPS queue or an emptied buffer to the Available Buffer queue and places the associated BRB on the Active BRB queue for a new buffer, preparing it for its next use.

The Line End routine provides many functions. On successful completion of an I/O operation, it routes the last emptied buffer to the LPS queue and restarts I/O for the LRC check, or it routes the last filled buffer to the LPS queue and goes to the LPS Control routine. For an error completion, the error condition is posted in the error-halfword of the LCB, and the LPS Control routine is entered. For a program check condition (an indication that the next BRB does not have a buffer assigned), a buffer is obtained through the PCI routine and the I/O is restarted. For a negative response to a poll operation, the polling list pointer is updated and the I/O restarted.

The ARU-Internal routine posts to the ARU-Send queue an ALCB, which contains a user-provided address chain.

The ARU-Receive routine posts to an MS-Process queue an ALCB, which contains an input message previously processed by the user's ARU/LPS routines.

The ARU-Send routine, according to the type of ALCB, performs the following functions. For a 7770 ALCB, it prepares the channel program to send an audio answer, and posts the ALCB representing a Start I/O request to the LPS queue. For a 7772 ALCB, it requires a DCV buffer, if necessary, before control is passed to the 7772 Disk Read routine for exploitation of the user-provided address chain.

The 7772 Line PCI routine, according to the type of the associated channel command, performs the following functions. When a PCI occurs on a Write command, the 7772 ALCB is routed to the 7772 Disk Read routine. When a PCI occurs on a Read command, the routine releases, if necessary, the DCV buffer associated with the ALCB to the corresponding DCV buffer queue.

The Audio Line End routine performs several functions. On completion of a 7772 enable operation, it prepares an invitational or informational channel program and posts the ALCB to the ARU-Send queue. On completion of an input operation, it posts the ALCB to the LPS queue to process the input message. On completion of an output operation, it updates the ALCB channel program according to initial conditions, posts the ALCB representing a Start I/O request to the LPS queue, and, if necessary, releases a 7772 DCV buffer to the corresponding DCV buffer queue.

QTAM PHYSICAL INPUT/OUTPUT MODULE

Module Name: IJLQRW (Chart RW, RX, RY)
This module consists of two routines:

1. the Activate routine, and
2. The Channel Program Generator (CPG) routine

These routines are discussed in the following paragraphs.

ACTIVATE ROUTINE (CHART RW)

Entry Point: This routine is entered at IJLQRW by the LPS Control routine, the EOB or EOBLC routine, and the expansions of the ENDSEND and POSTSEND macro instructions. Upon entry, it is expected that the register contents are:

Register 4 (IJLQLCBR) contains the LCB address.

Register 6 (IJLQBFRR) contains a buffer address.

Register 7 (IJLQLPSR) contains a user LPS routine address.

Register 8 (IJLQTBLR) contains the terminal table entry address.

Register 9 (IJLQEOSR) contains the end of segment address.

These registers plus registers 0, 13, and 15 are preserved by this routine; the others are destroyed.

Function: The Activate routine creates a channel program in a BRB and/or performs initialization functions for the CPG routine. It then exits by a branch to the CPG routine to initiate the I/O operation, or by a branch to the LPS Control routine, at IJLQIP20 in module IJLQIP, when a buffer has been added to a previously started line I/O operation.

In operation, the BRB to which the buffer is to be assigned is located from the LCCW field in the LCB. The chain field of the BRB to which LCCW points addresses the current BRB. For a "first" BRB, LCCW points to LPCI-9 (a pseudo BRB), and LPCI (the chain field of this pseudo BRB) is initialized to the address of the first real BRB. The buffer text address is placed in the BRB/CCW data field and a read or write operation code is inserted in the command code field. The handling thereafter depends upon the function of the BRB.

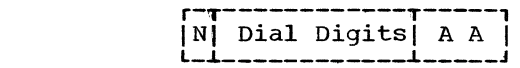
First BRB

The BRB may be a "first" BRB for receiving or sending. For this BRB, the channel program must be started through the Channel Program Generator routine. This includes an initial read or write operation and a read or write continue operation after an EOB.

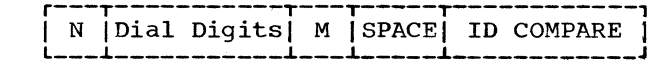
The BRB is set up to perform the I/O operation but is not used for it. The channel program is generated by the Channel Program Generator routine. For an initial operation, the data address passed to the CPG routine is the message header portion of the buffer. The count is the end of the segment minus the beginning address.

For a write initial, register 1 is set to point to the addressing characters (nonswitched terminal) or dial digits (switched terminal), or CCB/ECB (IBM 2260 Local) which are in the terminal table entry. If it is a nonswitched line, the device access area format is:

where A = addressing character and P = polling character. For a switched 1050 the format is:



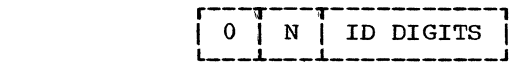
where N = the number of dial digits. For a TWX terminal, the format is:



where M = the length of the TWX terminal identification sequence. The dial digits are sent to address the TWX terminal, and the responding ID sequence is read into the SPACE area and compared against the ID COMPARE sequence (the compare is performed in the Line End routine). For an IBM 2260 or 1053 Local terminal, the device access area consists of a 16-byte CCB followed by an eight-byte ECB.

For a read initial on a nonswitched remote polled line, register 1 is set to point to the current polling list entry. The polling list entry is a two-byte offset to the terminal entry in the terminal table which can be accessed to obtain the polling characters. The last two bytes in the poll list contain binary zeros. For a switched 1050 line, the polling characters are in the polling list. The format is:

The zero byte specifies an answer list. For a TWX line the format of the polling list is:



For an IBM 2260 Local, register 1 is set to point to the CCB in the terminal table entry.

For a read initial on an autopolled line, the polling characters are in the polling list which has the following format:

where PP are the polling characters (one for an IBM 1030, two for other terminals), and I is the index byte used to identify the terminal in the polling list.

For a read initial, idle characters are placed in the field reserved at the beginning of the message area in the buffer, as specified in the LPSTART macro. The incoming message is temporarily routed to the error queue. The channel program is then generated. For a switched line for which connection has previously been established (the "Don't Dial" flag is on in the LCB), a write conversational channel program is specified.

For a write initial or write conversational, the status of the terminal is checked before entering the CPG routine. If the terminal is not receiving, the error is posted and the user's LPS routine is entered via register IJLQLPSR.

A read or write continue is requested by the EOB or EOBLC routine. In this case, the address and count for the data transfer is modified by the EOB offset field in the LCB to point to the position after the EOB character.

Non-First BRB

A "non-first" BRB is associated with an I/O operation which was started previous to the assignment of this buffer to this BRB. The address of the buffer text area and the count are placed in the BRB/CCW. The PCI and CD flags are set to one in the CCW. If it is the last segment of a message, the CD flag is turned off. The operation with which this BRB is associated:

1. may be currently in progress,
2. may be on an IBM 2260 Local read operation and thus not yet have been started, or
3. may have been terminated with a channel program check (invalid TIC address), indicating that a buffer was not supplied in time.

In cases 1 and 2, the TIC in the previous BRB is made valid. If the previous BRB was a "first" BRB, the channel program is chained to this BRB. (If it was not a "first" BRB, the two BRBs are already chained.) The BRB status is tested for an indication of the last BRB in the ring for an IBM 2260 Local read operation. If this is not the case, a branch is made to the LPS control routine. If this is the case, a branch is made to start the channel program before returning to the LPS Control routine.

In the case of a channel program check, the BRB address is moved into the CCB channel program address area, EXCP is issued to restart the I/O, and a branch is made to the LPS control routine.

Pause BRB

The Pause BRBs are special BRBs that send fill characters as specified by the user in a PAUSE macro. The first Pause BRB sends text, if there is text, as well as fill characters. The PAUSE routine creates CCWs with address and count fields. The count field for the first BRB/CCW in the chain is computed from the beginning of text address and the address in the next BRB/CCW (filled in by the PAUSE routine). All succeeding Pause BRBs are skipped over.

CHANNEL PROGRAM GENERATOR ROUTINE (CHARTS RX, RY)

Entry Point: This routine is entered at RW1CPG01 from the Activate routine. The address and count for the read or write data CCWs are passed from the Activate routine. The polling or addressing list address (or dial digits address) is passed in register 1 from the Activate routine. The count for the polling or addressing CCW is contained in the model CCW. The dial or TWX count is in the polling list passed.

Function: This routine creates a channel program in the CCB and issues EXCP to start the I/O. The channel programs created are Read/Write Initial, Read/Write Continue, Write Conversational, and Read Repeat. For initial contact on a switched line, the Answer or Call segment of the channel program is also generated. Exit from this routine is made by a branch to the LPS control routine (IJLQIP20 in module IJLQIP). This occurs when the end of the polling list is encountered or after the start of the I/O operation. It permits the initiation of I/O on other lines and concurrent processing of messages in the LPS sections or message processing programs, if any.

In operation, the desired channel program is created from model channel programs contained in a device I/O module.

The special characters for the device are also contained in the device I/C module. The address of the Device I/O module for a line is in the DTF table.

A model channel program consists of the skeleton information for every CCW needed to perform a given operation. The information includes the operation code, flag byte, TP operation code, and a count byte which may or may not be used.

Eleven subroutines in the CPG routine create CCWs:

1. Model Expander 1 (RW1SG101): inserts the address and count fields into the read response to polling, disable and enable CCWs.

2. Model Expander 2 (RW1SG201): inserts the address and count fields into read or write data CCWs.

3. Model Expander 3 (RW1SG301): inserts the address and count fields into read response to addressing CCWs.

4. Model Expander 4 (RW1SG401): inserts the address and count fields into special character CCWs.

5. Model Expander 5 (RW1SG501): inserts the addressing and count fields into write polling/addressing characters CCWs, dial CCWs or Read or Write TWX ID CCWs.

6. Model Expander 6 (RW1SG601): inserts the address and count fields into the poll CCWs.

7. Model Expander 7 (RW1SG701): inserts the address field into the transfer-in-channel CCWs for automatic polling (Auto Poll).

8. Model Expander 8 (RW1SG801): inserts the address of the sense field (IJLQLSEN) of the LCB into the Sense CCW of the Read Initial channel program.

9. Model Expander 9 (RW1SG901): inserts the addressing and count fields into the Write CPU-ID and Read Terminal-ID CCWs. Moreover, in the case of the Write CPU-ID CCW, this expander checks whether WRU=YES is specified in the DTFQT macro instruction.

   a. If WRU=YES is specified, the Command Chaining flag is set on, the next CCW (Write WRU) is executed, and an identification sequence exchange is performed.

   b. If WRU=YES is not specified, the Command Chaining flag remains off. In this case, the computer identification is sent to the terminal, but no identification sequence exchange is performed.

10. Model Expander 10 (RW1SGA01): is associated with the TIC command of the Write Initial channel program to insert the transfer address into this TIC command. This transfer address is:

    a. the address of CCW no. 3 (Write CPU-ID) if the WRU macro instruction is present in the Send Header subgroup of the LPS; or

    b. the address of CCW no. 6 (Write Area) if the WRU macro instruction is not present in the Send Header subgroup of the LPS.

11. Model expander 11 (RW1SGB01): inserts the address of the letter shift preceding the mark character string into the address field of the CCW. For a line defined as an output-only line, this subroutine adds to the CCW count the number of mark characters to be sent.

If an initial operation is requested on a switched line, a call or answer segment channel program is generated. The read or write initial segment is generated following the call or answer segment.

If a read initial operation is requested on a nonswitched line, the polling list is scanned until an entry is found for a terminal that is in a receiving status. If the end of the polling list is encountered, the buffer (empty) is posted to the LPS queue, and an exit is made to the LPS Control routine.

After the operation segment of the channel program has been generated, a TIC to the second BRB in the ring is placed after the last CCW in the CCB. If a buffer has not been requested for the second BRB, the PCI flag is set in the last CCW in the CCB. EXCP is issued to start the I/O and exit is made to the LPS Control routine.

Channel program generation for an IBM 2260 Local network differs from remote operation in two respects. First, because this is a contention device, polling is unnecessary. Instead, the terminal requesting attention is identified via the CCB for the line, which is included in the terminal table entry. Second, in order to efficiently handle the higher data rate of a local device, the channel program for

receive is not started until all buffers are assigned.

For a write initial, the CCB for the line is accessed in the terminal table entry. An SVC 25 is issued to remove the CCB from the DOS channel queue. The appropriate channel program (as specified by the WRT60 operand of the MODE macro) is generated and started in the same manner as for remote devices. No addressing is required because the device is connected directly to the channel. The supervisor automatically restarts any write that encounters a busy device.

For a read initial, the CCB in the terminal table is accessed for generation of the channel program. The appropriate read CCW (as specified by the RTYPE operand in the DTFQT macro instruction) is generated. The PCI flag is turned off. The channel program is not started at this time. Instead, control returns to the LPS Control routine to continue assigning buffers for the read.

CHANNEL PROGRAMS

The device I/O modules contain the model channel programs which the CPG routine uses to build the executable channel programs. There is one device I/O module for each terminal type supported. See Figure 6.

QTAM frequently places a TP operation code into bits 40-47 of a CCW. These codes are used to determine the action to be taken when an interrupt occurs. The TP operation codes used are contained in Figure 7.

Module Name	Model Channel Programs For:
IJLQM0	IBM 1030
IJLQM1	IBM 1060
IJLQM2	IBM 2260 Remote (2848)
IJLQM3	AT&T 83B3
IJLQM4	WU Plan 115A
IJLQM5	IBM 1050 Switched/Nonswitched
IJLQM6	IBM 1050 Nonswitched
IJLQM8	AT&T TWX (Models 33/35)
IJLQM9	IBM 2260 Local
IJLQN0	IBM 2740 (Basic)
IJLQN1	IBM 2740 (Basic/Dial)
IJLQN2	IBM 2740 (Station Control)
IJLQN3	IBM 2740 (Station Control and Checking)
IJLQN4	IBM 2740 (Dial, Transmit Control and Checking)
IJLQN5	IBM 2740 (Checking)
IJLQN6	IBM 2740 (Dial and Checking)
IJLQN7	IBM 2740 (Dial and Transmit Control)
IJLQN8	World Trade telegraph terminal

Figure 6. Device I/O Modules

Code	Meaning
X'00'	Not Used
X'01'	Read Response to Write polling characters (POLL-RESTART).
X'02'	Read Response to Write addressing characters (MULTI-ADDR).
X'03'	Read or write data
X'04'	RE-START after Enable.
X'05'	TWX or WITA identification check
X'06'	Read with skip.
X'07'	Read response to Write text (IBM).
X'08'	Read text in buffer.
X'09'	RE-SET.
X'0A'	Write Break (contention on WTTA lines)

Figure 7. TP Operation Codes

Each device I/O module contains a table of offsets at the beginning that defines the valid operations for the device. Each byte in the table contains the offset to the model channel program for the specified operation. The offsets and the corresponding model channel programs are defined in Figure 8. If the operation is invalid for the device, the byte contains X'FF'.

Offset Code	Operation
X'00'	Autopoll Read Initial
X'01'	Read Initial
X'02'	Write Initial
X'03'	Read Continue
X'04'	Write Continue
X'05'	Read Short
X'06'	Write Conversational
X'07'	Read Repeat
X'08'	Answer Segment
X'09'	Call Segment
X'0A'	Special Characters
X'0B'	Write Erase
X'0C'	Write At Line Address

Figure 8. Table of Offsets to Model Channel Programs

In the following channel operation descriptions,

Table refers to the special character table in the device I/O module

List refers to the polling, addressing, dial, or TWX ID list passed by the Activate routine,

Area and Length are the address and count fields passed by the Activate routine.

IBM 1030--Terminal to CPU

Read Initial:

Command	Operation	Address	Flags	TP Code	Count
1	Write © © ©	Table	CD,SLI	0	3
2	Write polling characters	List	CC,SLI	0	1
3	Read response character	Area	CD	Polling Restart	2
4	Read message block	Area+2	CD	0	length-2

Command 1: Sends three C s to set all terminals on the line in control mode.

Command 2: Polls a device by sending its polling characters.

Command 3: The effect of the command is as follows:

1. A negative response (only one character received) causes an interruption with a wrong-length indication. The Line End routine detects this condition and initializes the channel program to poll the device specified by the next entry in the polling list. Control is then returned to the Supervisor for execution of commands 2 through 4. This procedure continues until (a) a positive response is received, or (b) the last nonskipped entry in the polling list has been polled.

2. On positive response, the response character and the first byte of text are read. Data chaining then takes effect to Command 4.

Command 4: Reads text until the count is exhausted and chains through a TIC placed after it to the second BRB/CCW in the ring. If a buffer was not requested for the second BRB/CCW, the PCI flag would cause an interrupt on Command 4 and the PCI routine would be entered to obtain a buffer. If the buffer had been requested but not assigned, the TIC command would have an invalid address that would cause a program check interrupt. The Line End routine would recognize this condition and obtain the buffer.

The preceding example is the channel program generated for a read-initial operation to poll the IBM 1030. The channel program for polling the other terminals supported may be slightly different, but the polling technique is the same. Note that the IBM 1030 uses only one polling character in Command 2; therefore, the count is 1.

When an IBM 1031 Input Station is polled, each input unit of the station takes part in sending the message. The 1031 determines the order in which these input units will be contacted. The input units of the 1031 are the card reader, the badge reader, the manual control unit, and the cartridge reader.

Auto Poll Read Initial:

Command	Operation	Address	Flags	TP Code	Count
1	Write © © ©	Table	CC,SLI	0	3
2	Poll	Any entry in polling list	CC,SLI	0	k(n)
3	TIC	2nd POLL Command	SLI	0	1
4	TIC	Read Response			
5	Poll	1st entry in polling list	CC,SLI	0	k(n)
6	TIC	2nd POLL Command	SLI	0	1
7	Read Response	Area	CD	1	2
8	Read Message Block	Area+1	CD	3	Length-2

Command 1: Sends three © 's to set all terminals on the line in control mode.

Command 2: Starts polling of the terminals on the line, beginning at any entry in the polling list.

k = 2 for IBM 1030, k = 3 for other devices.

n = number of terminal entries between the starting entry and the end of the polling list.

Command 3: On a negative response to polling at the end of the polling list, this TIC command will be executed to start the second POLL command.

Command 4: On a positive response to polling, this TIC command will be executed to start the READ response command.

Command 5: Starts polling of the terminals on the line, beginning at the top of the polling list.

k = 2 for IBM 1030, k = 3 for other devices.

n = total number of entries in the polling list.

Command 6: On a negative response to polling at the end of the polling list, this TIC command will be executed to restart previous (2nd) POLL command.

Command 7: Reads the index byte and the first byte of text in the first two bytes of the buffer area. Data chaining then takes effect to command 8.

Command 8: Reads text until the count is exhausted and chains through a TIC placed after it to the second BRB/CCW in the ring. If a buffer was not requested for the second BRB/CCW, the PCI flag causes an interrupt on command 8 and the PCI routine is entered to obtain a buffer. If the buffer was requested but not assigned, the TIC command has an invalid address which causes a program check interrupt. The Line End routine recognizes this condition and obtains the buffer.

The preceding example is the channel program generated for an Auto Poll read initial operation to poll an IBM 1030. The channel program for polling the other terminals supported is the same.

Read Continue: Consists of a Write (Y) plus a read initial operation. A read continue operation is sent after an EOB character has been encountered during a read operation. It restarts polling on the line.

Read Repeat: Consists of a Write (N) plus a read initial operation. Read Repeat is issued after an EOB character has been encountered during a read operation, EOBLC is specified, and an error in transmission occurred. It sends a negative response (N) and chains to a read initial sequence which restarts polling.

IBM 1030--CPU to Terminal

Each 1033 printer on a line is addressed with a single addressing character obtained from a terminal table entry.

Write Initial: A write-initial operation addresses and then sends a message to the 1033 printer.

Operation	Address	Flags	TP Code	Count
1. Write Ⓒ Ⓒ Ⓒ Ⓢ	Table	CD,SLI	0	4
2. Write addressing character	List	CD	0	1
3. Write 1	Table	CC,SLI	0	2
4. Read response to addressing	Area	CC	0	1
5. Write Data	Area	CD	Read data	Length

C Places the line in control mode.

S Indicates to the 1031A control stations that the address specifies a 1033 printer.

1 Conditions the 1031A to answer.

The 1031A replies with either Ⓨ (ready) or Ⓝ (not ready). Command chaining occurs only on the receipt of Ⓨ. On negative response to addressing, the operation is terminated. On positive response, Command 4 chains to Command 5 to begin the data transfer.

Write Continue: Same as step 5 of the write-initial operation. A Write Continue is requested after an EOB character when sending.

IBM 1060--Terminal to CPU

Each IBM 1062 Teller Terminal is polled with a two-character code. The first of these two characters identifies the IBM 1061 Teller Terminal (one or two per 1061).

Read Initial: Used to begin polling on the line. The channel program is:

Operation	Address	Flags	TP Code	Count
1. Write Ⓒ Ⓒ Ⓒ	Table	CD,SLI	0	3
2. Write polling characters	List	CC,SLI	0	2
3. Read response character	Area	CD	Poll-restart	2
4. Read message block	Area+2	CD	0	Length-2

Auto Poll Read Initial: Used to begin polling on the Auto Poll line. The channel program is:

Command	Operation	Address	Flags	TP Code	Count
1	Write Ⓒ Ⓒ Ⓒ	Table	CC,SLI	0	3
2	Poll	Any entry in polling list	CC,SLI	0	k(n)
3	TIC	2nd POLL Command	SLI	0	1
4	TIC	Read Response			
5	Poll	1st entry in polling list	CC,SLI	0	k(n)
6	TIC	2nd POLL Command	SLI	0	1
7	Read Response	Area	CD	1	2
8	Read Message Block	Area+1	CD	3	Length-2

Read Continue [=Write Ⓨ + Read Initial]: Read continue is issued after an EOB is encountered during a successful read-initial operation to execute a single command to send a positive response Ⓨ to the 1061 and chain to a read-initial sequence to restart polling on the line.

Read Repeat [=Write Ⓝ + Read Initial]: Read Repeat is issued after a Read operation is terminated by an incoming EOB and the block was received in error. The channel program writes the negative response Ⓝ to the 1061, then chains to a read-initial sequence to restart polling on the line. Read Repeat is performed only when the EOBLC macro is specified.

IBM 1060--CPU to Terminal

Each 1062 on a line is addressed with a two-character code. The first of the two addressing characters identifies the 1061 control unit. The second character identifies the 1062 terminal.

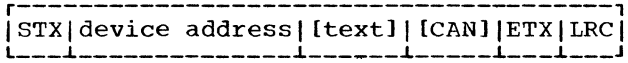
Write Initial: A write-initial channel program is issued to address and send a message to a 1062. The channel program is:

Operation	Address	Flags	TP Code	Count
1. Write ⓒ ⓒ ⓒ	Table	CD,SLI	0	3
2. Write addressing character	List	CC,SLI	0	2
3. Read response to addressing	Area	CC	0	1
4. Write message block	Area	CD	0	Length

IBM 2260 Remote (2848)

PROGRAMMING INFORMATION

1. The format of a message received from a 2260 through a READ is the following:



   STX: Start of text character.

   Device Address: Characters that identify the sending unit (display station or printer).

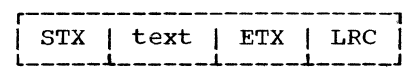
   Text: The printer status has a zero-length text.

   CAN: This character is sent only if the display control unit detects an internal operation error when transmitting the message.

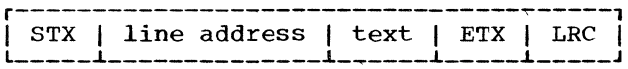
   ETX: End of text character.

   LRC: Longitudinal redundancy check character. The LRC accumulation is started by the STX character and terminated by the ETX character.

2. The format of a message sent to a 2260 through a WRITE is the following:



3. When Write-at-Line-Address is specified, the first byte of the text indicates the line address.



When further messages are sent without re-addressing, the display station interprets the first character after STX as a line address.

IBM 2260 Remote--Terminal to CPU

Each IBM 2260 is polled with a two-character code.

Read Initial (DISPLAY): A Read-initial operation is issued to start or restart polling on a line. The channel program is:

Operation	Address	Flags	TP Code	Count
1. Write EOT sequence	Table	CD	0	3
2. Write polling characters or printer request chars.	List	CD	0	2
3. Write READ MI code (2848 alert)	Table	CC,SLI	0	1
4. Read response character	Area	CD	Poll-restart	2
5. Read message	Area+2	CD	0	Length-2

The characters specified in the polling list may initiate a specific poll of a 2260 display station or a general poll of a 2848 display control unit.

Specific Poll of a Display Station: On positive response (STX), Command 4 chains to Command 5, and the message is read. On negative response (EOT), an interruption occurs. The Line End appendage routine detects this condition via the TP op code, initializes the channel program to poll the next entry within the list, and returns control to the Supervisor.

General Poll of a Display Control Unit: If the printer has a status pending as a result of a previous request (see Write Initial), this message will be transmitted. Command 4 chains to Command 5. If the printer is not ready, the display stations are scanned for a message. If a message is pending, it is sent. If the printer is not ready and no message is waiting for transmission, a negative response EOT is received. The channel program is interrupted. Line End detects this condition, updates the channel program, and returns control via the Supervisor to poll the next entry within the polling list.

Read Continue [WRITE ACK + Read Message]: READ Continue is requested from the EOB routine after a successful read-initial operation. A WRITE command sends a positive response (ACK) to the station and chains to a READ. If the previous operation was a specific poll of a display station, that station answers with EOT, which ends the operation. If the previous operation was a general poll of a display control, a message is received if one is pending; otherwise, an EOT ends the operation.

Read Repeat [WRITE NAK + READ Message] (Display Only): A Read Repeat is requested from the EOBLC routine for retransmission of a message block that is received in error. The Write command sends a negative response to the display station and chains to a read command to retry the transmission.

IBM 2260--CPU to Terminal

Two characters are used to address the IBM 2260.

Write Initial (Display or Printer):

Operation	Address	Flags	TP Code	Count
1. Write EOT sequence	Table	CD	0	3
2. Write addressing characters	List	CD	0	2
3. Write WRITE code	Table	CC,SLI	0	1
4. Read response	Area	CC	0	1
5. Write STX	Table	CD	0	1
6. Write message	Area	CD	0	Length

The EOT sequence resets the IBM 2848 Display Control address selection to control mode, nonselected status and causes all 2848s on the line to prepare to receive an addressing sequence. The addressing characters indicate a display station or a printer.

When a printer is addressed, Command 4 reads the response to the addressing sequence. A negative response (EOT or NAK) interrupts the channel program: NAK indicates that the printer is not ready ; EOT indicates that the printer is ready but the buffer is not empty. In both cases a printer request is set.

A positive response (ACK) indicates that the printer is ready and available. Command 4 chains to Command 5, and the message is sent. If a transmission error occurs, the 1053 operation is aborted as soon as the data parity error is detected, a quote symbol is printed out, and the printer buffer is cleared. EOBLC causes transmission retry by requesting a Write continue.

When a Display Station is addressed, Command 4 reads the response to the addressing sequence. That response is normally positive (ACK). Command 4 chains always to Command 5. The characters are displayed on the CRT, starting at the cursor position.

If a transmission error occurs, the transfer is retried from EOBLC by requesting a Write Continue. However, the data will be displayed starting at the cursor position, and the erroneous message will not be cleared unless the write erase option is specified in the user's LPS (MODE macro).

Write at Line Address (Display Only): Same as Write Initial, except Command 3 is the write WRITE LINE code. The MODE macro causes a Write at Line Address operation, which positions the cursor to the start of a specified line. If a transmission error occurs, sending is retried with a Write Continue. Data will be displayed starting at the same line.

Write Erase (Display Only): Same as Write Initial, except Command 3 is write ERASE code. Write Erase is specified by the MODE macro to erase the CRT before the next segment is sent.

Write Continue [=Write Message] (Display or Printer): A Write Continue is issued when an error is encountered while sending and the EOBLC (or EOB) macro has been specified. If the previous operation was a Write Initial addressing a printer, the Write Continue retries transmission of the message.

If the previous operation was a Write-at-Line-Address, or a Write-at-Line-Address followed by one or several Write Continues, the cursor is positioned at the beginning of the line specified in the message. In all other cases, the characters are displayed on the station starting at the cursor position.

IBM 2260 LOCAL (2848)

Programming Information

1. A message received from an IBM 2260 through a read consists entirely of text. The transmission code is in EBCDIC. The transmitted text consists of all characters displayed between the START and EOM symbols and excluding any characters to the right of the first NL symbol on a line. (The NL symbol itself is transmitted.) There are no line control characters for this device.

   A zero-length text is possible and can arise in the following ways:

   a. The terminal operator enters a null message (START and EOM only).

   b. The operator attempts to enter a message with no START symbol.

   c. The operator prepares a valid message, but the START symbol is erased between the time the ENTER key is depressed and QTAM issues the read.

   When a zero-length message is received, the 'zero-length message'bit (bit 4) is set in the error halfword.

2. A message sent to an IBM 2260 Local through a write consists entirely of text, except when the write-with-line-address mode is specified in the MODE macro instruction. A zero-length message is allowed and may be used with the erase/write MODE option to effect erasure of the display.

3. When write-at-line-address mode is being used, the first byte of the user header portion of the message must be the line address code. This byte is transmitted but is not displayed. Line address codes for the IBM 2260 Local terminal are shown in Figure 8A. Other values for the line address byte give unpredictable results.

4. Messages to an IBM 2260 Local which exceed the line width automatically continue on the next line, wrapping around from the last line to the first if necessary. Messages to the IBM 1053 printer should contain NL or EOM characters for carrier control.

5. The WRT60 operand of the MODE macro instruction is ignored for messages to the IBM 1053 printer. A write initial command is issued and the line address code, if present, is printed as data.

Line Address Code	Display Line Number
X'F0'	1
X'F1'	2
X'F2'	3
X'F3'	4
X'F3'	5
X'F4'	6
X'F6'	8
X'F7'	9
X'F9'	10
X'FA'	11
X'FB'	12

IBM 2260 LOCAL--TERMINAL TO CPU

Terminal to CPU transmission occurs as follows:

1. At OPEN time, or following the completion of a successful read or write, the CCB for the line is placed on the Supervisor channel queue by the IBM 2260 Local Appendage. This initializes the Supervisor to pass Attention interrupts from the terminal to the IBM 2260 Local Appendage.

2. When the terminal operator has a message to send, he keys a START symbol, followed by the message text, and then presses the ENTER key. This causes an Attention interrupt, and the Supervisor passes control to the IBM 2260 Local Appendage. This appendage removes the CCB from the channel queue and posts it to the Attention queue maintained in the LCB for the line group.

3. When QTAM recognizes the CCB on the Attention queue, a number of buffers sufficient to contain the maximum size message (as specified in the BUFNO operand of the DTFQT macro instruction) are obtained, and a channel program is built to read the entire message.

4. When all buffers have been assigned, an EXCP is issued to read the message.

Read Initial: A read initial is issued to read a message from the IBM 2260 Local. All characters between the START and EOM symbols, except characters to the right of a NL symbol, are transfered. When the message is read, the START symbol is cleared and the keyboard is unlocked. If insufficient buffers have been specified in the BUFNO operand, a partial message is received and the 'insufficient buffers' bit (bit 11) is set in the error halfword. The channel program is:

Operation	Address	Flags	TP Code	Count
Read DS MI	Area	CD,SLI	63	Length

Read Short: A read short replaces a read initial for all terminals in the line group if RTYPE=SHORT is specified in the DTFQT macro instruction. The read short option allows the user to take advantage of the faster hardware end-sequence for this command. Read short does not clear the START symbol after the message has been read, but it does unlock the keyboard. The channel program is identical to read initial except that the command code is Short Read DS MI.

IBM 2260 Local--CPU to Terminal

CPU-to-terminal transmission occurs as follows:

1. QTAM issues an SVC 25 to remove the terminal CCB from the channel queue. This causes the Supervisor to ignore any subsequent Attention interrupts from that device.

2. A write channel program is executed to write to tne IBM 2260 Local or to the IBM 1053 printer.

3. If the device is an IBM 2260 Local, the message is displayed beginning at a point determined by the WRT60 operand of the MODE macro instruction. The message overwrites existing characters in the display screen, including messages in preparation or messages entered but not yet read by QTAM.

4. If the device is an IBM 1053 printer, the message is transferred whenever the printer buffer becomes free. The Supervisor restarts the channel program automatically if the printer is busy, but the line group is not freed for other messages until the message has been sent.

5. If the printer is not ready (intervention required), the operation is posted complete-with-error.

Write Initial: A write initial is issued to send a message to an IBM 2260 Local or to an IBM 1053 printer. If the message is to an IBM 2260 Local, the display begins at the current location of the cursor. Messages to the printer start at the current location of the IBM Selectric type element (normally at a new line). The channel program is:

Operation	Address	Flags	TP Code	Count
Write Buffer	Area	CD,SLT	03	Length

Erase/Write: Erase/Write may be specified instead of Write Initial by the MODE macro instruction. The command is valid only to an IBM 2260. The screen is erased and the message is displayed beginning in the upper left-hand display position. The channel program is:

Operation	Address	Flags	TP Code	Count
1. Erase	--	CC,SLI	00	01*
2. Write DS Buffer	Area	CD,SLI	03	Length

*Dummy count

Write-at-line-address: Write-at-line-address may be specified in place of write initial by the MODE macro instruction. The command is valid only to an IBM 2260. The first byte of text is interpreted as the line address, and the message is displayed beginning at that line. The channel program is the same as write initial except that the command code is write-at-line-address.

AT&T 83B3--Programming Information

1. An AT&T 83B3 Teletypewriter Selective Calling System Control Station controls up to 38 teletypewriters on a line. The teletypes may be:

   a. Model 28 Receive Only Typing Reperforator (ROTR);

   b. Model 28 Automatic Send Receive (ASR); or

   c. A combination of the two models.

2. Transmission between the 270x and the terminal is in Baudot code. The shift character conversion is a standard feature.

0 1 2 3 4 5 6 7	System/360 byte
- - S 1 2 3 4 5	Shifted Baudot code

3. The Transmitter Start Code (TSC) is a two-character code the function of which is similar to polling characters in IBM equipment. The Call Directing Code (CDC) is a two-character code the function of which is similar to addressing characters in IBM equipment.

4. Response to TSC (polling)

   a. Positive response is the message itself.

   b. Negative response is V. The channel-end and device-end status bits are set. Unlike IBM terminals, there is no unit exception with negative response. When the terminal is not ready, no response character is sent. A 2 second timeout occurs and unit check is set in the status byte.

5. Response to CDC (addressing)

   a. Positive response is V or M.

   b. No response is a negative response = 2 second timeout and unit check.

AT&T 83B3--Terminal to CPU

Read Initial: A Read Initial is issued to start or restart polling. The channel program is:

Operation	Address	Flags	TP Code	Count
1. Write EOT sequence	Table	CD,SLI	0	3
2. Write TSC	List	CC,SLI	0	2
3. Read response	Area	CD	Poll-restart	2
4. Read Message	Area+2	CD	Read Data	Length-2

AT&T 83B3--CPU to Terminal

Each terminal on a line has a unique two-character Call Directing Code (CDC).

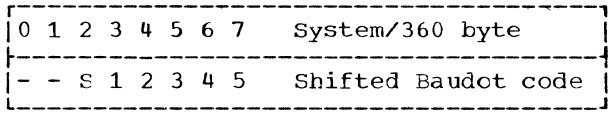
Write Initial: A Write Initial is issued to send a message to a station. The channel program is:

Operation	Address	Flags	TP Code	Count
1. Write EOT sequence	Table	CD,SLI	0	3
2. Write CDC	List	CD	0	2
3. Write shift character (ltrs)	Table	CC,SLI	0	1
4. Read response character	Area	CC	Multi-addressing	1
5. Write message	Area	CD	Write Data	Length

Western Union 115A--Programming Information

1. Plan 115A is a multidrop system. Up to 20 way stations can be accommodated on a line. A way station is a Model 28 Teletypewriter Automatic Send Receive (ASR) associated with a way station selector.

2. Transmission between the 270x and the terminal is in Baudot code. The shift character conversion is a standard feature.



3. The invitation-to-send code is similar to polling characters:

   The first character is an "X".

   The second character identifies the station.

   The call code is similar to addressing characters:

   The first character is a line control character 'circuit call".

   The second character identifies the station.

4. Response to invitation to send (polling):

   a. Positive response is the message itself.

   b. Negative response is V, which generates the channel-end and device-end status (no unit exception is set as with IBM terminals).

   c. No response: 2 second timeout and unit check.

5. Response to call (addressing):

   a. Positive response is 'V.'

   b. When the terminal is not ready, no response character is received.

6. The EOT sequence is ↑H which sets channel end, device end, and unit exception status. (↑= upshift or "FIGS"; = downshift or "LTRS.")

7. The first character of the message must be an EOA, which generates a space. The last portion of the message is an EOT sequence, which disconnects the station.

WU Plan 115A--Terminal to CPU

Each terminal on a line is "invited to send" with a two-character code. The first character is always X. The second character identifies the terminal.

Read Initial: Read Initial is issued to start or restart polling on a line. The channel program is:

Operation	Address	Flags	TP Code	Count
1. Write EOT sequence	Table	CD,SLI	0	9
2. Write "Invitation Code"	List	CC,SLI	0	2
3. Read Response	Area	CD	Poll-restart	2
4. Read Message	Area+2	CD	Read Data	Length-2

WU Plan 115A--CPU to Terminal

Each terminal on a line is addressed by a two-character call code. The first character is the line control character circuit call. The second character identifies the terminal.

Write Initial: A Write Initial is issued to send a message. The channel program is:

Operation	Address	Flags	TP Code	Count
1. Write EOT sequence	Table	CD,SLI	0	9
2. Write call code	List	CC,SLI	0	2
3. Read response	Area	CC	Multi-addressing	1
4. Write message	Area	CD	Write Data	Length

IBM 1050, Nonswitched--Terminal to CPU

Each component on a line is polled with a two-character code. The first of these two polling characters identifies the terminal; the other either selects a device or is the common polling character (0).

Read Initial: Read Initial is used to start or restart polling on a line. The channel program is:

Operation	Address	Flags	TP Code	Count
1. Write Ⓒ Ⓒ Ⓒ	Table	CD,SLI	0	3
2. Write polling characters	List	CC,SLI	0	2
3. Read response	Area	CD	Poll-restart	2
4. Read message block	Area+2	CD	0	Length-2

Auto Poll Read Initial: Used to begin polling on the Auto Poll line. The channel program is:

Command	Operation	Address	Flags	TP Code	Count
1	Write Ⓒ Ⓒ Ⓒ	Table	CC,SLI	0	3
2	Poll	Any entry in polling list	CC,SLI	0	k(n)
3	TIC	2nd POLL Command	SLI	0	1
4	TIC	Read Response			
5	Poll	1st entry in polling list	CC,SLI	0	k(n)
6	TIC	2nd POLL Command	SLI	0	1
7	Read Response	Area	CD	1	2
8	Read Message Block	Area+1	CD	3	Length-2

Read Continue [=WRITE Ⓨ + READ Data]: The Read Continue is requested from the EOB or EOBLC routine after a successful Read Initial or Read Continue operation to read subsequent blocks from the same component. The channel program writes the positive response character and chains to a read command. The data received is either followed by a Ⓑ or Ⓒ alone. To restart polling on the line, another Read Initial must be generated.

Read Repeat [=WRITE Ⓝ + READ Message block]: The Read Repeat is requested from the EOBLC routine, in the event of a transmission error, to reread a message block. The channel program writes the negative response character and chains to a read command that specifies the same storage area as in the previous operation.

IBM 1050, Nonswitched--CPU to Terminal

Each terminal on a line is addressed with a two-character code. The first of these two addressing characters identifies the terminal. The second selects one or all of the components of that terminal.

Write Initial: Write Initial is generated to send a message to a terminal. The channel program is:

Operation	Address	Flags	TP Code	Count
1. Write Ⓒ Ⓒ Ⓒ	Table	CD	0	3
2. Write addressing characters	List	CC,SLI	0	2
3. Read response to addressing	Area	CC	Multi-addressing	1
4. Write Ⓓ	Table	CD	0	1
5. Write message	Area	CD	0	Length

Write Continue: Write Continue is requested from the EOB or EOBLC routine to write another message block after a successful write-initial or write-continue operation. The channel program writes a message block and chains to the BRB/CCW to continue the operation.

IBM 1050, Switched--Terminal to CPU

Terminal to CPU transmission occurs as follows:

1. Program issues a Read Initial referencing an answering-polling list to enable a line. Prior to the enable, no calls can be received by the CPU.

2. Terminal operator calls the CPU when he has something to send. This completes the enable command, causing a device end-channel end (DE-CE) interruption.

3. QTAM polls the terminal that dialed, using the polling characters supplied in the polling list. If the terminal sends a positive response, a read CCW is executed to transfer the message block. The answering-polling list must contain polling characters of all the terminals that may call the CPU on this line. Normally, the common polling character 0 is used to poll all components on the terminal.

Read Initial: A Read Initial is issued to start polling. The channel program is:

Operation	Address	Flags	TP Code	Count
1. Disable		CC,SLI	0	1
2. Enable		SLI	Restart	1
3. Write pad characters	Table	CC,SLI	0	15

Commands 4-7 are the same as commands 1-4 of read initial for 1050 nonswitched.

Read Continue [WRITE Ⓨ + READ Data]: A Read Continue is requested from the EOB or EOBLC routine after a successful Read Initial or Read Continue to read the next block of data from the same component. The channel program writes the positive response character and chains to a read command.

Read Repeat [WRITE Ⓝ + READ Data]: The Read Repeat is requested from the EOBLC routine after an unsuccessful read operation for retransmission of the data which was received in error. The channel program writes the negative response character and then chains to a read command that specifies the same storage as the previous operation.

IBM 1050, Switched--CPU to Terminal

CPU-to-terminal transmission occurs as follows:

1. The program issues a Write Initial referencing a call-addressing list to dial the terminal and address the component.
2. If a positive response is received, the message is sent. If a negative response, the operation is posted complete-with-error.

Write Initial: A Write Initial is generated to send the first block of a message. The channel program is:

Operation	Address	Flags	TP Code	Count
1. Disable		CC,SLI	0	1
2. Dial the call digits	List	CC,SLI	0	n
3. Write Ⓒ Ⓒ Ⓒ	Table	CD	0	3
4. Write addressing characters	List	CC,SLI	0	2
5. Read response to addressing	Area	CC	Multi-addressing	1
6. Write message block	Area	CC,SLI	0	Length

n = number of dial digits as specified in the terminal table entry.

Write Continue (Same as Step 6 of Write Initial): The Write Continue is requested from the EOB or EOBLC routine after a successful Write Initial, Write Conversational, or Write Continue operation to send the next message block.

Write Conversational (Same as Steps 3-6 of Write Initial): This option provides the facility to readdress a terminal on a line with which connection is already established.

AT&T TWX 33/35--Terminal to CPU

TWX terminals are not polled. Line connection is established on terminal request; the terminal dials the CPU through a switched network, and the CPU answers the call.

Read Initial: The Read Initial channel program is used when transmission is originated by the terminal operator calling the CPU. The type of polling list is an answer list which specifies that line connection must be established by the terminal, before data may be transmitted. The channel program is:

Operation	Address	Flags	TP Code	Count
1. Disable		CC,SLI	0	1
2. Enable		SLI	Restart	1
3. Write CPU-ID sequence	List	CC,SLI	0	m
4. Read Response	Area	CD	Poll-restart	2
5. Read data	Area-2	CD	Read data	Length-2

m = count of characters in the CPU ID sequence specified in the polling list for the line.

TWX Model 33/35--CPU to Terminal

TWX terminals are not addressed. Line connection is established on CPU request; the CPU calls the remote terminal through a switched network.

Write Initial: The Write Initial channel program is used when transmission is to be originated by the CPU calling the terminal. The type of terminal list is a call list which specifies that line connection must be established by the CPU, before data may be transmitted. The channel program is:

Operation	Address	Flags	TP Code	Count
1. Disable		CC, SLI	0	1
2. Call the dial-digits	List	CC, SLI	0	n
3. Read terminal ID Sequence	List	CC, SLI	TWX ID	m
4. Write data	Area	CD	Write data	Length

n = specifies the number of dial digits in the terminal table entry.

m = specifies the number of characters in the TWX terminal ID sequence.

Write Conversational [WRITE Data]: Write Conversational can be used following Read Initial, Write Initial, or a previous Write Conversational. The channel program consists of a single command. The data transmitted must not end with a turn-around character.

IBM 2740 Remote Terminals

There are eight different types of IBM 2740 terminals, depending on the features provided. Each terminal uses a different set of commands. Eight modules are provided (IJLQN0 - IJLQN7), one for each terminal (see Figure 9).

IBM 2740 Terminal Type	Features Provided	Associated Module
274A	Basic Type	IJLQN0
274B	Basic, Dial	IJLQN1
274C	Station Control	IJLQN2
274D	Station Control and Checking	IJLQN3
274E	Dial, Transmit Control and Checking	IJLQN4
274F	Checking	IJLQN5
274G	Dial with Checking	IJLQN6
274H	Dial, Transmit Control	IJLQN7

Figure 9. Types of IBM 2740 Terminals with Associated Modules

IBM 274A (Basic) - Terminal to CPU

Read Initial:

Command	Operation	Address	Flags	TP Code	Count
1	Write © © ©	Table	CC,SLI	0	3
2	Prepare for Receipt of Data		CC,SLI	0	1
3	Sense	LSEN	CC,SLI	0	2
4	Read Data	Area	CD	Read Data	Length

Command 1: Sends three © 's to set all terminals on the line in control mode.

Command 2: Indicates to the CPU that data is arriving. (No data transfer occurs with this command.)

Command 3: Indicates to QTAM whether or not the Prepare command has completed for STOPLN-STARTLN logic.

Command 4: Reads text until the count is exhausted and chains through a TIC placed after it to the second BRB/CCW in the ring. If a buffer is not requested for the second BRB/CCW, the PCI flag causes an interrupt on command 4, and the PCI routine is entered to obtain a buffer. If the buffer is not requested but not assigned, the TIC command has an invalid address which causes a program check interrupt. The Line End routine recognizes this condition and obtains the buffer.

IBM 274A (Basic) - CPU to Terminal

Write Initial:

Command	Operation	Address	Flags	TP Code	Count
1	Write Ⓓ followed by 15 idle characters	Table	CD	0	16
2	Write Data	Area	CD	Write Data	Length

Command 1: Write Ⓓ indicates that the following characters will be text. Transmission of idle characters insures that the terminal has time to prepare to receive.

Command 2: Writes text until the count is exhausted.

IBM 274B (Basic/Dial) - Terminal to CPU

Read Initial:

Command	Operation	Address	Flags	TP Code	Count
1	Disable		CC,SLI	0	1
2	Enable		CC,SLI	Restart	1
3	Prepare for Receipt of Data		CC,SLI	0	1
4	Sense	LSEN	CC,SLI	0	2
5	Read Data	Area	CD	Read data	Length

IBM 274B (Basic/Dial) - CPU to Terminal

Write Initial:

Command	Operation	Address	Flags	TP Code	Count
1	Disable		CC,SLI	0	1
2	Dial		CC,SLI	0	Length
3	Write Pad Characters	Table	CC,SLI	0	15
4	Write Ⓓ followed by 15 idle characters	Table	CD	0	16
5	Write Data	Area	CD	Write Data	Length

Write Conversational: Same as Commands 4 and 5 of Write Initial.

IBM 274C (Station Control) - Terminal to CPU

Read Initial:

Command	Operation	Address	Flags	TP Code	Count
1	Write Ⓒ Ⓒ Ⓒ	Table	CD	0	3
2	Write Polling Character	List	CD	0	1
3	Write Space	Table	CC,SLI	0	1
4	Read Response Character	Area	CD	Poll Restart	2
5	Read Data	Area	CD	Read Data	Length-2

Auto Poll Read Initial:

Command	Operation	Address	Flags	TP Code	Count
1	Write ⓒ ⓒ ⓒ	Table	CC,SLI	0	3
2	Poll	Any entry in polling list	CC,SLI	0	k(n)
3	TIC	2nd POLL Command	SLI	0	1
4	TIC	Read Response			
5	Poll	1st entry in polling list	CC,SLI	0	k(n)
6	TIC	2nd POLL Command	SLI	0	1
7	Read Response	Area	CD	1	2
8	Read Message Block	Area+1	CD	3	Length-2

IBM 274C (Station Control) - CPU to Terminal

Write Initial:

Command	Operation	Address	Flags	TP Code	Count
1	Write Ⓒ Ⓒ Ⓒ Ⓢ	Table	CD	0	4
2	Write Addressing Character	List	CD	0	1
3	Write Space	Table	CC,SLI	0	1
4	Read Response Character	Area	CC	Multiaddressing	1
5	Write Data	Area	CD	Write Data	Length

IBM 274D (Station Control and Checking) - Terminal to CPU

Read Initial:

Command	Operation	Address	Flags	TP Code	Count
1	Write Ⓒ Ⓒ Ⓒ	Table	CD	0	3
2	Write Polling Character	List	CD	0	1
3	Write Space	Table	CC,SLI	0	1
4	Read Response Character	Area	CD	Poll Restart	2
5	Read Data	Area+2	CD	Read Data	Length-2

Auto Poll Read Initial:

Command	Operation	Address	Flags	TP Code	Count
1	Write Ⓒ Ⓒ Ⓒ	Table	CC,SLI	0	3
2	Poll	Any entry in polling list	CC,SLI	0	k(n)
3	TIC	2nd POLL Command	SLI	0	1
4	TIC	Read Response			
5	Poll	1st entry in polling list	CC,SLI	0	k(n)
6	TIC	2nd POLL Command	SLI	0	1
7	Read Response	Area	CD	1	2
8	Read Message Block	Area+1	CD	3	Length-2

Read Continue:

Command	Operation	Address	Flags	TP Code	Count
1	Write Ⓨ	Table	CC,SLI	0	1
2	Read Data	Area	CD	Read Data	Length

Read Repeat: Write Ⓝ followed by Command 2 of Read Continue.

IBM 274D (Station Control and Checking) - CPU to Terminal

Write Initial:

Command	Operation	Address	Flags	TP Code	Count
1	Write Ⓒ Ⓒ Ⓒ Ⓢ	Table	CD	0	4
2	Write Addressing Character	List	CD	0	1
3	Write Space	Table	CC,SLI	0	1
4	Read Response Character	Area	CC	Multiaddressing	1
5	Write Ⓓ followed by 15 idle characters	Table	CD	0	16
6	Write Data	Area	CD	Write Data	Length

Write Continue: Same as Command 6 of Write Initial.

IBM 274E (Dial, Transmit Control and Checking) - Terminal to CPU

Read Initial:

Command	Operation	Address	Flags	TP Code	Count
1	Disable		CC, SLI	0	1
2	Enable		SLI	Restart	1
3	Write /, space	Table	CC, SLI	0	2
4	Read Response Character	Area	CD	Poll Restart	2
5	Read Data	Area	CD	Read Data	Length-2

Read Continue:

Command	Operation	Address	Flags	TP Code	Count
1	Write (Y)	Table	CC, SLI	0	1
2	Read Data	Area	CD	Read Text	Length

Read Repeat: Write (N) followed by Command 2 of Read Continue.

IBM 274E (Dial, Transmit Control and Checking) - CPU to terminal

Write Initial:

Command	Operation	Address	Flags	TP Code	Count
1	Disable		CC,SLI	0	1
2	Dial		CC,SLI	0	Length
3	Write pad Characters	Table	CC,SLI	0	15
4	Write Ⓓ followed by 15 idle characters	Table	CD	0	16
5	Write Data	Area	CD	Write Data	Length

Write Continue: Same as Command 5 of Write Initial.

Write Conversational: Same as Command 5 of Write Initial.

IBM 274F (Checking) - Terminal to CPU

Read Initial:

Command	Operation	Address	Flags	TP Code	Count
1	Write Ⓒ Ⓒ Ⓒ	Table	CC,SLI	0	3
2	Prepare for Receipt of Data		CC,SLI	0	1
3	Sense	LSEN	CC,SLI	0	2
4	Read Data	Area	CD	Read Data	Length

Read Continue: Write Ⓨ followed by Command 3 of Read Initial.

Read Repeat: Write Ⓝ followed by Command 3 of Read Initial.

IBM 274F (Checking) - CPU to Terminal

Write Initial:

Command	Operation	Address	Flags	TP Code	Count
1	Write Ⓓ followed by 15 idle characters	Table	CD	0	16
2	Write Data	Area	CD	Write Data	Length

Write Continue: Same as Command 2 of Write Initial.

Write Conversational: The same Commands as Write Initial.

IBM 274G (Dial and Checking) - Terminal to CPU

Read Initial:

Command	Operation	Address	Flags	TP Code	Count
1	Disable		CC,SLI	0	1
2	Enable		CC,SLI	Restart	1
3	Write Ⓒ Ⓒ Ⓒ	Table	CC,SLI	0	3
4	Prepare for Receipt of Data		CC,SLI	0	1
5	Sense	LSEN	CC,SLI	0	2
6	Read Data	Area	CD	Read Data	Length

Read Continue: Write Ⓨ followed by Command 5 of Read Initial.

Read Repeat: Write Ⓝ followed by Command 5 of Read Initial.

IBM 274G (Dial and Checking) - CPU to Terminal

Write Initial:

Command	Operation	Address	Flags	TP Code	Count
1	Disable		CC,SLI	0	1
2	Dial		CC,SLI	0	Length
3	Write pad Characters	Table	CC,SLI	0	15
4	Write Ⓓ followed by 15 idle characters	Table	CD	0	1
5	Write Data	Area	CD	Write Data	Length

Write Continue: Same as Command 5 of Write Initial.

Write Conversational: Same as Commands 4 and 5 of Write Initial.

IBM 274H (Dial and Transmit Control) - Terminal to CPU

Read Initial:

Command	Operation	Address	Flags	TP Code	Count
1	Disable		CC,SLI	0	1
2	Enable		SLI	Restart	1
3	Write /,space	Table	CC,SLI	0	2
4	Read Response Character	Area	CD	Poll Restart	2
5	Read Data	Area+2	CD	Read Data	Length-2

IBM 274H (Dial and Transmit Control) - CPU to Terminal

Write Initial:

Command	Operation	Address	Flags	TP Code	Count
1	Disable		CC,SLI	0	1
2	Dial		CC,SLI	0	Length
3	Write Pad Characters	Table	CC,SLI	0	15
4	Write Data	Area	CD	Write Data	Length

Write Conversational: Same as Command 4 of Write Initial.

World Trade Telegrapn Terminals

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 an inner channel program automatically associated with the second part to perform identification sequence exchanges, whenever requested, during message transmission.
- The second part (Read or Write channel program proper) is set up to receive input messages or to send output messages.

Read Initial: A Read Initial is issued to enable the CPU to receive a character sent by the terminal. The channel program is:

Command	Operation	Address	Flags	TP Code	Count
1	Write CPU-ID sequence (Note 5)	List	CD SLI	5	n
2	Write (Note 3)	WRU	CC SLI	0	1
3	Read Terminal-ID (Note 6)	List	SLI	5	Length-1
4	Prepare	0	CC SLI	0	1
5	Sense (Note 1)	LSEN	CC SLI	0	1
6	Read	Area	SLI	3	Length

Read Continue: A Read Continue is issued after an EOM signal to read the next input message, or after a WRU signal to read the rest of the input message. The channel program is:

Command	Operation	Address	Flags	TP Code	Count
1	Write CPU-ID sequence (Note 5)	List	CD SLI	5	n
2	Write (Note 3)	WRU	CC SLI	0	1
3	Read Terminal-ID (Note 6)	List	SLI	5	Length-1
4	Read	Area	SLI	3	Length

Write Initial: A Write Initial is issued to send an output message. The channel program is:

Command	Operation	Address	Flags	TP Code	Count
1	Write (Note 2)	Table	CD SLI	5	1+m
2	Write (Note 3)	Table	CD SLI	5	12
3	TIC (Note 4)				
4	Write CPU-ID sequence (Note 5)	List	CD SLI	5	n
5	Write (Note 6)	WRU	CC SLI	0	1
6	Read Terminal-ID (Note 7)	List	SLI	5	Length-1
7	Write	Area	SLI	3	Length

Note 1: LSEN is the address of the sense byte in the LCB. This byte is set to X'FF' before the Read Initial channel program is set up. When the SENSE command is executed (that is, when the terminal starts sending an input message), the LSEN byte is overlaid by the adapter sense byte.

Note 2: One LTRS character is sent which may be followed by "m" padding characters if the line is defined as an output line only with the "motor-off" option.

Note 3: Twelve (12) LTRS characters must be sent at the beginning of each output message.

Note 4: The transfer address in the TIC CCW is that of CCW no. 3 or that of CCW no. 6, depending on whether or not the WRU macro instruction is present in the Send-Header subgroup of the LPS.

Note 5: The computer identification (CPU-ID) defined in the POLL macro instruction associated with the line is sent to the terminal.

Note 6: When the Automatic Answerback Unit feature is installed on the terminal, the CPU sends the WRU signal to the terminal which, in turn, sends its identification sequence to the CPU.

Note 7: The terminal identification is read into the area reserved by the TERM macro instruction associated with the line.

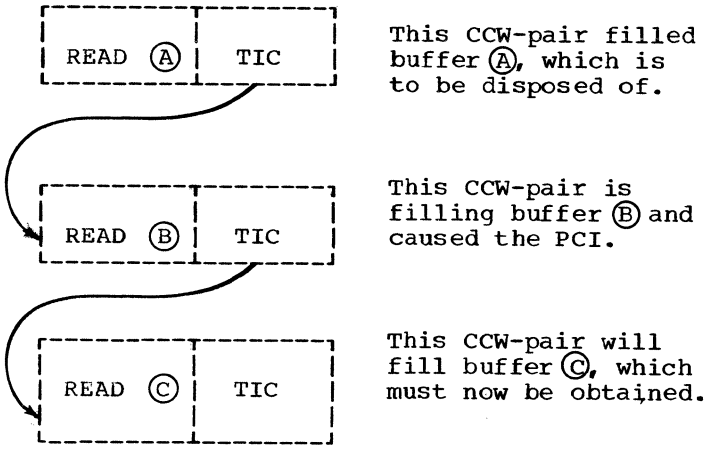
LINE-PCI AND PROGRAM CHECK MODULE

Module Name: IJLQLC (Chart LB)

Entry: This module is entered from IJLQLA when a PCI or CPC has occurred.

This module disposes of the buffer filled or emptied by the channel command preceding that which caused the PCI, and places a request for the buffer that is to be emptied or filled by the CCW when it is again encountered in the ring.

Graphically:



This CCW-pair filled buffer Ⓐ, which is to be disposed of.

This CCW-pair is filling buffer Ⓑ and caused the PCI.

This CCW-pair will fill buffer Ⓒ, which must now be obtained.

For receive operations, buffer Ⓐ is routed to the Interim-LPS queue. If the line is an Auto Poll line, the first byte in this buffer contains the index of the responding terminal in the polling list and is used to update the polling list pointer in the LCB. (If the index is not that of the terminal on which the polling operation was started, the message count is reset to zero in the LCB.) For send operations, the buffer is routed to the Available-Buffer queue. In either case, the request for buffer Ⓒ is routed to the Active Buffer Request queue.

This module 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 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 accessed. 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 nonzero 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 (IJLQLCPA) 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 made valid, in the period between the program check's generation 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 routine (at NOTINQ) to release the buffer filled by the CCW immediately preceding the TIC that caused the program check.

LINE APPENDAGE MODULE

Module Name: IJLQLA (Chart EG-EN)

Entry: This appendage is entered from the supervisor.

1. When a programmed-controlled interrupt (PCI) or channel program check (CPC) occurs during the execution of QTAM channel command for the line.

2. When a channel end occurs during line I/O operation.

When entered because of a PCI or CPC, control is passed to IJLQLC after initialization of the register.

Normally the routine routes a message-filled buffer to the LPS queue or exits to the supervisor to restore the channel program.

When it is 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 end of the polling list has been reached, the routine sets the "cleanup" code in the LCB (LSTA = X'80').

For autopolled lines, when it is entered because the end of the polling list is reached (the channel program must have been previously modified to allow this interruption), the routine sets the LCB polling list pointer to the dummy "end-of-list" entry and sets on the LCB cleanup code.

Error recovery procedures are provided to perform tests on the status and sense bits to determine if a transmission error occurred on the operation. If so, the routine branches to an appropriate subroutine which attempts to recover from the error.

Generally, if there has been no text transfer, the channel program is retried. If there is an error after three attempts to execute the channel program, the error is considered permanent. If a timeout or intervention required error occurs on a switched line, the CSW is saved in the LSAV field of the LCB. A CCW to disable the line is set up as the first CCW in the channel program. A disable return (X'80') is set in the LERR+1 field of the LCB, and exit is made to the supervisor to execute the disable.

A condition that should not happen is also considered a permanent error, and the "should not occur" bit, bit 7, is set in the error halfword in the LCB.

For all permanent errors, a message is written to the operator. The routine executes a branch-and-link to the Message Writer subtask (IJLQMW), passing parameters in standard registers. The Message Writer subtask builds an information block from the information in the LCB, and inserts a dummy ECB at the top of the Ready Queue, then returns to the Line Appendage routine. The routine releases the LCB and issues a QPOST to free the buffer. LPS Control recognizes the dummy ECB, and either:

1. Issues a FETCH (SVC 2) to call Phase 1 of the Message Writer routine into the transient area, passing the address of the information block in register 0, or

2. If the Message Writer is presently writing a message, stacks the message in a save area, issues a QPOST to post a dummy element to itself with a low priority, and exits to Qdispatch. When the Implementation module regains control, it again tests to determine if the Message Writer is occupied and repeats the sequence.

For certain types of error (data check, intervention required, and nontext timeouts), threshold line error counters are added to the cumulative counters, the threshold counters are reset, and an operator message is provided.

Error Recovery Procedure Subroutines

For all channel status errors except unit exception (attention, status modifier, control unit end, program check, protection check, channel data check, channel control check, interface control check or chaining check), no retry is performed and an error message is immediately sent.

A unit exception on a read, read response to poll, or enable is handled as other channel status errors. A unit exception on a write command is retried two times, with these exceptions:

1. If a Teletype I adapter is on the line, a break command is issued and it is treated as a permanent error. No message is sent, however, unless the error occurred on the second retry.

2. If an IBM 2701 terminal control unit is on the line, a read skip is issued and return is to the Read Skip subroutine (see Read Skip Subroutine).

3. If a World Trade Telegraph Adapter is on the line, there is a contention situation. A break command is issued, and on completion of this command the write command is retried.

When a unit check occurs, a SENSE command is issued to obtain the sense bits. Upon return, these bits are tested to determine which of the following subroutines to enter.

Data Check Subroutine: Generally, the channel program is retried two times. If it fails three times, it is treated as a permanent error: An error message is sent, and the line is handled as if it had returned a negative response to polling.

If there was a text transfer, it is treated as a permanent error, but no error message is sent unless other errors have occurred on the same channel program.

If a World Trade Telegraph Adapter is on the line, two conditions can occur:

1. If the failing CCW is a write, there is a contention situation. A break command is issued, and on completion of this command the write command is retried.

2. If the failing CCW is a break, the command is retried 15 times, after which it is treated as a permanent error.

If the failing CCW is a write on a line with any but a Teletype I adapter, the "should not occur" flag is set in the error halfword, and it is treated as a permanent error.

If a permanent error occurs when an IBM 2701 terminal control unit is on the line, a Diagnostic Write/Read is performed. If the Diagnostic Write/Read does not complete successfully, bit 13 is set in the error halfword.

A data check causes the threshold line error counter for data checks to be incremented. The counter is incremented for each nontext failure (except "should not occur" errors), including both retries.

Timeout Subroutine: Generally, the channel program is retried two times. If it fails three times, it is treated as a permanent error: An error message is sent and the line is handled as if it had returned a negative response to polling.

If there was a text transfer, it is treated as a permanent error, but no error message is sent unless this is a return from a third retry.

For a read response to polling on a TWX terminal, it is treated as a permanent error, but no error message is sent.

For a read response to polling on any other switched terminal:

1. If receiving, retry is started with the third CCW.

2. If sending, the first CCW is set to a disable and retry is started on the first CCW. If the failing CCW is a prepare, retry is started with the prepare CCW. For all other cases involving retry, retry is started on the first CCW.

A timeout causes the threshold line error counter for timeouts to be incremented. The counter is incremented for each failure, including all retries.

Intervention Required Subroutine: Generally, the channel program is retried two times. If it fails three times, it is treated as a permanent error: An error message is sent, and the line is handled as if it had returned a negative response to polling.

If there was a text transfer, the timeout bit, bit 9, is set in the error halfword and it is treated as a permanent error, but no message is sent unless this is a return from a third retry. If this occurs on a switched line, a disable is issued to break the connection.

An intervention required error causes the threshold line error counter for intervention required errors to be incremented. The counter is incremented for each failure, including all retries.

Lost Data Subroutine: Generally, the channel program is retried two times. If it fails three times, it is treated as a permanent error: An error message is sent, and the line is handled as if it had returned a negative response to polling.

If the failing CCW is not a text transfer, the channel program is retried, starting at the first CCW for a nonswitched line and starting at the third CCW for a switched line.

If the failing CCW is a text transfer and the residual count is not equal to 0, it is treated as a permanent error.

If the failing CCW is a text transfer and the residual count is equal to 0, a read skip command is executed, and return is to the Read Skip subroutine.

Bus Out and Overrun Subroutine: Generally, the channel program is retried two times. If it fails three times, it is treated as a permanent error: An error message is sent, and the line is handled as if it had returned a negative response to polling.

If a bus out check has occurred on a write CCW, the subroutine tests if:

1. The next CCW is a read, or

2. A type III adapter is on the line.

If either is the case, a read skip is executed. Subsequent actions taken are the same as those taken following execution of a read skip in the Lost Data Subroutine.

If a bus out check has occurred on a write CCW and either a type I or type II adapter is on the line, it is considered a permanent error but no message is sent.

If a bus out check has occurred on a CCW that is not text transfer, the channel program is retried. For a TWX terminal, retry must start with the first CCW. For any other switched terminal, retry starts with the third CCW, bypassing the disable-dial CCWs.

If a bus out check has occurred on a dial CCW, retry must begin with the first CCW.

For an overrun check on a text transfer CCW, the retry counter in the error halfword (bits 14 and 15) is updated and retry is performed.

For an overrun check on a CCW that is not text transfer, the channel program is retried. For a TWX terminal, retry must start with the first CCW. For any other switched terminal, retry starts with the third CCW, bypassing the disable-dial CCWs.

Command Reject and Equipment Check Subroutine: If the error is a command reject, the channel program is retried two times. If it fails three times, it is treated as a permanent error: An error message is sent, and the line is handled as if it had returned a negative response to polling.

If the error is an equipment check, the "control unit failure" bit, bit 13, is set in the error halfword. For a switched line, the line is disabled. The condition is considered a permanent error.

Read Skip Subroutine: This subroutine is entered on return from issuing a read skip in one of the other error subroutines.

The status bytes from the read skip are tested for any errors, and the sense bytes are tested for command reject, bus out check, equipment check, overrun or residual count equal to 0. If any of these conditions has occurred, the "should not occur" bit, bit 7, is set in the error halfword, and it is treated as a permanent error. Otherwise, the sense byte of the read skip is saved and the original sense byte and CSW are restored.

If the original error was on a switched line, a test is made to determine if a timeout or intervention required error occurred on the read skip. If yes, the line is disabled and it is treated as a permanent error.

If no, a test is made to determine if the original error was on a text transfer. If yes, it is treated as a permanent error but no message is sent. If no, retry is performed.

If the original error was on a nonswitched line, a test is made to determine if it was the third error. If yes, it is treated as a permanent error.

If no, a test is made to determine if a timeout or intervention required error occurred on the read skip. If yes, an indicator is set for the End of Block and Line Correction routine to perform reselection. If no, the indicator is not set. Both cases are treated as permanent errors but no message is sent.

DECISION TABLES FOR ERROR RECOVERY PROCEDURES

Flowcharts are not provided for error recovery procedures within the Normal Line End Appendage and ERP module (IJLQEP). The following decision tables can be used to determine what action is taken for the various errors that may occur.

Table 1. Initial Selection Errors

Conditions	Actions
Attention	B
Control Unit End	B
Status Modifier	B
Unit Check	
Bus Out Check	C
Command Reject	C
Any other Abnormal Status or Sense	B

Table 2. Errors After Initial Selection

Priority	Status Conditions	Action
1	Channel Data Check	D
2	Busy	D
2	Attention	D
2	Control Unit End	D
2	Status Modifier	D
3	Unit Check	See Table 3
4	Chaining Check	D
4	Program Check	D
4	Protection Check	D
5	Unit Exception	See Table 4
6	Incorrect Length	D

Table 3. Sense Byte Analysis

Priority	Sense Bit	Action
1	Equipment Check	D
2	Lost Data	See Table 5
3	Time Out	See Table 6
4	Intervention Required	See Table 7
5	Bus Out Check	See Table 8
6	Data Check	See Table 9
7	Overrun	See Table 10
8	Command Reject	C

Table 4. Unit Exception

Command	Device Type	Action
WRITE	83B3,115A	E
WRITE	World Trade telegraph terminal	N
WRITE	All Others	F
ALL OTHERS	All	D

Table 5. Lost Data

Command	Type	Action
DIAL	--	G
READ	TWX ID Response	G
READ	TEXT	H
READ	All Others	C
ALL OTHERS	All	D

Table 6. Time Out

Command	Type	Action
DIAL	--	G
DISABLE	--	C
PREPARE	--	C
READ	TEXT	J
READ	TEXT Response	J
READ	TWX ID Response	M
READ	All Others	I
ALL OTHERS	All	D

Table 7. Intervention Required

Command	Type	Action
DIAL	--	G
PREPARE	--	G
WRITE	--	D
READ	Prior to Text	I
READ	Text or After	J
ALL OTHERS	All	D

Table 8. Bus Out Check

Command	Type	Action
DIAL	--	G
WRITE	TEXT (IBM Adapter Types 1, 2, 3)	F
WRITE	TEXT - All Others	J
WRITE	Prior to Text	I
WRITE	After Text	C
ALL OTHERS	All	D

Table 9. Data Check

Command	Type	Action
WRITE	All except World Trade telegraph terminals	K
WRITE	World Trade telegraph terminal	N
BREAK	World Trade telegraph terminal	O
READ	TWX ID Response	G
READ	TEXT	L
READ	Response to Text	L
READ	All Others	I
ALL OTHERS	All	D

Table 10. Overrun

Command	Type	Action
READ	TWX ID Response	G
READ	Prior to Text	I
READ	Text or After	L
ALL OTHERS	All	D

ACTIONS

A An intervention required message is provided indicating that the device is unavailable; for example, control unit has power off.

B The failing CCW is retried. On the third occurrence of this condition an operator message is provided indicating a "should not occur" error.

C The failing CCW is retried. On the third occurrence of this condition an operator message is provided.

D An operator message is provided. If the error occurred on a switched line, a disable CCW is executed before return.

E A Break command is issued and the initial channel program is executed. On the third occurrence of this condition, an operator message is provided.

F A Read skip command is issued and an initial channel program is executed. On the third occurrence of this condition, an operator message is provided.

G The channel program is retried. On the third occurrence of this condition, the line is disabled and an operator message is provided.

H If the residual count is not 0, go to action D. If the residual count is 0, a Read skip command is issued and an initial type channel program is executed if EOBLC has been specified. On the third occurrence of this condition, an operator message is provided.

I The channel program is retried after the Dial or Enable sequence, if present. On the third occurrence of this condition, an operator message is provided.

J An initial channel program is executed if EOBLC has been specified. On the third occurrence of this condition an operator message is provided.

K For Teletype Type I adapters go to action I. Otherwise a Diagnostic Write/Read operation is performed on nonswitched lines if the control unit is a 2701, and an operator message is provided.

L A repeat channel program is executed if EOBLC has been specified. On the third occurrence of this condition an operator message is provided.

M The error is ignored and return is made to normal processing.

N A contention situation occurred. A break command chained to a write command (Carriage return plus Line Feed) is issued.

O The break command is retried 15 times before the error is treated as a permanent error and a message is sent to the operator.

AUDIO LINE APPENDAGE MODULE

Module Name: IJLQAA

Entry Point: This appendage is entered from the Supervisor at IJLQAA

1. when a program-controlled interrupt (PCI) occurs during the execution of a 7772 channel command for the line; or

2. when a channel-end condition occurs during an I/O operation.

Entry from the Supervisor is into an analysis subroutine which determines whether the 7772 Line PCI routine or the Audio Line-End routine is to be executed. The ARU-Internal, ARU-Receive, and ARU-Send routines packaged in the Audio Line Appendage module are discussed first.

ARU-Internal Routine (Chart A1)

This routine is entered at IJLQAA70 in problem program state through a branch generated on expansion of a CHECKARU or REPEAT macro. The ALCB containing a user-provided address chain is posted (via an SVC 31) to the ARU-Send queue before return is made to the beginning of the LPS Control routine (at IJLQIP20).

ARU-Receive Routine (Chart A1)

This routine is entered at IJLQAA20 in problem program state through a branch generated on expansion of a POSTARU macro. The ALCB priority is set to X'FC', and the ALCB containing an input message is posted (via an SVC 31) to the corresponding MS process queue defined in the message processing program for the GET audio messages. When the message processing program is not opened, the ALCB is queued in an ALCB waiting chain located in the expansion of the corresponding PROCESS macro instruction. In any case, this routine exits to the beginning of the LPS Control routine (at IJLQIP20).

ARU-Send Routine (Chart A2)

This routine is entered on activation of the ARU-Send subtask at IJLQAA51+6 when an ALCB containing a user-provided address chain is posted (via an SVC 31) to the ARU-Send queue by the ARU-Internal or the Audio PUT routine. The routine is also entered when an ALCB representing a 7772 line requiring an initial write operation is posted Via a branch to the POST subroutine) to the ARU-Send queue by the Audio Line End routine.

Depending on the type of audio line, the routine performs the following functions. For a 7770 line, it updates the ALCB channel program before branching to the Audio Line-End routine to request a Start I/O operation on the line. For a 7772 line, it first checks if a DCV buffer pool has been provided for the line group. If not, the routine branches to the 7772 Disk Read routine. If yes, a DCV buffer is required. When no DCV buffer is available, the 7772 ALCB is queued in the ALCB waiting chain of the DCV buffer queue, and the address of the 7772 DCV Buffer subtask is placed in the subtask field of this queue (if this has not been formerly done by a previous ALCB in this chain). Then, the routine exits to the Qdispatch subroutine. When a DCV buffer is available, it is dequeued from the element chain of the DCV buffer queue and allocated to the ALCB. Then, the routine branches to the 7772 Disk Read routine.

IBM 7772 Line PCI Routine (Chart A3)

This portion of the Audio Line Appendage is entered from an analysis subroutine when a program controlled interrupt (PCI) has occurred during the execution of a read or write channel command for a 7772 line.

When the PCI has occurred on a write command, this routine branches to the 7772 Disk Read routine to prepare the sending of the next DCV word. When the PCI has occurred on a read command, with or without a channel-end condition, the following action is taken.

- No channel-end: If a DCV buffer was allocated to the 7772 ALCB, it is posted to the DCV buffer queue via a branch to the QPOST subroutine. If not, the routine branches to the Qdispatch subroutine.

- Channel-end on reception of a nonconsistent message (no input character, or an EOT character only): The channel program is updated for a Disable command, the status of the ALCB is set to indicate DISABLE (LSTS=X'04'), and the routine branches to the Audio Line-End routine to request a Start I/O operation on the 7772 line.

- Channel-end on reception of a consistent message: The length of the input message is computed, then stored in the ALCB, and the status of the ALCB is set to indicate READ (LSTS=X'02'). When a 7772 DCV buffer was allocated to the ALCB, the exit to the Qdispatch subroutine is changed to return to the 7772 Line PCI routine at the SPECIAL entry, and this DCV buffer is posted to the DCV Buffer queue via a branch to the POST subroutine. When no DCV buffer was allocated, or when the 7772 Line PCI routine is reentered at SPECIAL, the routine branches to the Audio Line-End routine to post the ALCB to the LPS queue for input processing.

Audio Line-End Routine (Charts A4, A5)

This portion of the Audio Line Appendage consists of two subroutines entered from the analysis subroutine (Chart A3) when a channel-end condition occurs during the execution of a channel command for an audio line.

IBM 7770 Line-End Subroutine (Chart A4): Depending on which channel command causes the channel-end condition, the subroutine function is as follows:

- On a Disable command, the subroutine updates the channel program according to initial conditions and sets the status of the ALCB to indicate ENABLE (LSTS=X'01'). If the line must be stopped, exit is made to the Qdispatch subroutine. If not, the ALCB representing an enable request is posted to the LPS queue via a branch to the POST subroutine.

- On a Read command when a nonconsistent message (no input character or an EOT character only) is received, the subroutine updates the channel program to prepare the disabling of the line and sets the status of the ALCB to indicate DISABLE (LSTS=X'04'). Then, the ALCB representing a disable request is posted to the LPS queue via a branch to the POST subroutine.

- On a Read command when a consistent message is received, the subroutine computes the length of the input message, stores it in the ALCB, and sets the status of the ALCB to indicate READ (LSTS=X'02'). If an overlength condition is detected, the overlength bit is set on in the ALCB error byte. Then, the ALCB representing an input message to be processed by the user's LPS routines is posted to the LPS queue via a branch to the POST subroutine.

IBM 7772 Line-End Subroutine (Chart A5): Depending on which channel command causes the channel-end condition, the subroutine function is as follows. (The simultaneous occurrence of a PCI condition is described in IBM 7772 Line PCI Routine.)

- On an Enable command, the subroutine updates the channel program according to the initial write condition and sets the status of the ALCB to indicate WRITE AFTER DISABLE (LSTS=X'10'). Then, the ALCB is posted to the ARU Send queue via a branch to the POST subroutine to send the invitional or informational message.

- On a Disable command, the subroutine updates the channel program according to initial conditions, sets the status of the ALCB to indicate DISABLE (LSTS=X'04'), and, if a DCV buffer was allocated to the ALCB, returns this buffer to the DCV buffer queue. Then, the ALCB representing an enable request is posted to the LPS queue via a branch to the POST subroutine. When the subroutine detects that the line must be stopped after an Enable command, the disabling of the line is prepared. If this is detected after a Disable command, the enabling of the line is not requested.

- On a Read command, the subroutine computes the length of the input message, stores it in the ALCB, sets the status of the ALCB to indicate READ (LSTS=X'02'), and, if an overlength condition is detected, sets on the overlength bit in the ALCB error byte. Then, the ALCB representing an input message to be processed by the user's LPS routines is posted to the LPS queue via a branch to the POST subroutine. When the input message is nonconsistent, the disabling of the line is prepared and the ALCB posted to the LPS queue represents a disable request.

Audio Error Recovery Procedures

The audio error recovery procedures are provided to perform tests on the status and sense bits to determine if a transmission error occurred on the operation. If so, the routine attempts to recover from the error.

Generally, if there has been no data transfer, the channel program is retried. If there is an error after three attempts, the error is

considered permanent. A condition that should not happen is considered a permanent error, and the CSW is saved in the LACF field of the ALCB. A disable CCW is prepared and exit is made to the supervisor to execute the disable.

For all permanent errors, a message is written to the operator on the system console. The routine executes a branch-and-link to the Message Writer subtask (IJLQMW), passing parameters in standard registers. The Message Writer subtask builds an information block from the information in the ALCB, and inserts a dummy ECB at the top of the Ready Queue, then returns to the Audio Line Appendage routine. The LPS Control routine recognizes the dummy ECB, and either:

1. Issues a FETCH (SVC 2) to call the Audio Message Writer routine (IJLQWA) into the logical transient area, passing the address of the information block in register 0, or

2. If the Message Writer is presently writing a message, the message is stacked and the availability of the Message Writer will be tested each time the Implementation module regains control.

For each transmission, the threshold line error counters are updated, and if one of them reaches its threshold value, these counters are added to the cumulative counters, the threshold counters are reset, and an operator message is provided.

For the errors occurring on read commands (channel data check or overrun) the "error on read" bit (X'20') is set in the line error byte.

For all channel status errors except unit exception (program check, protection check, chaining check, or channel data check) no retry is performed and an error message is immediatly sent.

A unit exception detected on a read or write command is not an error, but indicates the caller is not expecting the end of the transaction.

When a unit check occurs, a SENSE command is issued to obtain the sense bits. Upon return, these bits are tested to determine which action is to be performed.

DECISION TABLES FOR AUDIO ERROR RECOVERY PROCEDURES

Charts are not provided for error recovery procedures within the Audio Line Appendage module (IJLQAA). The following decision tables can be used to determine what action is taken for the various errors that may occur.

Table 11. Initial Selection Errors

Conditions	Action
Unit Check	
Bus Out Check	A
Command Reject	A
Any other Abnormal Status or Sense	B

Table 12. Errors After Initial Selection

Priority	Status Conditions	Action
1	Channel Data Check	See Table 3
2	Unit Check	See Table 4
3	Protection Check	C
3	Program Check	C
3	Chaining Check	C
4	Incorrect Length	C
5	Any other Abnormal Status	C

Table 13. Channel Data Check

Command	Control Unit Type	Action
READ	7770, 7772	D
WRITE	7770	E
WRITE	7772	F

Table 14. Sense Byte Analysis

Priority	Sense Conditions	Action
1	Equipment Check	C
2	Bus Out Check (not alone)	
	WRITE (7770)	E
	WRITE (7772)	F
	READ	C
3	Data Check (not alone)	
	WRITE (7770)	E
	OTHERS	C
4	Overrun	
	WRITE (7770)	E
	WRITE (7772)	C
	READ	D
5	All other Sense Conditions	C

ACTIONS

A. The failing CCW is retried. On the third occurrence of this condition an operator message is provided and a disable CCW is executed before return.

B. The failing CCW is retried. On the third occurrence of this condition an operator message is provided indicating a "should not occur" error and a disable CCW is executed before return.

C. An operator message is provided and a disable CCW is executed before return.

D. An operator message is provided and return is made to normal processing.

E. An operator message is provided. A noise message is sent on the line before restarting the channel program on the command chained to the failing write command.

F. An operator message is provided. The channel program is restarted on the command chained to the failing write command.

WTTA LINE APPENDAGE MODULE

Module Name: IJLQTA (Charts Y1, Y2, and Y3)

Entry points: This appendage 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 appendage can be re-entered from the QTAM Line Appendage routine IJLQLA.

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 different from the initial count, control is returned to the QTAM Line Appendage IJLQLA.

If the interrupted channel command is a Read CCW with equal initial and residual counts, the action taken depends on the type of the Read CCW, as follows:

1. If the interrupted channel command is the first Read CCW, the PCI is ignored and control is returned to IJLQLA.

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 into the CSW.
b. If this character is other than EOM, EOT, or WRU, control is returned to the QTAM Line Appendage IJLQLA.

WTTA LINE-END ROUTINE

The WTTA Line-End routine is entered when an I/O operation ends with a channel-end condition, or is re-entered from the QTAM Line Appendage IJLQLA. Moreover, if an I/O operation ends with channel-end and unit-check conditions, the result of the Sense operation performed by IJLQLA is analyzed to check whether an abnormal condition occurred and, if so, control is passed to the ERP routines.

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, of an exchange of identification sequences or of a Break channel program, 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 an LTRS character followed by n Mark characters (where "n" is the number specified in the DTFQT macro instruction). On completion of this Write channel program:

   a. If the command has correctly ended, the buffer prepared for input is posted to the Interim LPS Queue and the cleanup flag is set on in the LCB.
   b. If the command has not correctly ended, the Read channel program is restarted.

2. On completion of a Read channel program: The last character received in the corresponding buffer is analyzed:

   a. If this character is EOT, the EOT flag is set on 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. If the WRU character is in the first buffer, the Read CCW is updated to read the rest of this buffer, and the first part (identification sequence exchange) of the Read channel program is started. If the WRU character is in another buffer, the WRU flag is set on 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 identification sequences is requested at the end of the output message. If this exchange is requested, the first part (identification sequence exchange) of the Write channel program is started.
   b. If the I/O operation has ended with an abnormal-end condition (contention), a Write Break channel program is started if the interrupted CCW is not a Write Text CCW. If the interrupted CCW is a Write Text CCW, the transmission error bit is set on and the buffer is posted to the LPS Queue.

4. On completion of an exchange of identification sequences: If the terminal has sent its identification sequence (WRU=YES specified in DTFQT), a Write Break is started to inhibit the LTRS character, which is still sent after the terminal identification. On completion of the Break CCW, 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 was 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 was successful, the action taken depends on when the exchange was 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 different from WRU, the read channel program is restarted to read the rest of the input message.

5. On completion of a Break channel program: If the command has correctly ended, the interrupted Write CCW is restarted. If the command has not correctly ended, the Break channel program is restarted (there are only 15 retries). If the fifteenth retry has not correctly ended, the transmission error bit is set on and the buffer is posted to the LPS Queue if the line is sending, or to the Interim LPS Queue if the line is receiving (before posting, the buffer size is updated).

IBM 2260 LOCAL APPENDAGES

Module Name: IJLQLO (Charts LO, LP, and LQ)

Entry: This appendage is entered from the Supervisor when:

1. A program-controlled interrupt (PCI) occurs during execution of a QTAM command for the IBM 2260-2848 Local line group;

2. An I/O interrupt occurs ending execution of such a command; or

3. An I/O interrupt occurs with the Attention bit set in the CSW status, caused by the ENTER key being depressed at an IBM 2260 Local terminal. This interrupt is processed only if a command control block (CCB) for that IBM 2260 is in the DOS channel queue.

The status bits of the CSW are analyzed for normal or error conditions and entry is made to the proper subroutine.

The module is composed of two subroutines, in addition to preliminary analysis. These subroutines are the Attention subroutine and the Channel End subroutine. Error recovery procedures are included as a logical part of the Channel End subroutine. These subroutines are discussed seperately in the following paragraphs.

Whenever exit is made to a supervisory routine, the CSW status bits are set to indicate whether or not the CCB for the terminal is to

remainon the channel queue. If sending or receiving is to be permanently discontinued for the terminal, such as during a system close-down, the CCB is removed from the channel queue.

When entered because of a PCI or program check, control is passed to IJLQLC. PCI's can occur only on send operations. For receive operations, all buffers needed to read a message are obtained in the BRB-Ring routine prior to execution of the channel program.

Attention Subroutine

This subroutine is entered when an attention interrupt has occurred, indicating that a message has been entered at a terminal and is ready for processing. If this Read request occurs when a system close-down is in progress, or when the terminal is set for output only, it is ignored and return is made to the Supervisor to remove the CCB from the channel queue. In the case of a temporary stoppage of the terminal, return is made to the Supervisor, but the CCB remains on the channel queue.

If the Read request can be accepted, a test is made to determine if the LCB is available. If it is, it is posted to itself and linked to the top of the Ready Queue. This causes receiving to be initiated the next time Qdispatch gains control. Exit is then made to the Priority-Search subroutine in the QTAM nucleus to post the CCB for the terminal to the Attention queue for the line group.

Channel End Subroutine

Under normal ending conditions of a receive operation, the prefix of each buffer containing data is initialized and all buffers are linked to the head of the Ready Queue. Exit is then made to the Qdispatch subroutine, which routes the buffers to the user's LPS.

For send operations, a test is made to determine if an Attention interrupt has been posted for the terminal to which the message was just sent. If so, the CCB is removed from the Attention queue, because the message to be read when the Attention interrupt was serviced has been overlaid by the message just written to the terminal. The last buffer of a message sent is routed to the Send LPS so that the user may perform error checking.

In a receive operation, if the residual count is equal to the initial count and the interrupt occurs on the first CCW in the channel program, the terminal operator has depressed the ENTER key without having keyed in the START symbol. The 'zero length message bit' (bit 4) in the error halfword is set and the first buffer is routed to the LPS. If an incorrect length indication is received with the residual count equal to zero, the terminal operator has entered a message too long to fit into the buffers provided. The 'insufficient buffers' bit (bit 11) in the error halfword is set and the buffers filled are routed to the LPS.

Error recovery procedures are entered if a transmission error is indicated in the status or sense bits. Appropriate error bits are set in the error halfword and a message is provided either to the system console through the Message Writer subtask or to an operator control terminal if Operator Awareness is specified in the Message Control Program.

After completion of the error recovery procedures, the buffers are returned and exit is made to the LPS Control Routine as though normal end had occurred. Specific actions taken for various errors are shown in Tables 15 and 16.

Table 15. Initial Selection Errors

Priority	Status Condition	Action
1	Channel Data Check	1
2	Unit Exception	2
2	Chaining Check	2
7	Unit Check with Bus Out	4
9	Program Check	1
10	Protection Check	1

Table 16. Unit Check in Status

Priority	Sense Byte Condition	Action
5	Equipment Check-Read	5
	-All	
	Others	2
6	Intervention Required	
	-Printer	1
	-Others	2
7	Bus Out-Write	3
7	Bus Out-Read	2
8	Command Reject	1
4	All other sense bits	4

ACTIONS

1. An operator message is provided.
2. An operator message indicating a 'should not occur' error is provided.
3. On the second occurrence, an operator message is provided. Retry of the channel program is initiated by EOBLC.
4. The channel program is retried. On the second occurrence an operator message is provided.
5. An ERASE CCW is executed and an operator message is provided.

MESSAGE WRITER INITIATOR ROUTINE

Module Name: IJLQMW (Chart MW)

Entry Points: The Message Writer Initiator routine may be entered:

1. From the Line Appendage module when an error message is to be written on the system log. Entry is at IJLQMW. Line Appendage passes the address of the buffer in register 2, the address of the DTF table for the line in register 3, and the address of the LCB for the line in register 4.

2. At a log appendage within the routine that is entered upon an interrupt from the system log. Entry is at LOGAP.

3. From the 7772 Disk or Audio Line Appendage module when the sending of an error message is requested by the audio devices. Note that in this case, the Audio ERP Message Writer will be loaded via a FETCH in the logical transient area.

Function: Builds an information block containing all data needed to write the error message and places an ECB on the Ready Queue to cause the transient Message Writer routine to be called via a FETCH to write the message.

Upon entry from Line Appendage, the routine tests to determine if any one of six information blocks provided is available for use. If not, the routine places the buffer on a waiting queue and returns to Qdispatch. If an information block is available, the routine moves all data needed to write the error message from the LCB into the information block.

The routine tests to determine if a system log I/O operation is taking place. If it is, the routine returns to Line Appendage. Otherwise, the routine inserts a special ECB on the Ready Queue, and then returns to Line Appendage.

The LPS Control routine recognizes the ECB, and calls Phase 1 of the QTAM Message Writer routine into the logical transient area via a FETCH, passing to it the address of the information block.

Upon entry at the log appendage (LOGAP), the routine examines the information blocks to determine if another message is to be sent.

If not, the routine branches to Qdispatch. Otherwise, the special ECB is again placed on the Ready Queue before exit is made to Qdispatch.

External Routines Used: None.

Chart A1. ARU Internal and ARU Receive Routines

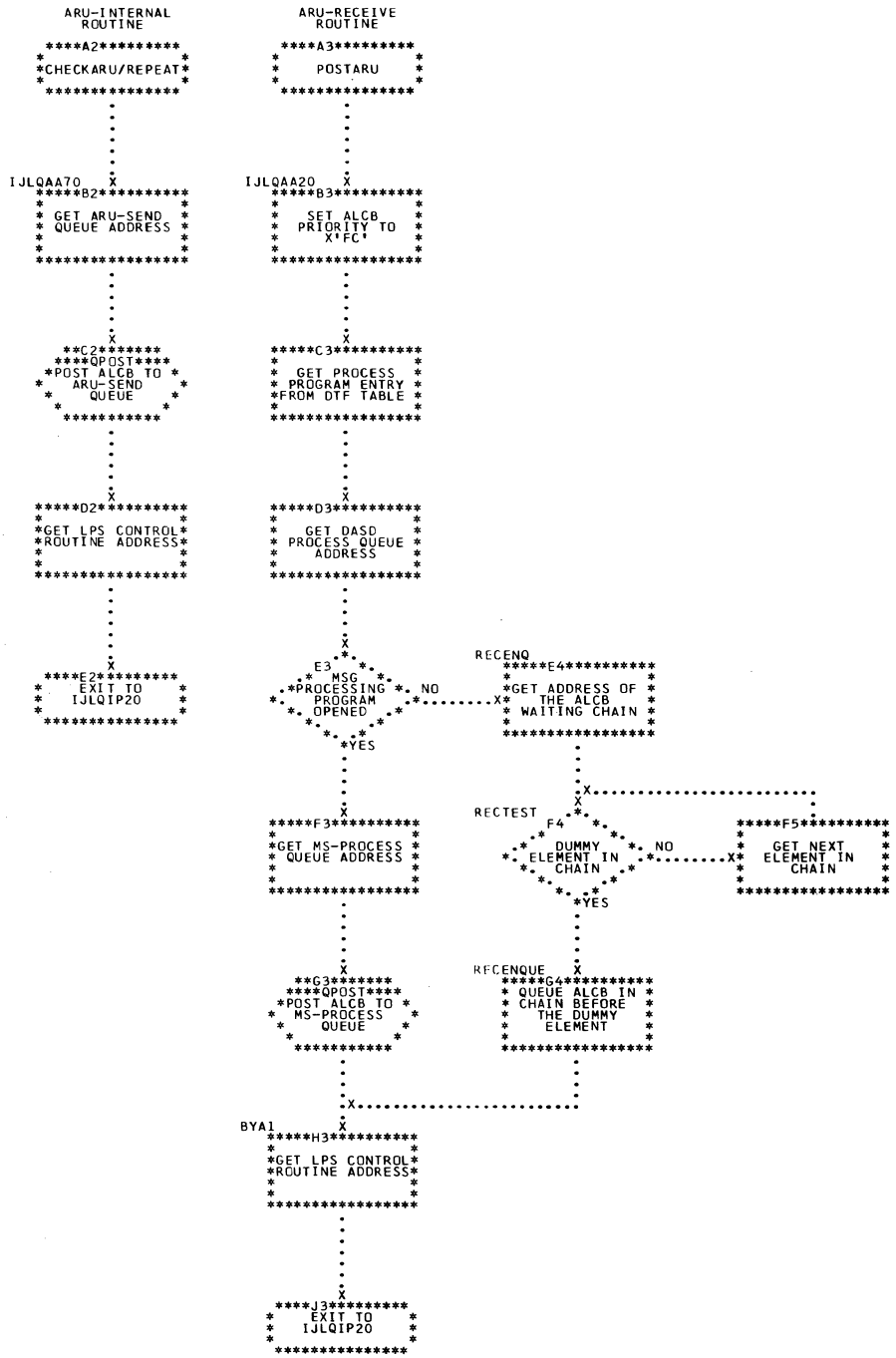


Chart A2. ARU Send Routine

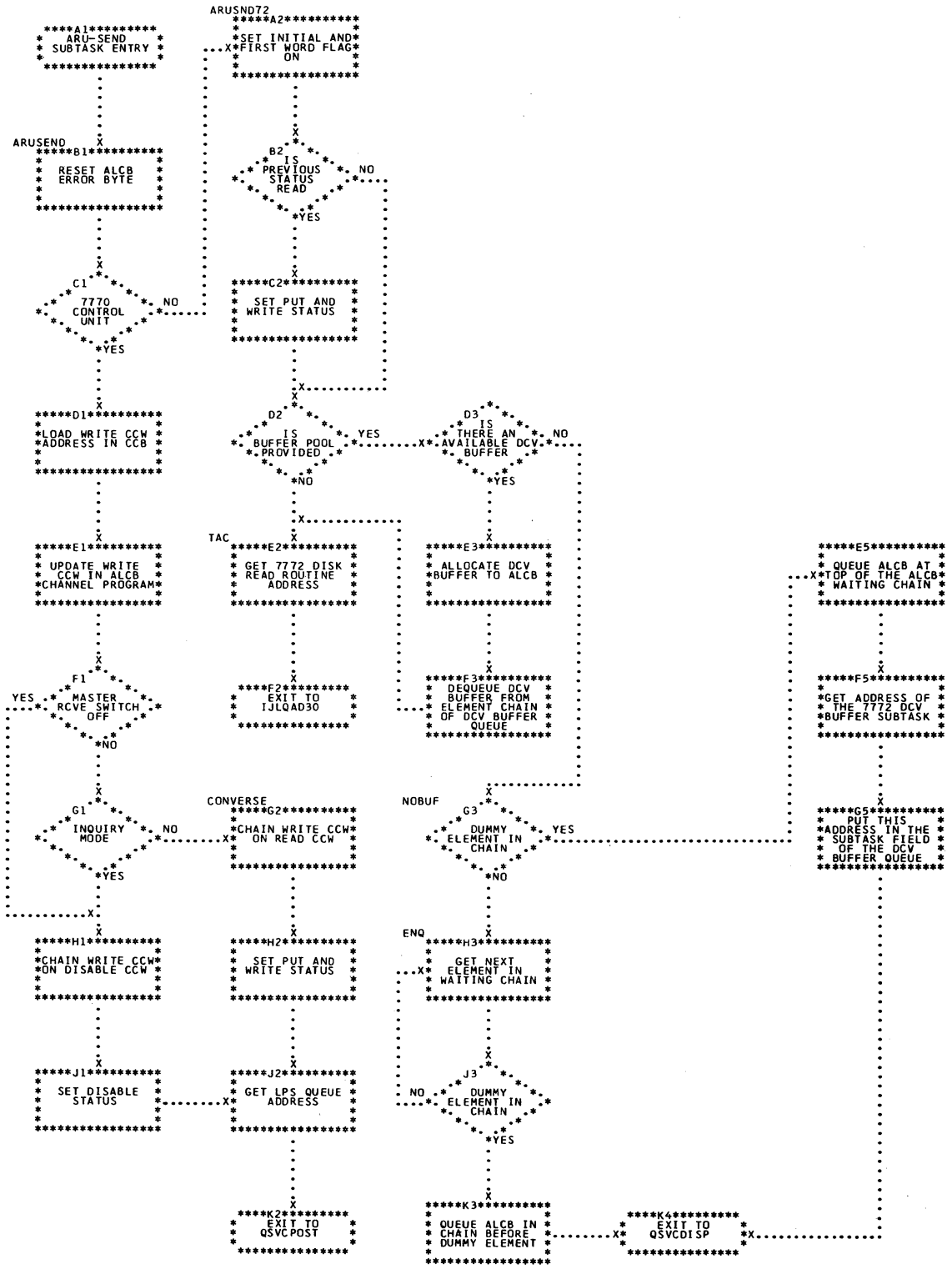


Chart A3. Analysis and IBM 7772 Line PCI Routines

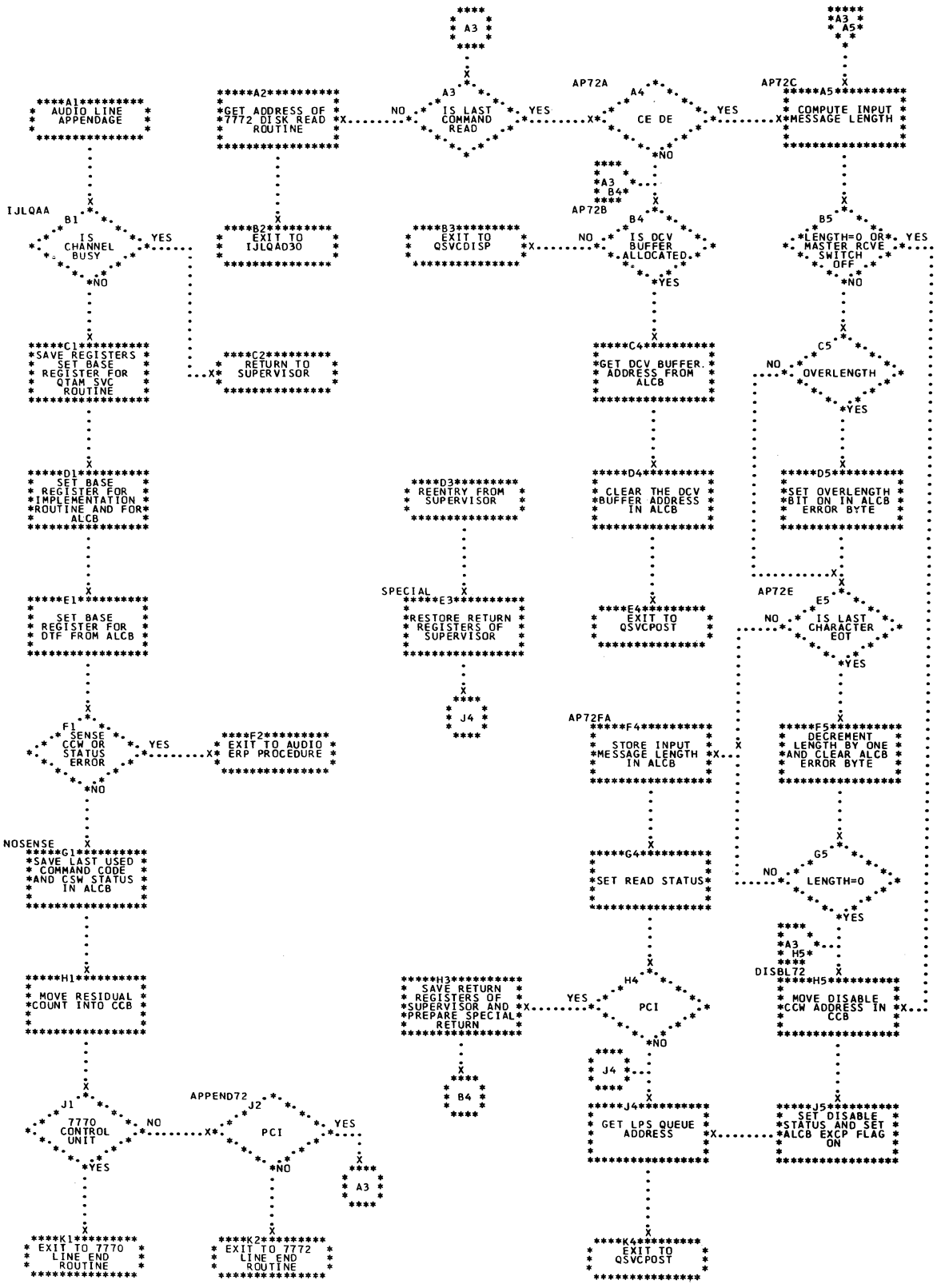


Chart A4. IBM 7770 Line End Routine

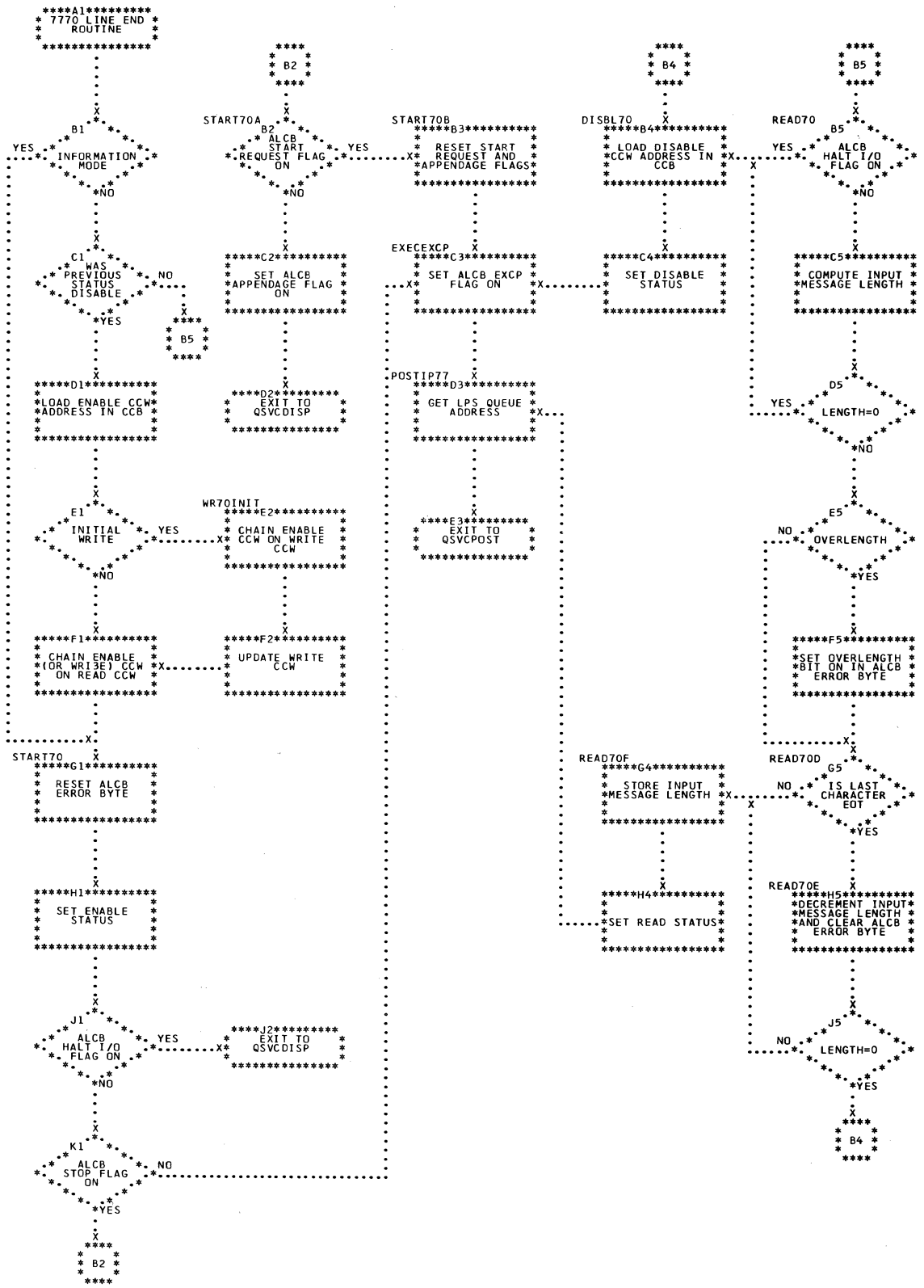


Chart A5. IBM 7772 Line End Routine

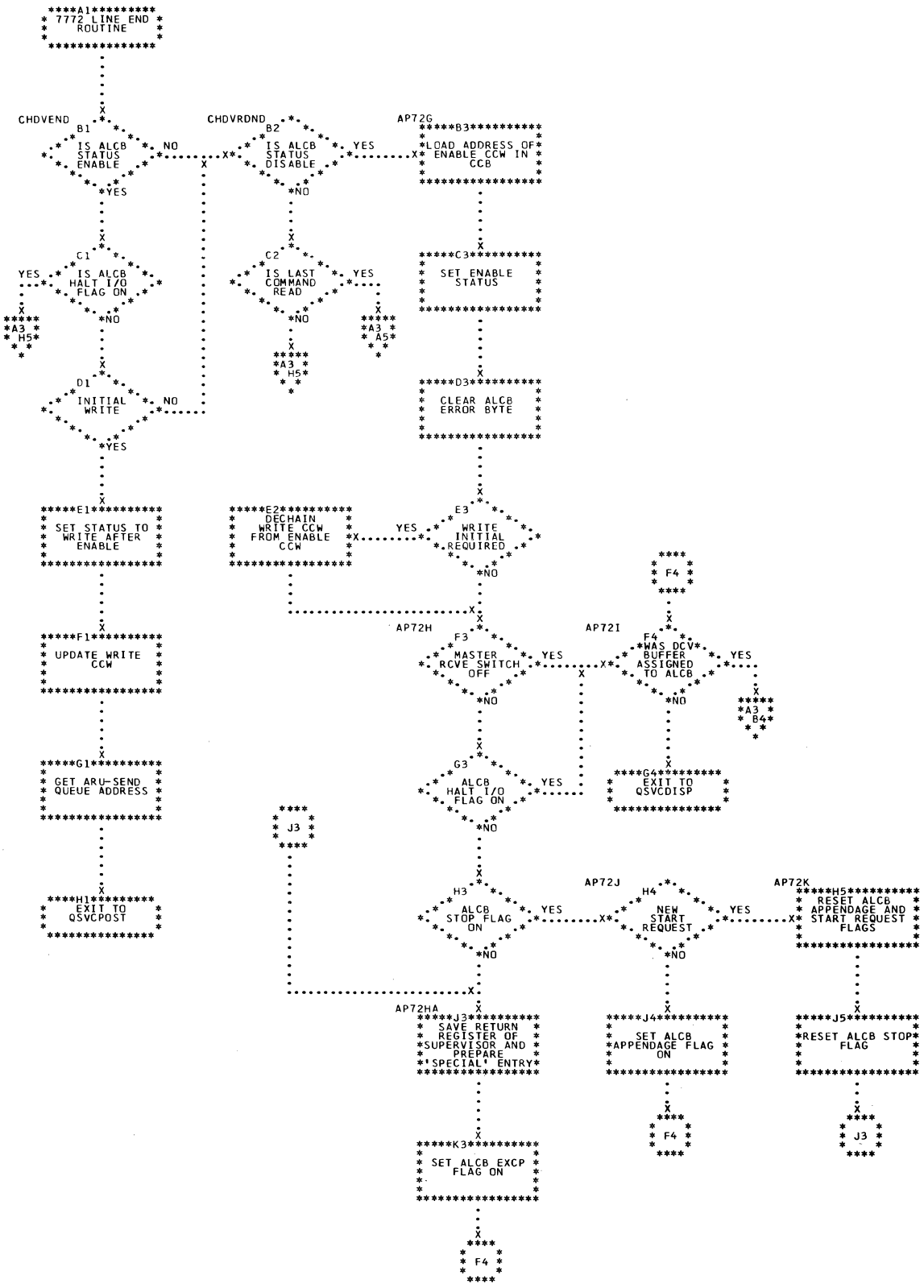


Chart BO. Breakoff Routine

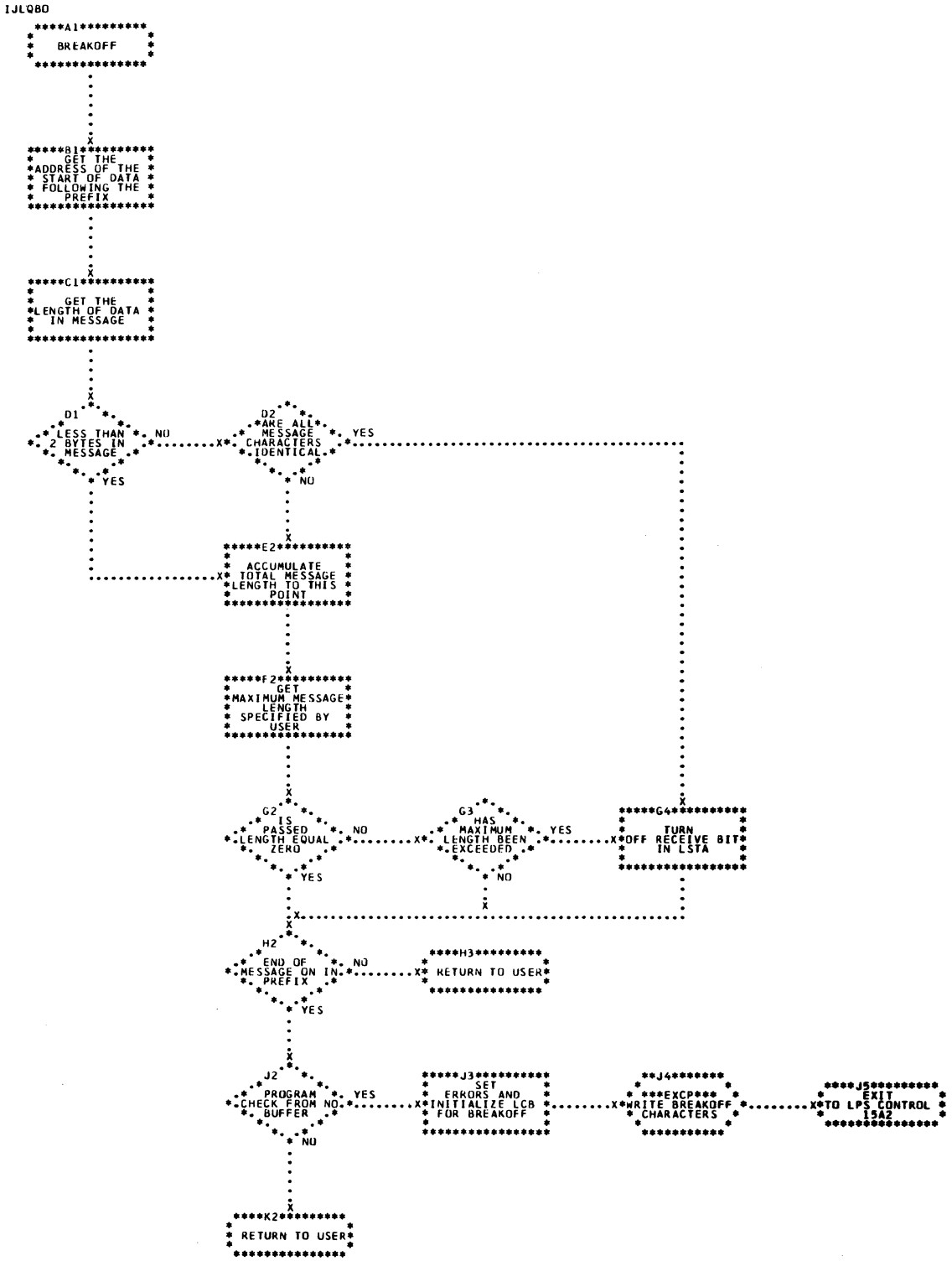


Chart CC. Copy Line Error Counters Routine

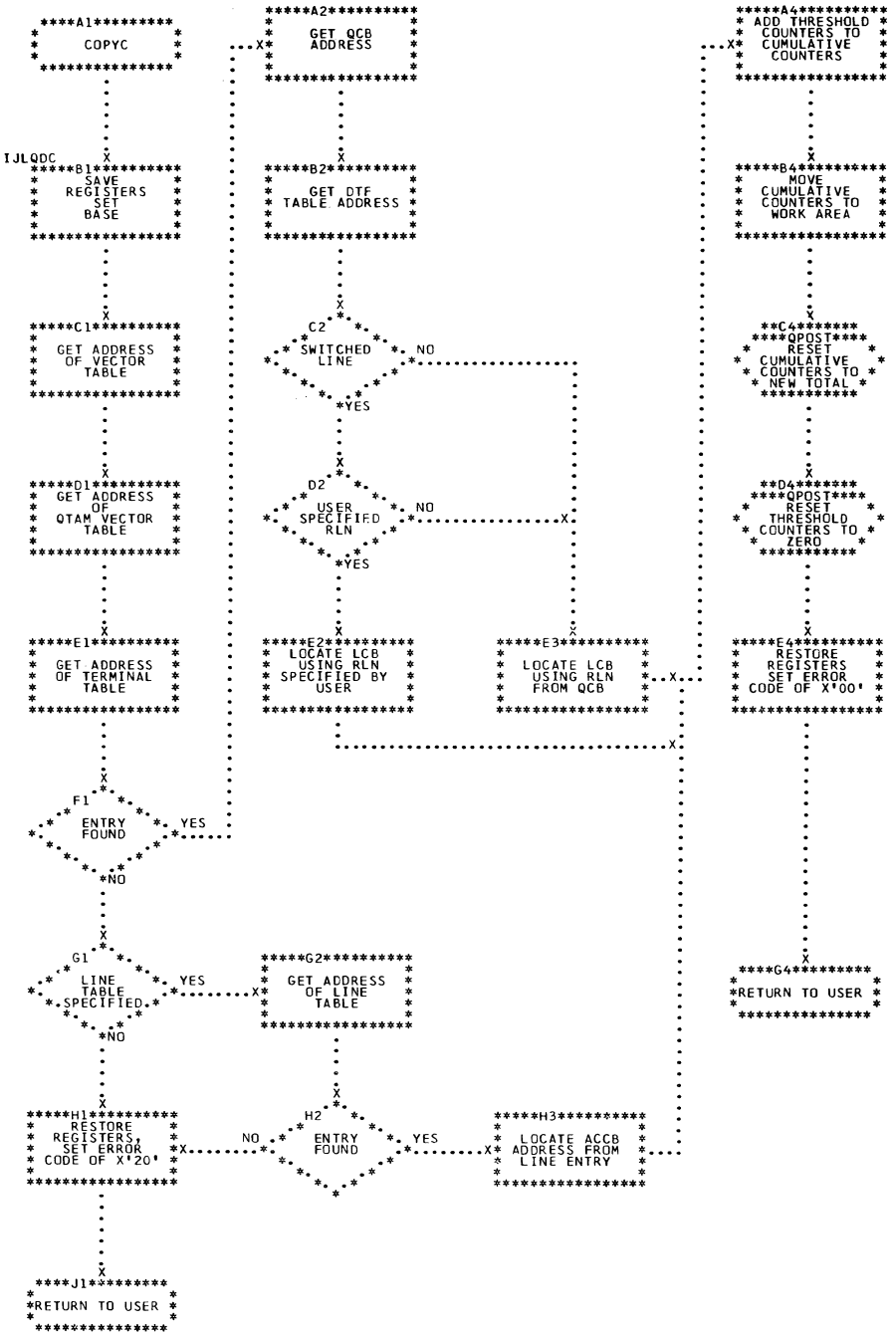


Chart CK. Checkpoint Routine

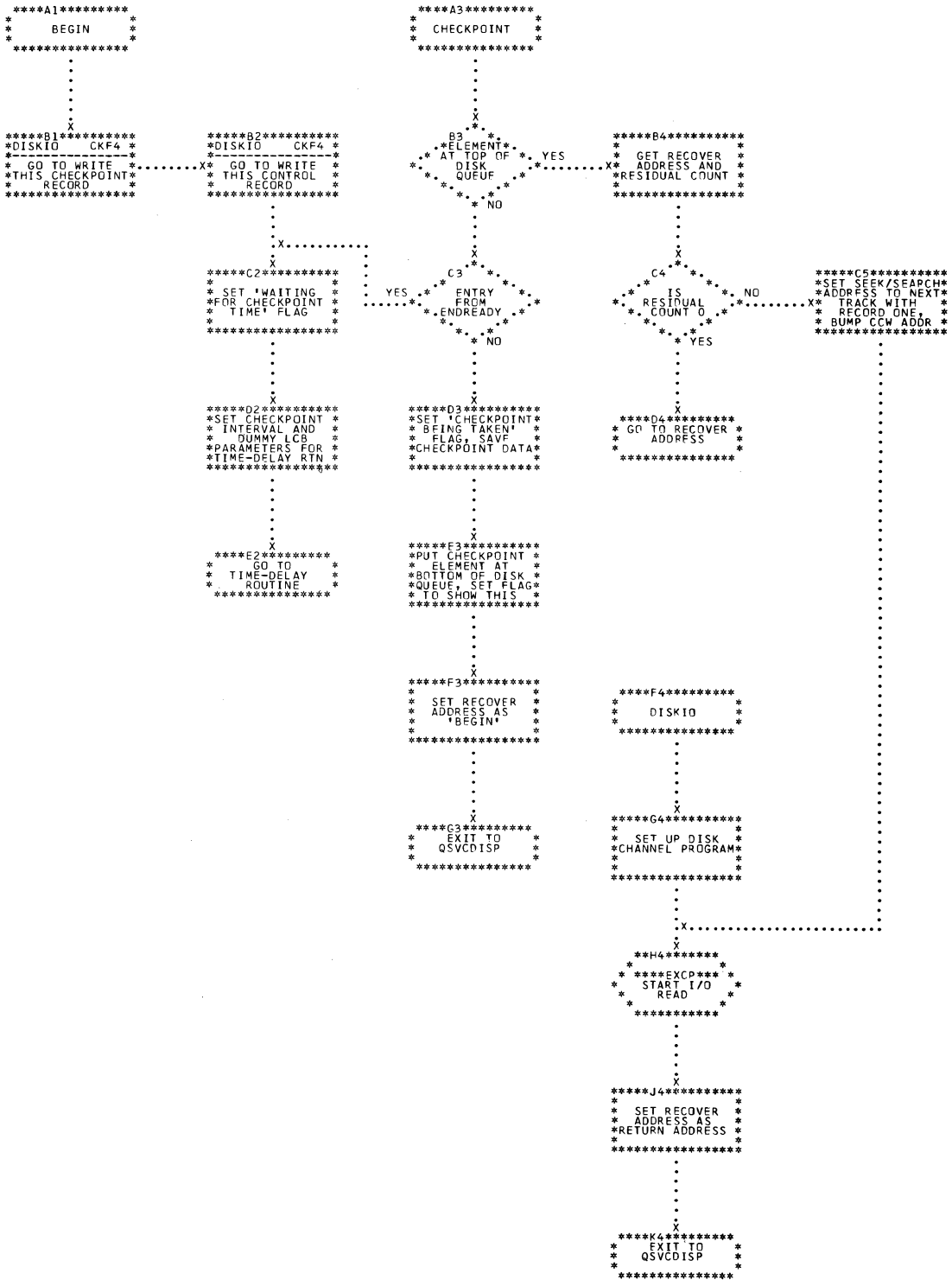


Chart CL. Change Line

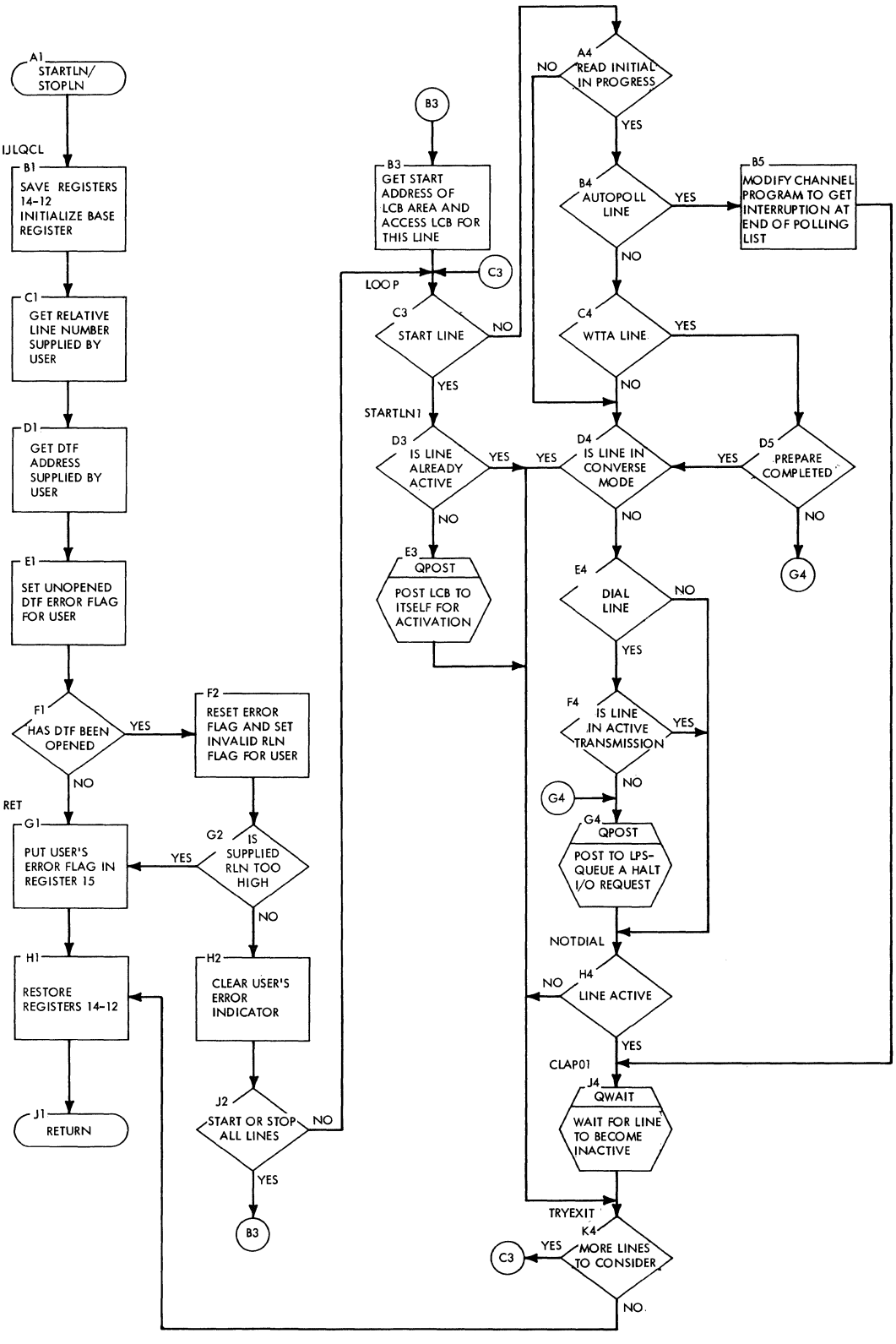


Chart CM. Cancel Message Routine

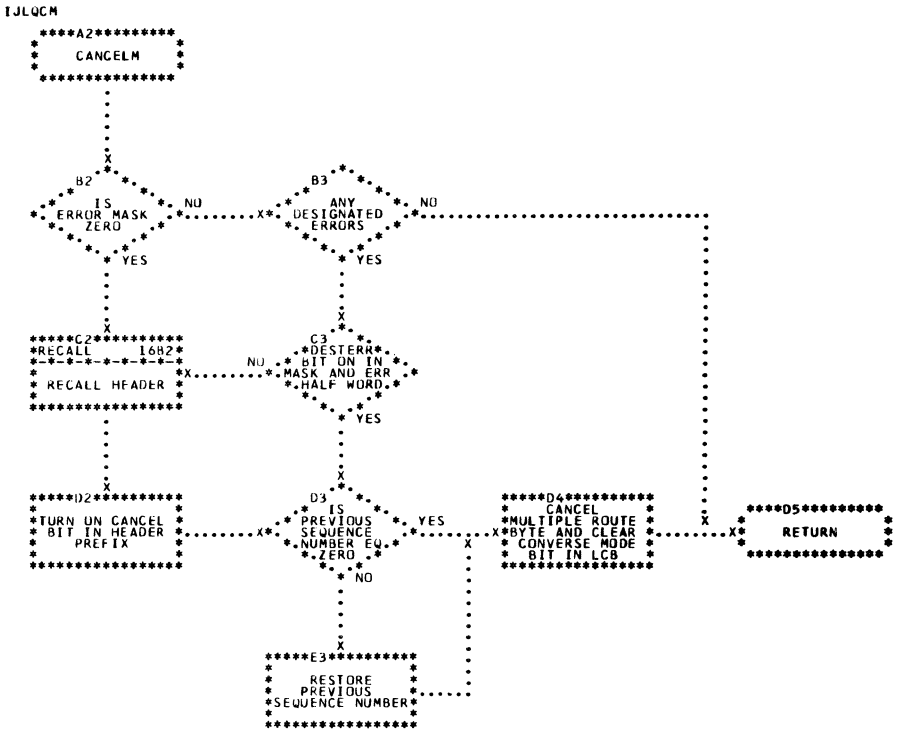


Chart CP. Change Polling List Routine

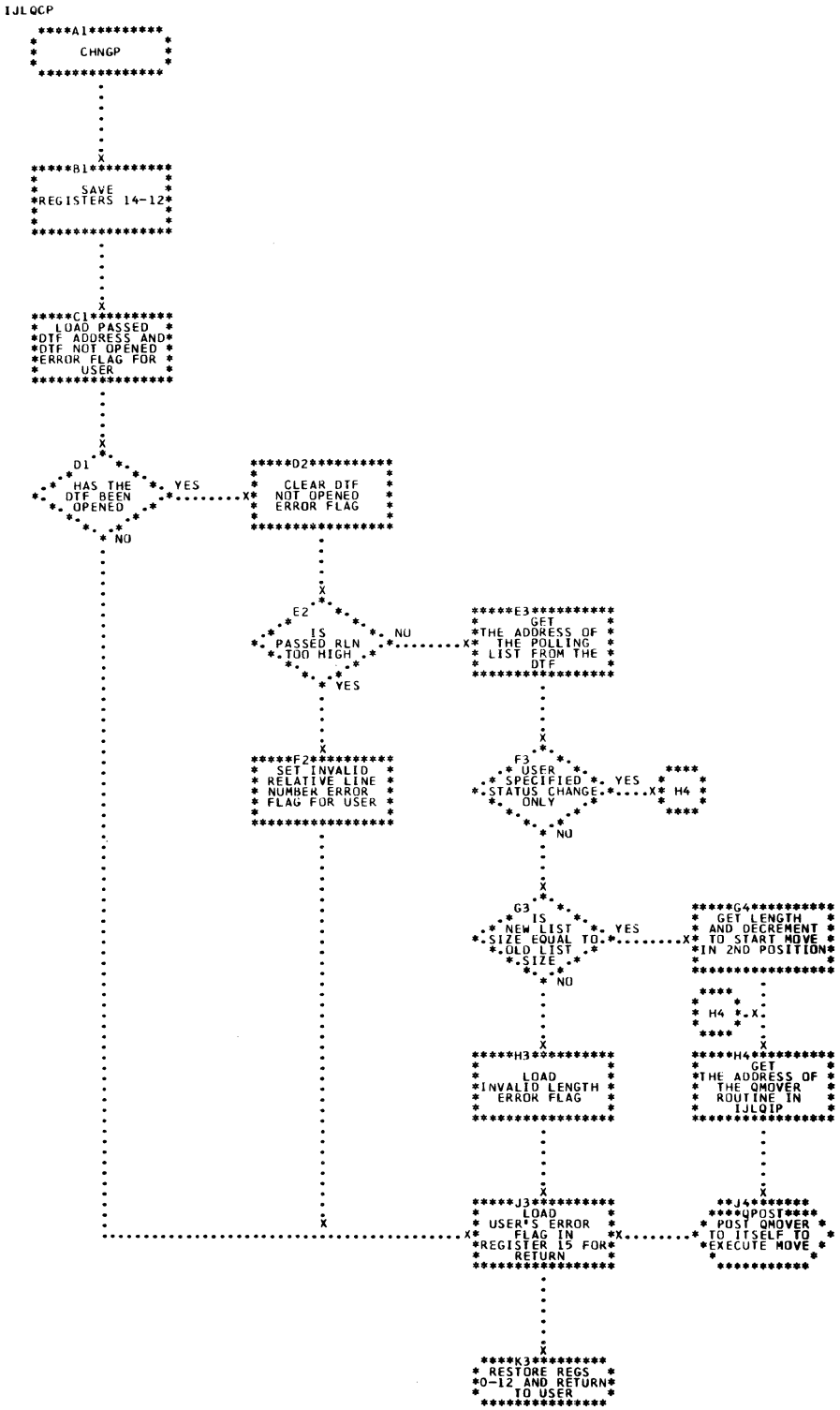


Chart CR. Checkpoint-Restart Routine

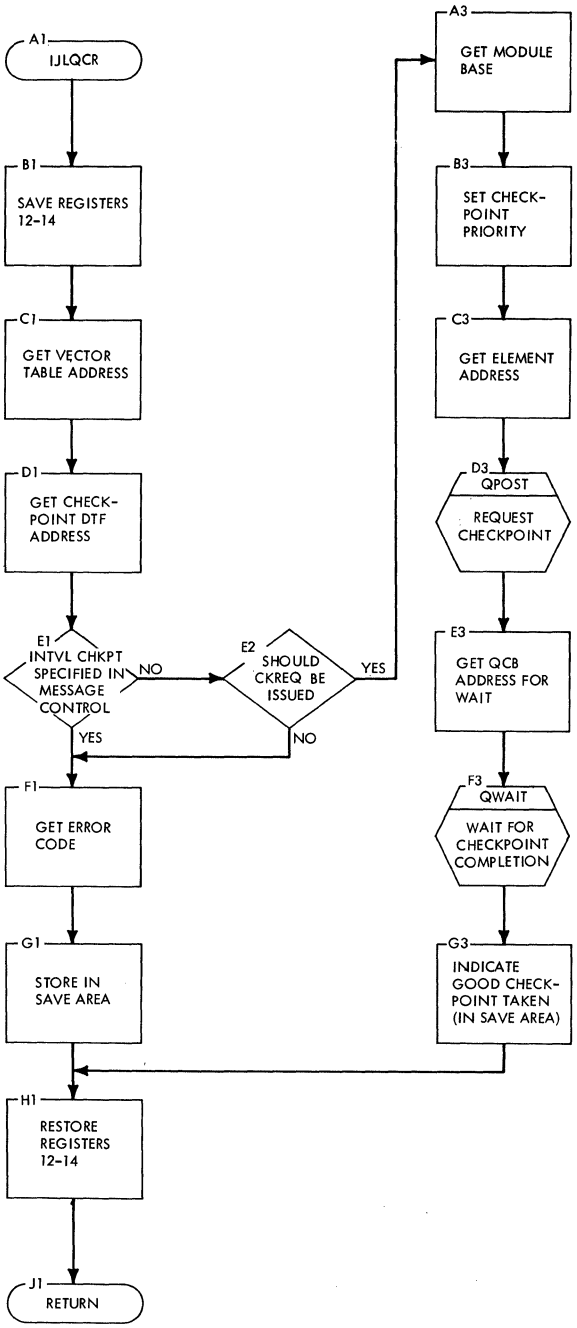


Chart CT. Change Terminal Table Entry Routine

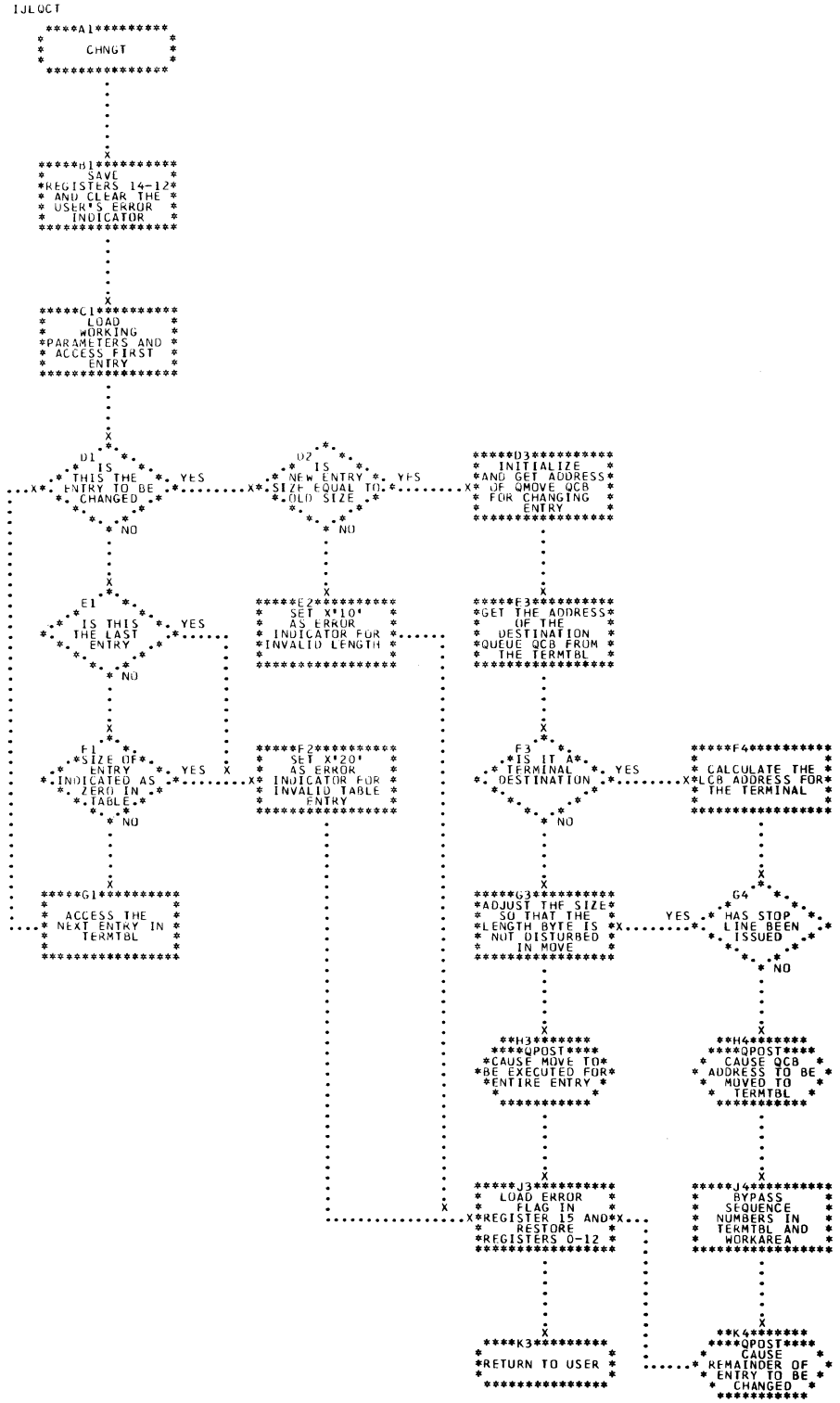


Chart C1. QTAM Close Routine (Phase 1)

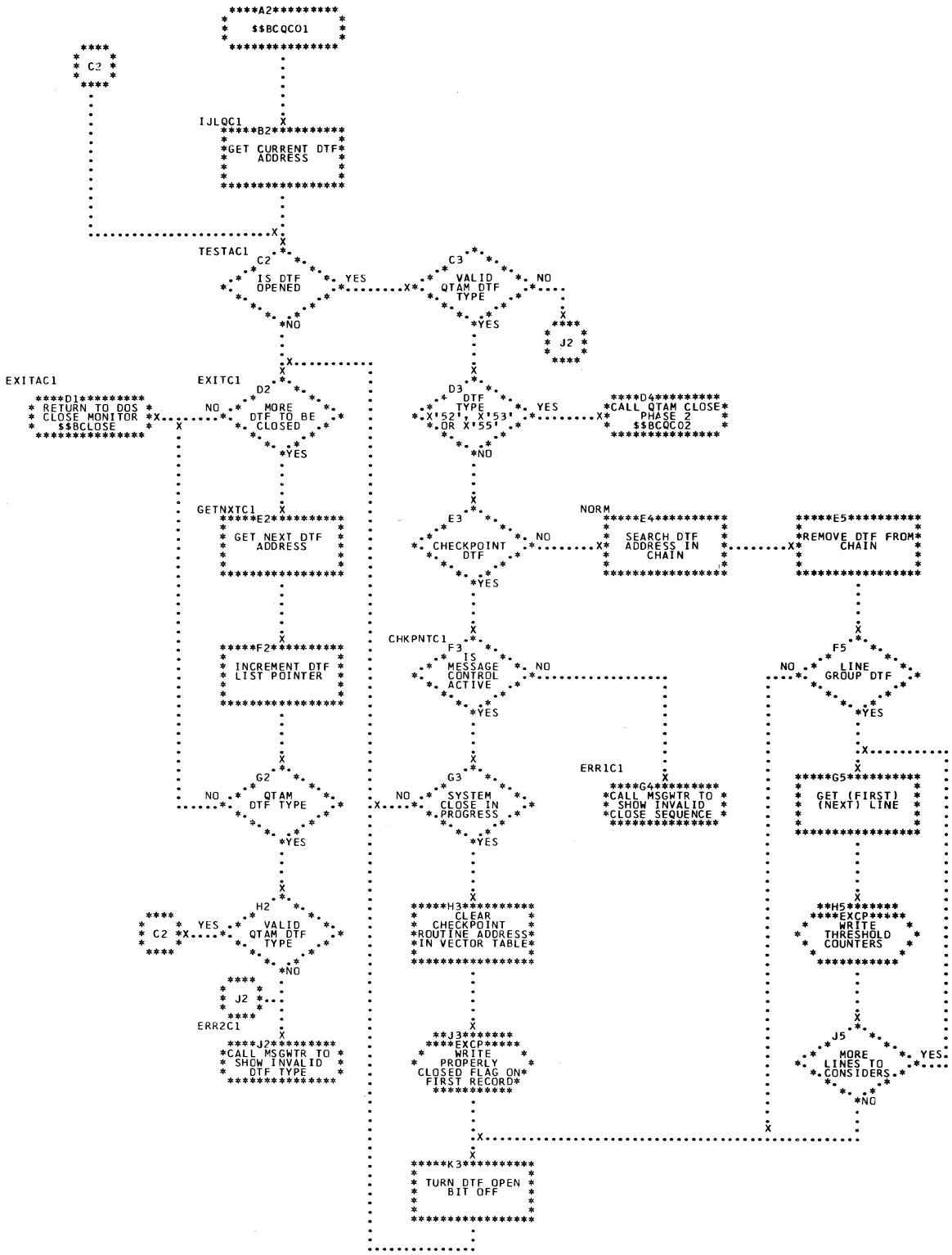


Chart C2. QTAM Close Routine (Phase 2)

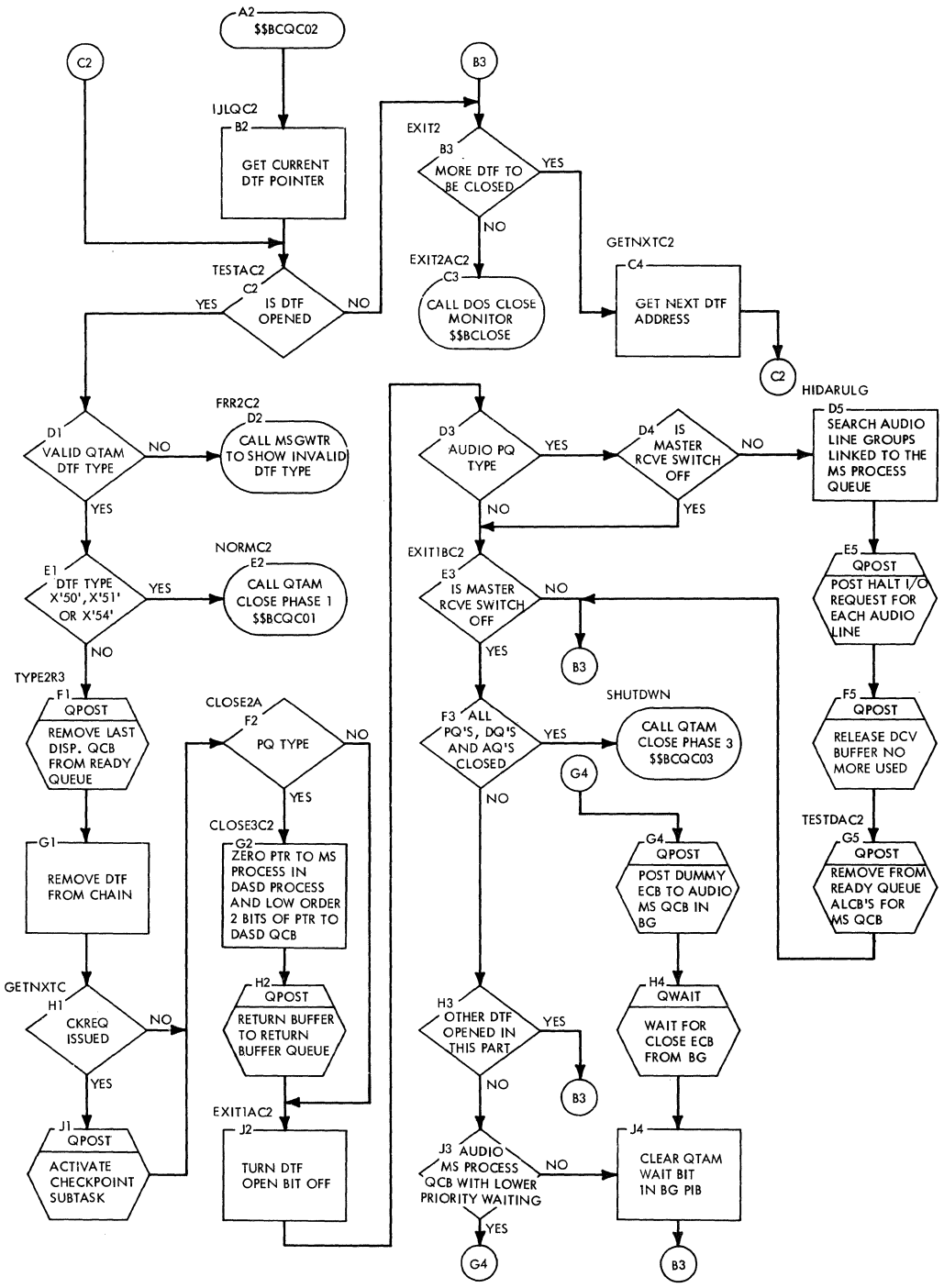


Chart C3. QTAM Close Routine (Phase 3)

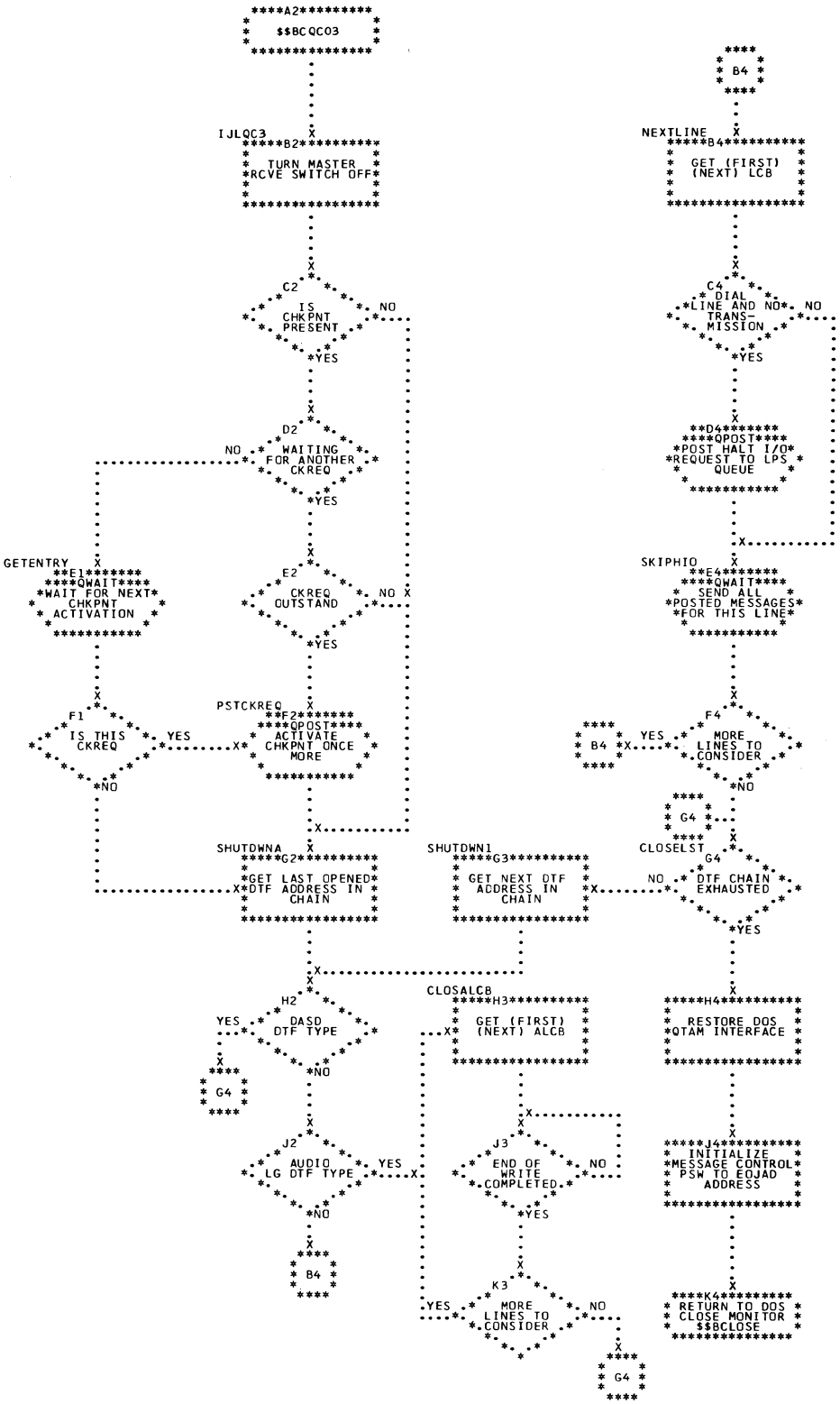


Chart DA. Disk-End Appendage (Part 1 of 2)

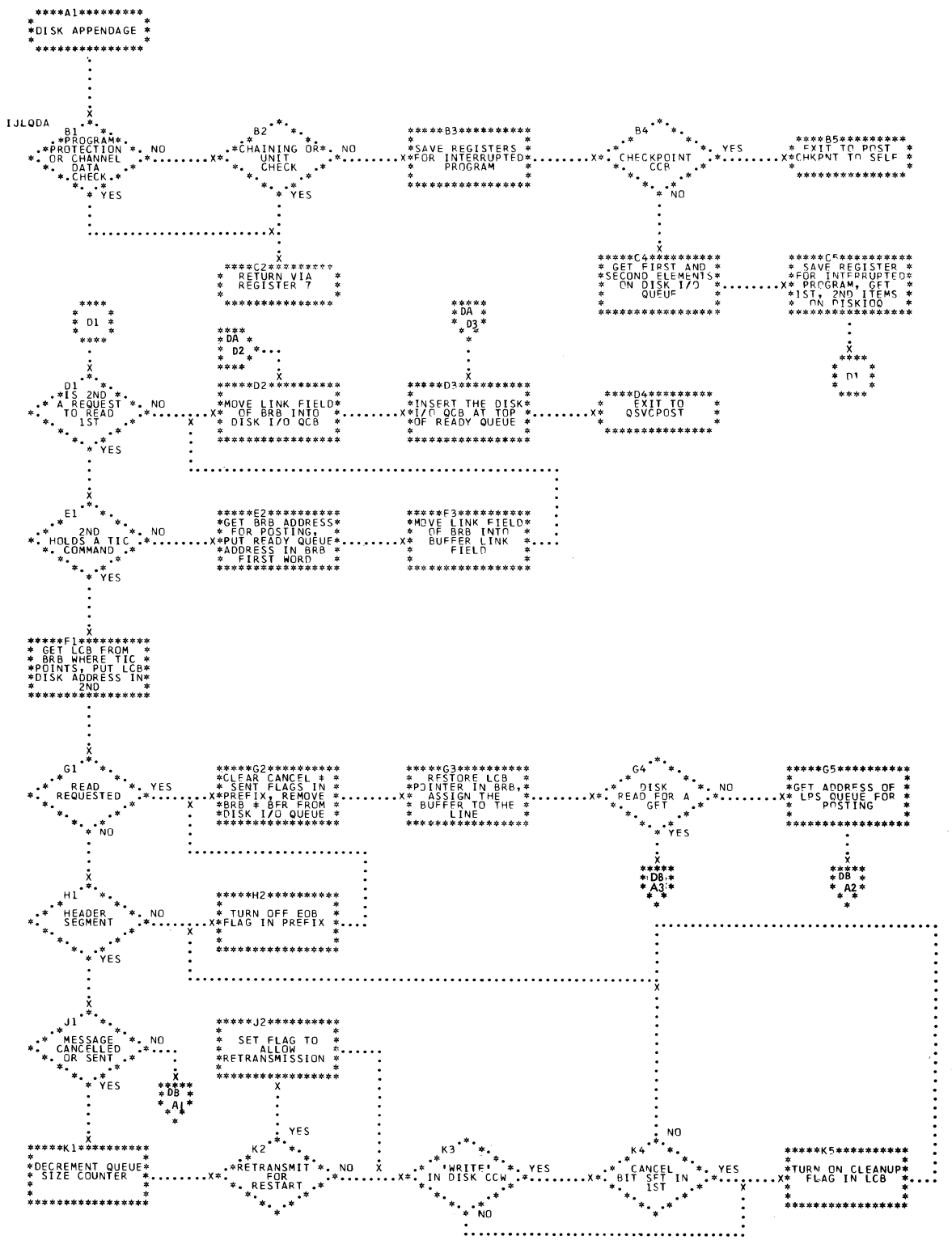


Chart DB. Disk-End Appendage (Part 2 of 2)

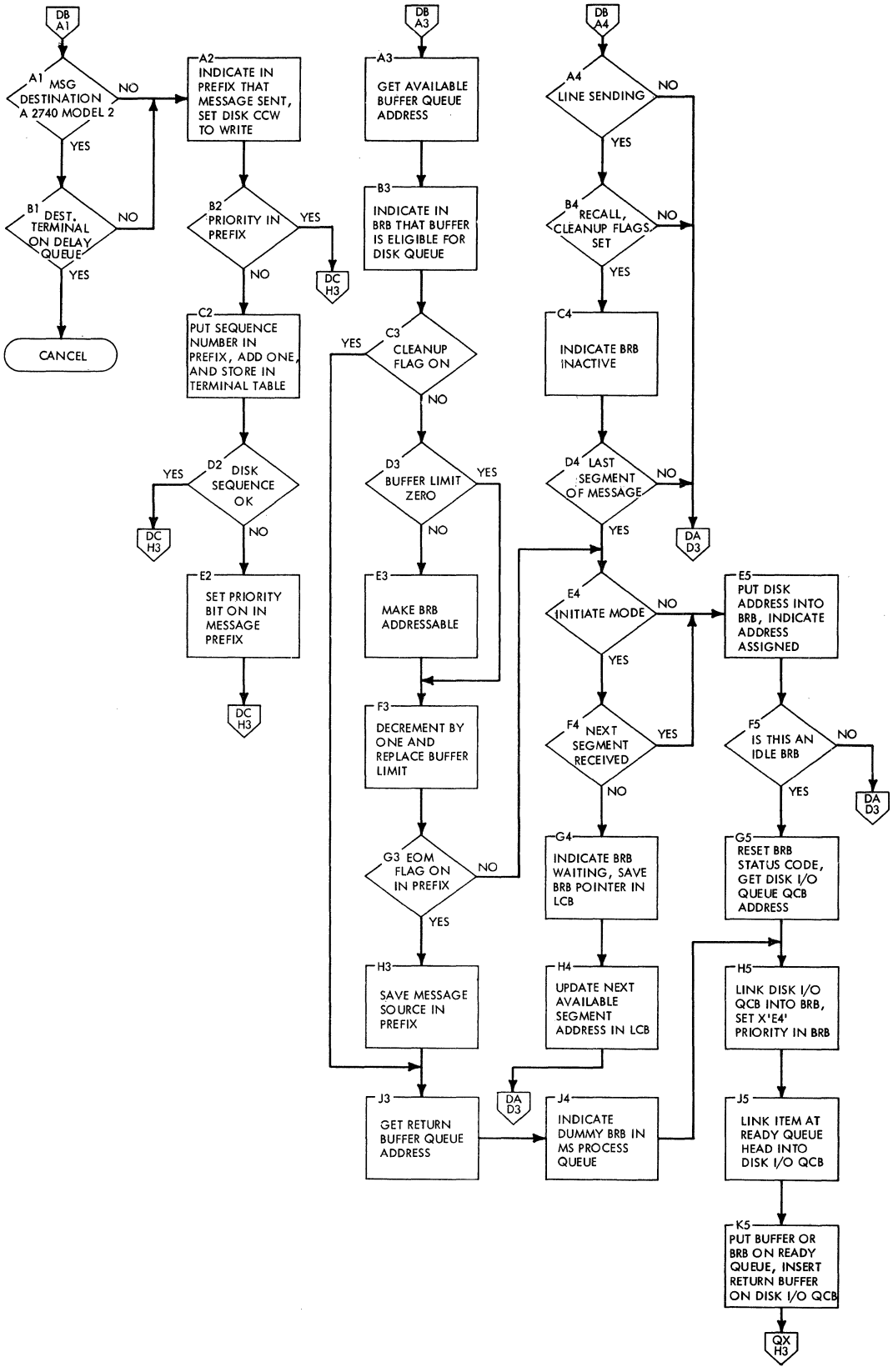


Chart DC. Disk I/O Routine

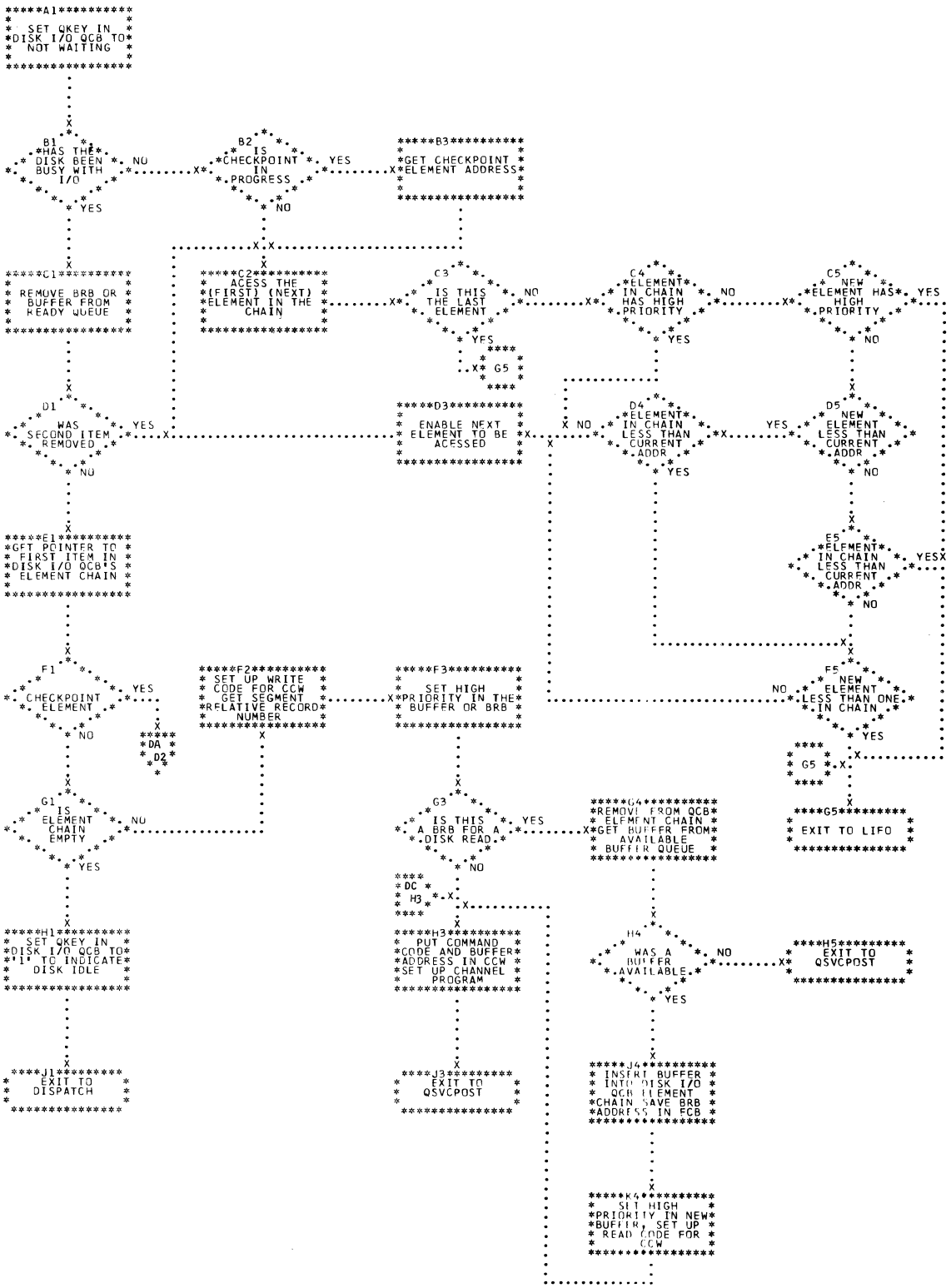


Chart DE. Copy Terminal Table Entry Routine

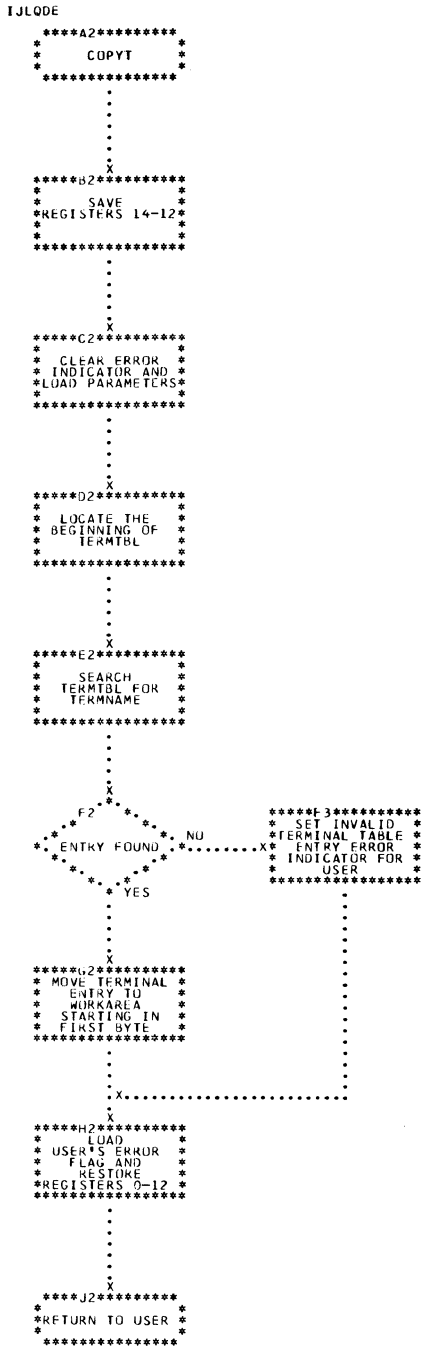


Chart DL. Distribution List Routine

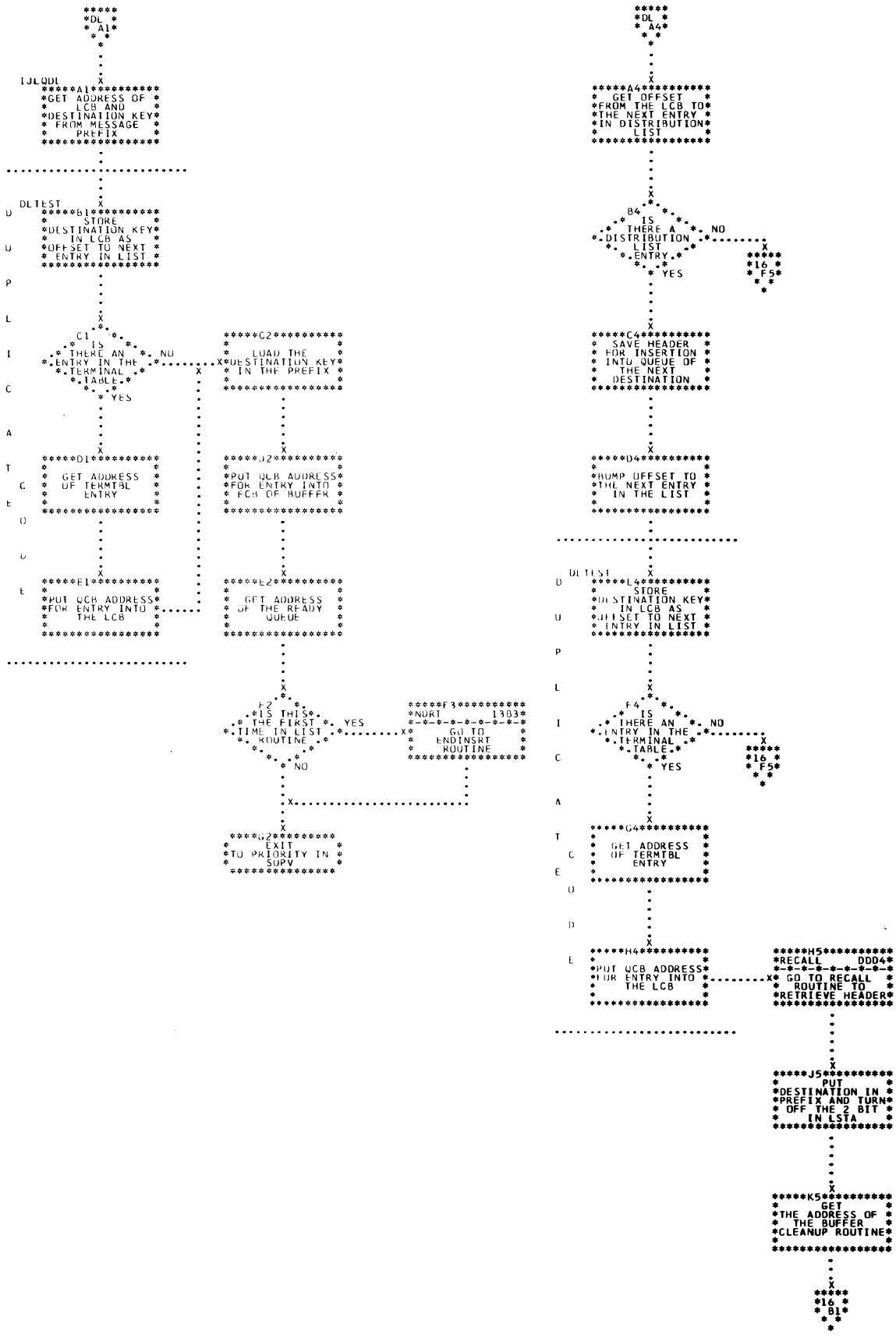


Chart DP. Copy Polling List Routine

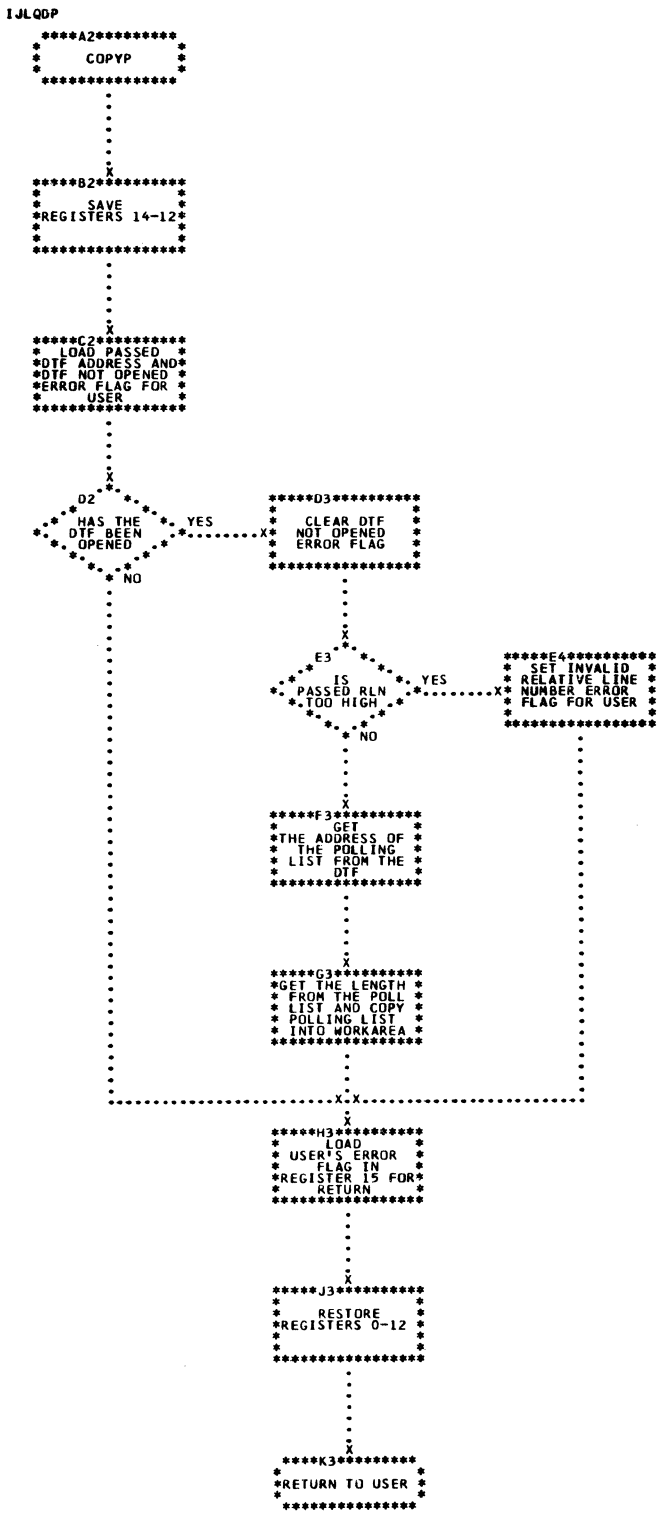


Chart DQ. Copy Queue Control Block Routine

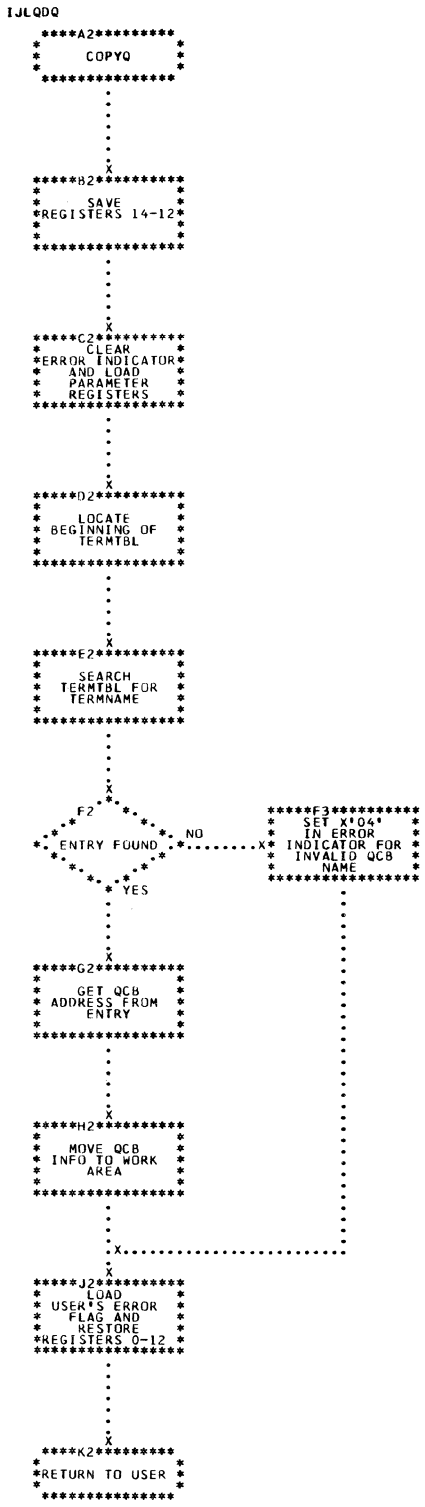


Chart DT. Date Stamp Routine

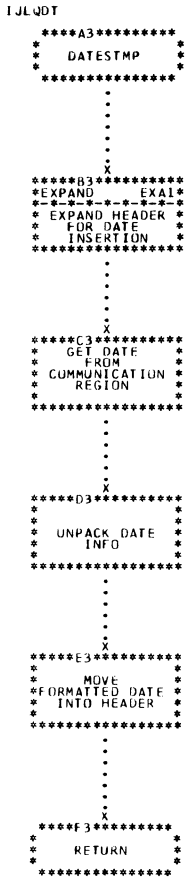


Chart D1. IBM 7772 Disk End Appendage

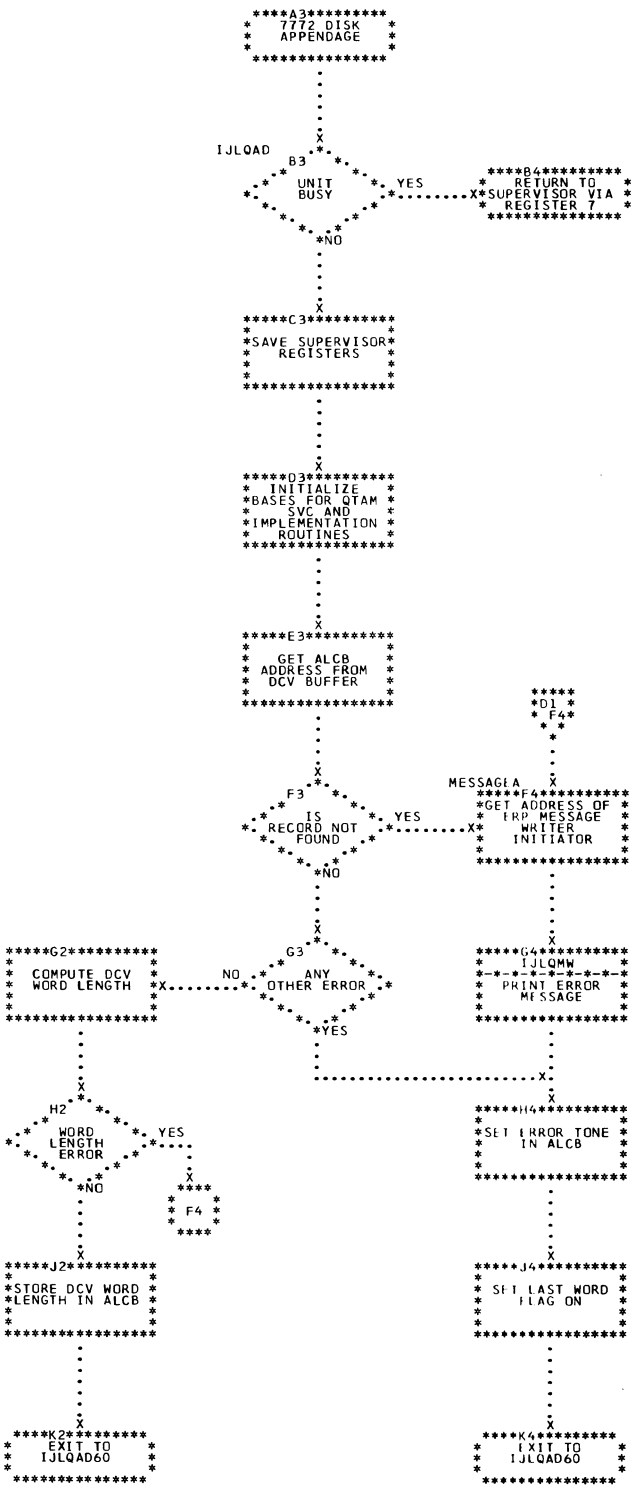


Chart D2. IBM 7772 Disk Read Routine

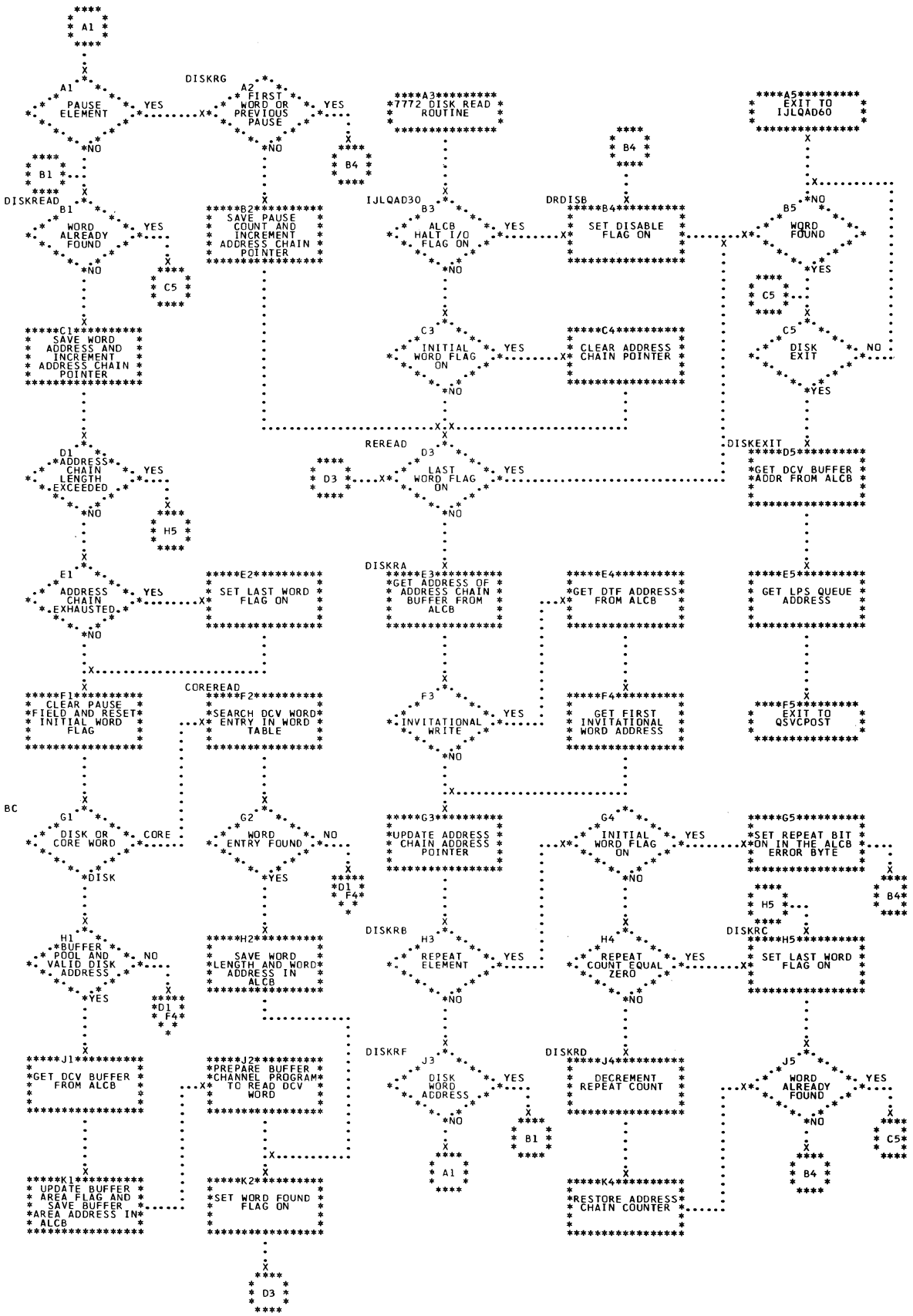


Chart D3. IBM 7772 Line Write Routine

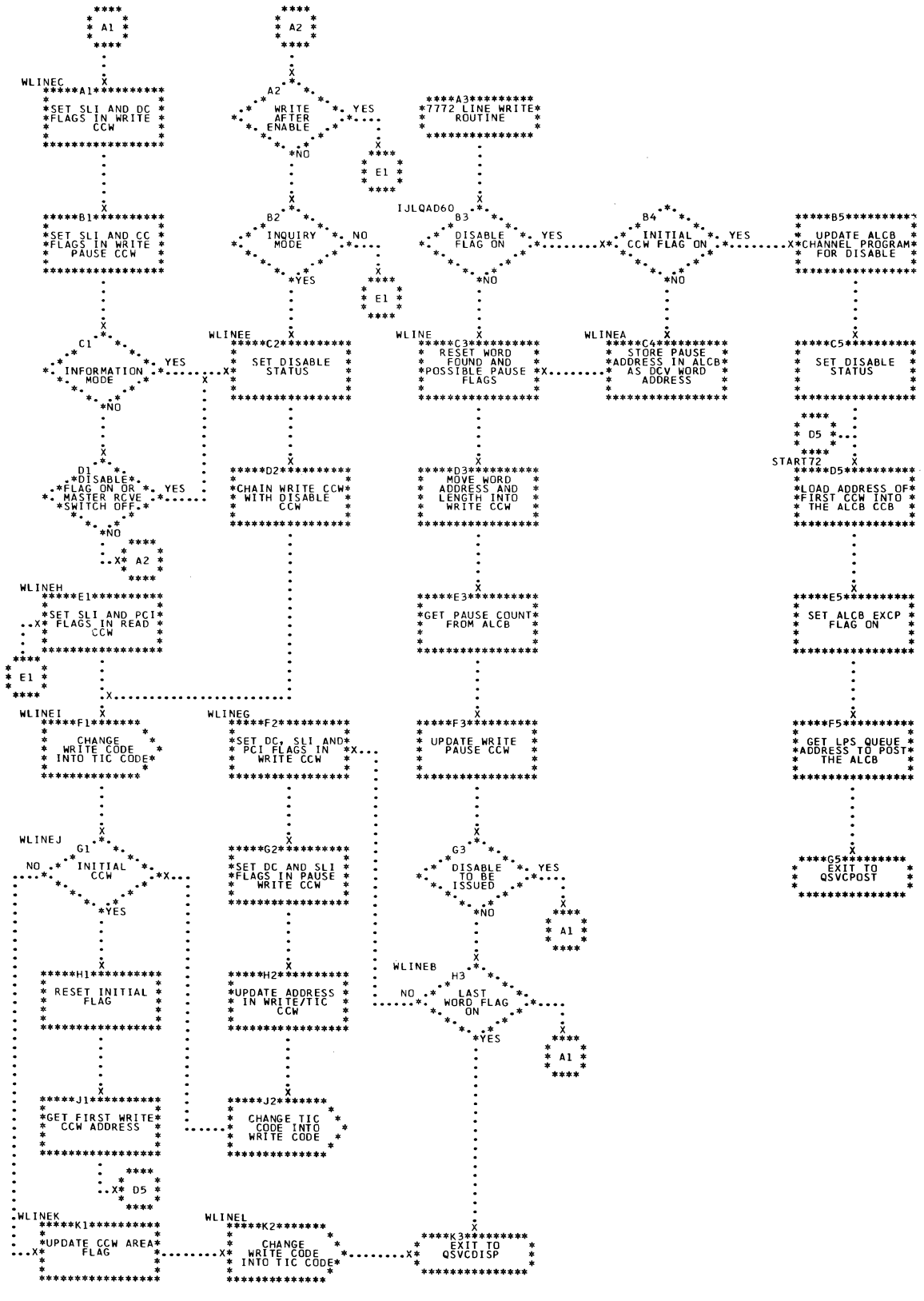


Chart D4. IBM 7772 DCV Buffer Routine

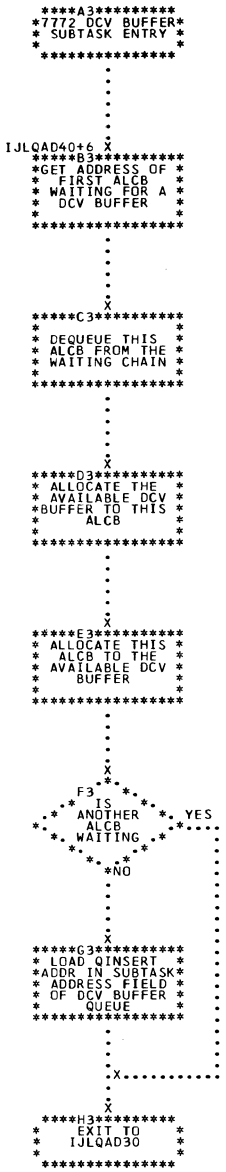


Chart EA. End-of-Address (EOA) Routine

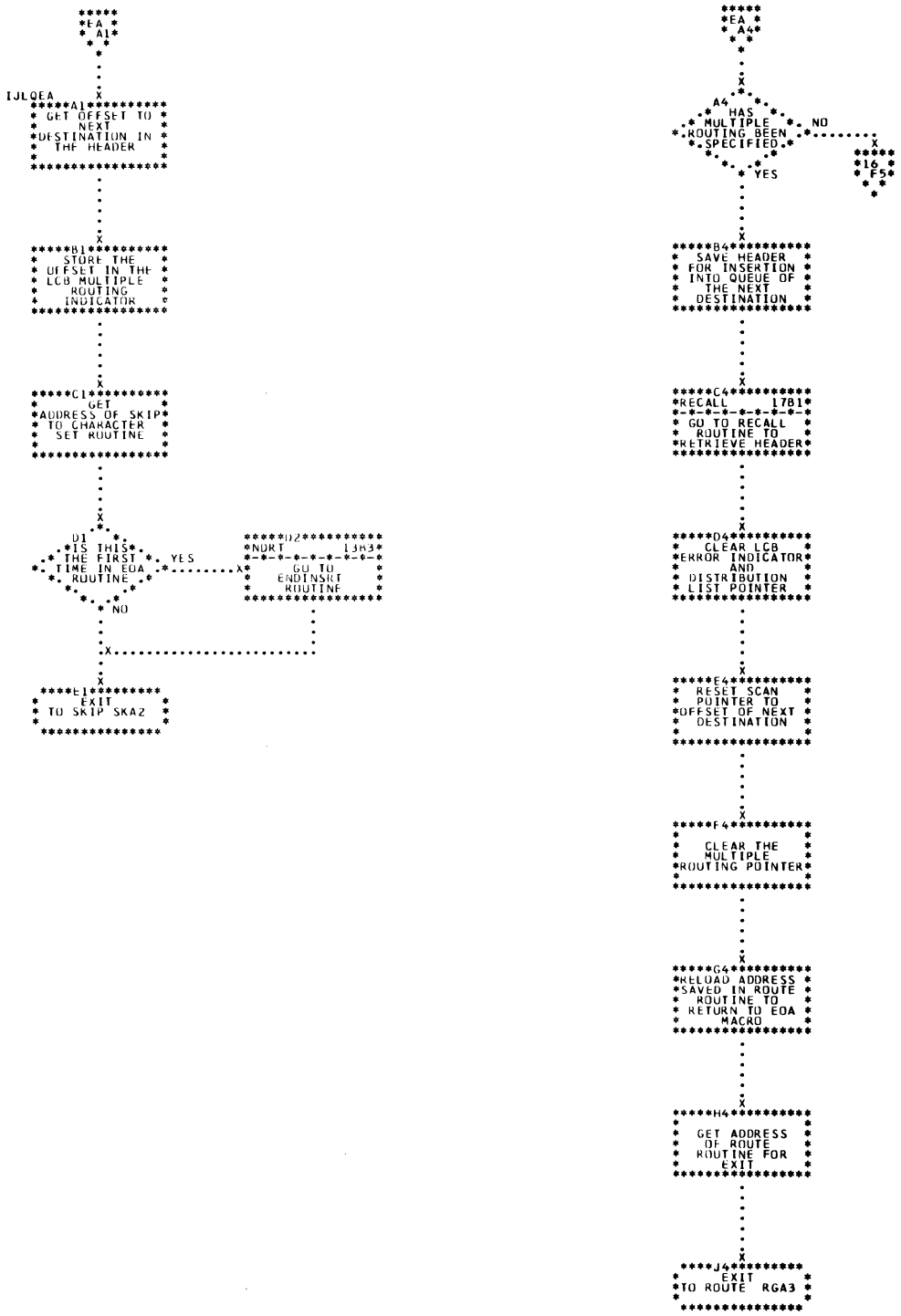


Chart EB. End-of-Block (EOB) Routine

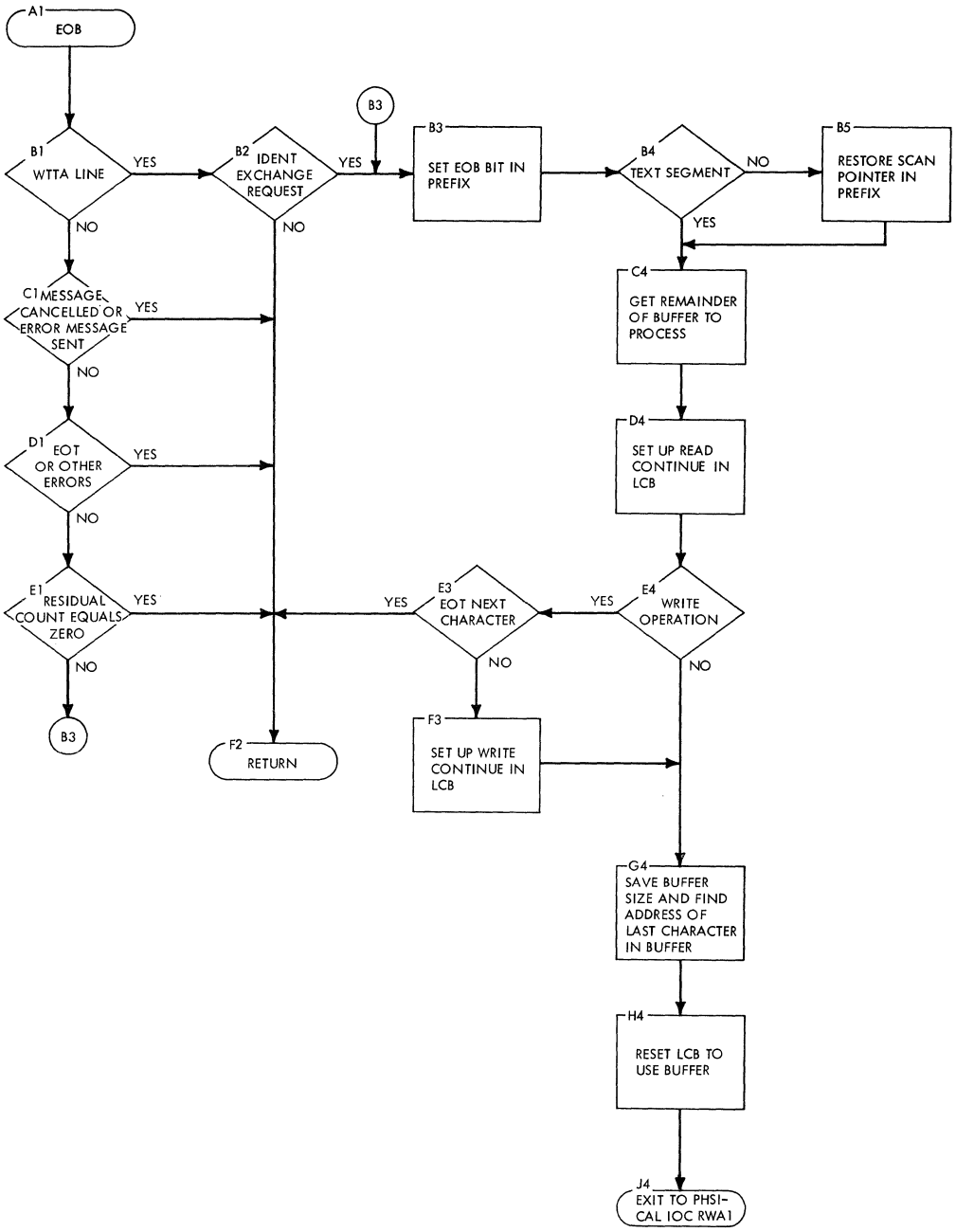


Chart EC. End-of-Block and Line Correction Routine

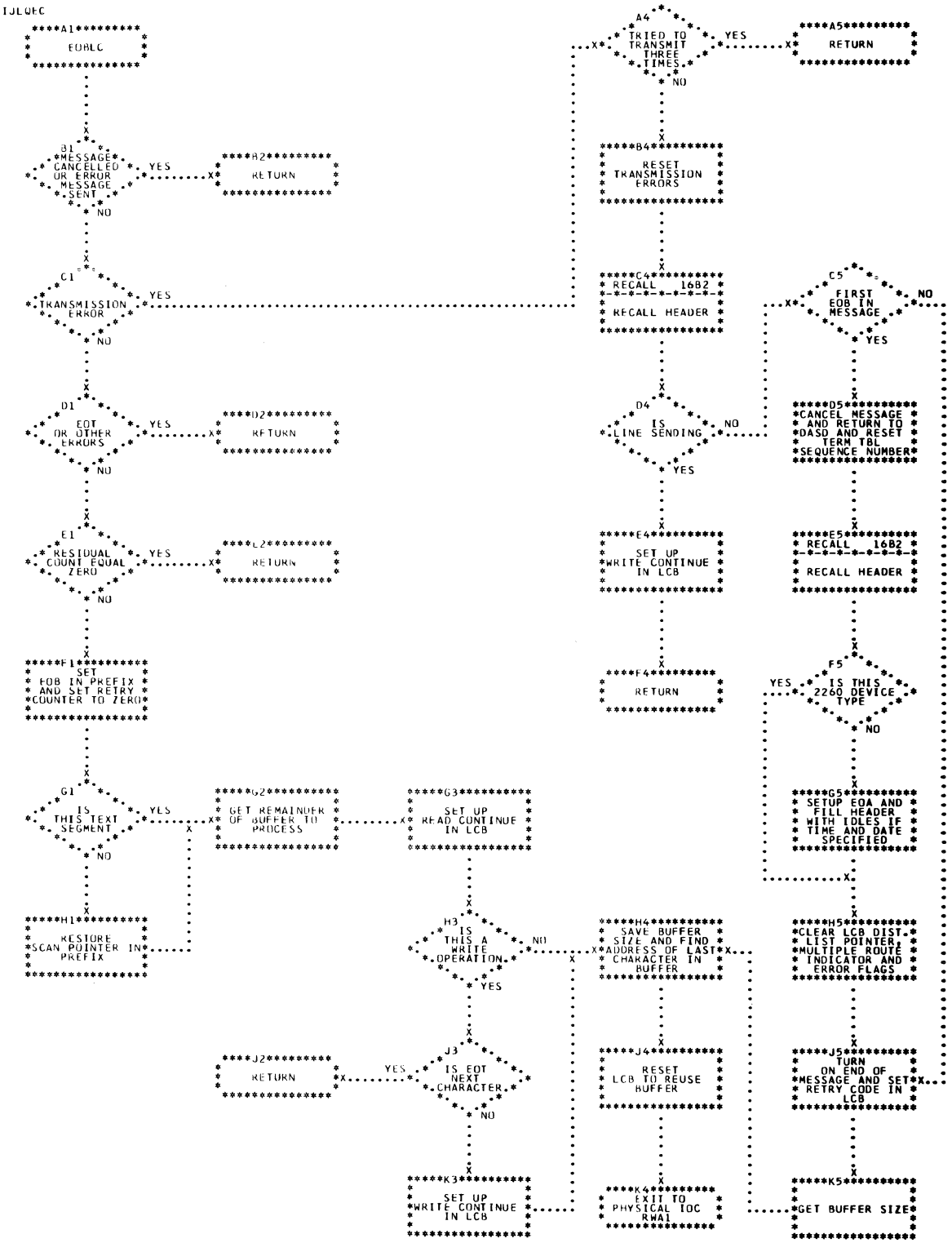


Chart EG. Line End Appendage (Part 1 of 9)

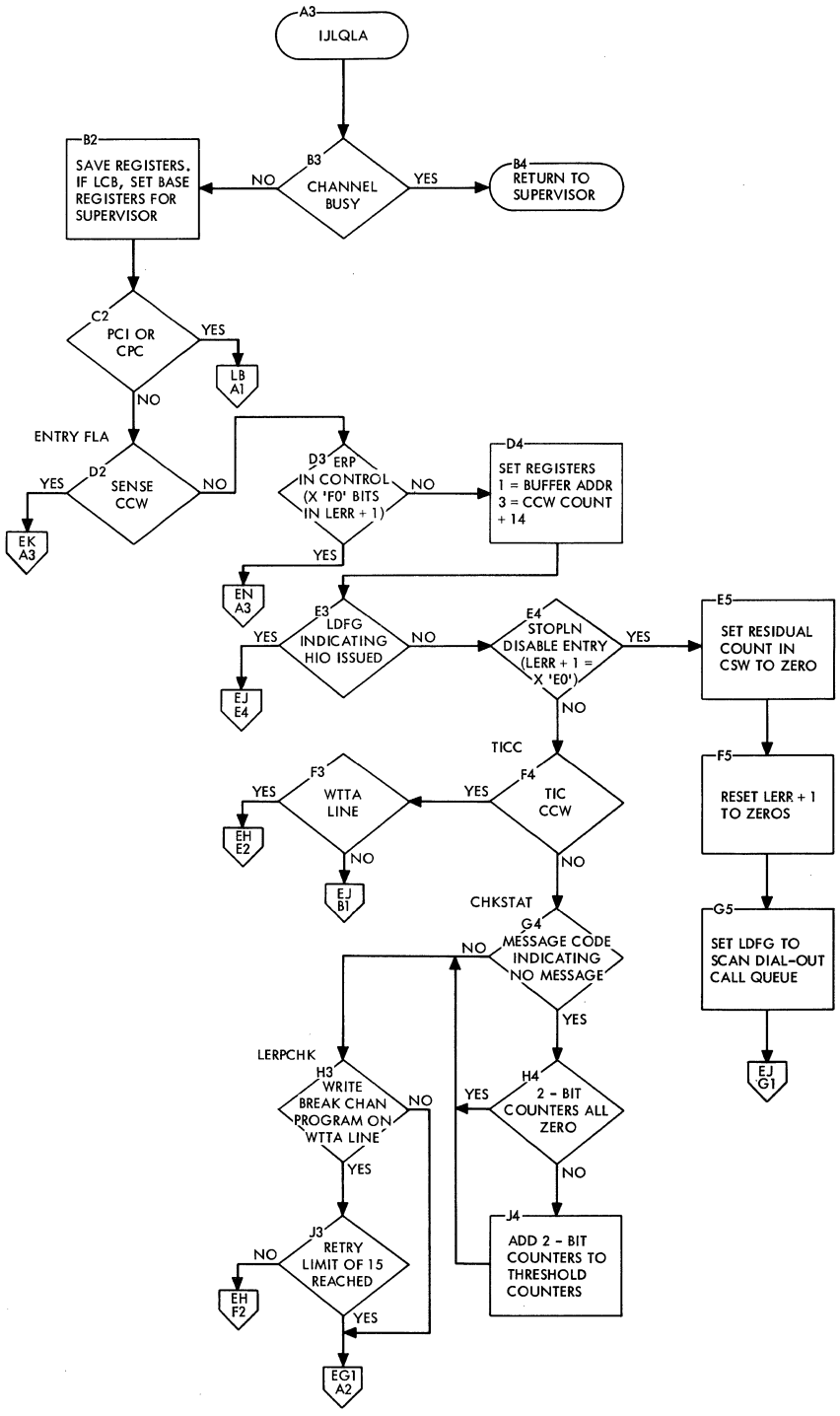


Chart EG1. Line End Appendage (Part 2 of 9)

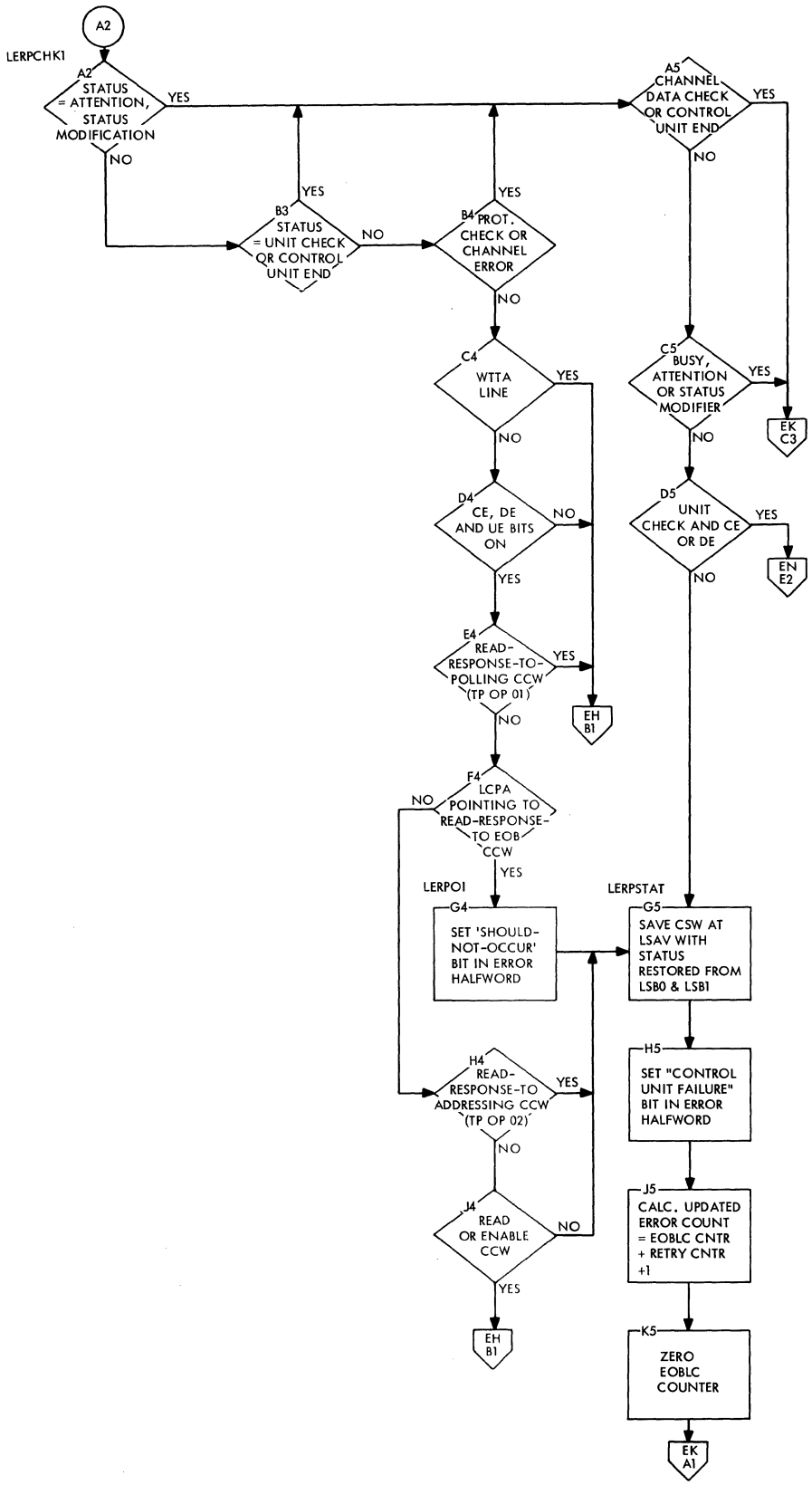


Chart EH. Line End Appendage (Part 3 of 9)

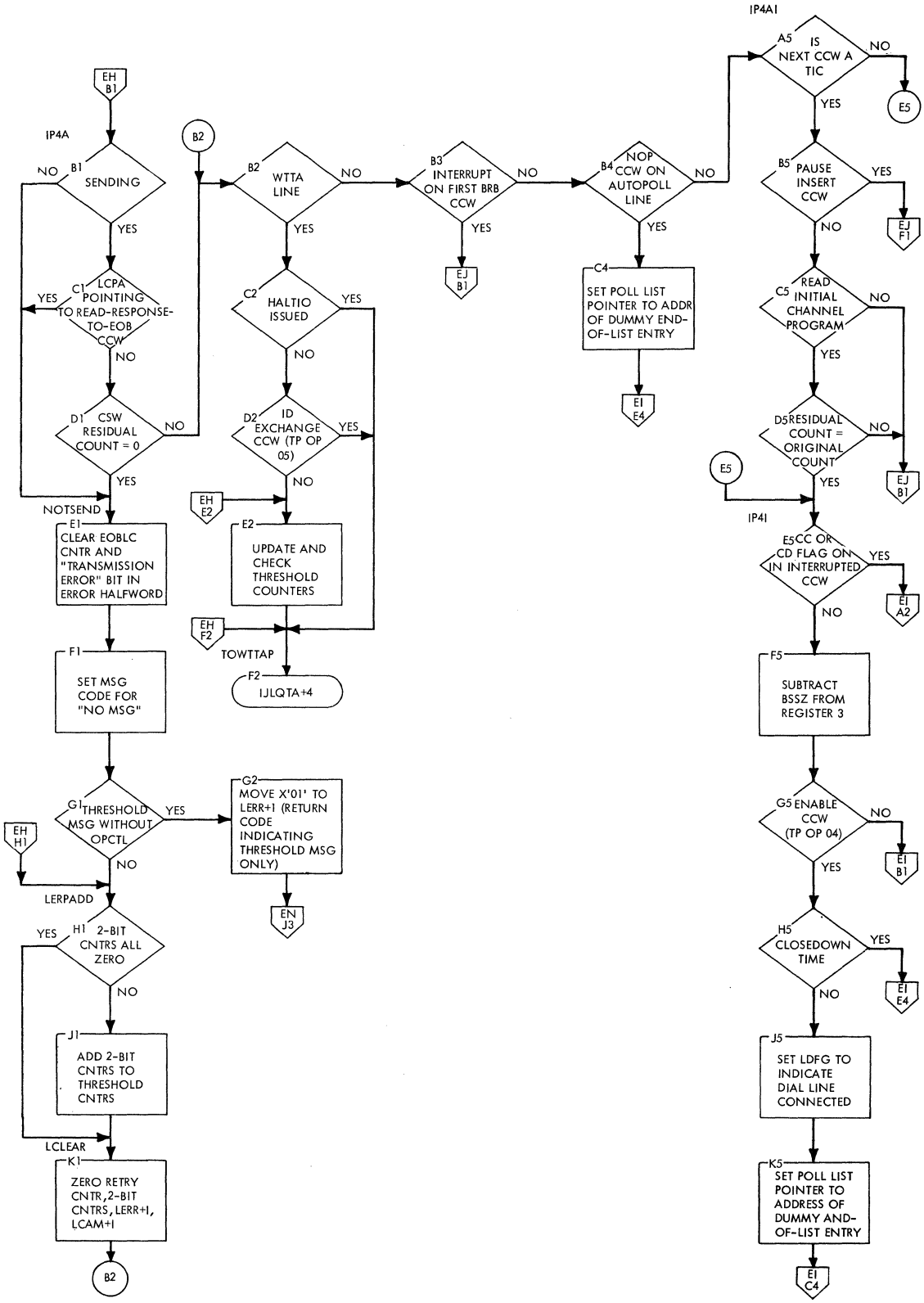


Chart EI. Line End Appendage (Part 4 of 9)

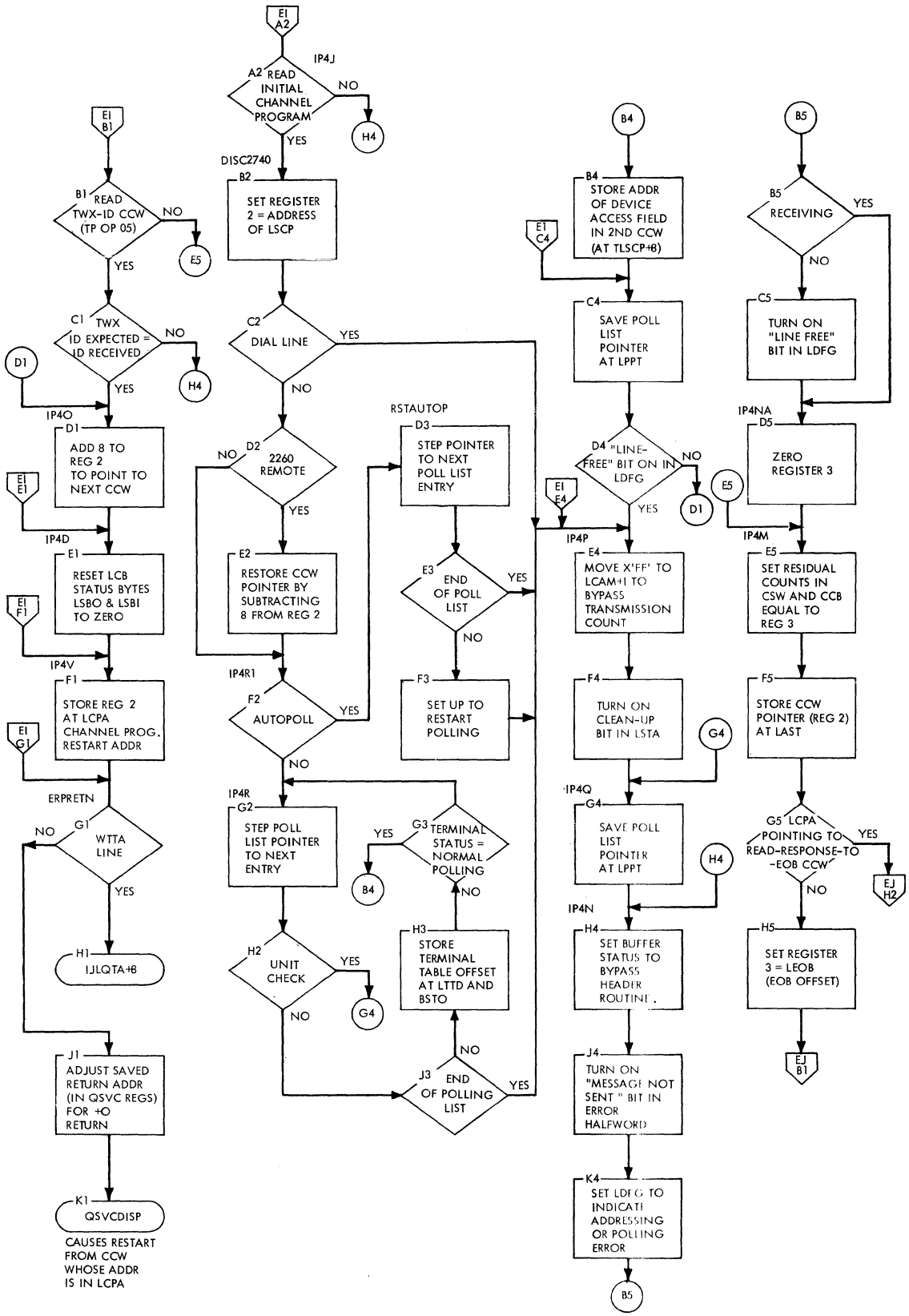


Chart EJ. Line End Appendage (Part 5 of 9)

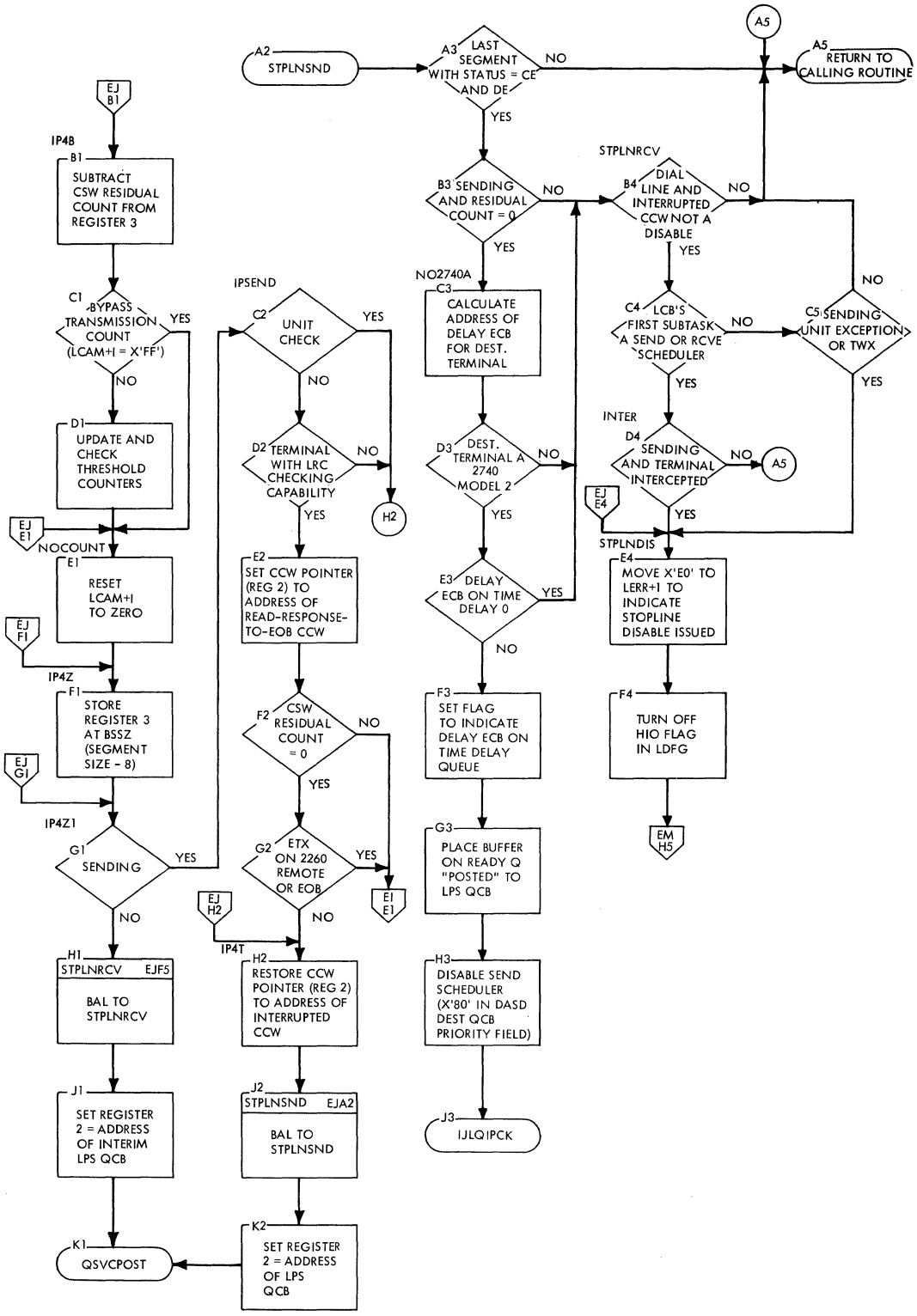


Chart EK. Line End Appendage (Part 6 of 9)

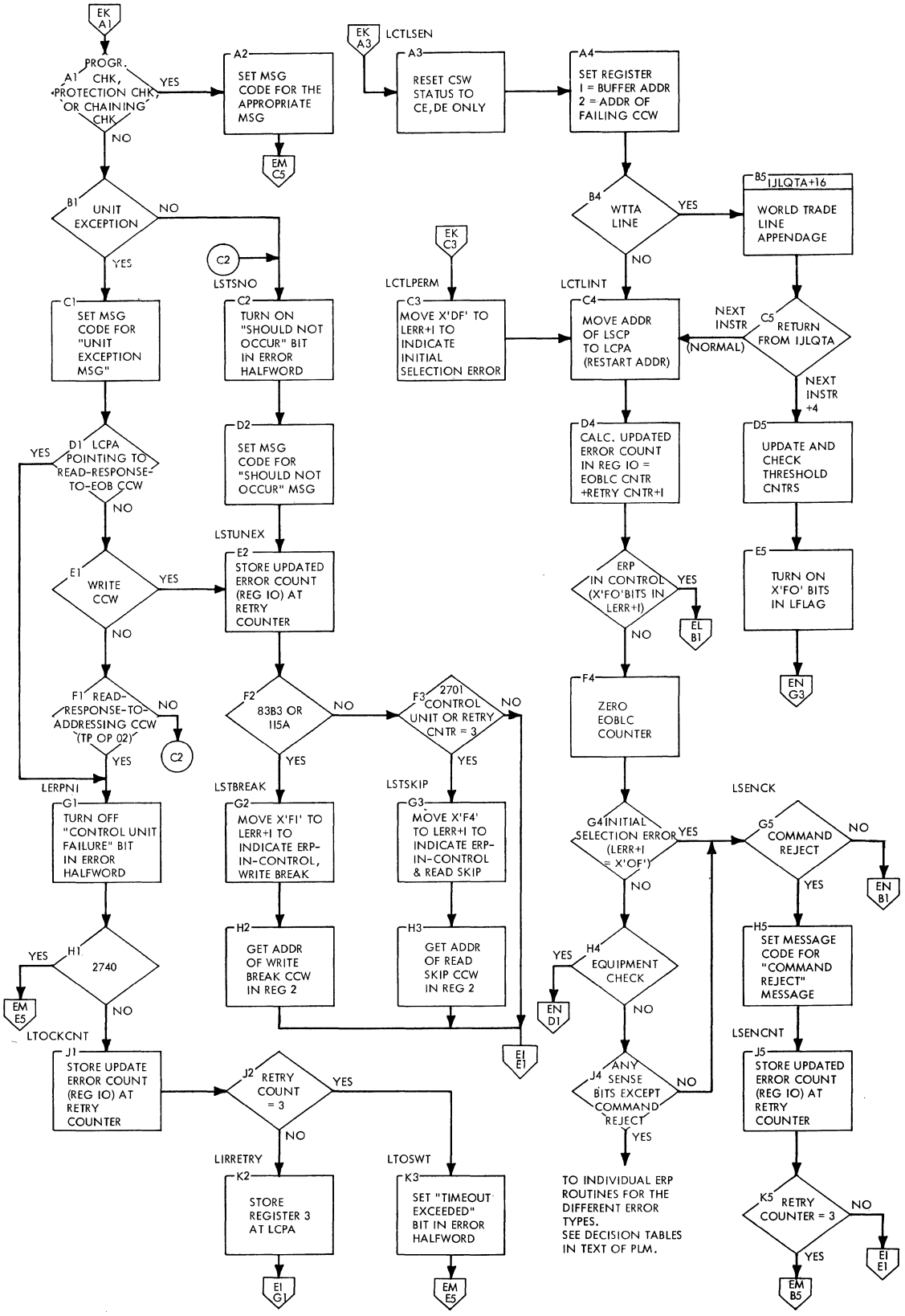


Chart EL. Line End Appendage (Part 7 of 9)

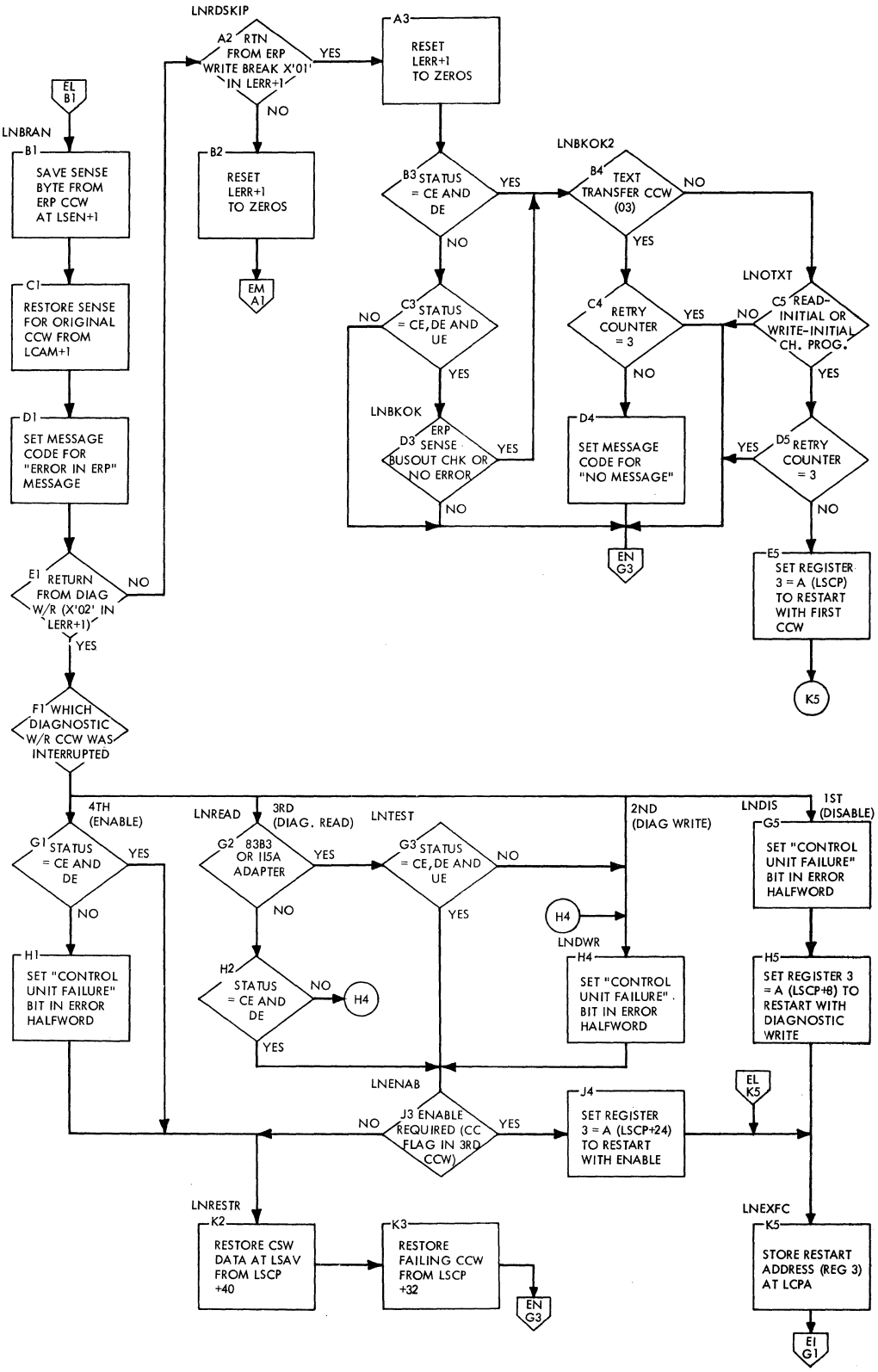


Chart EM. Line End Appendage (Part 8 of 9)

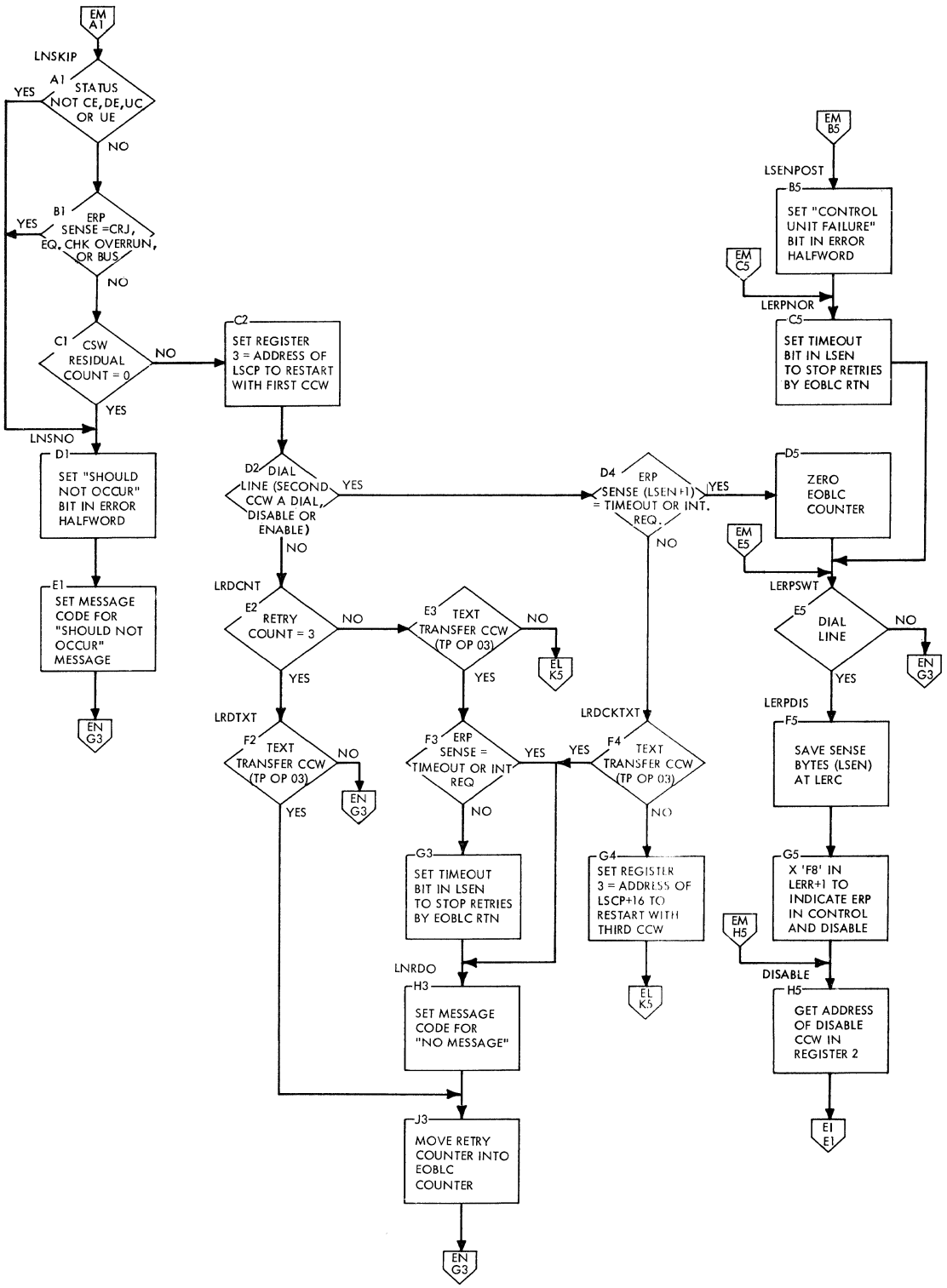


Chart EN. Line End Appendage (Part 9 of 9)

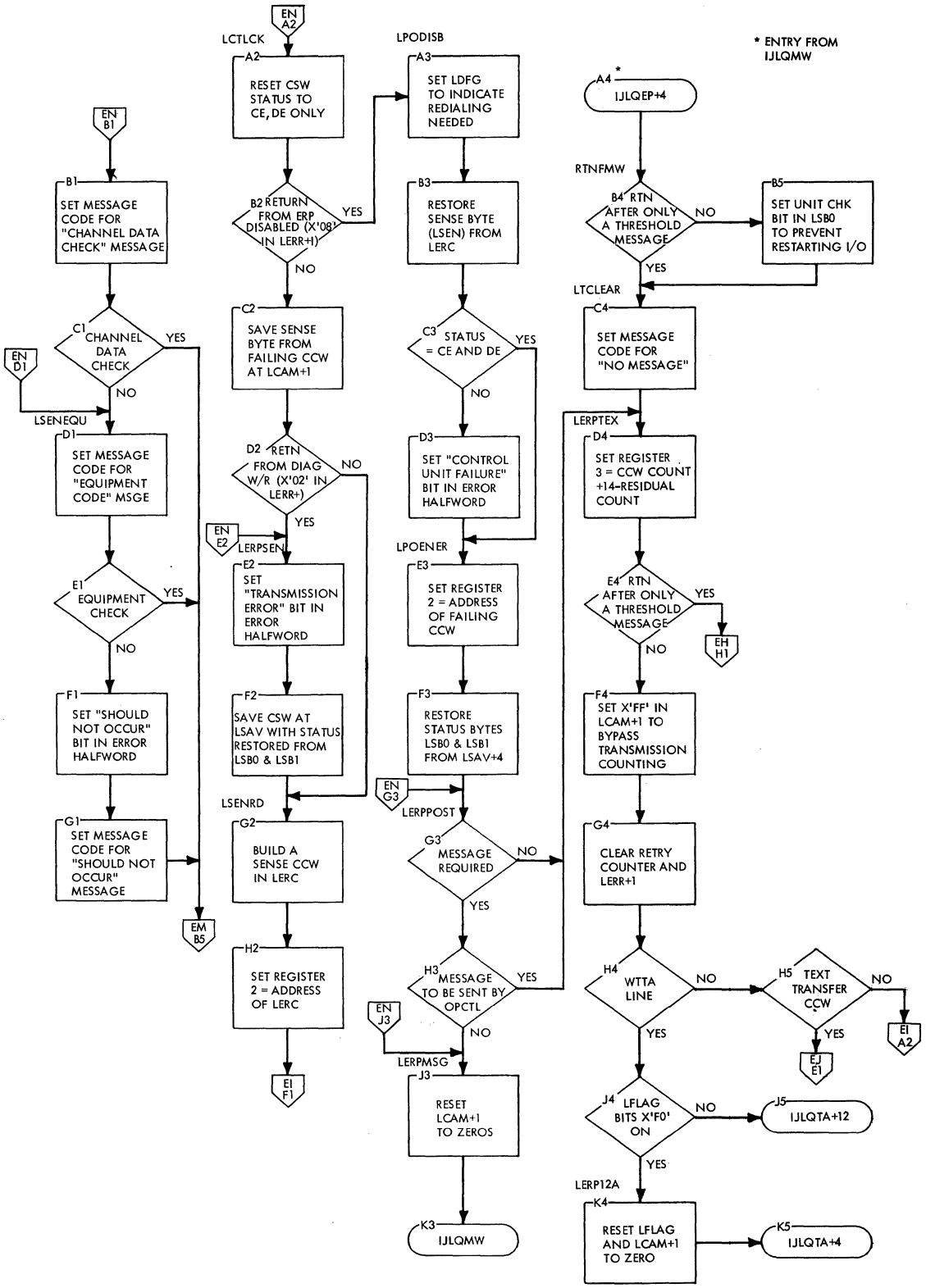


Chart ER. Error Message Routine

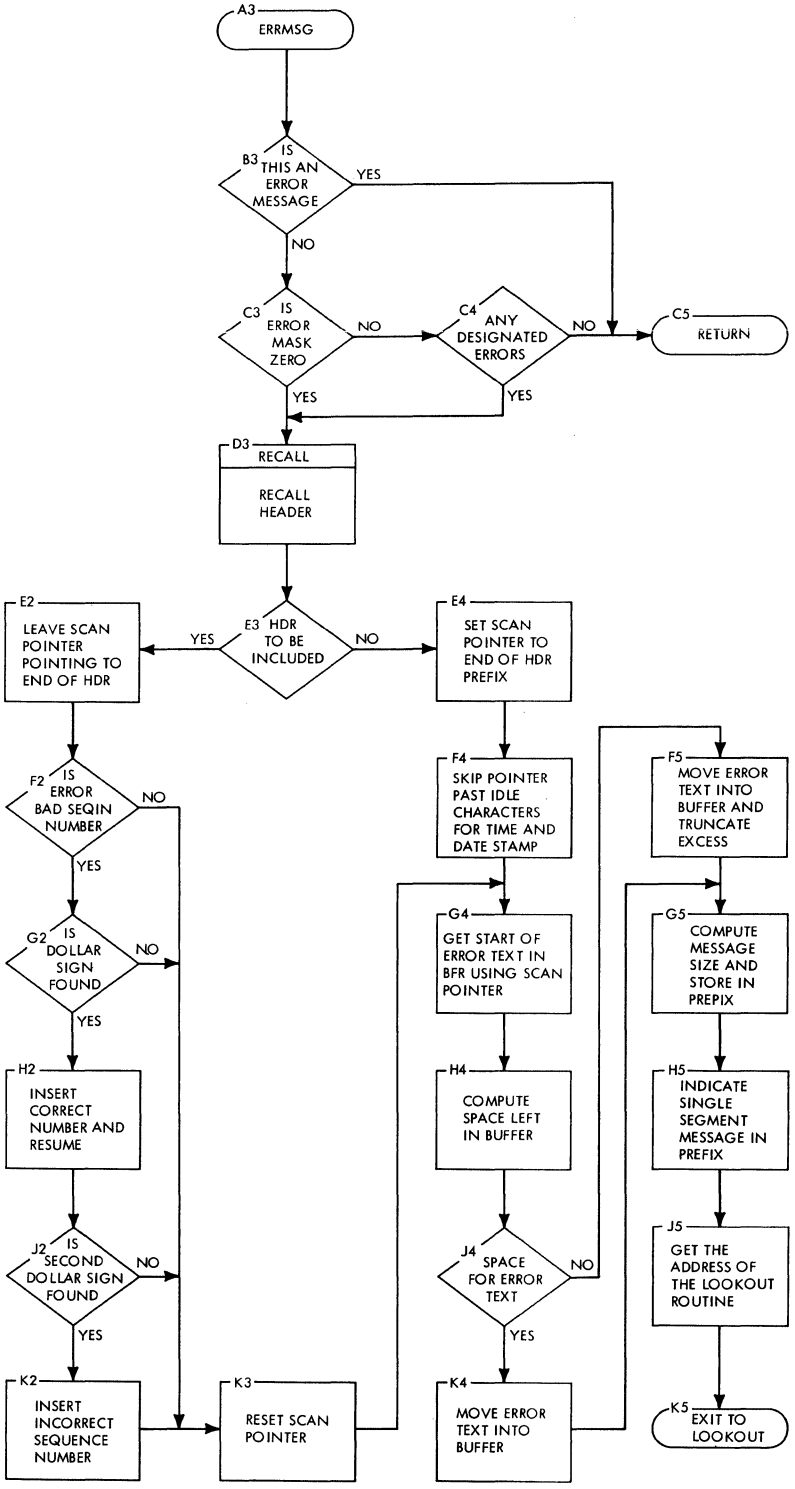


Chart EX. Expand Header Routine

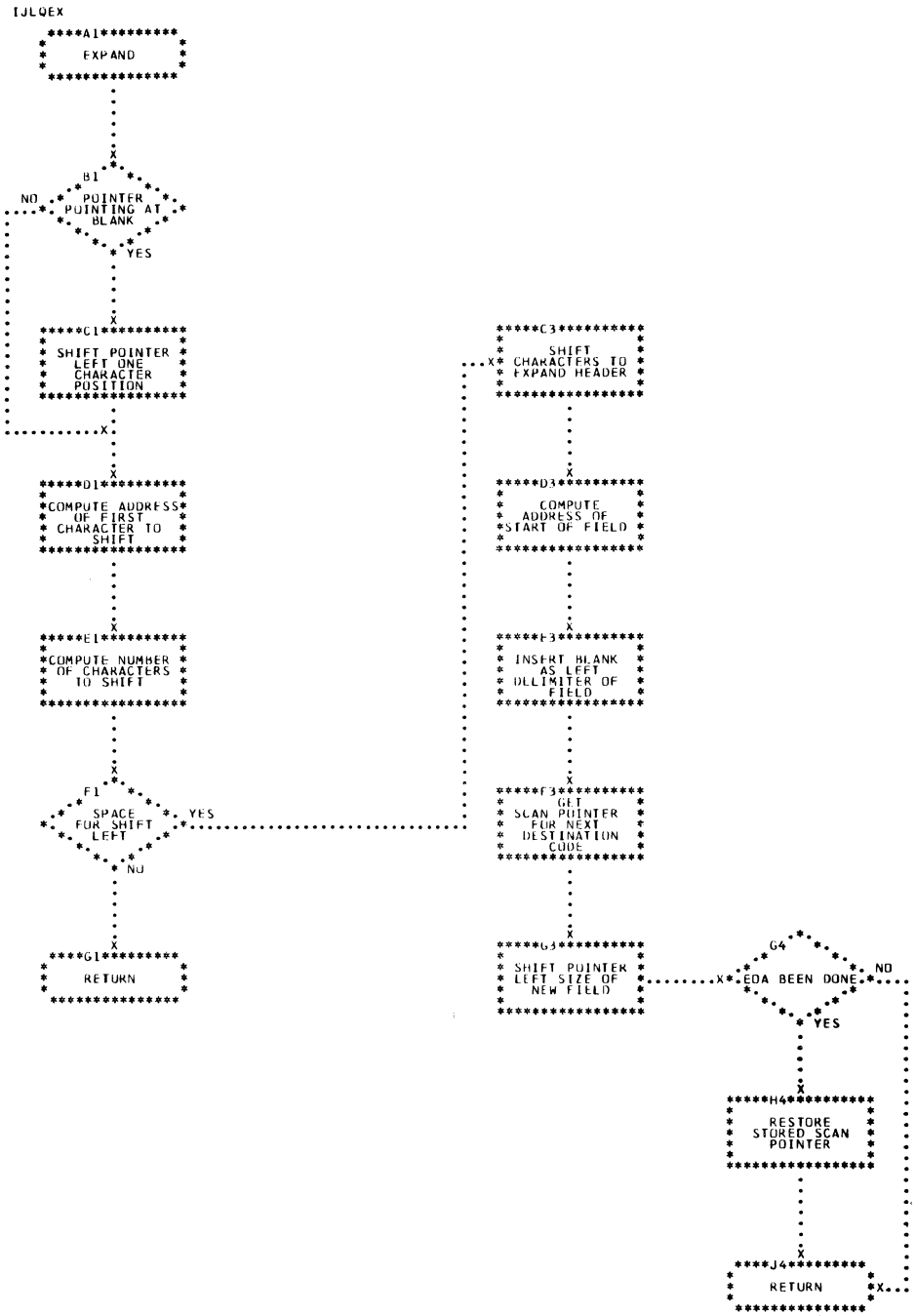


Chart FL. DTF Locator Routine

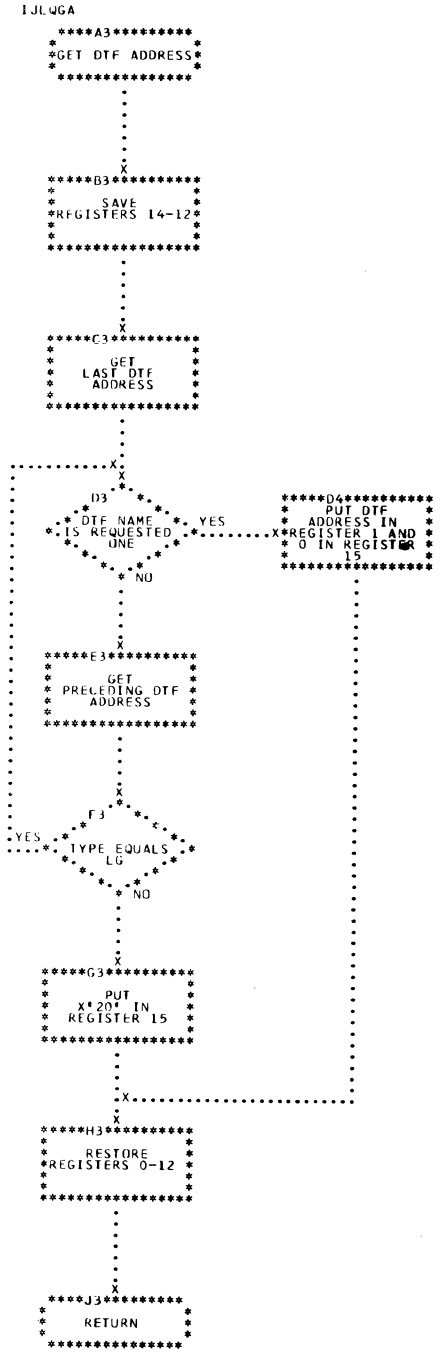


Chart GA. Get Audio Message Routine

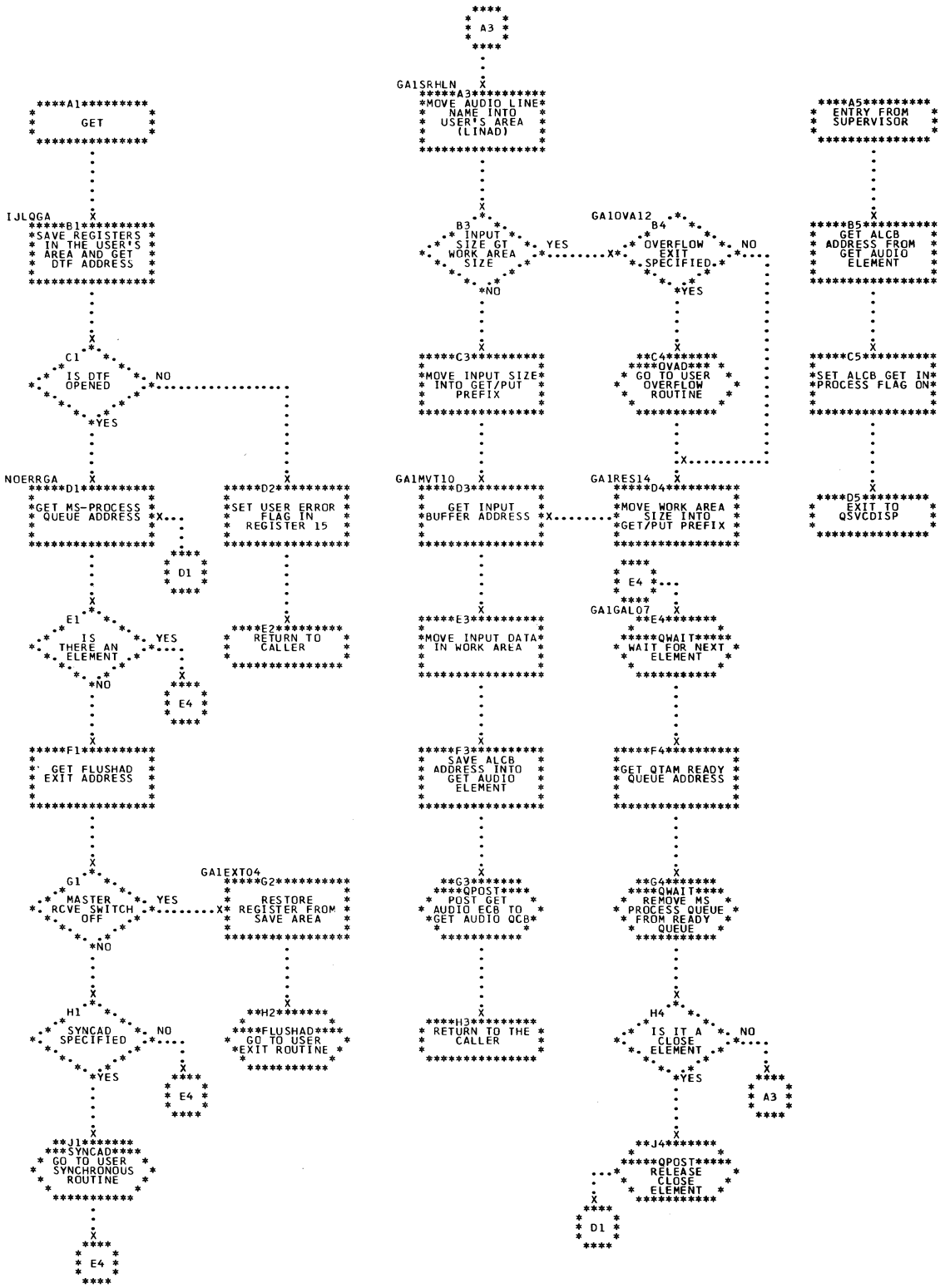


Chart GB. Get Nonaudio or Audio Message Routine (Part 1 of 2)

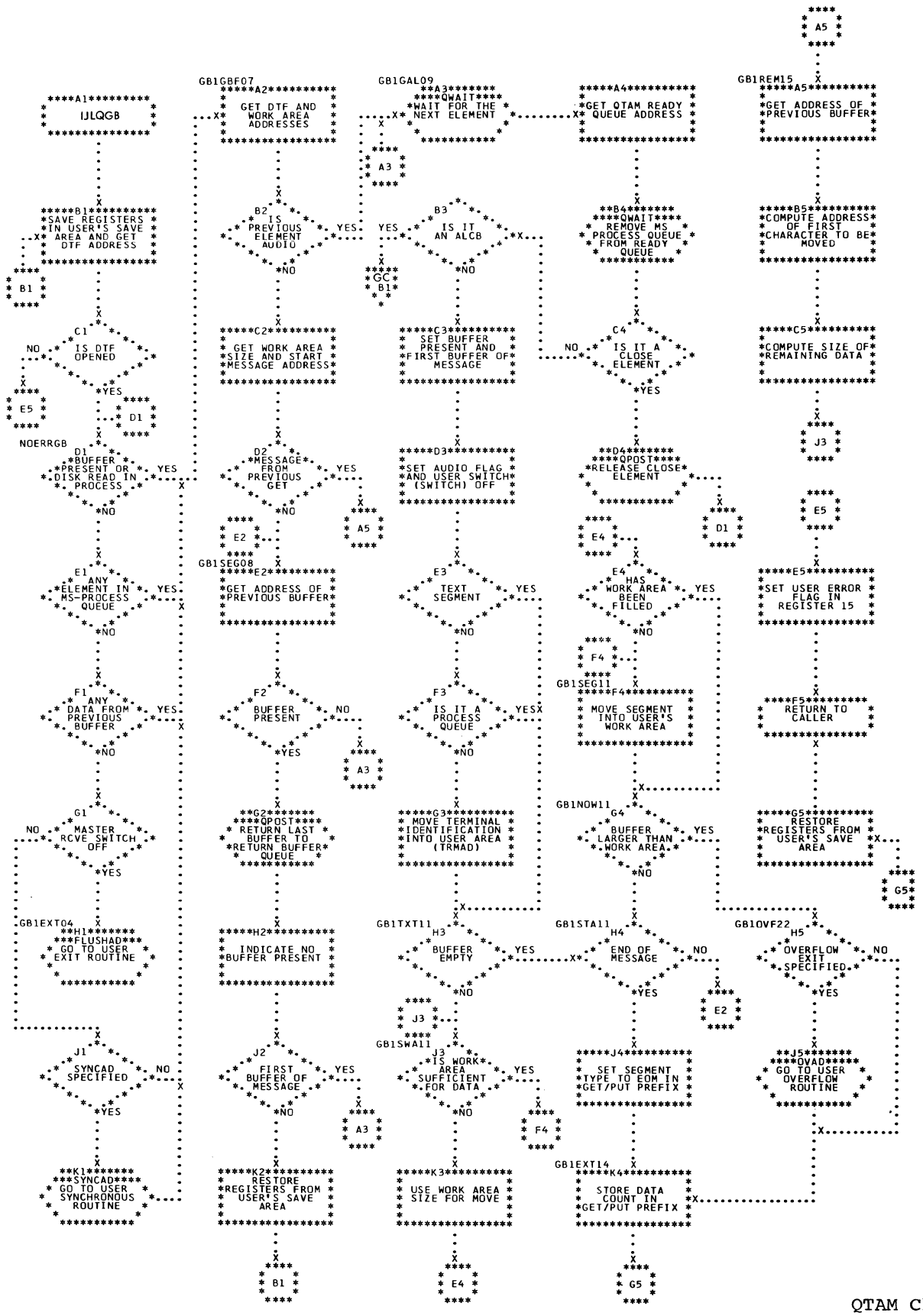


Chart GC. Get Nonaudio or Audio Message Routine (Part 2 of 2)

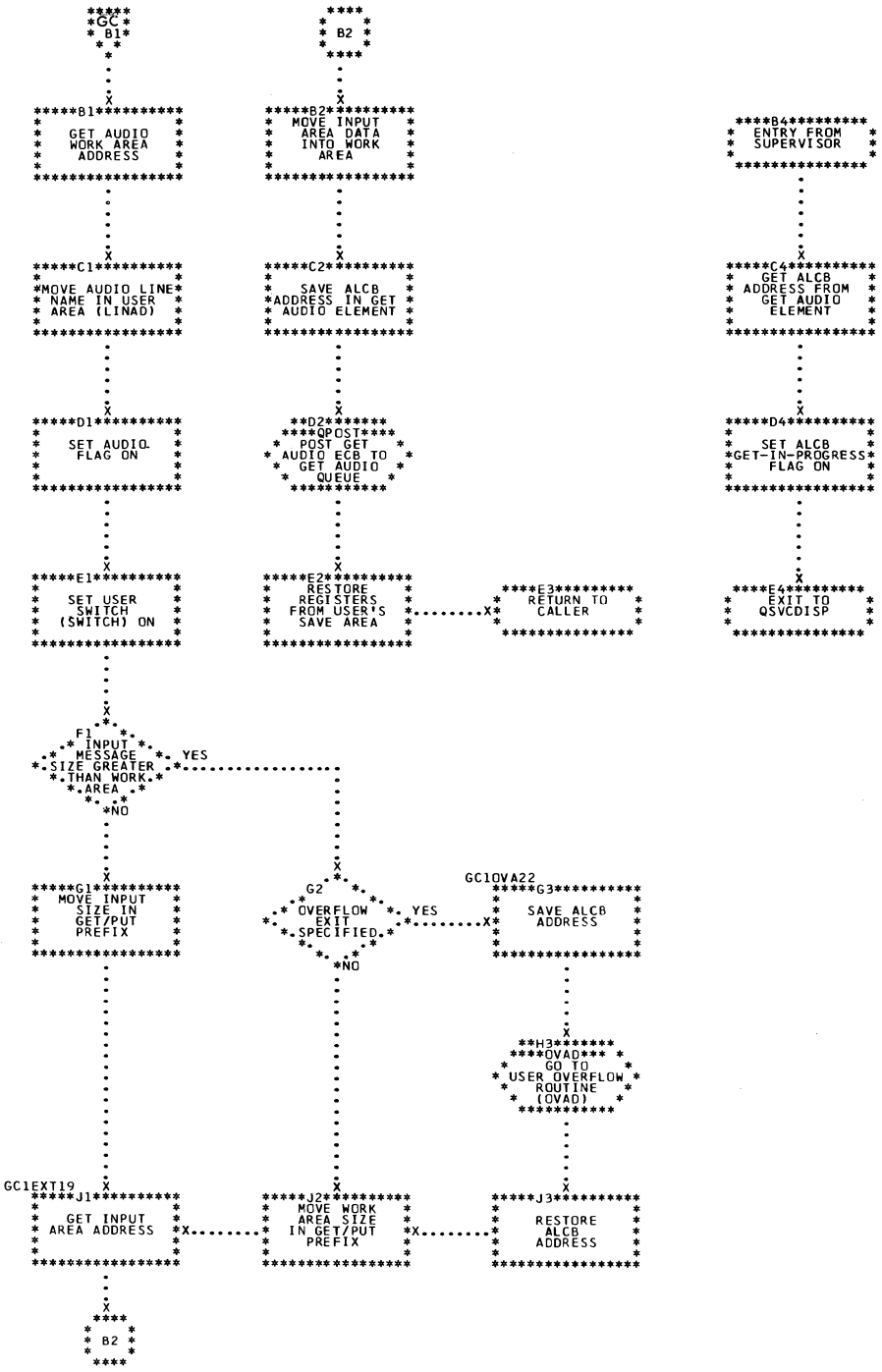


Chart GD. Get Record or Audio Message Routine (Part 1 of 2)

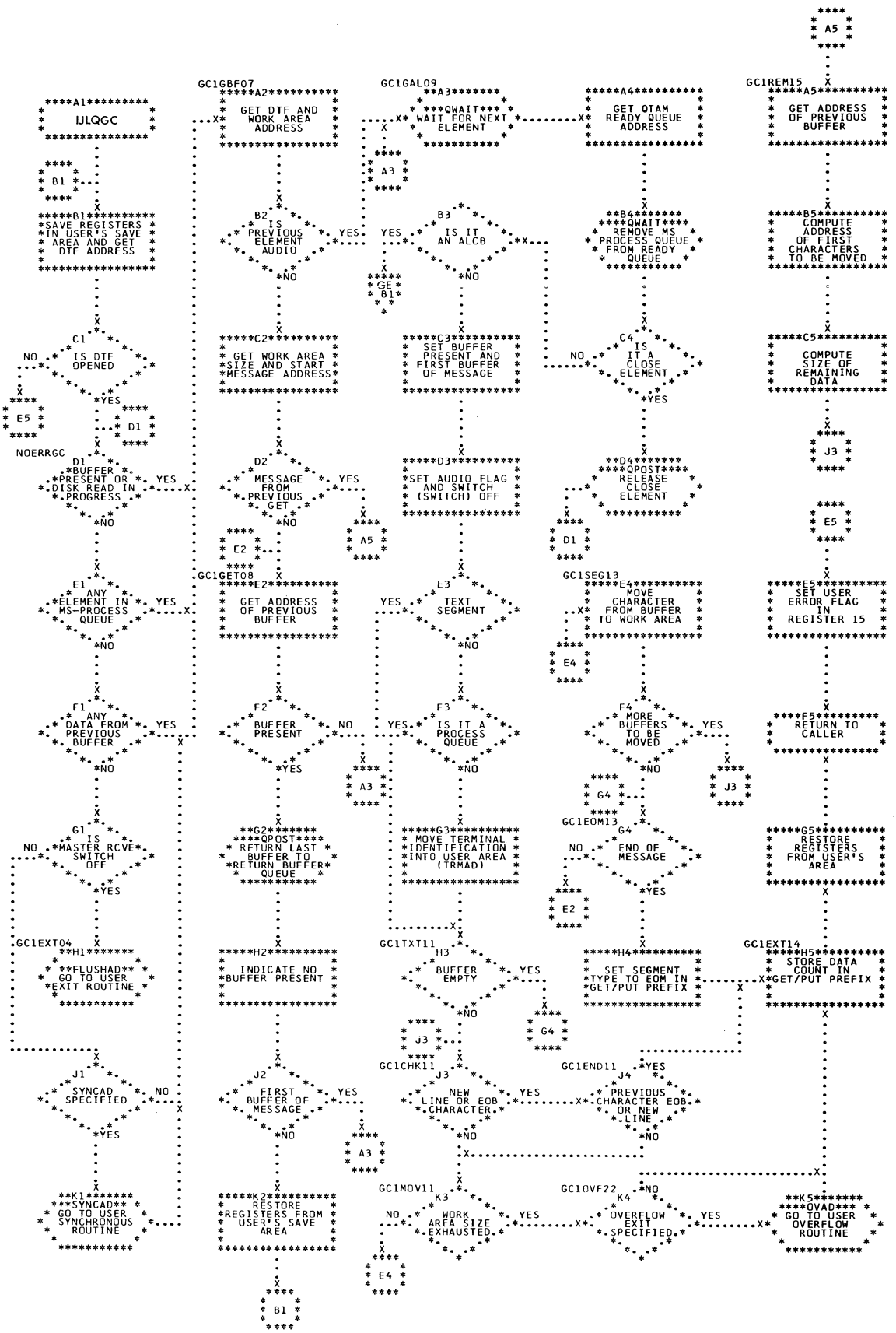


Chart GE. Get Record or Audio Message Routine (Part 2 of 2)

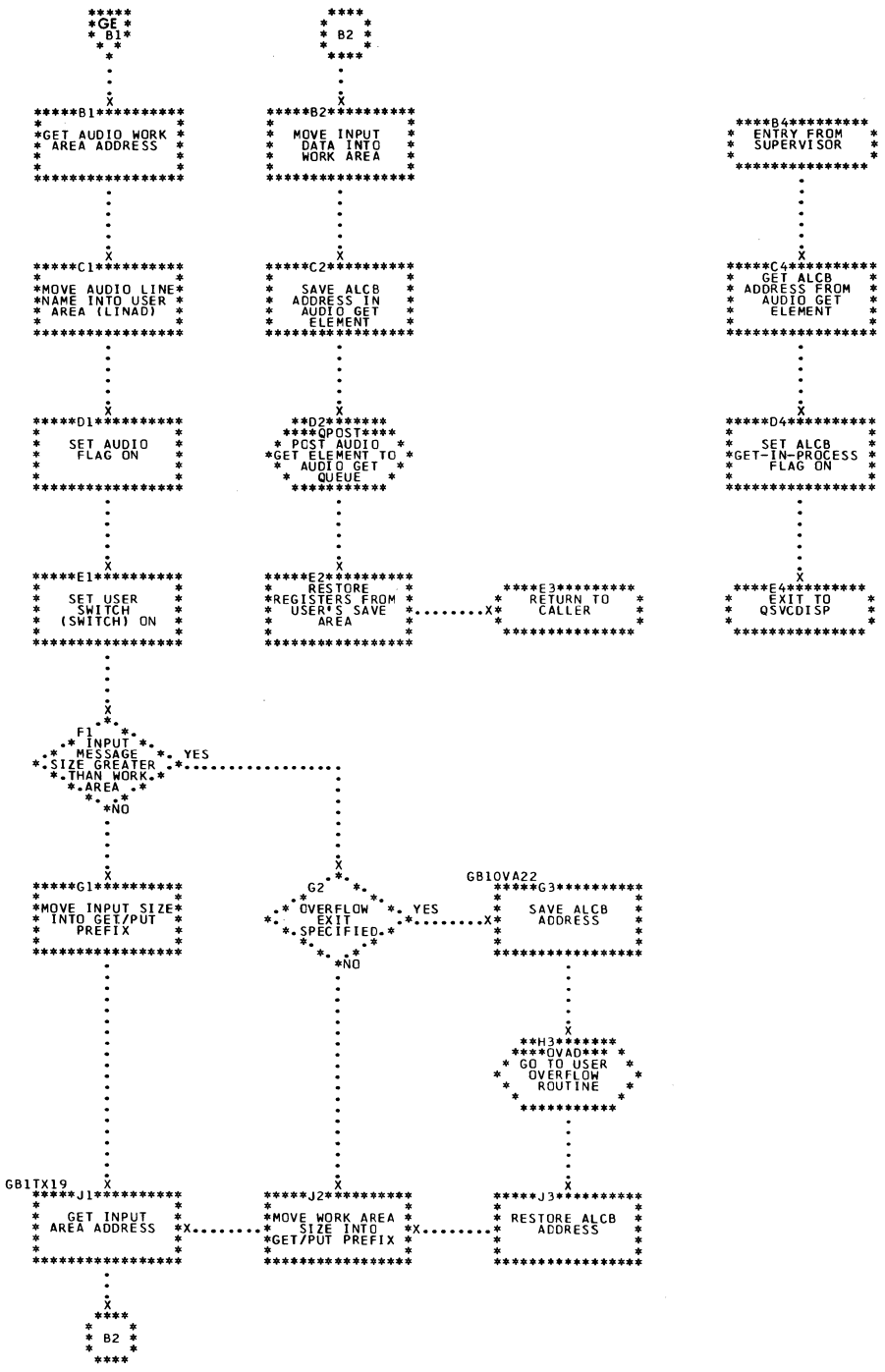


Chart GF. Get Segment or Audio Message Routine (Part 1 of 2)

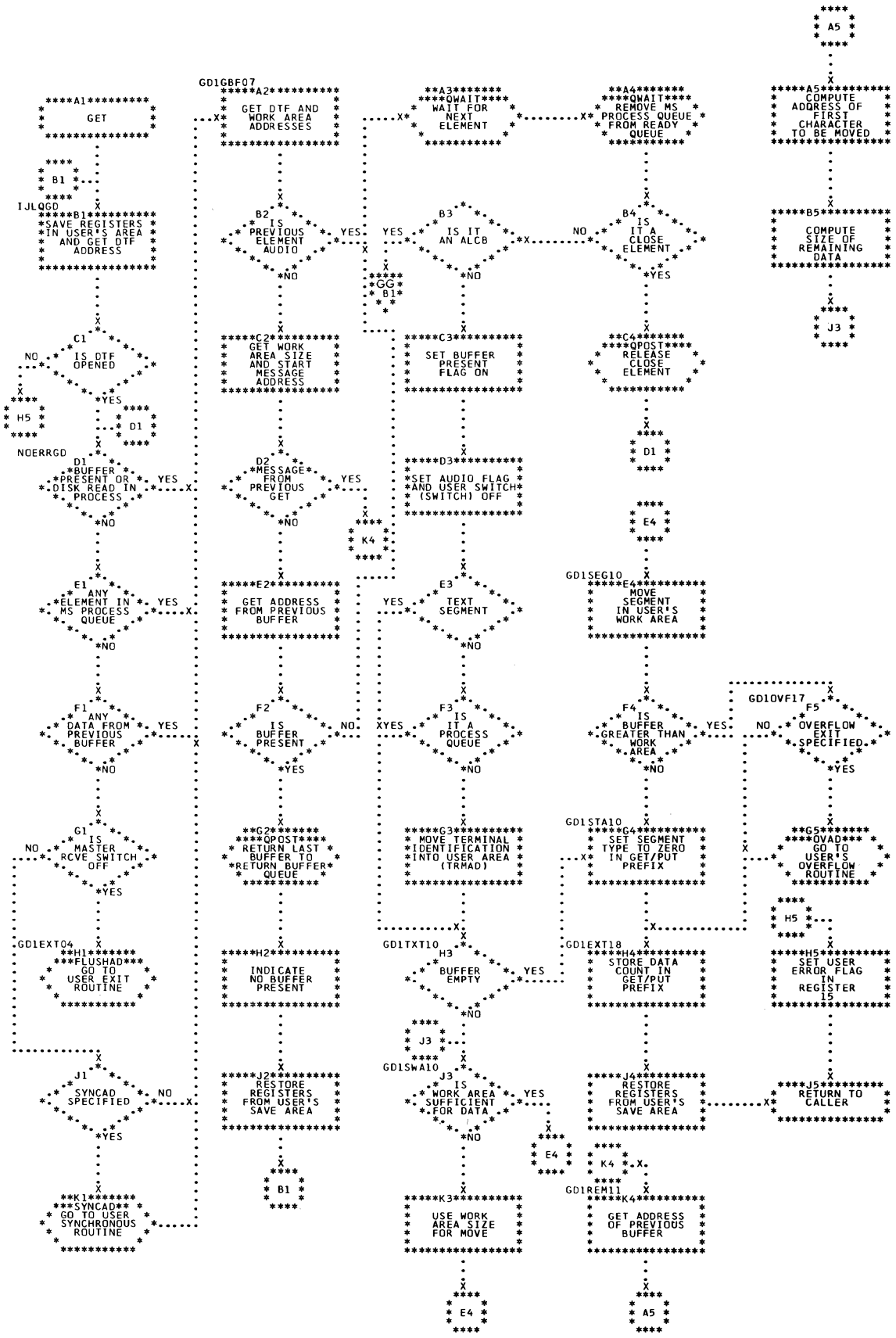


Chart GG. Get Segment or Audio Message Routine (Part 2 of 2)

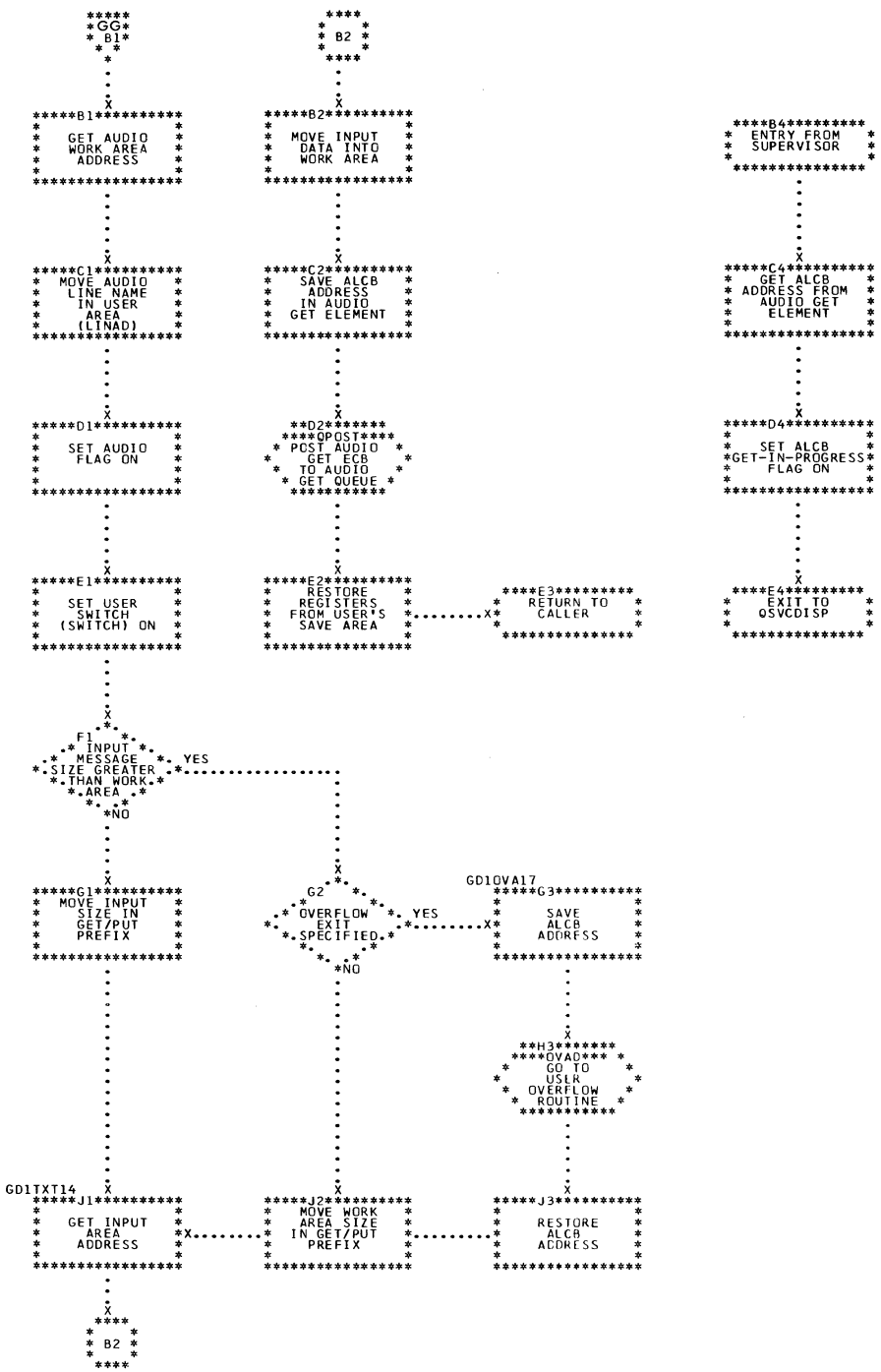


Chart GM. Get Message Routine (Part 1 of 2)

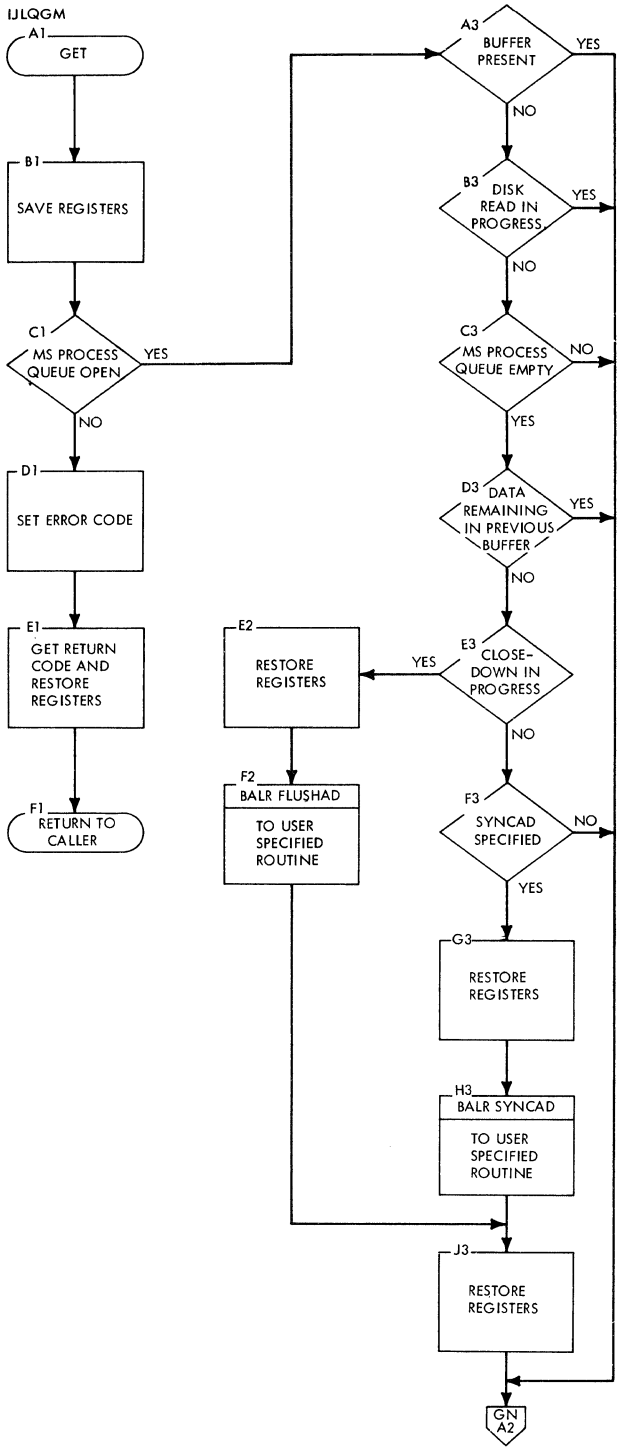


Chart GN. Get Message Routine (Part 2 of 2)

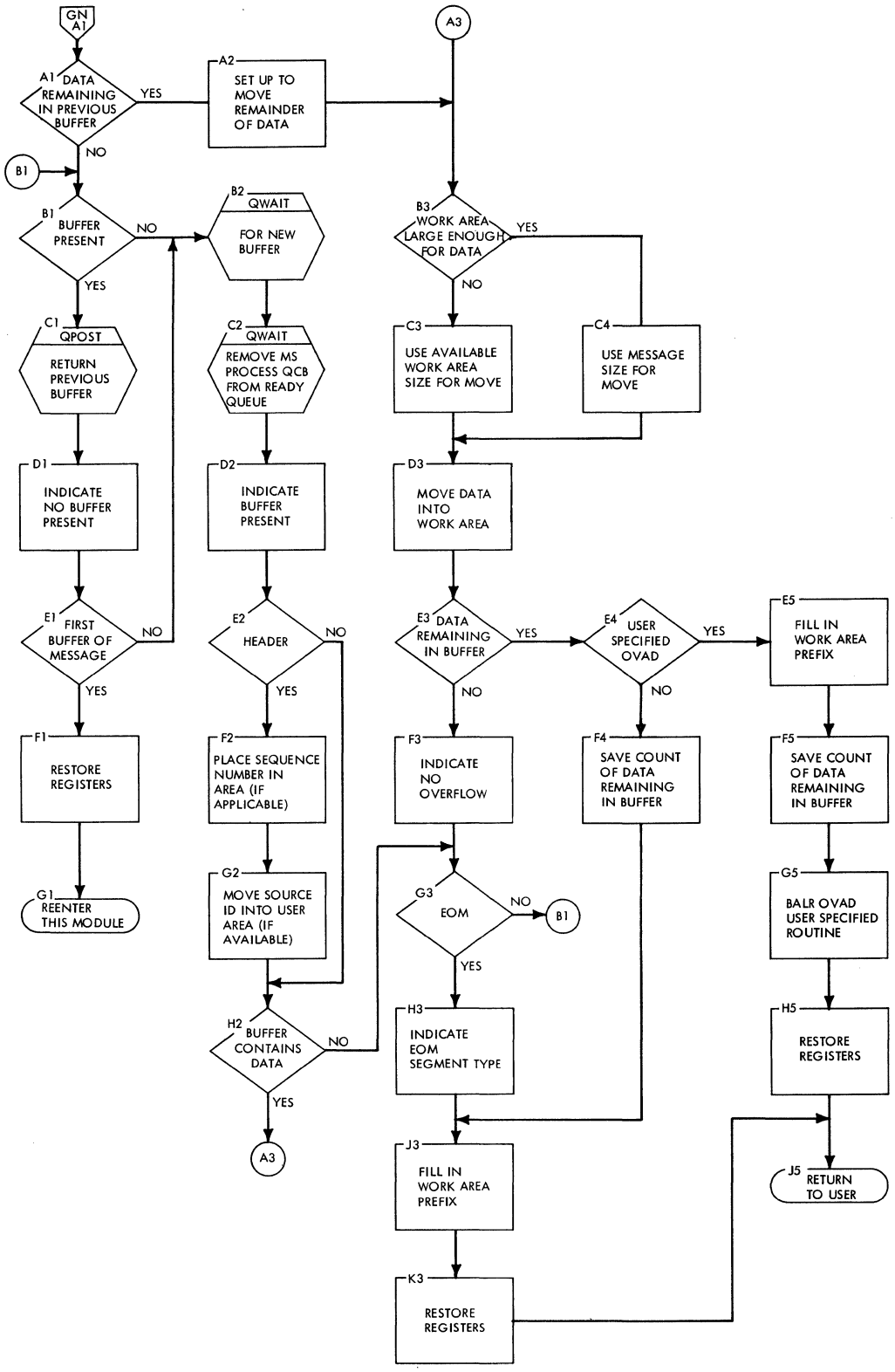


Chart GQ. Get Record Routine (Part 1 of 2)

IJLQGS

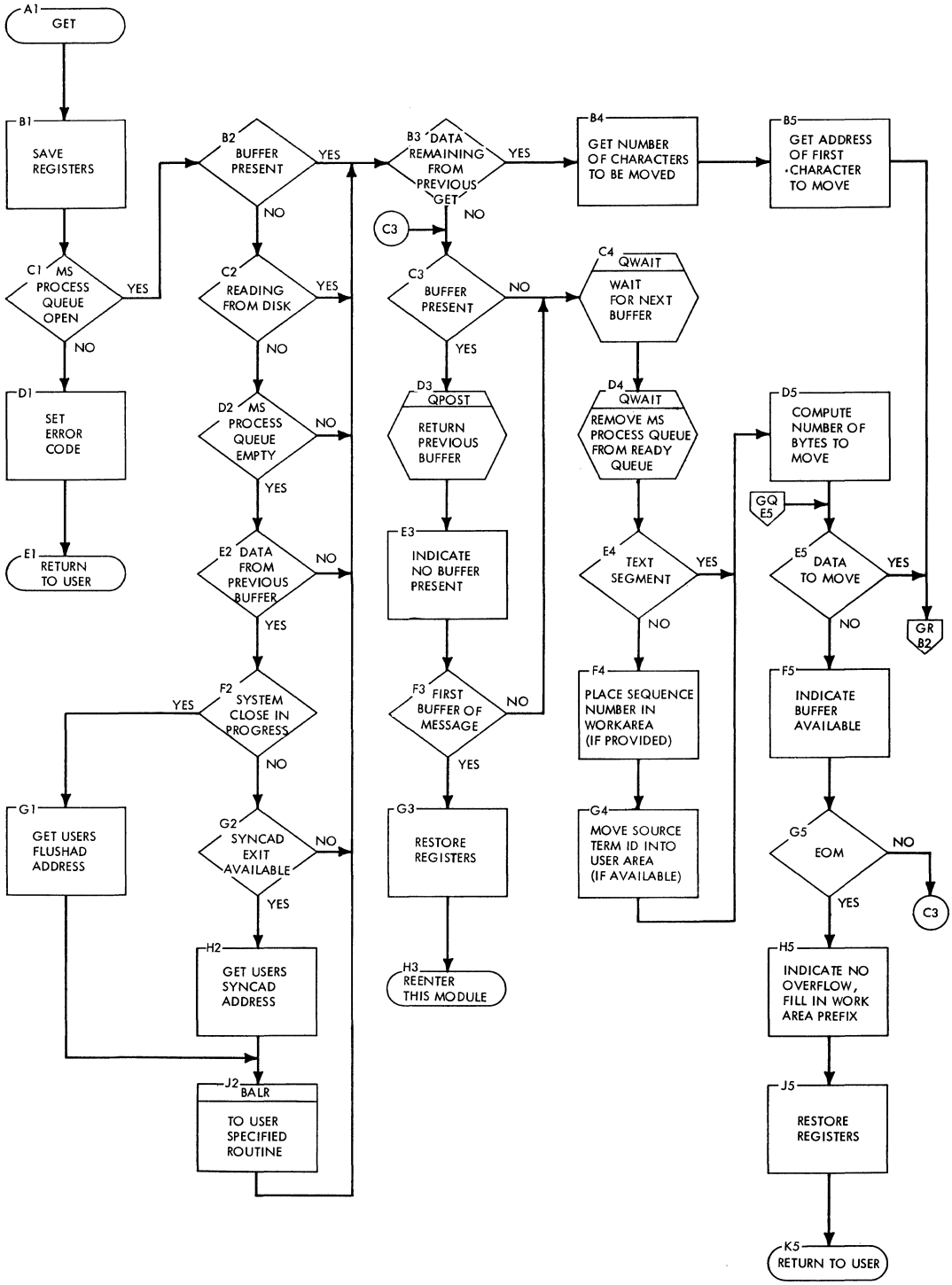


Chart GR. Get Record Routine (Part 2 of 2)

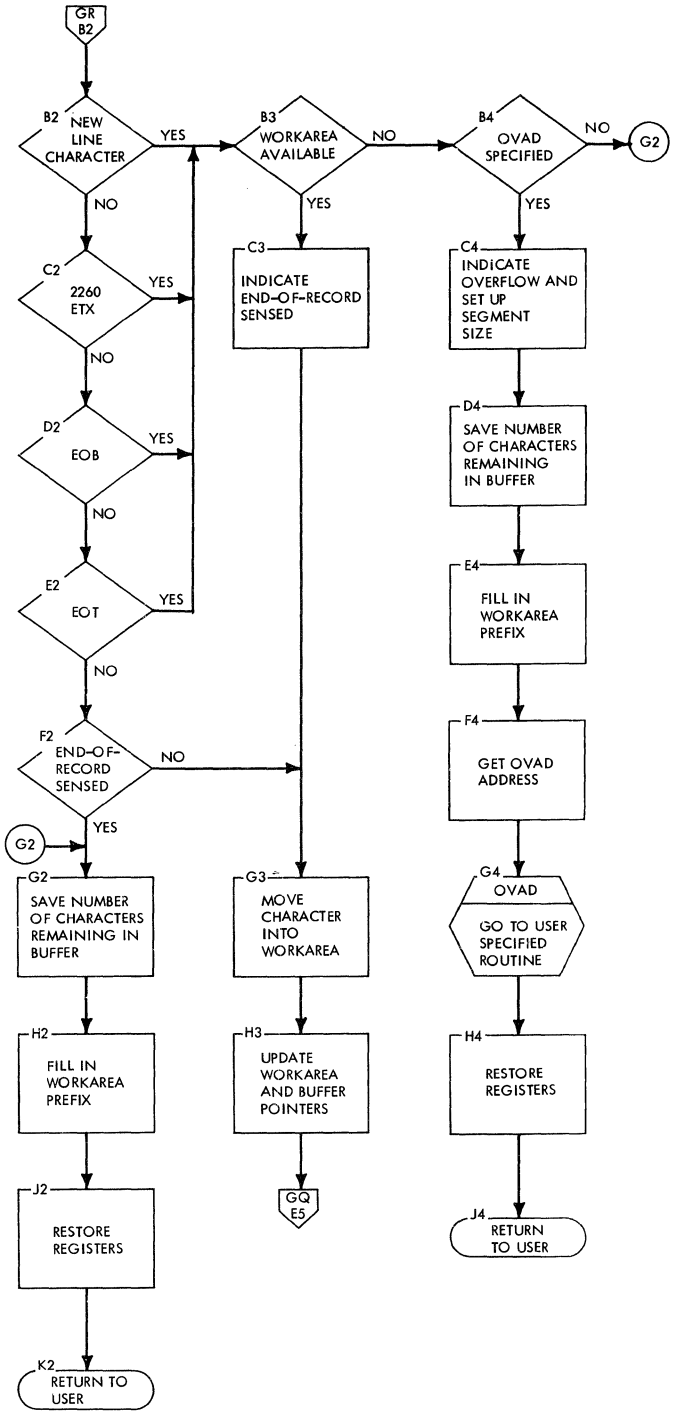


Chart GS. GET Segment Routine

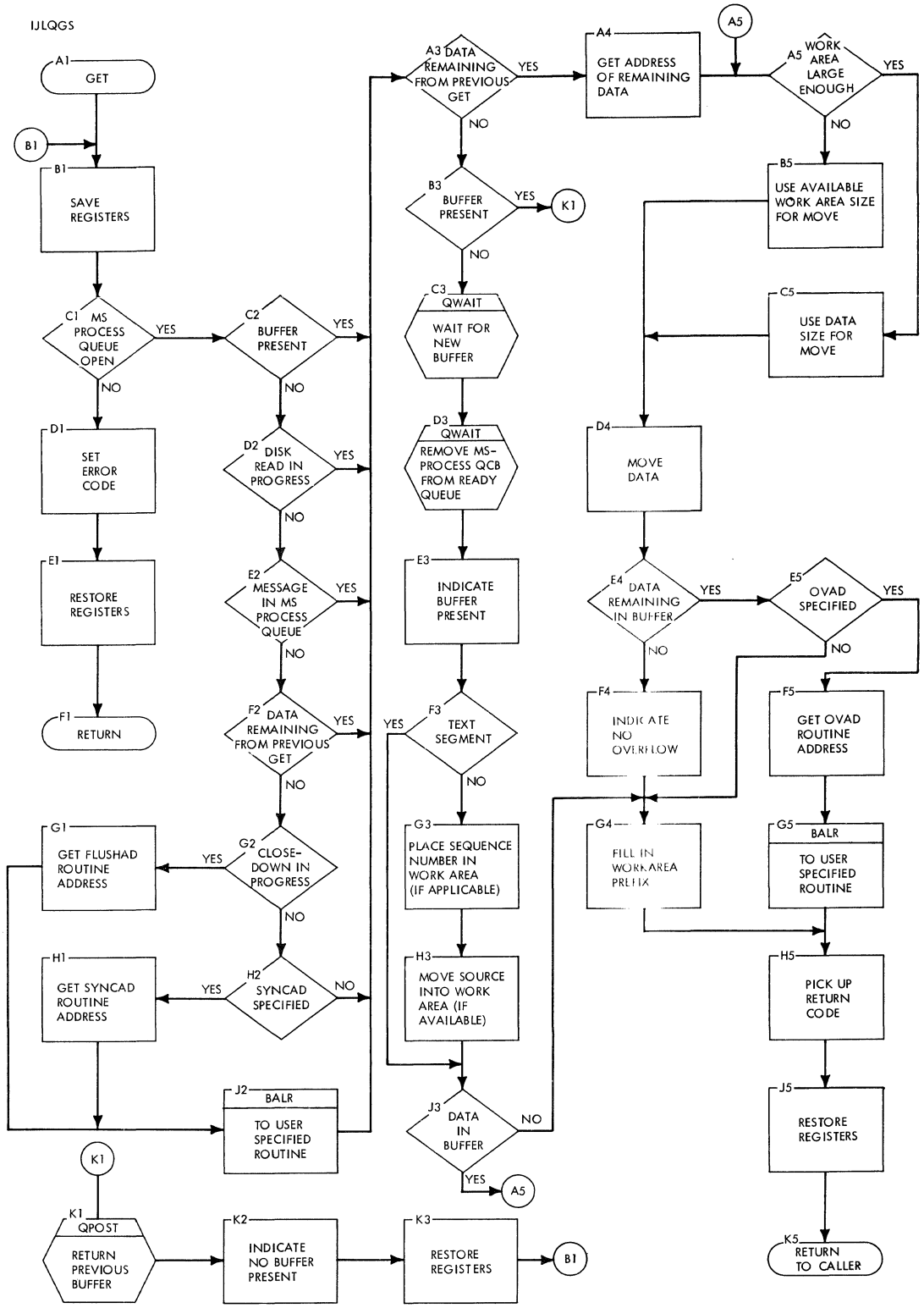


Chart IT. Intercept Message Routine

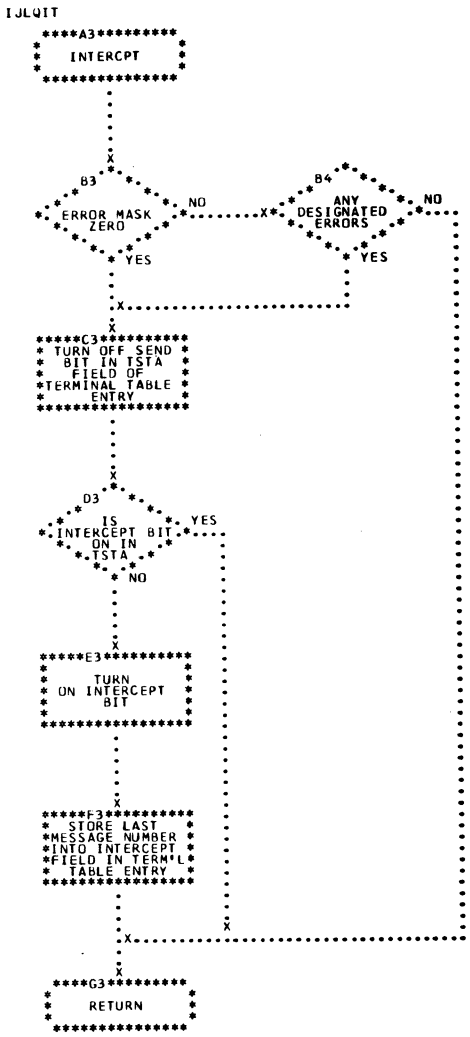


Chart LB. Line Appendage PCI and Program Check Module

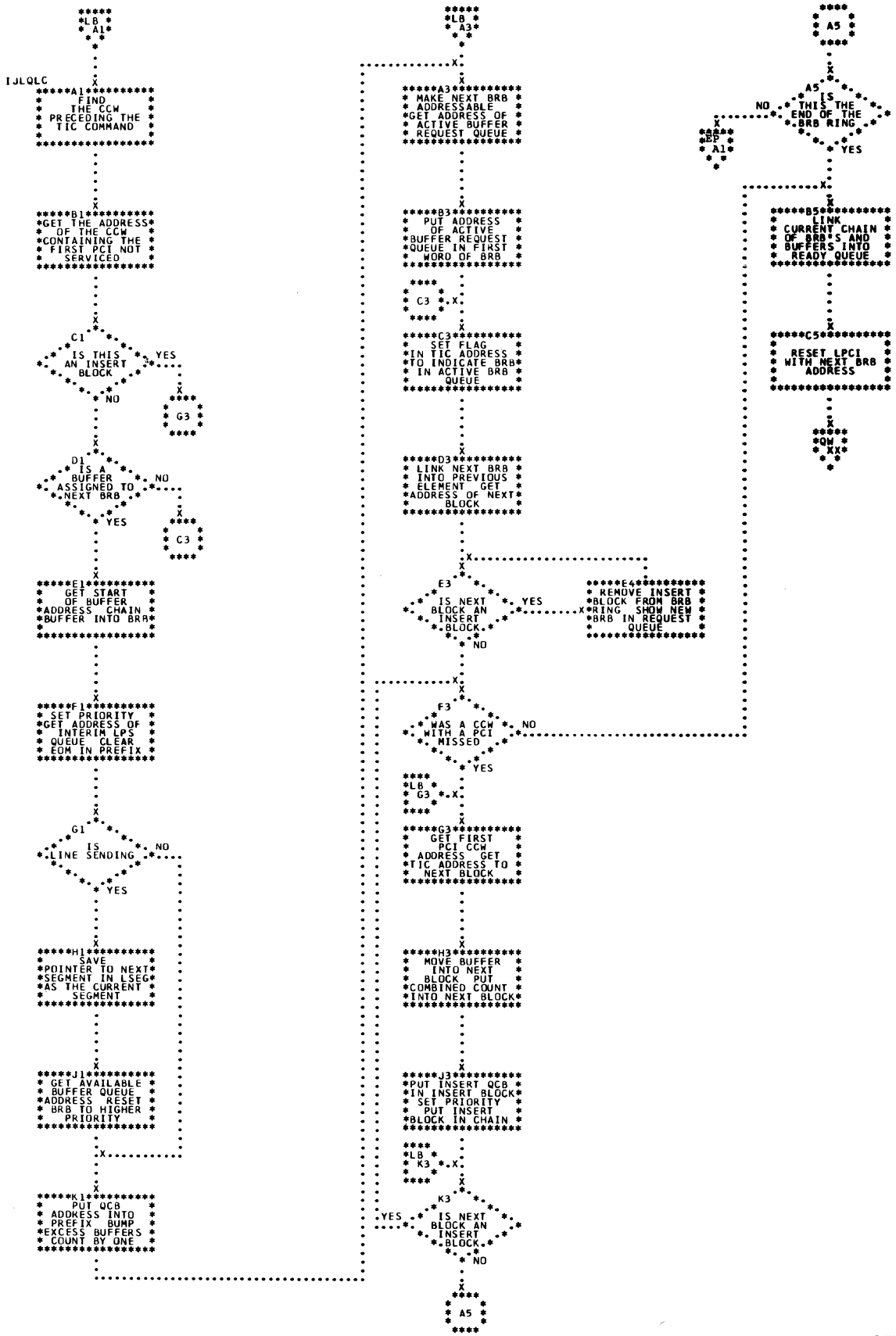


Chart LG. Audio Input Message Logging Routine

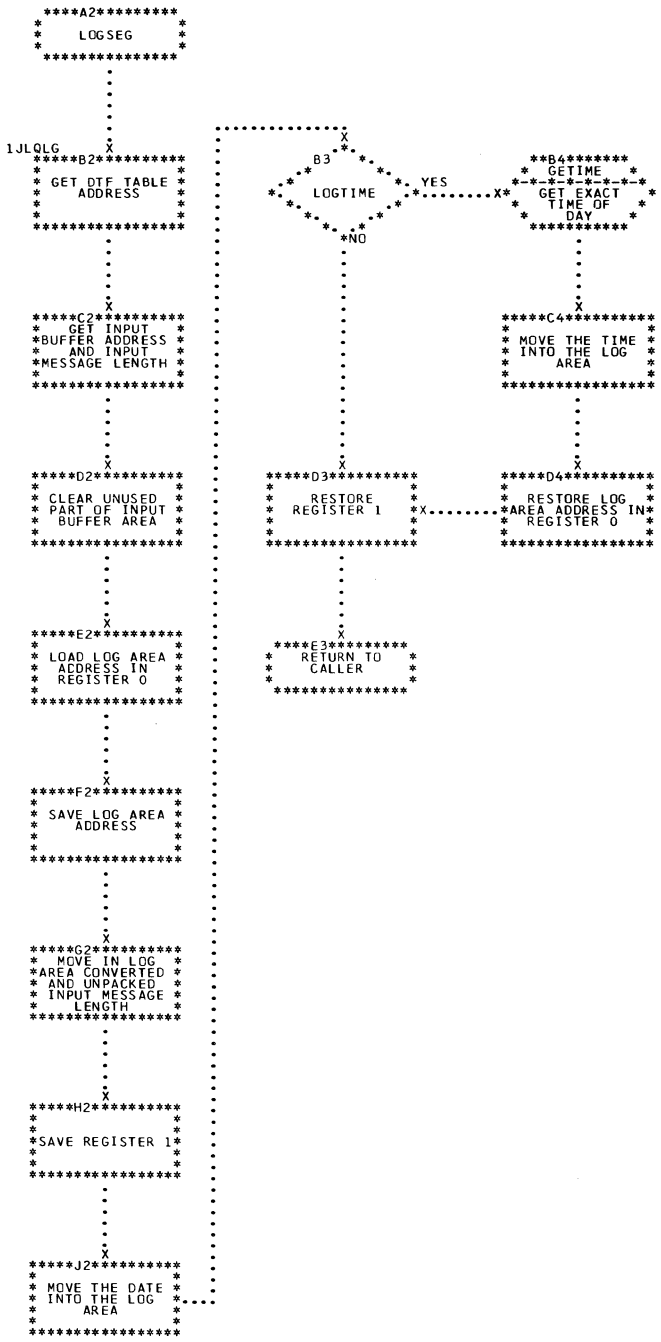


Chart LK. Lookup Terminal Table Entry Routine

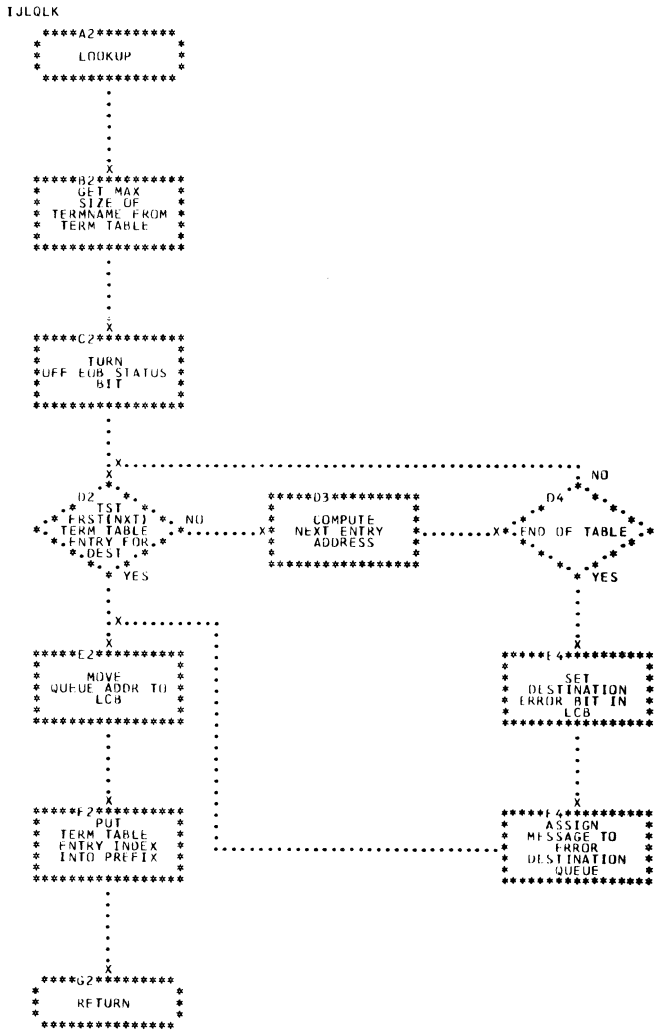


Chart LO. IBM 2260 Local Appendage (Part 1 of 3)

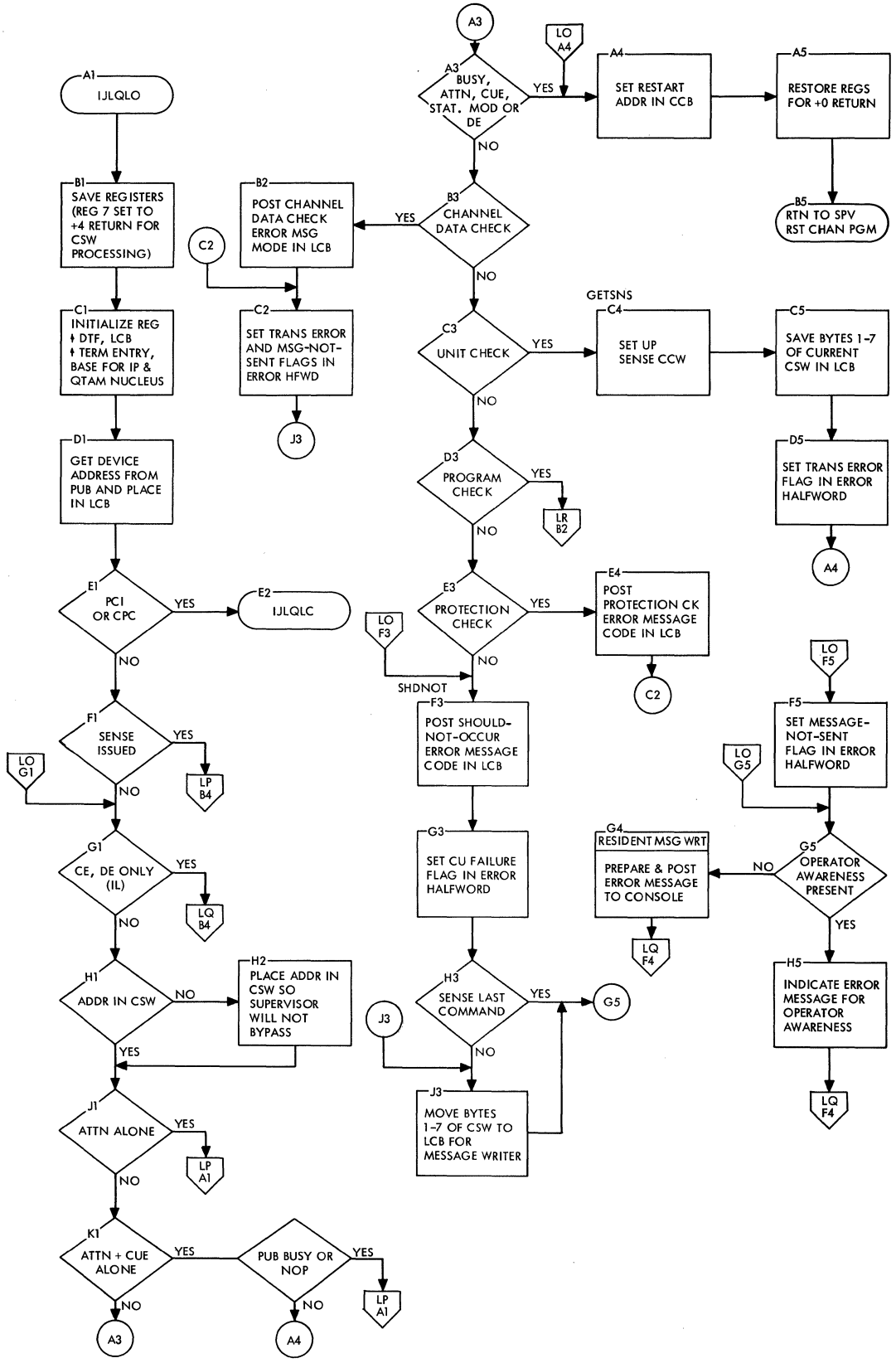


Chart LP. IBM 2260 Local Appendage (Part 2 of 3)

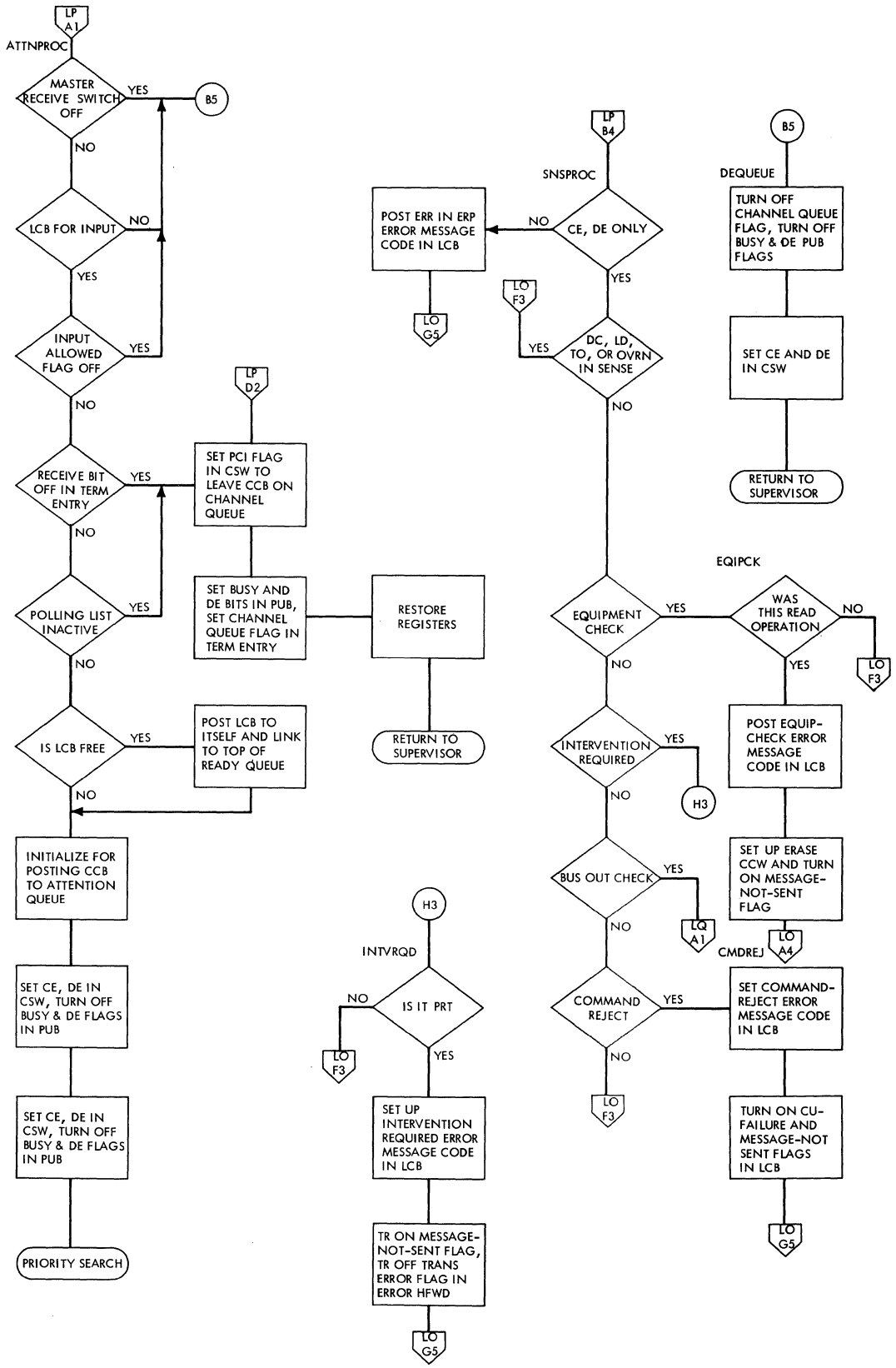


Chart LQ. IBM 2260 Local Appendage (Part 3 of 3)

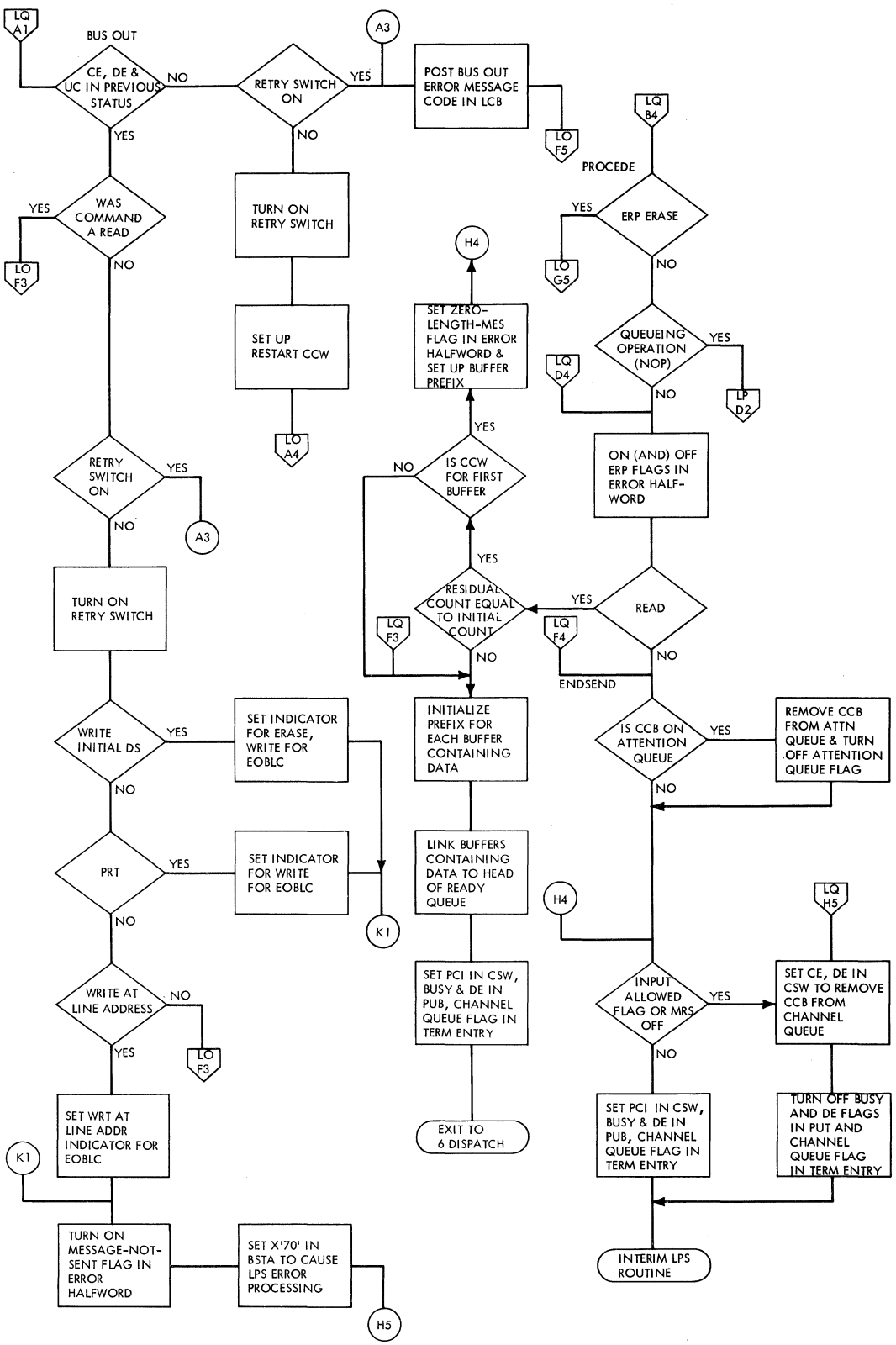


Chart MC. Conversational Mode Routine

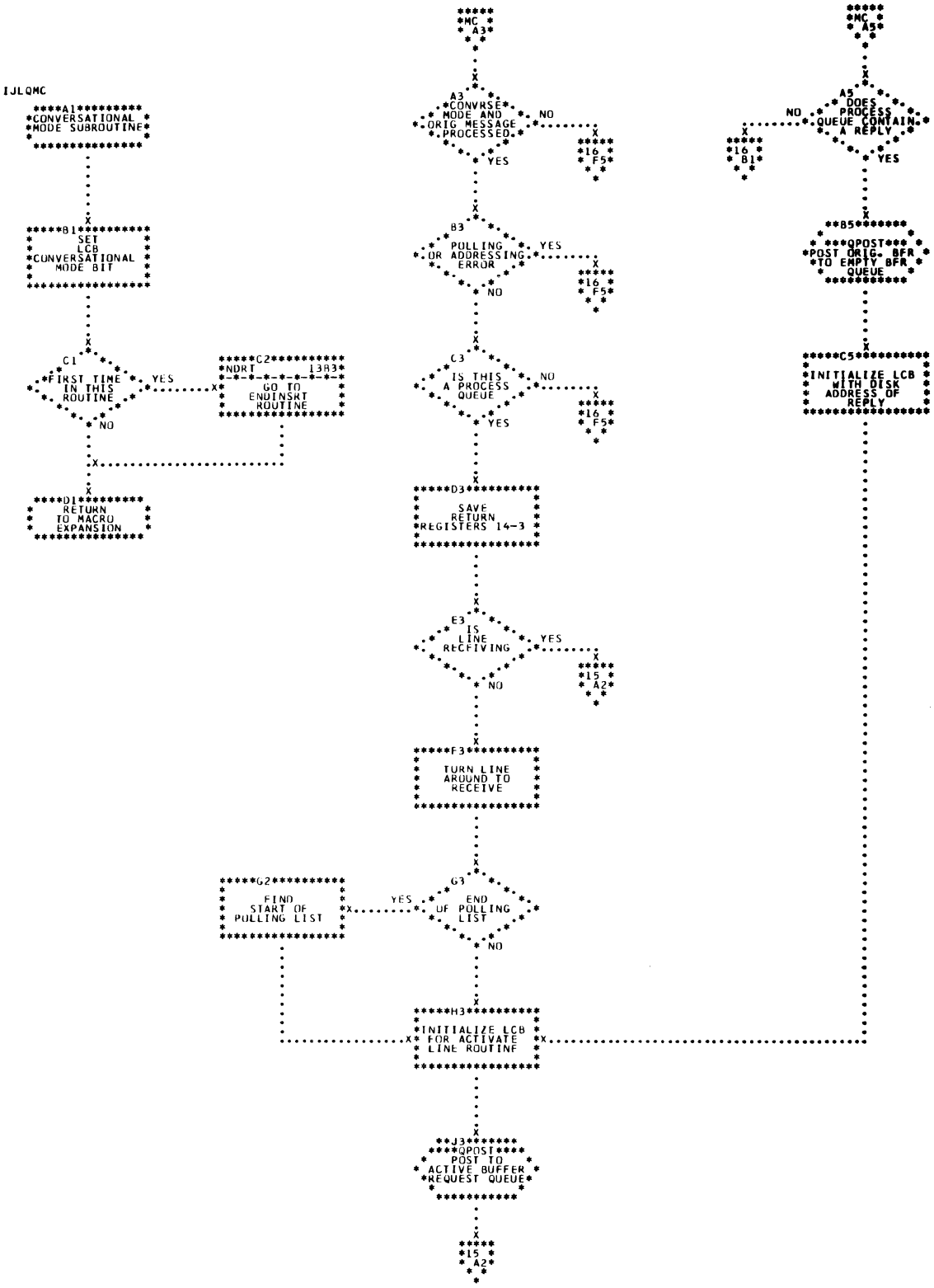


Chart MM. Message Mode Interface, Initiate Mode, and Priority Mode Routines

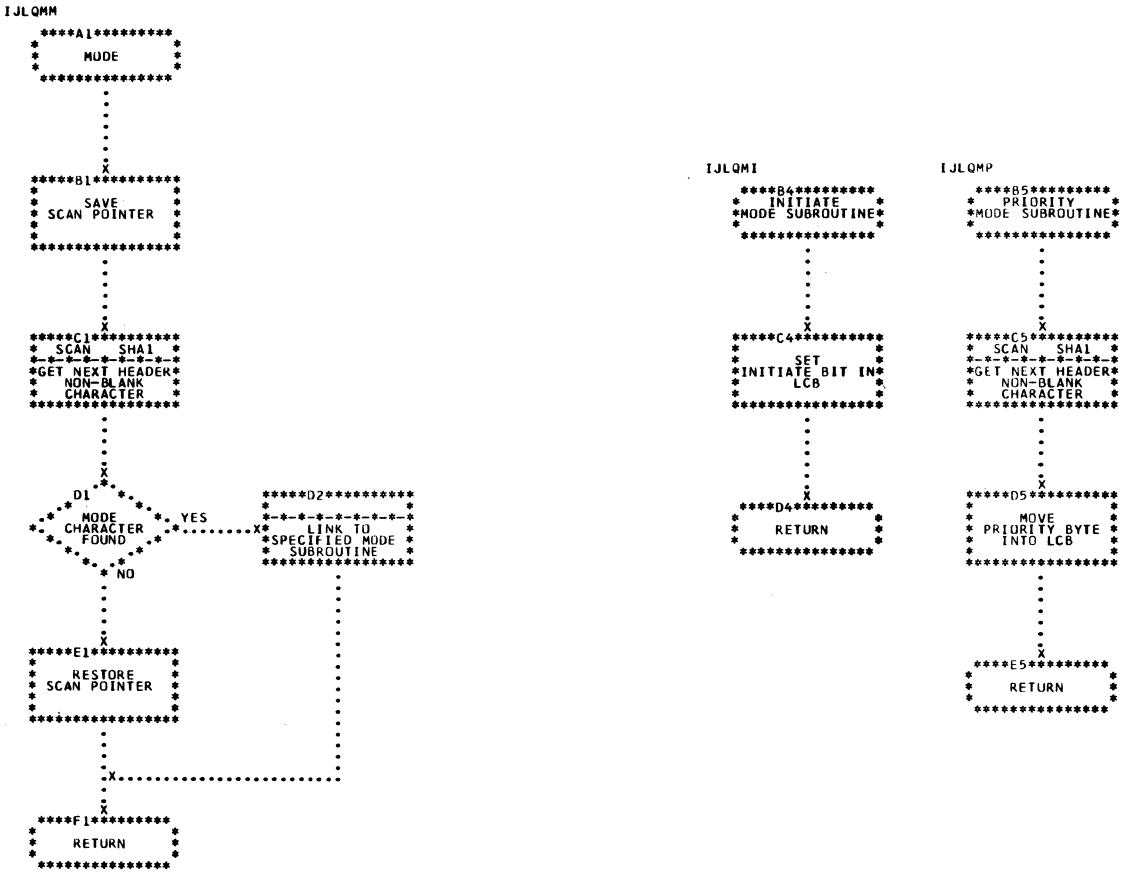


Chart MT. Message Type Routine

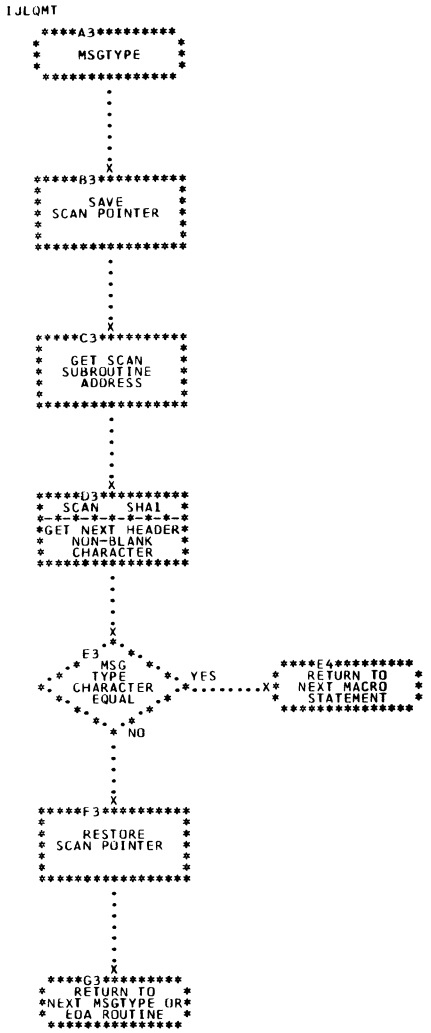


Chart MW. Message Writer Initiator Routine

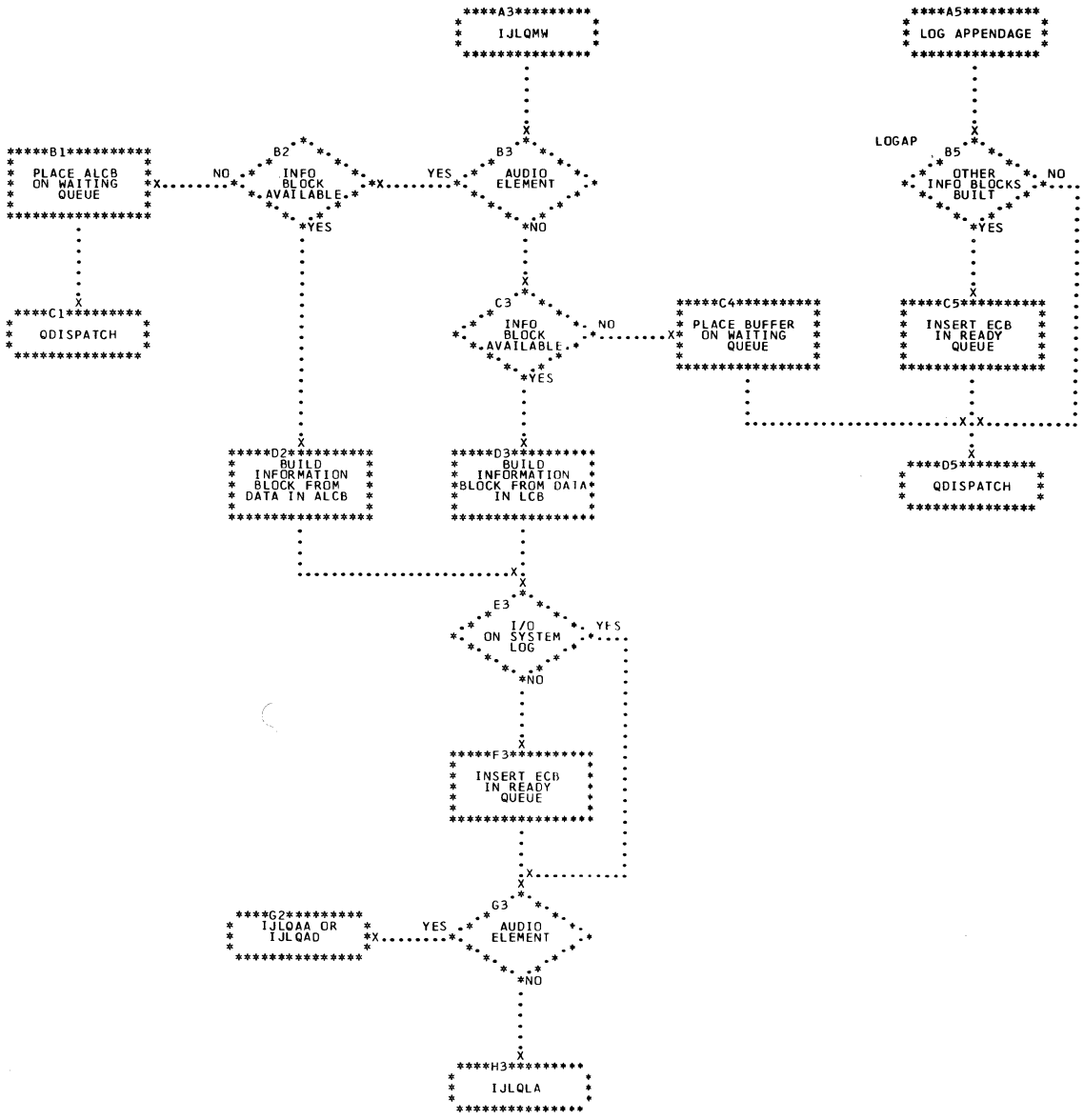


Chart OA. Operator Awareness Routine

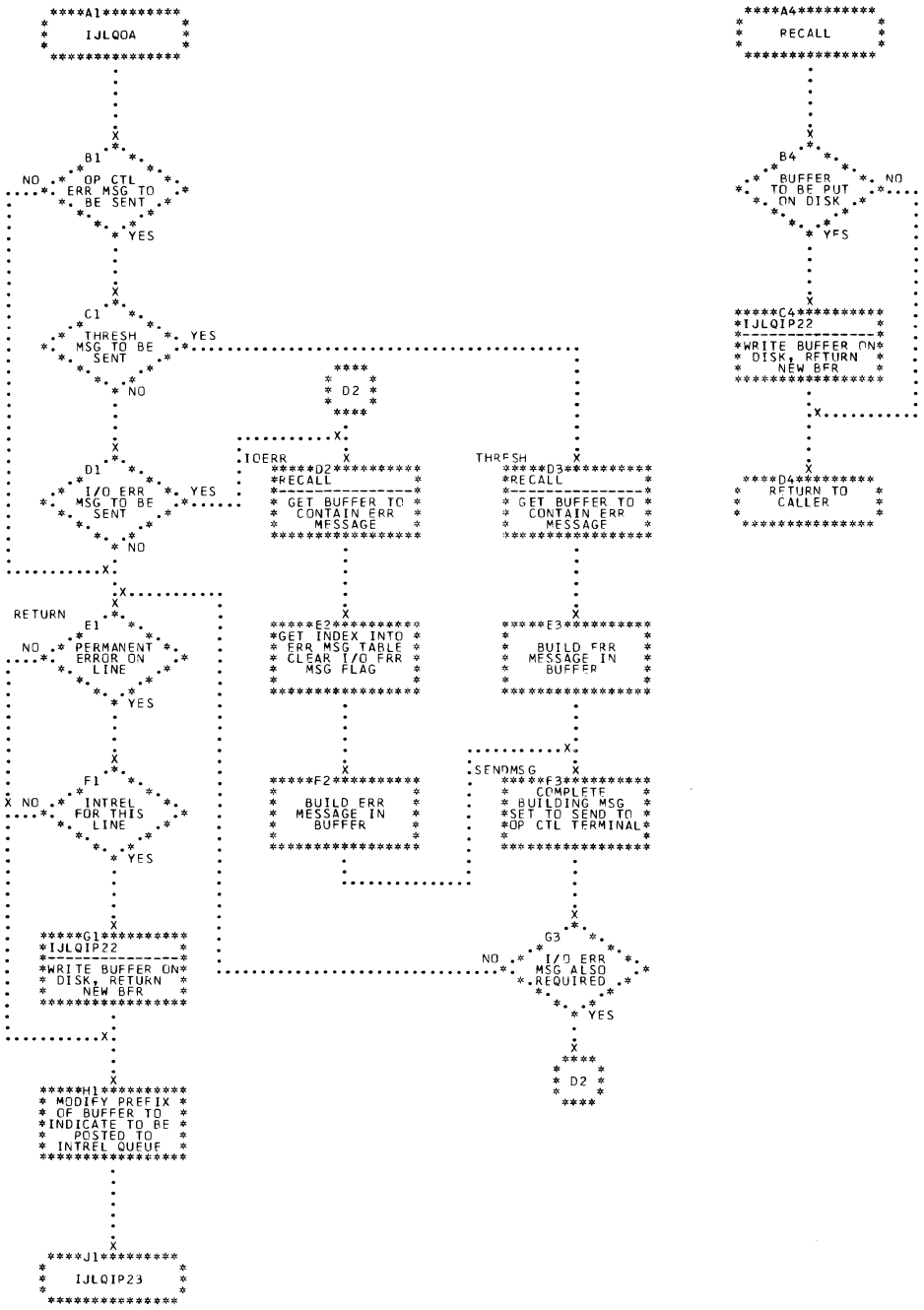


Chart OB. OBR/SDR Routine

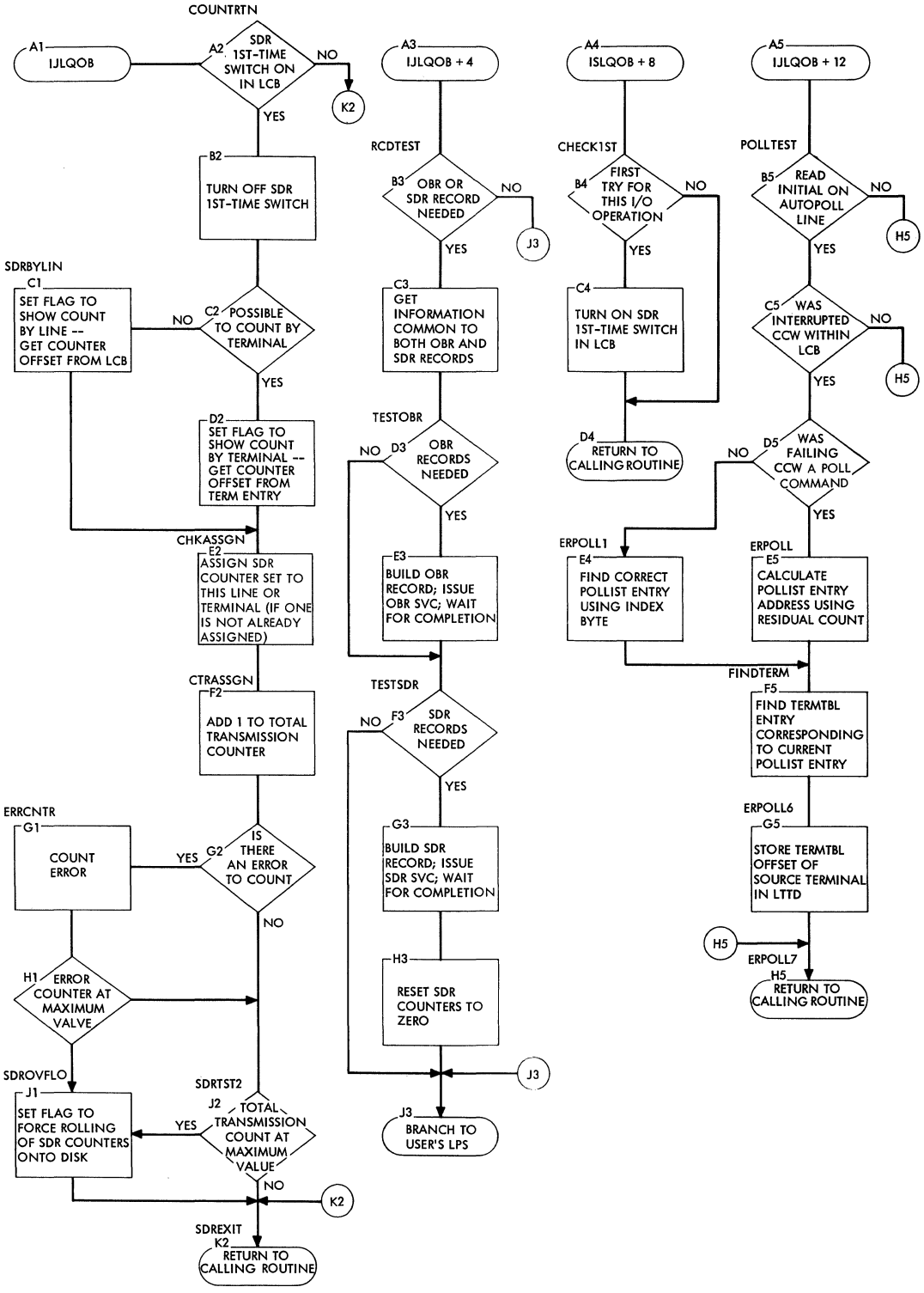


Chart OC. Operator Control Routine

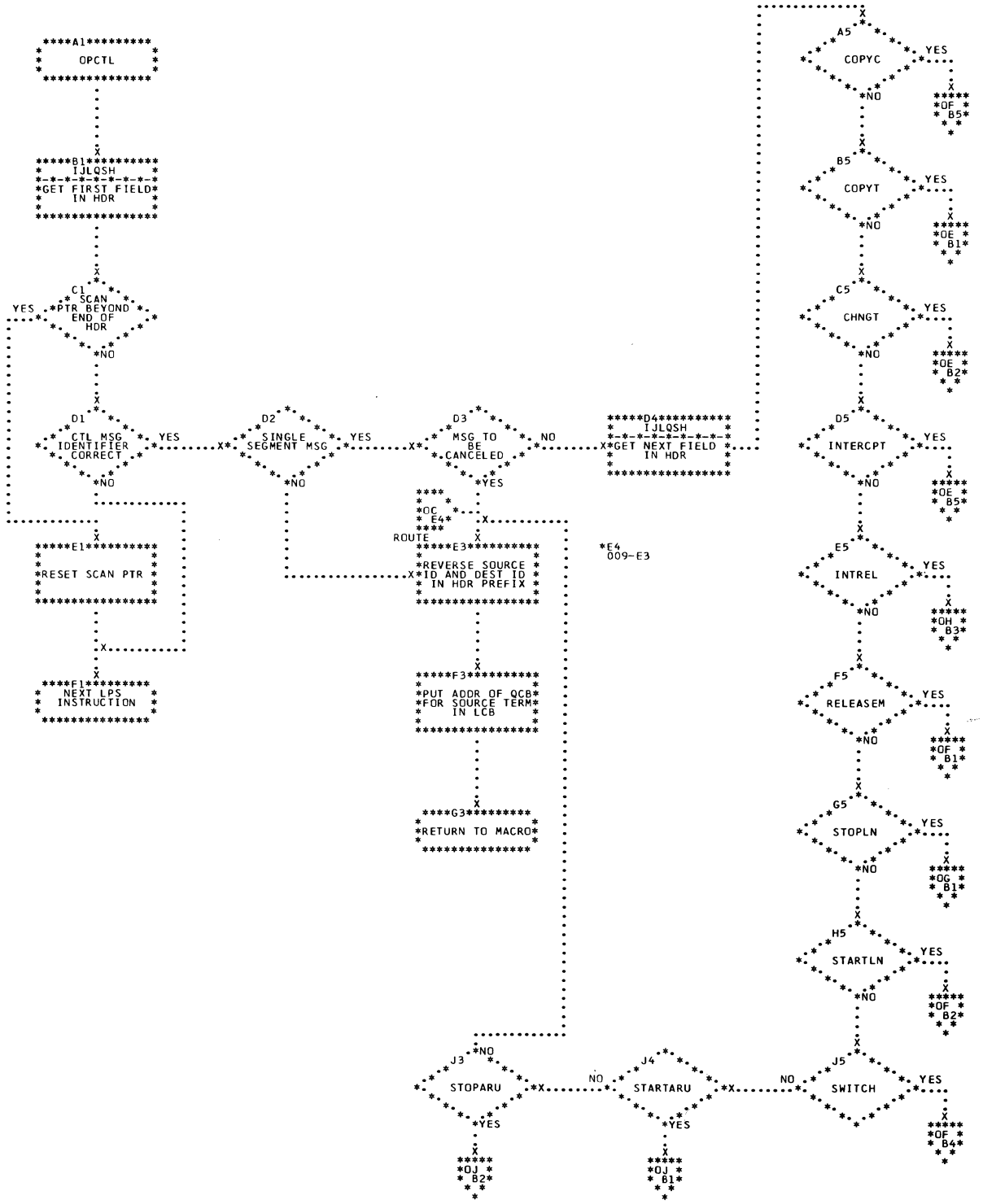


Chart OD. Operator Control Routine: Common Subroutines

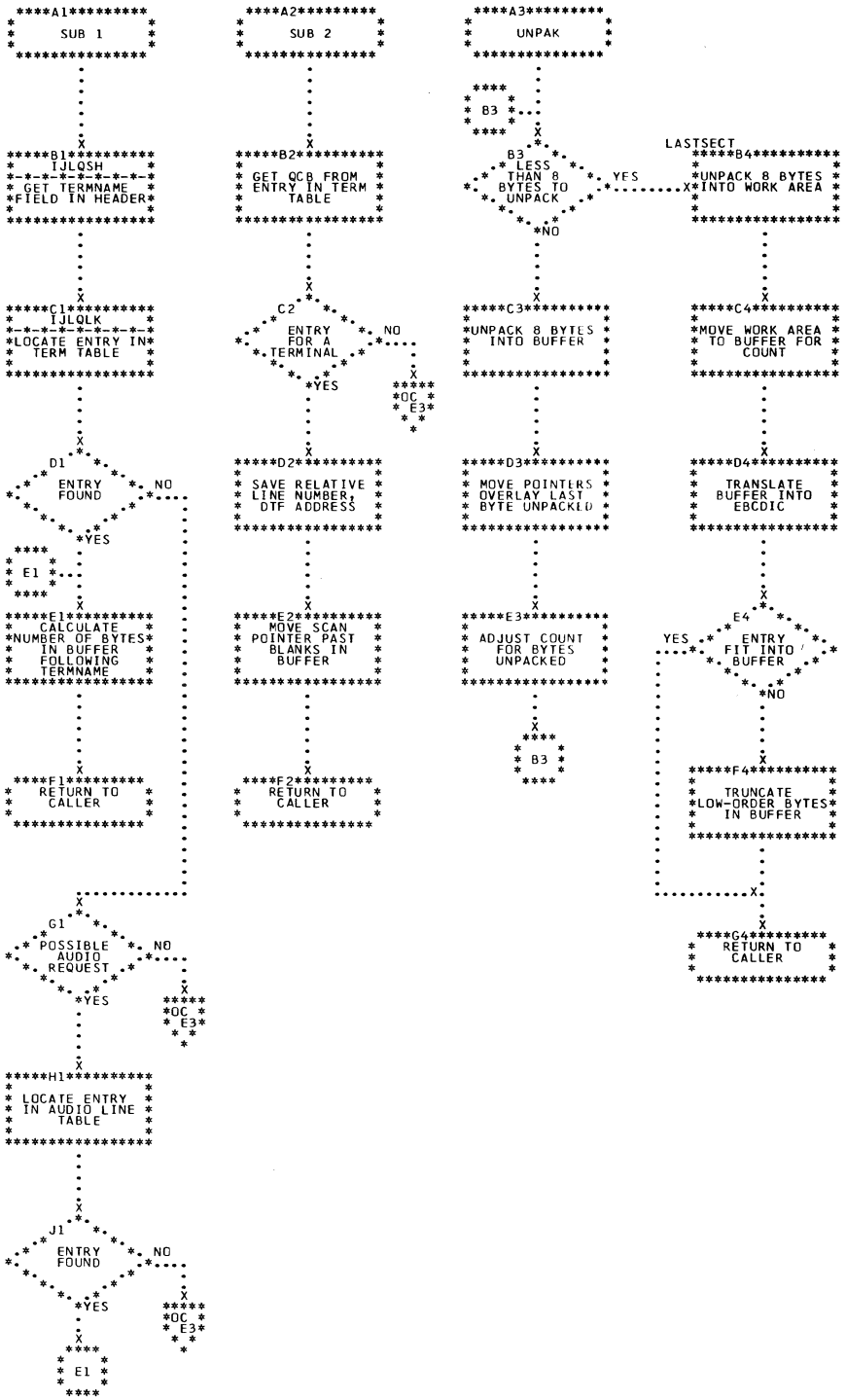


Chart OE. Operator Control Routine: COPYT, CHNGT, and INTRCPT

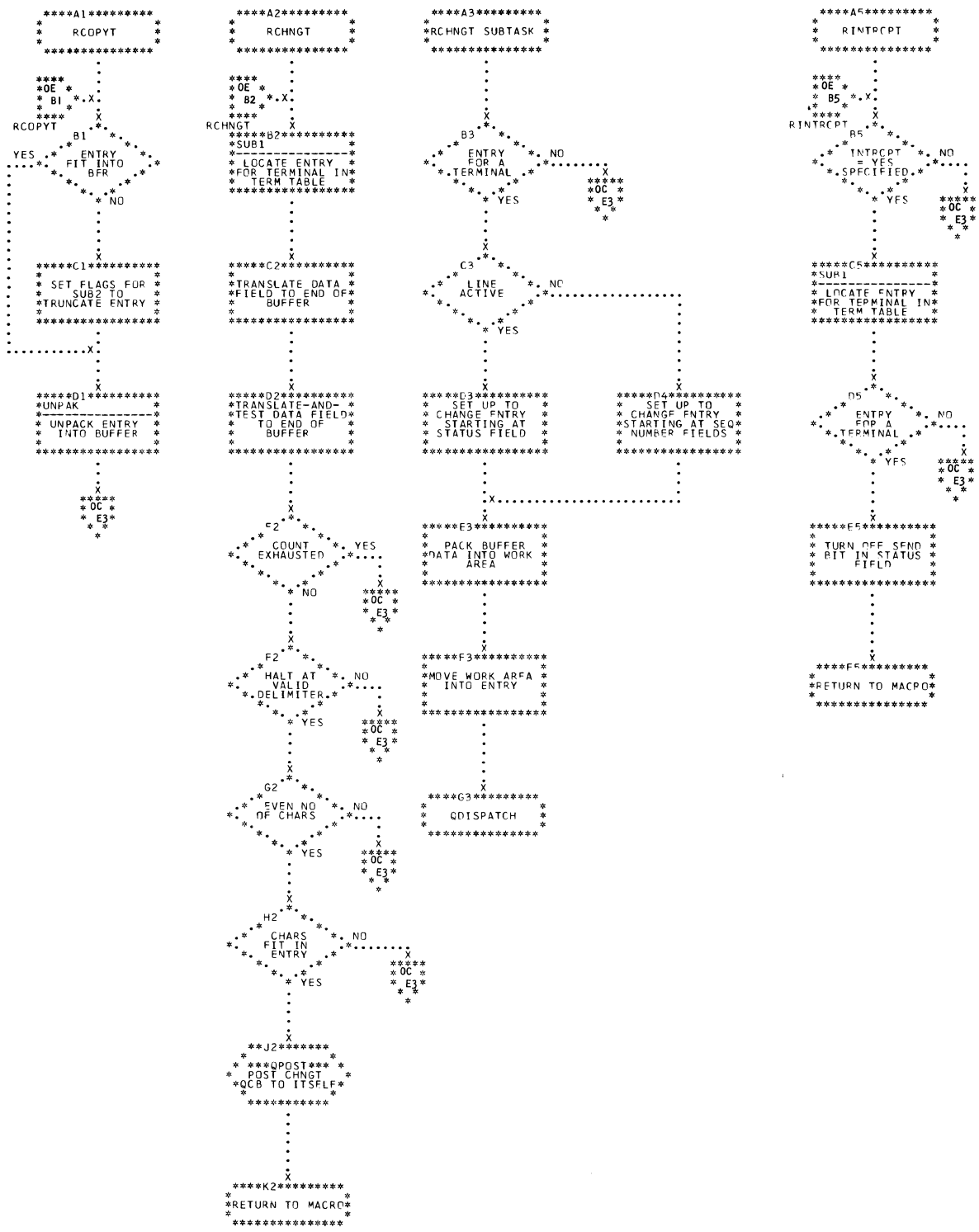


Chart OF. Operator Control Routine: RELEASEM, STARTLN, SWITCH, and COPYC

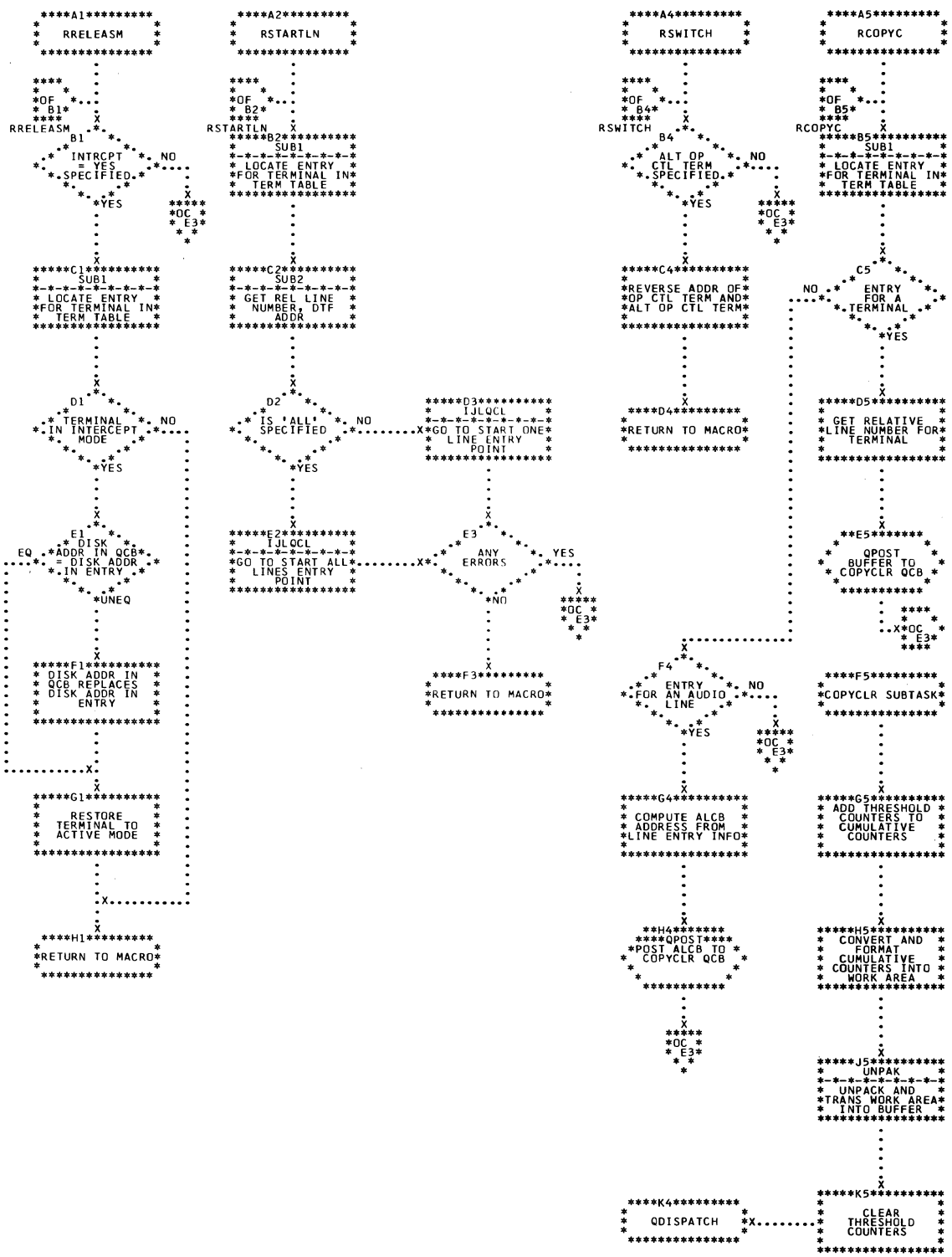


Chart OG. Operator Control Routine: STOPLN

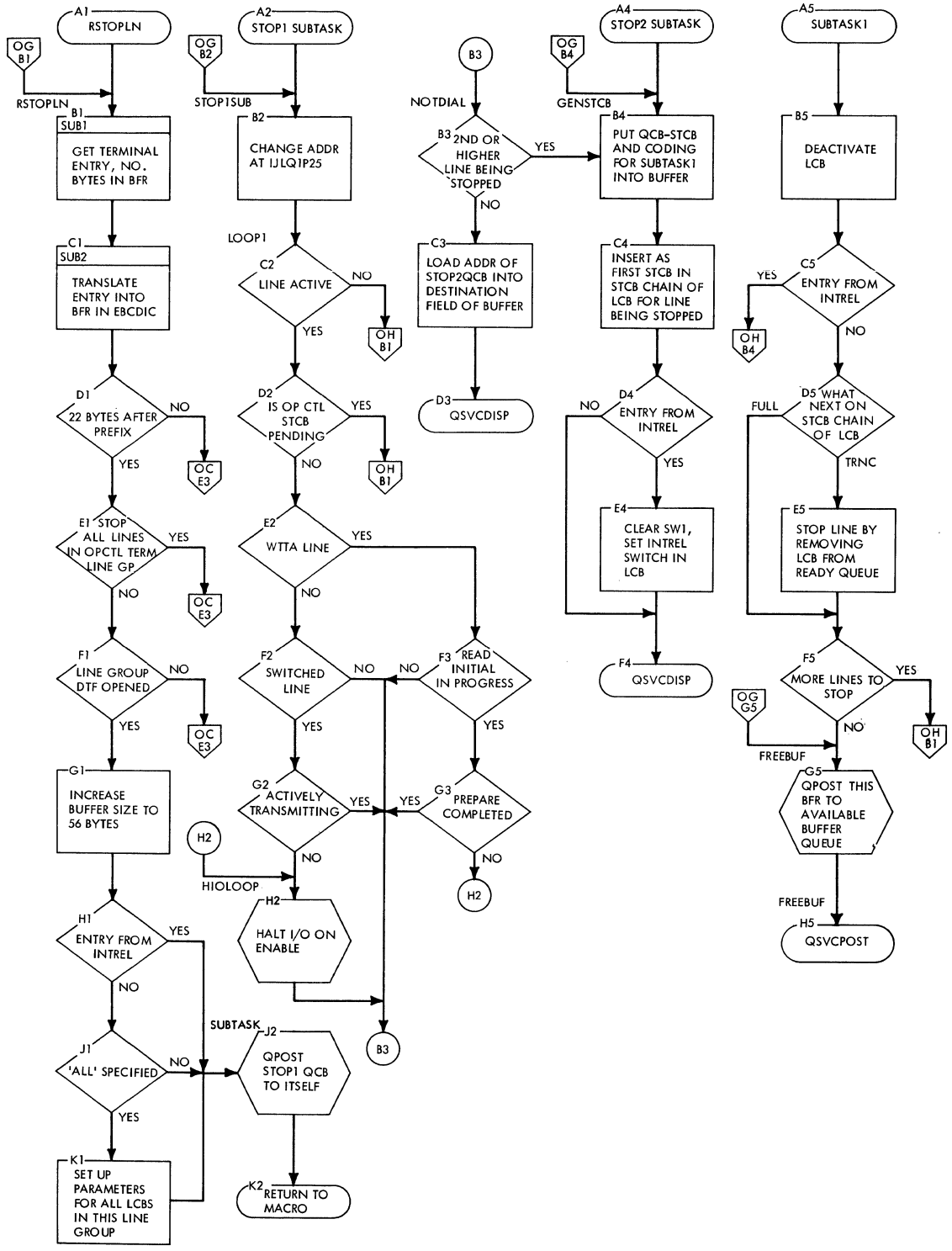


Chart OH. Operator Control Routine: STOPLN and INTREL

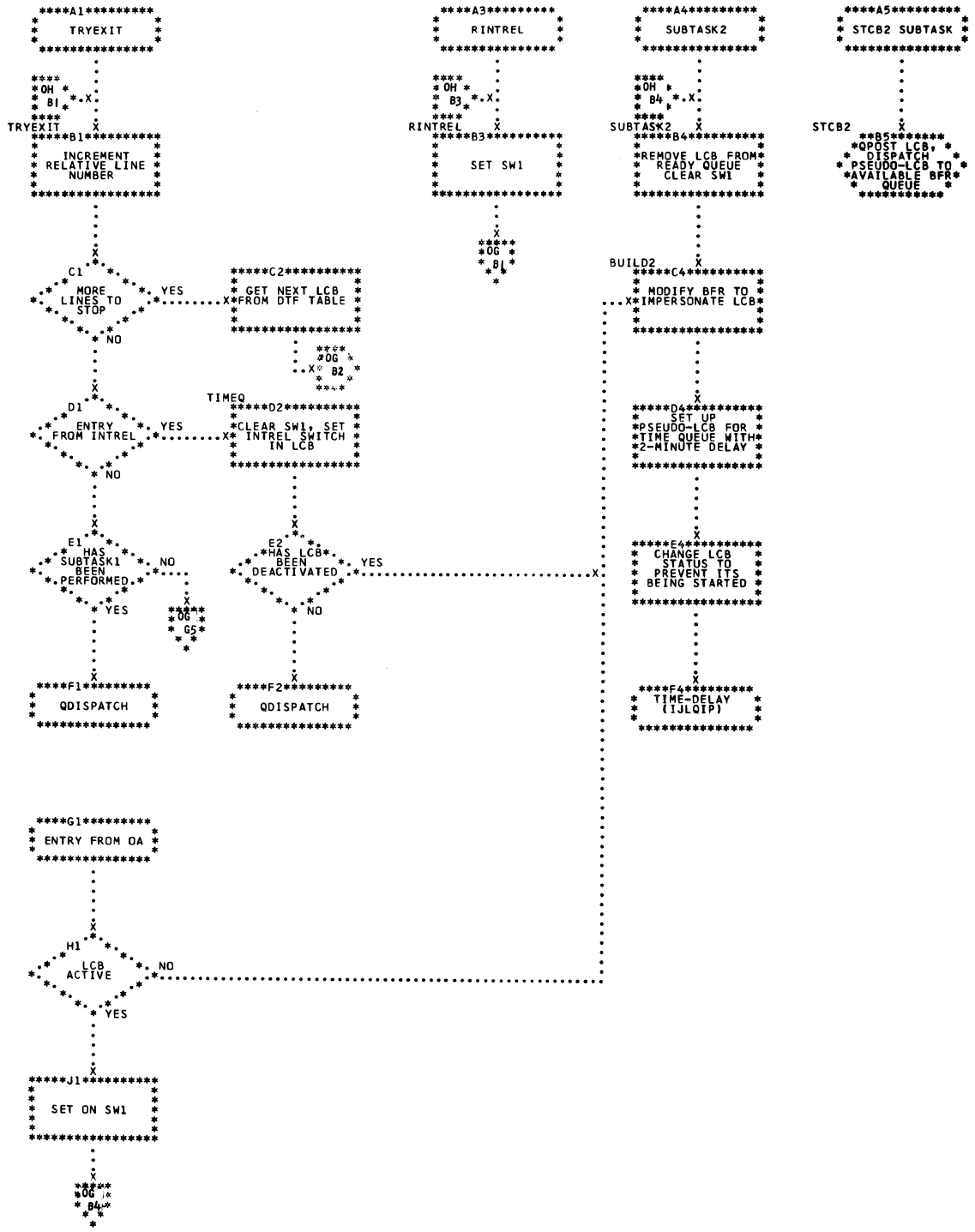


Chart OJ. Operator Control Routine: STARTARU and STOPARU

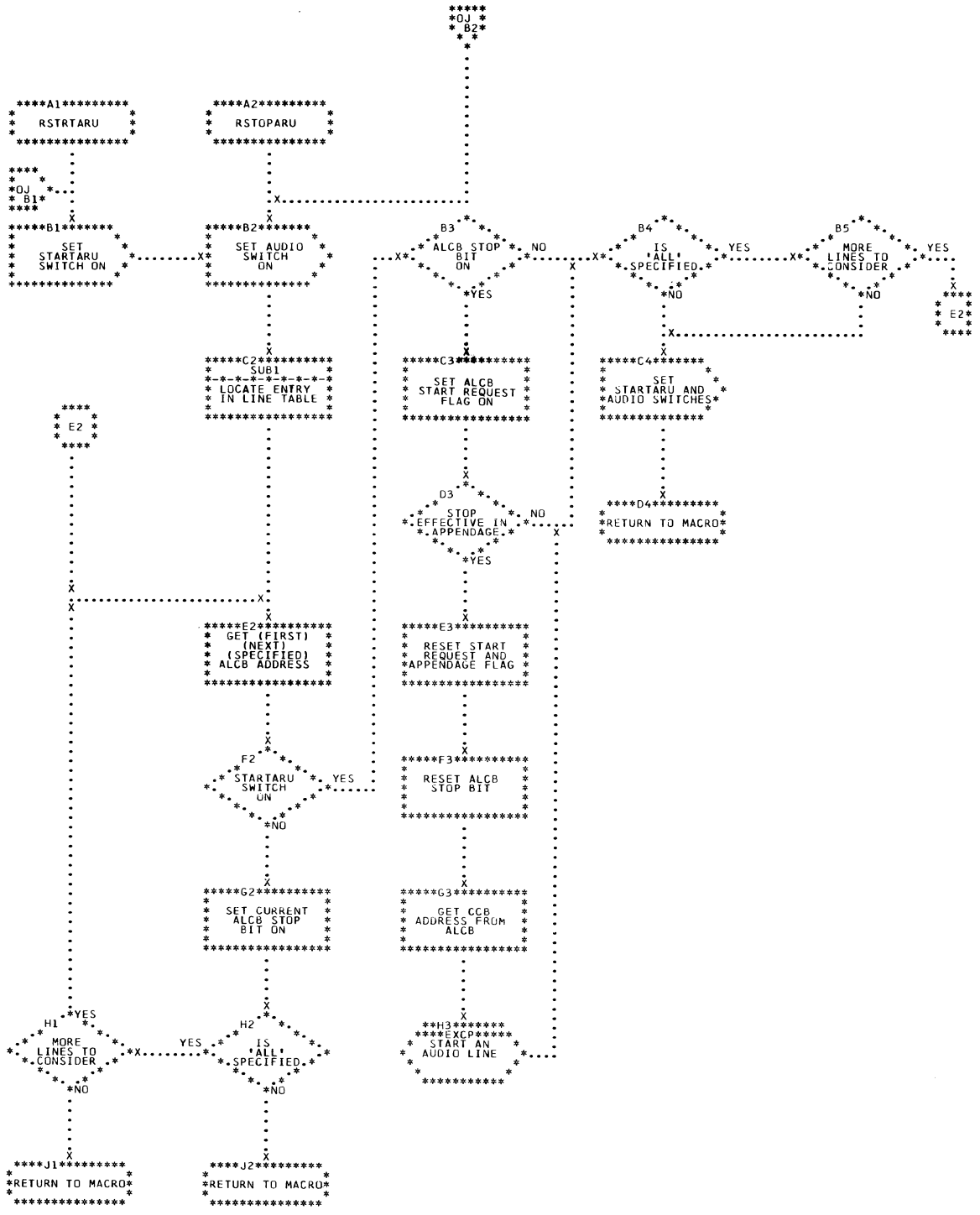


Chart 01. QTAM Open Monitor/Open DASD Message Queue File Routine (Phase 1)

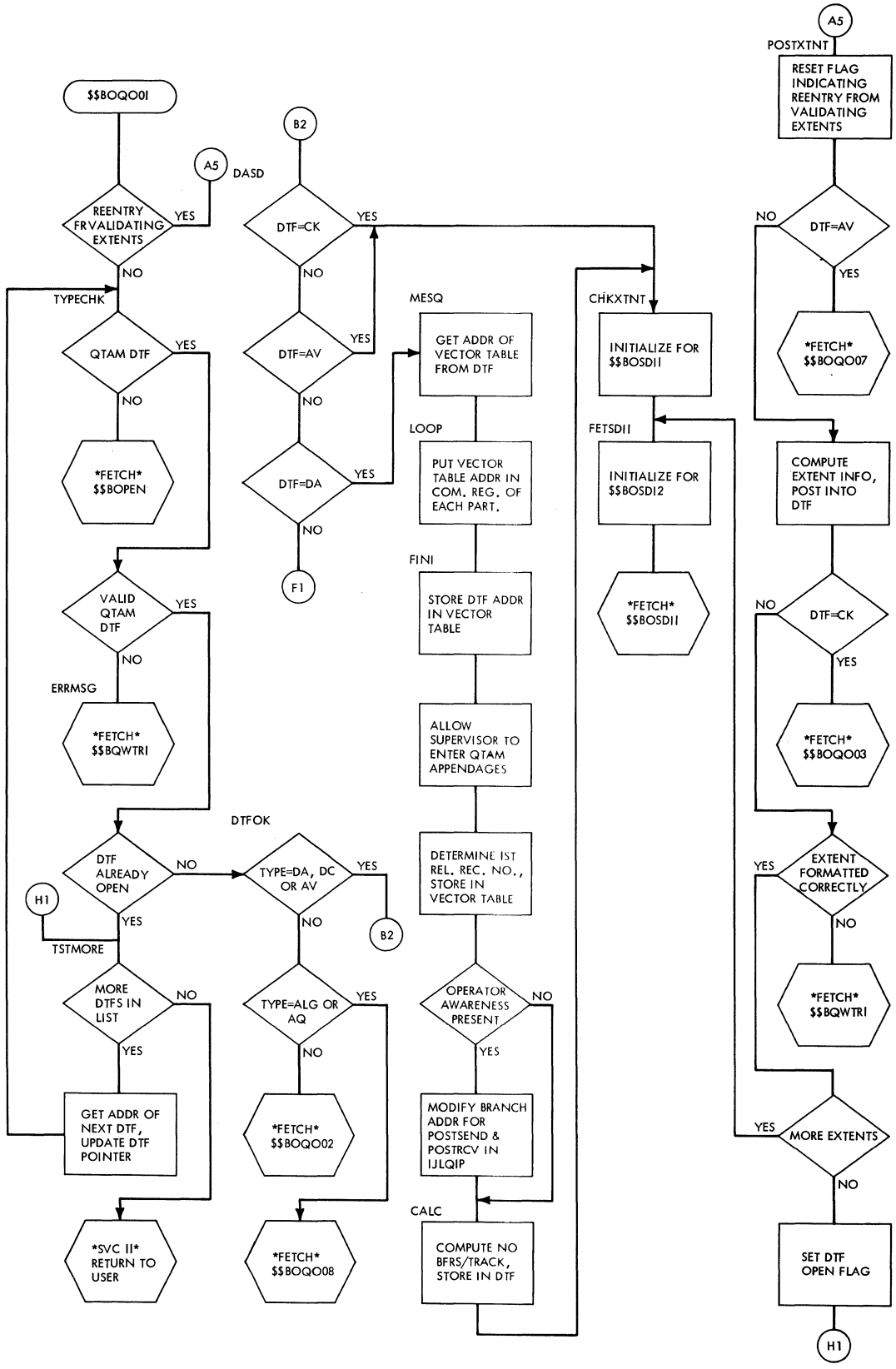


Chart 02. Open Nonaudio Line Group/MS Queue File Routine

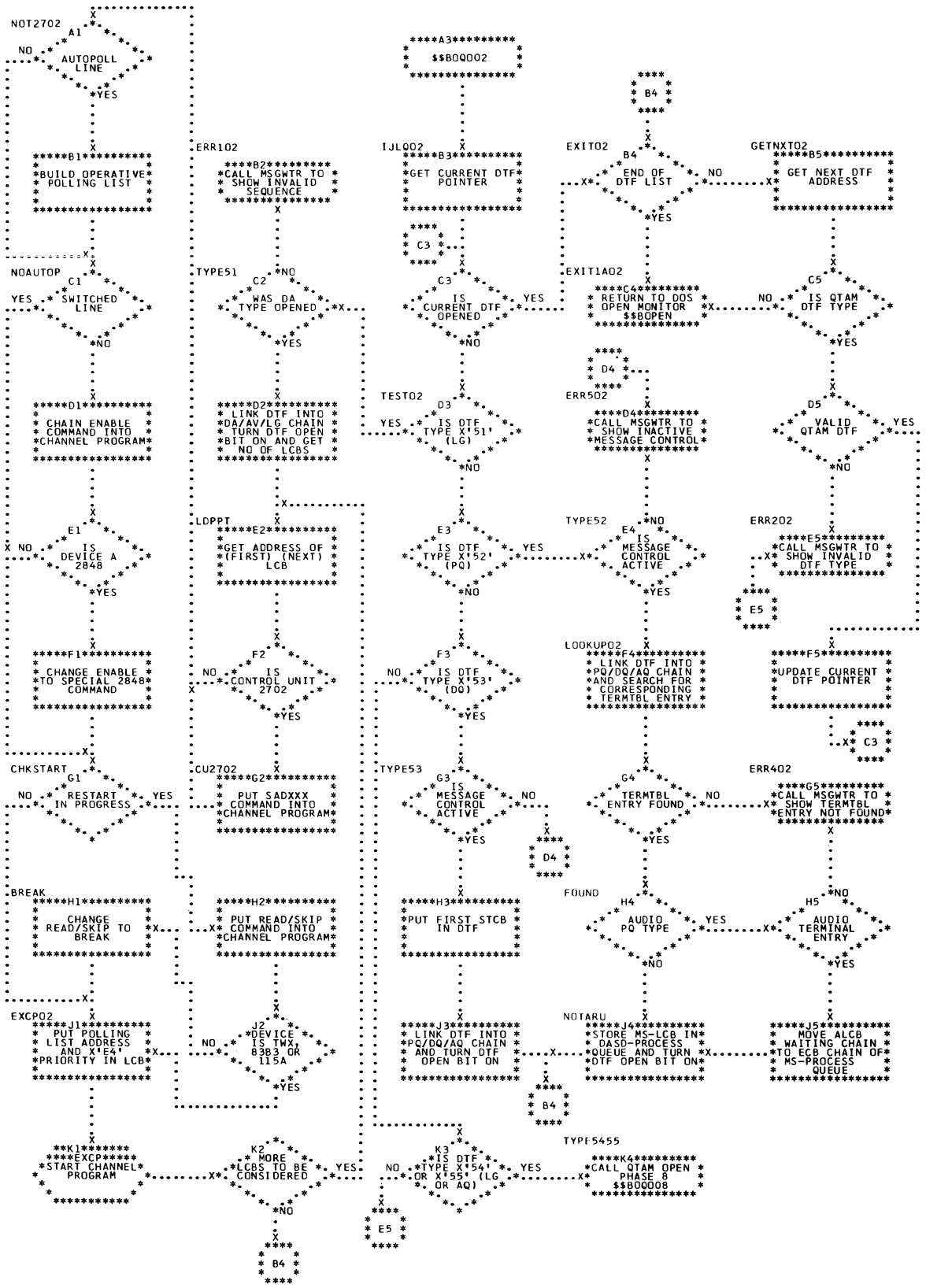


Chart 03. Open Checkpoint/Restart Routine (Phase 1)

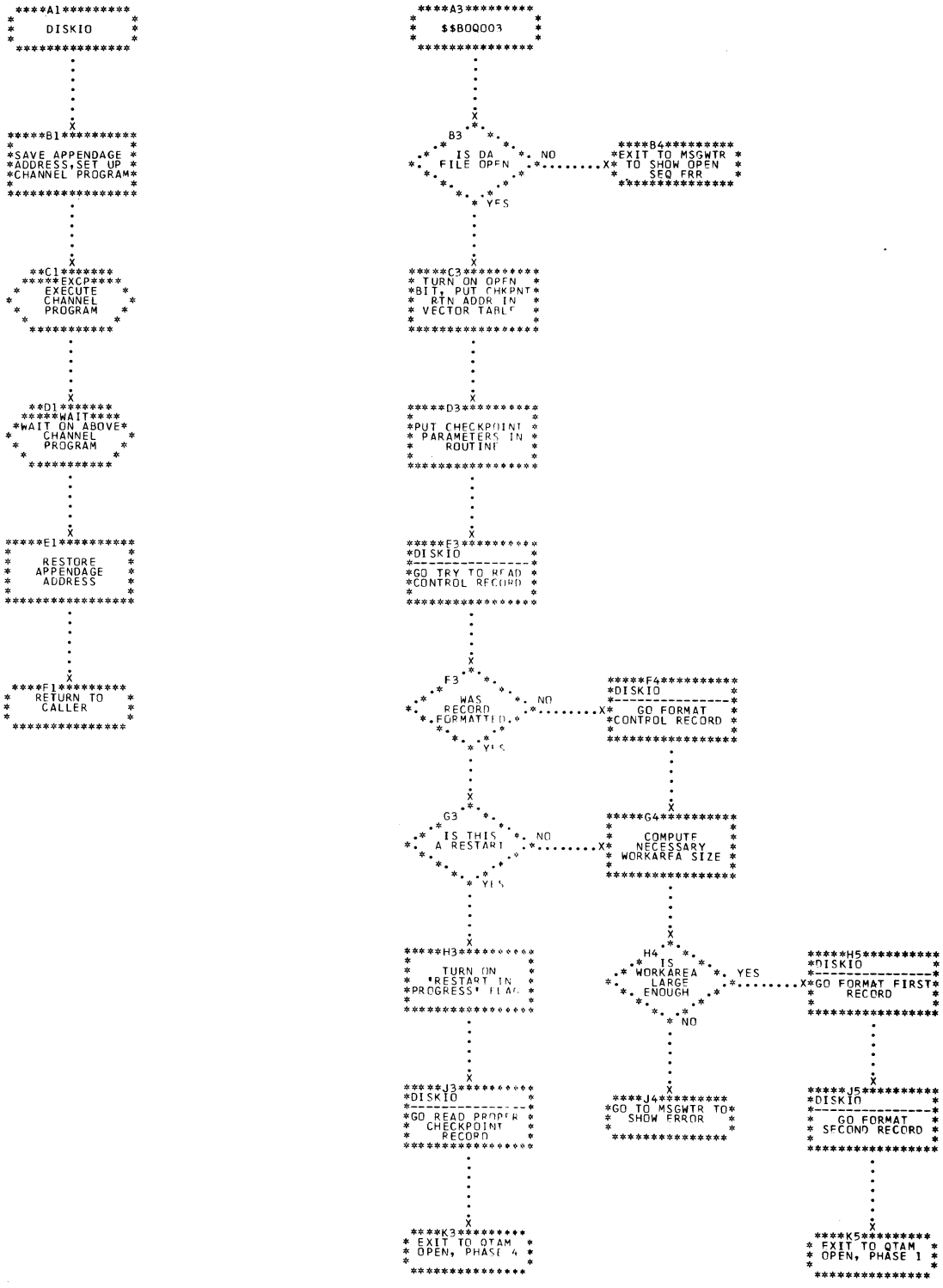


Chart 04. Open Checkpoint/Restart Routine (Phase 2)

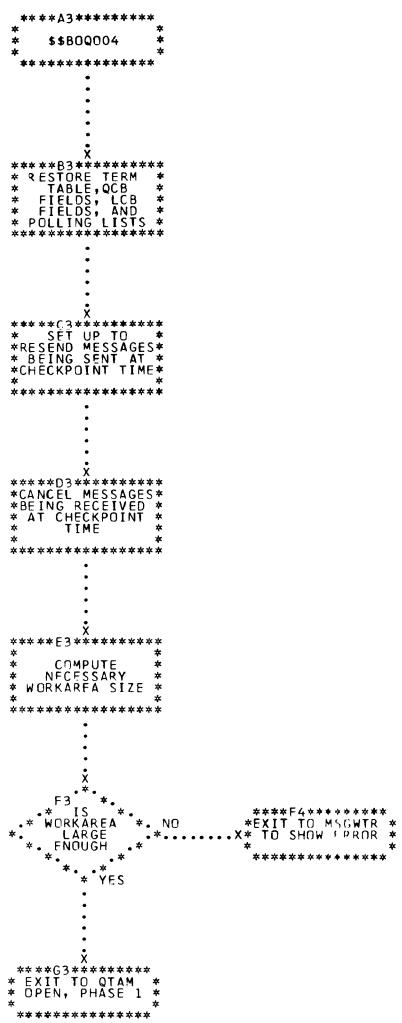


Chart 07. QTAM Open IBM 7772 DCV Vocabulary File Routine

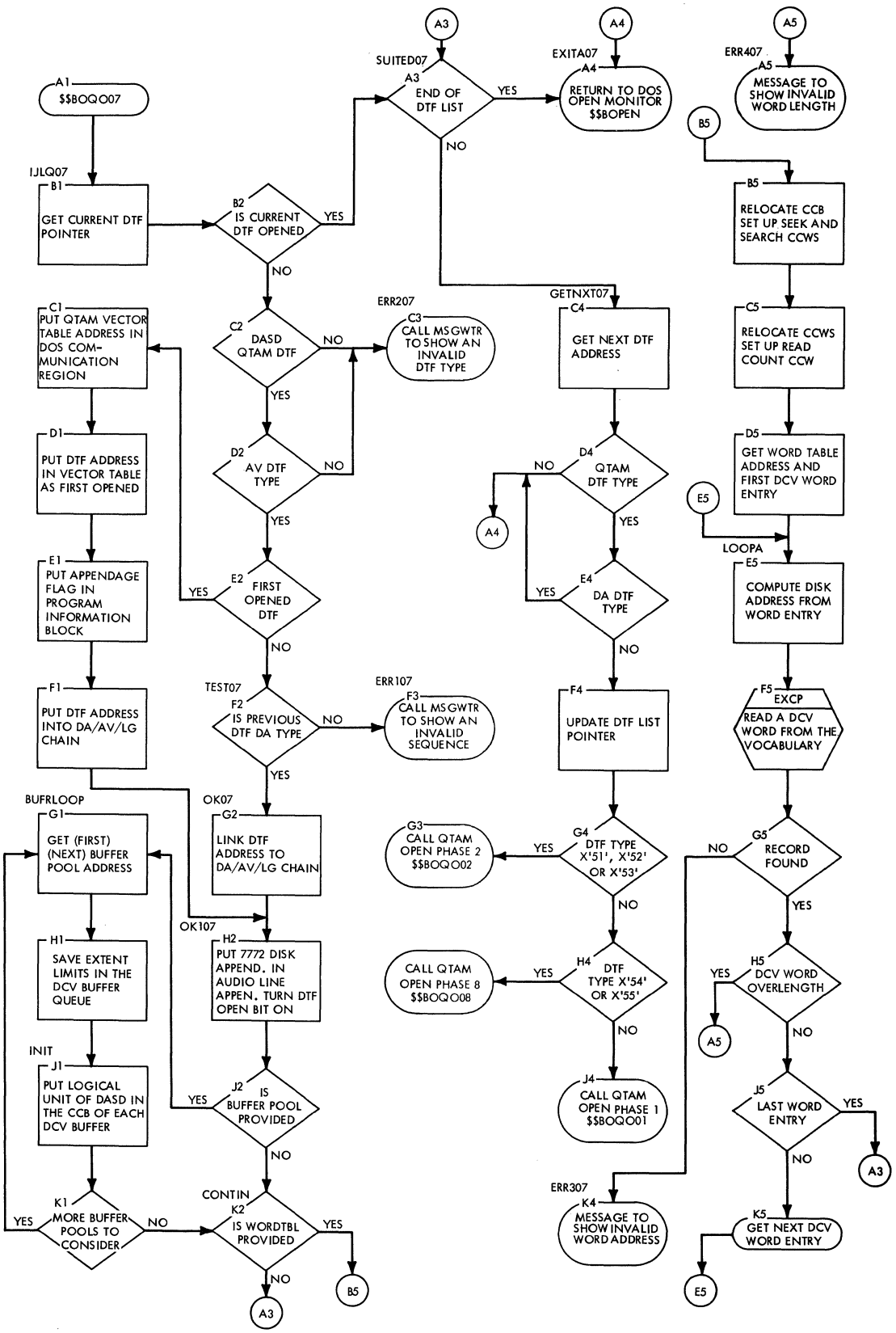


Chart 08. QTAM Open Audio Line Group/Output Queues File Routine

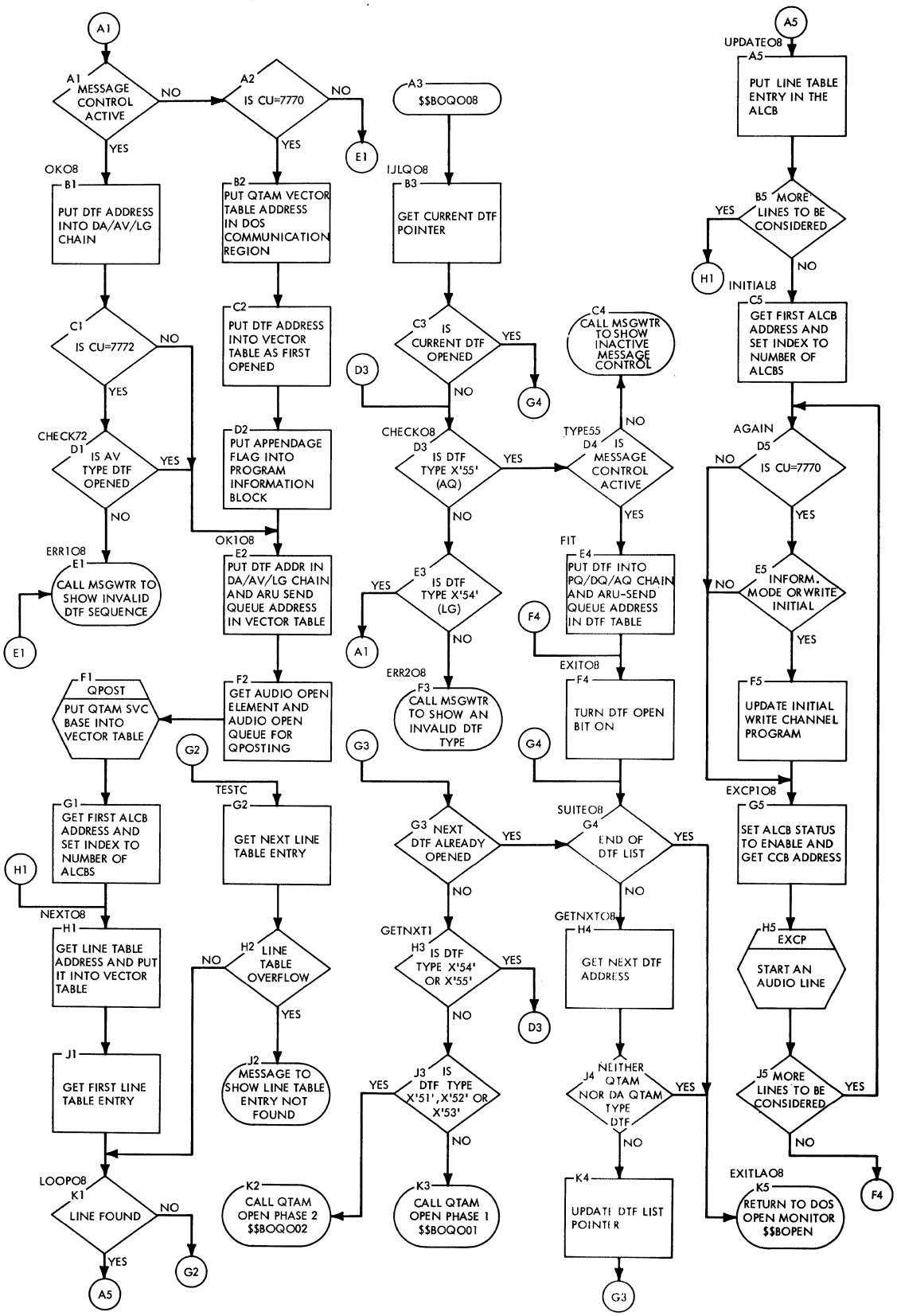


Chart PA. Put Audio Message Routine

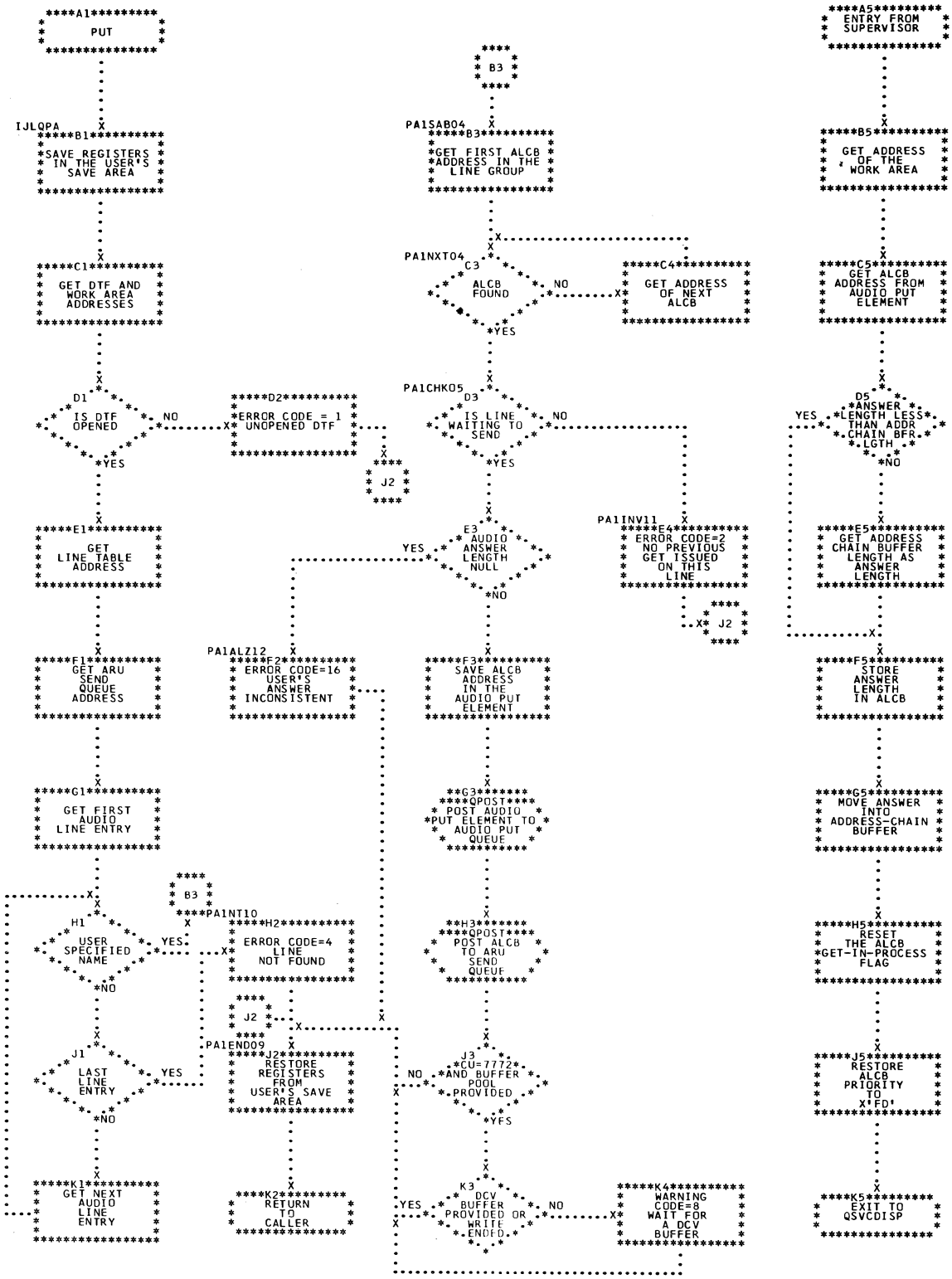


Chart PL. Polling Limit Control Routine

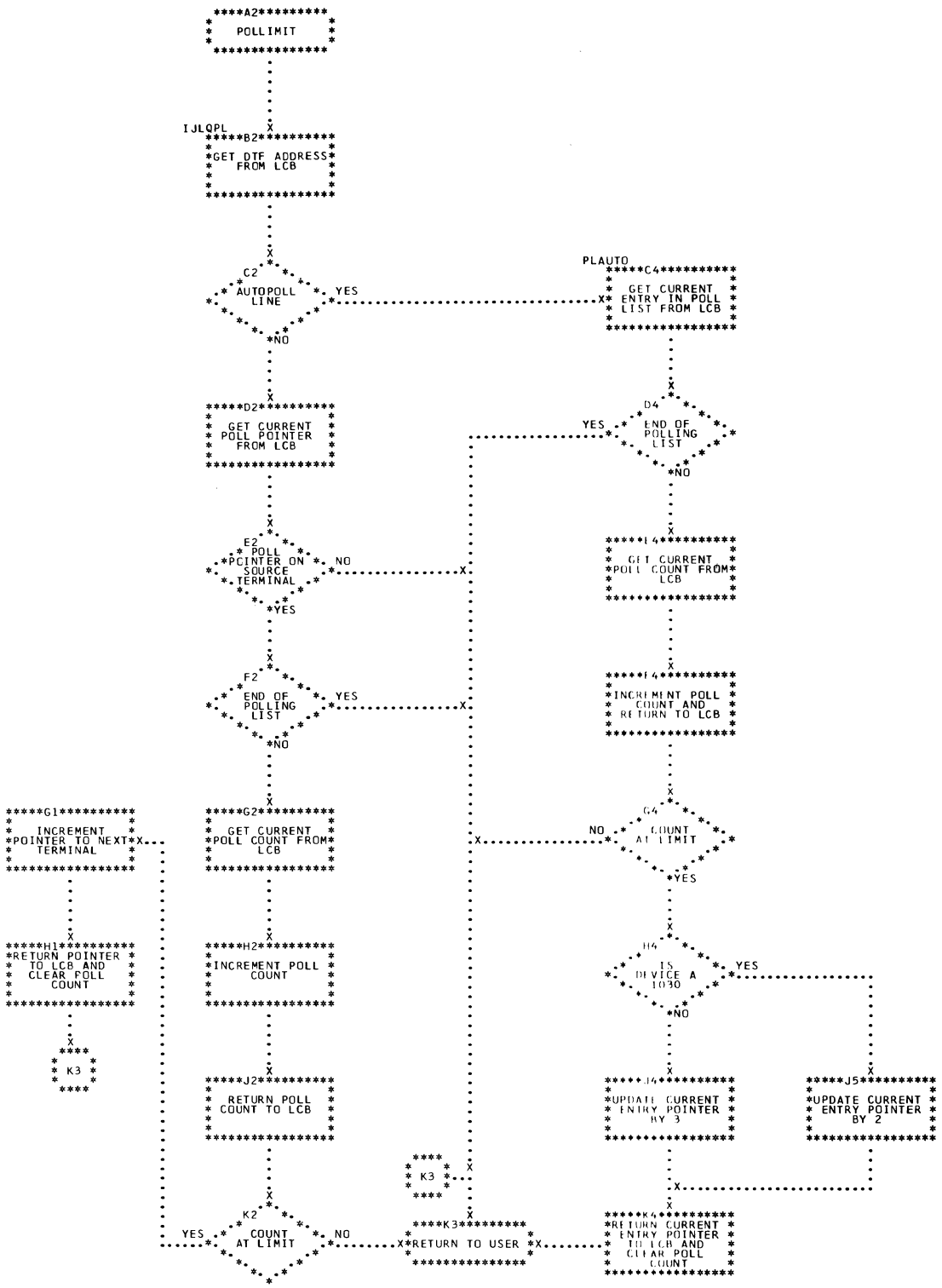


Chart PM. PUT Message Routine

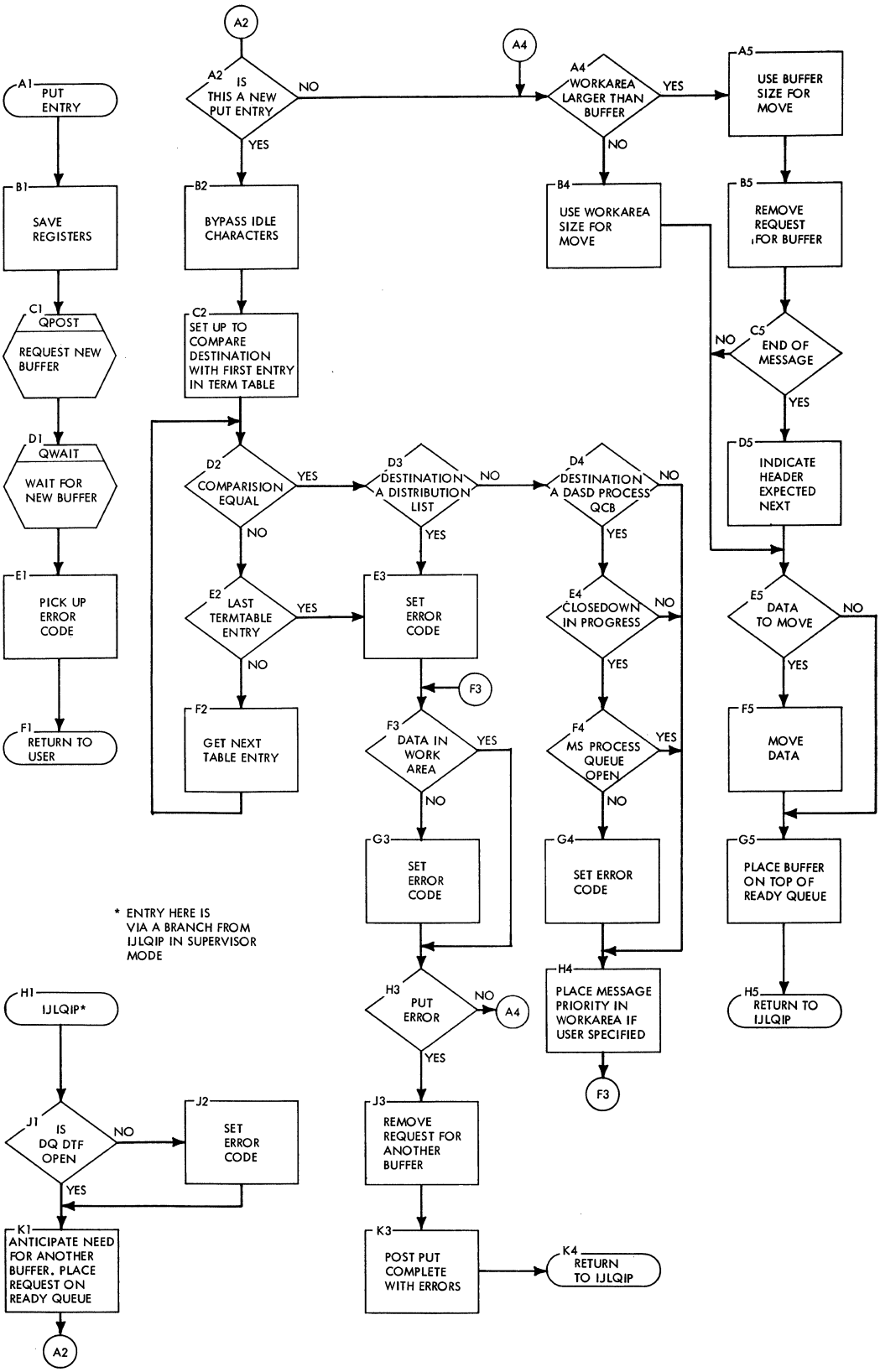


Chart PQ. Put Record Routine (Part 1 of 2)

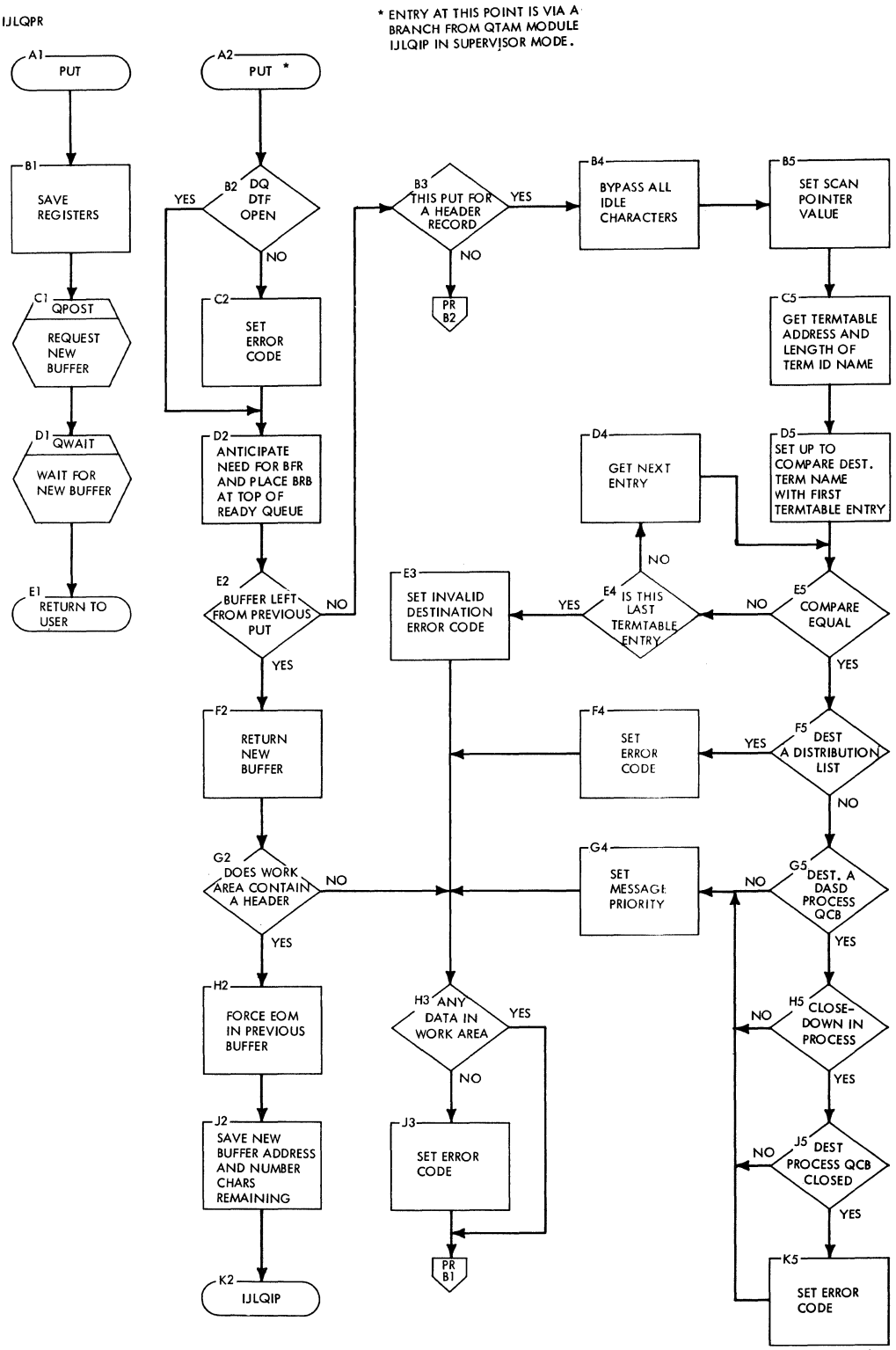


Chart PR. Put Record Routine (Part 2 of 2)

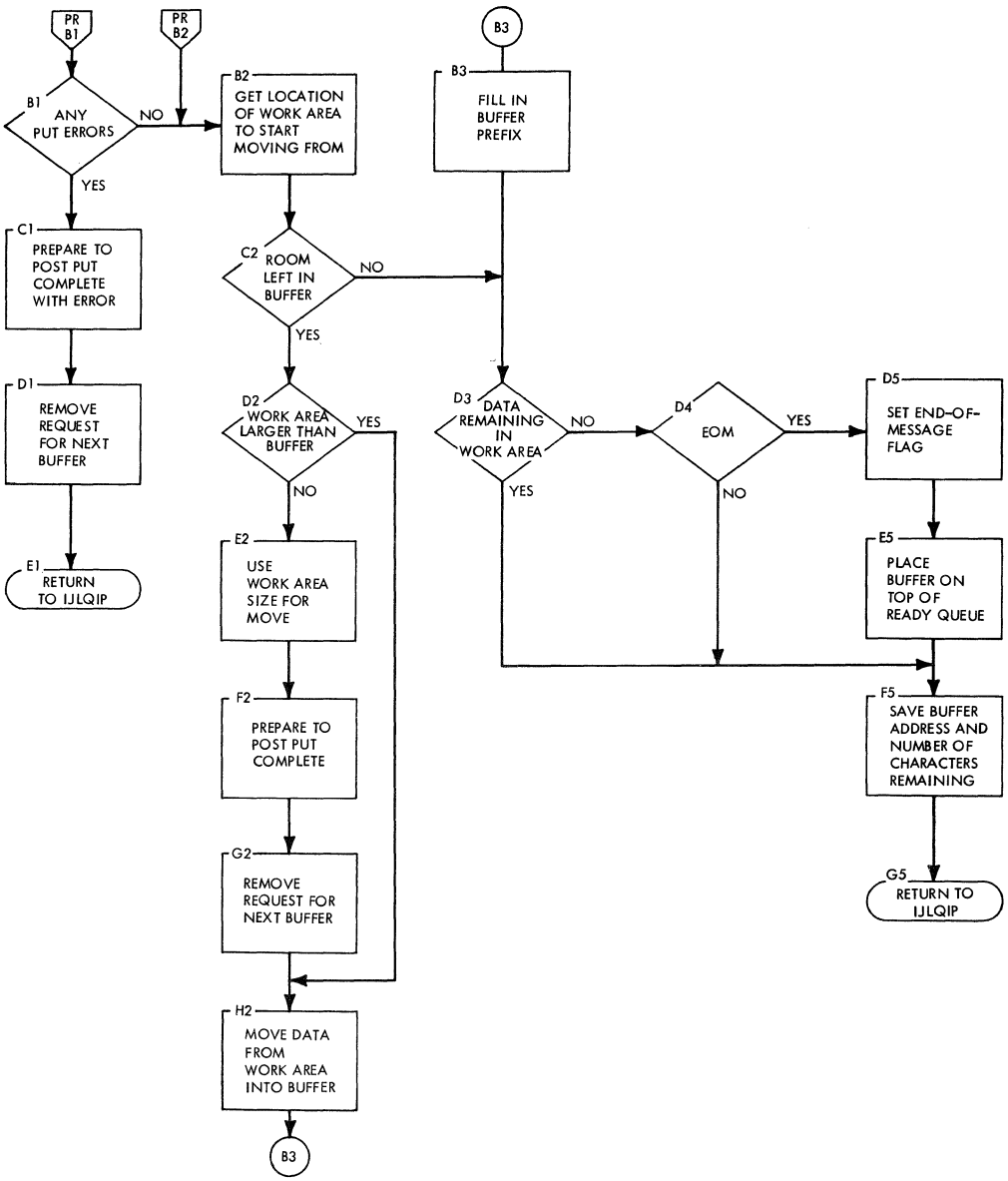


Chart PS. PUT Segment Routine

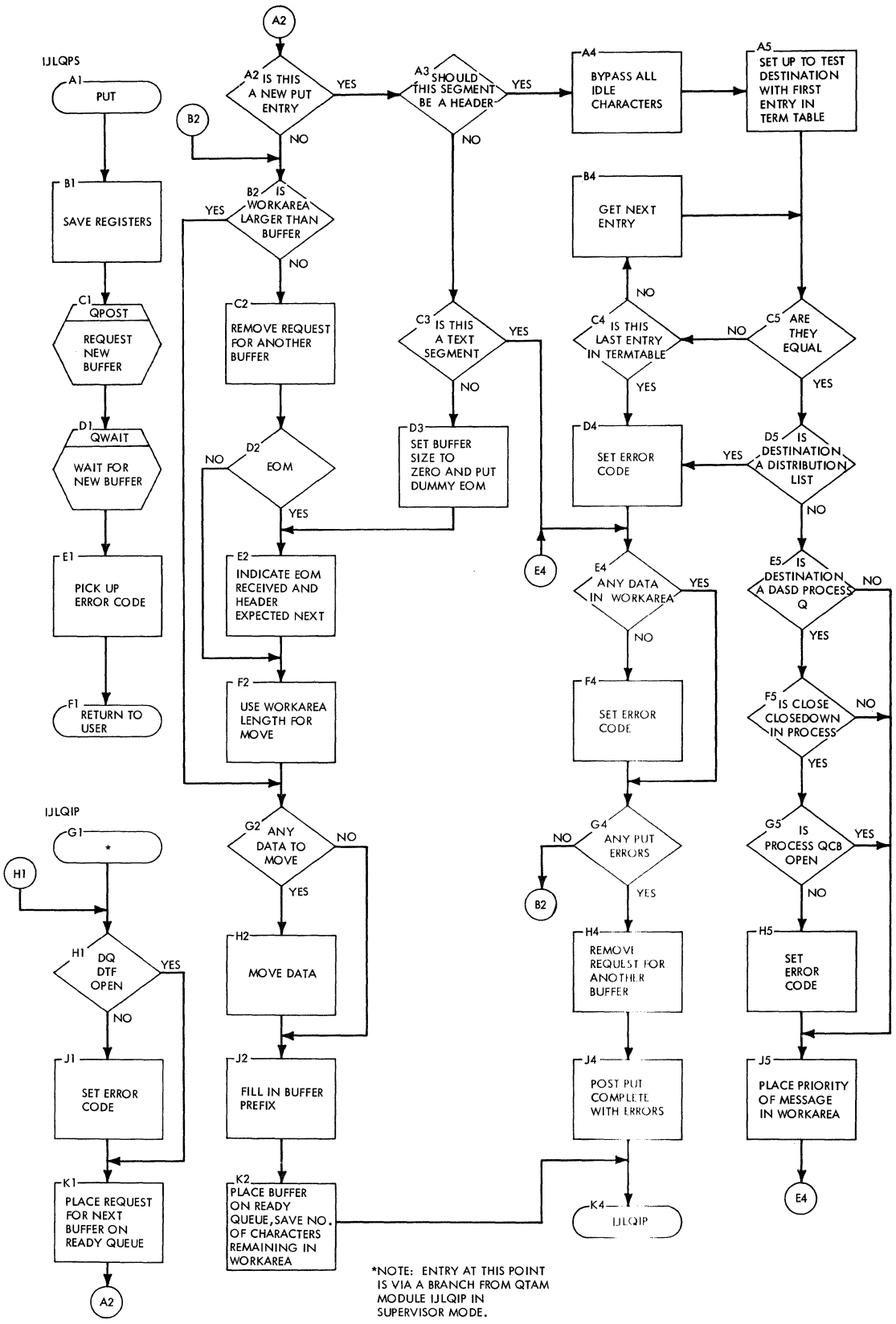


Chart PZ. Pause Routine

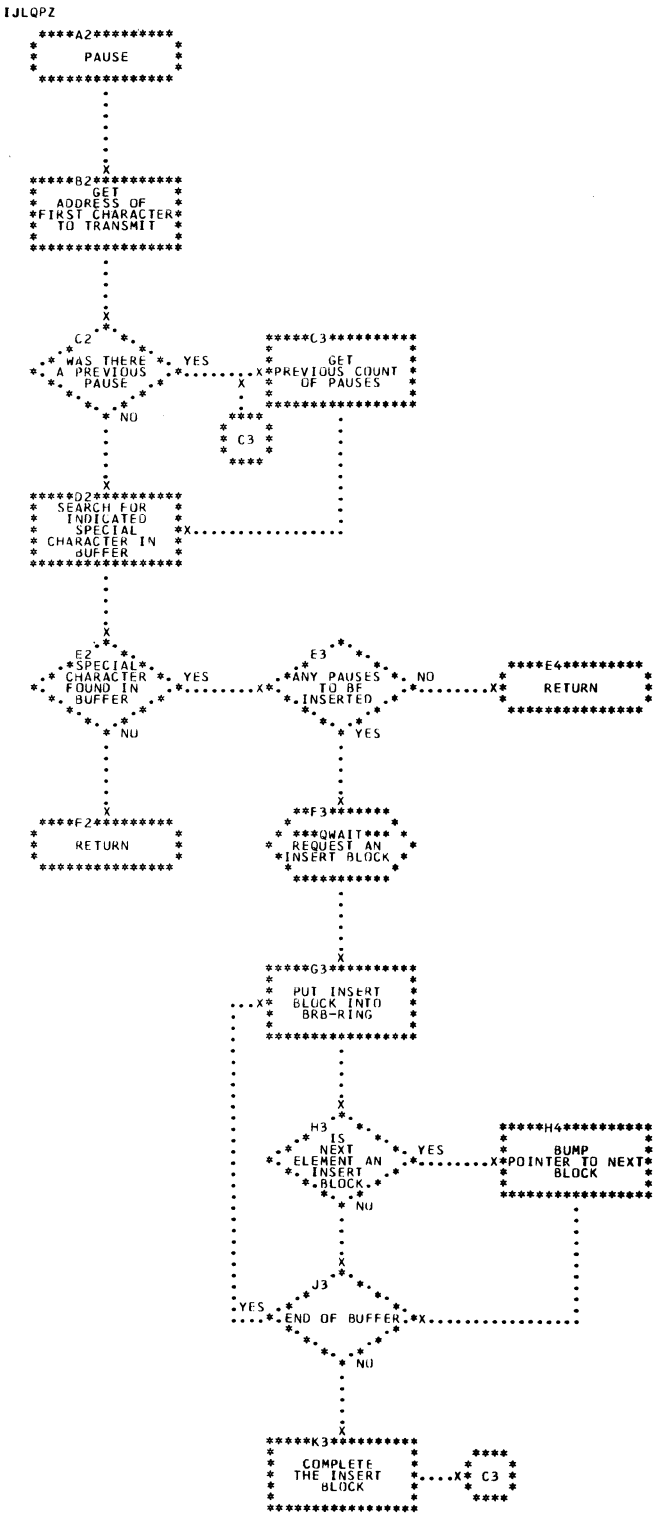


Chart QT. CLOSE Message Control

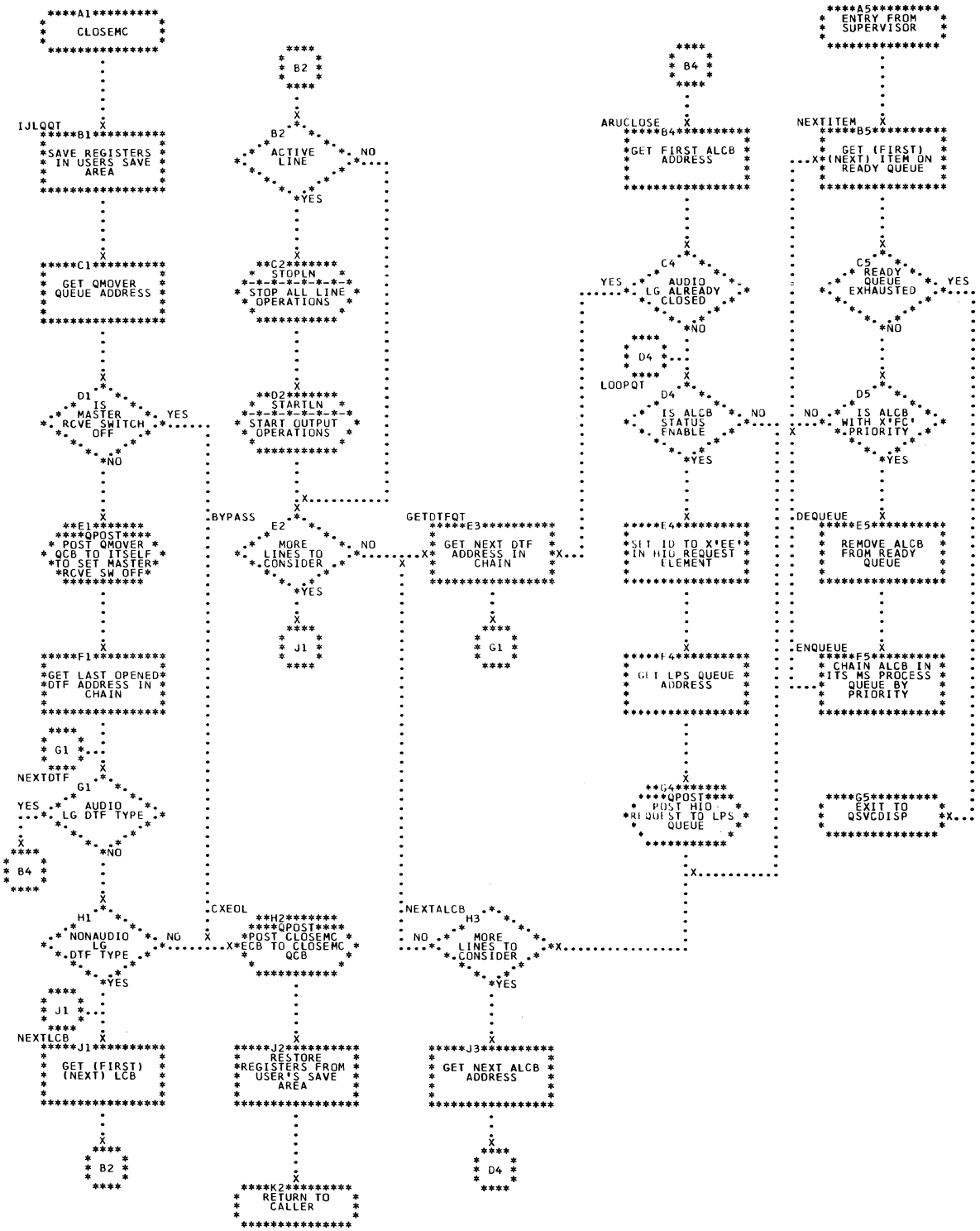


Chart QW. QTAM SVC/Subtask Control Routine (Part 1 of 3)

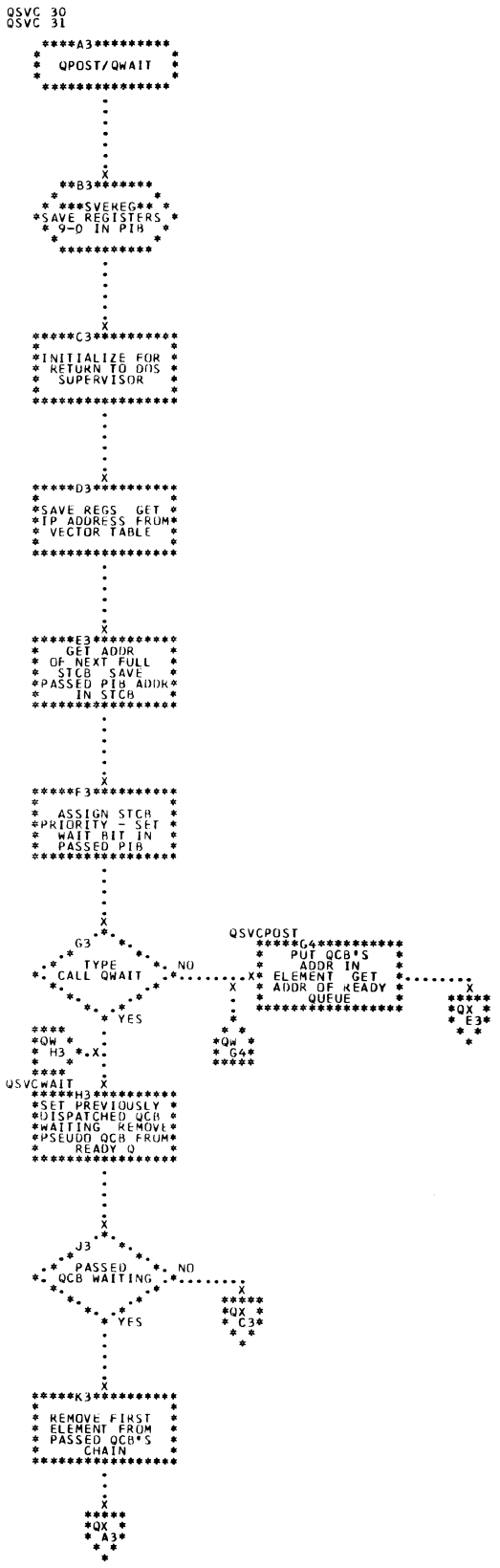


Chart QX. QTAM SVC/Subtask Control Routine (Part 2 of 3)

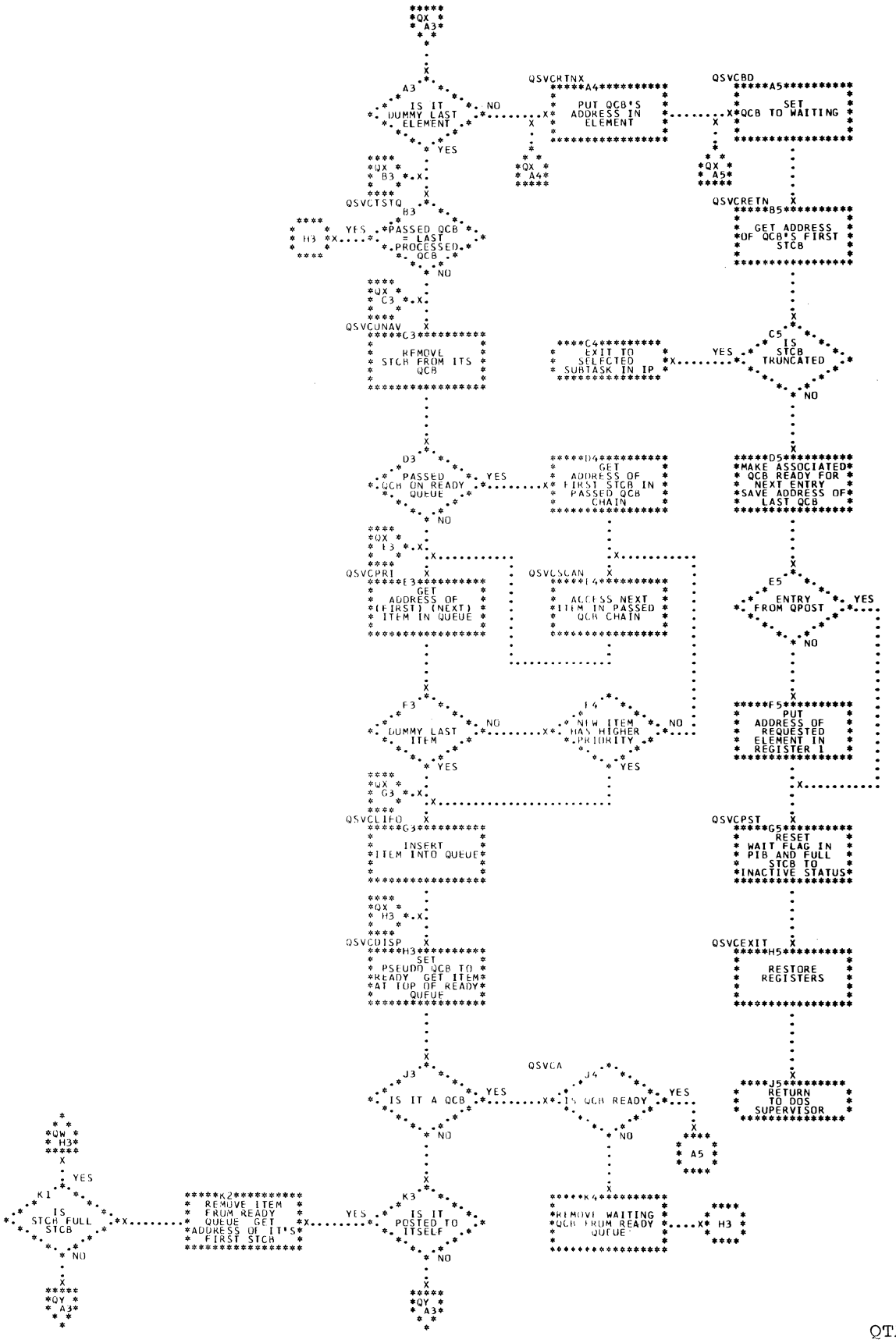


Chart QY. QTAM SVC/Subtask Control Routine (Part 3 of 3)

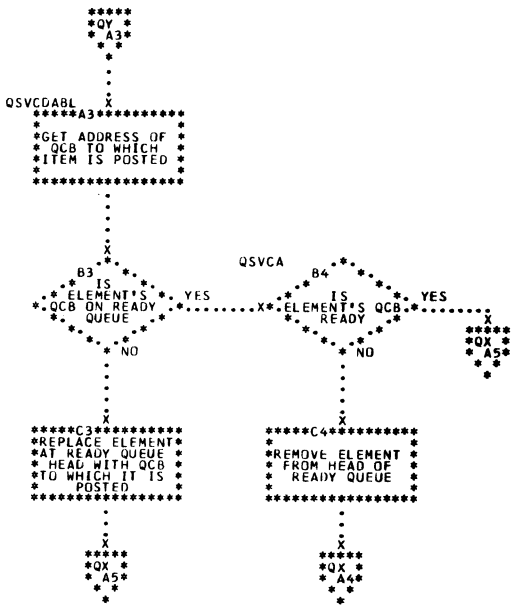


Chart RD. Retrieve DASD Routine

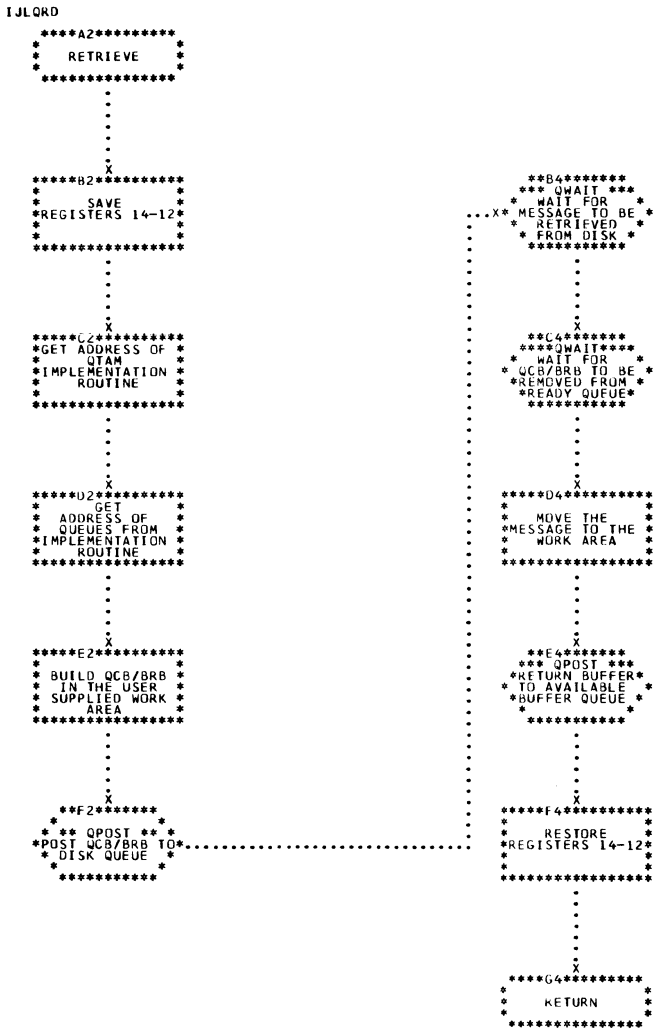


Chart RG. Route Message Routine

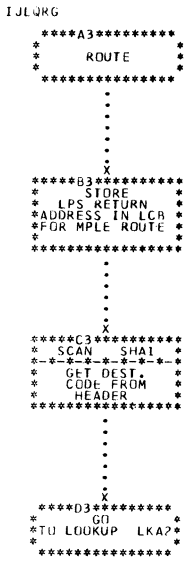


Chart RM. Release Message Routine

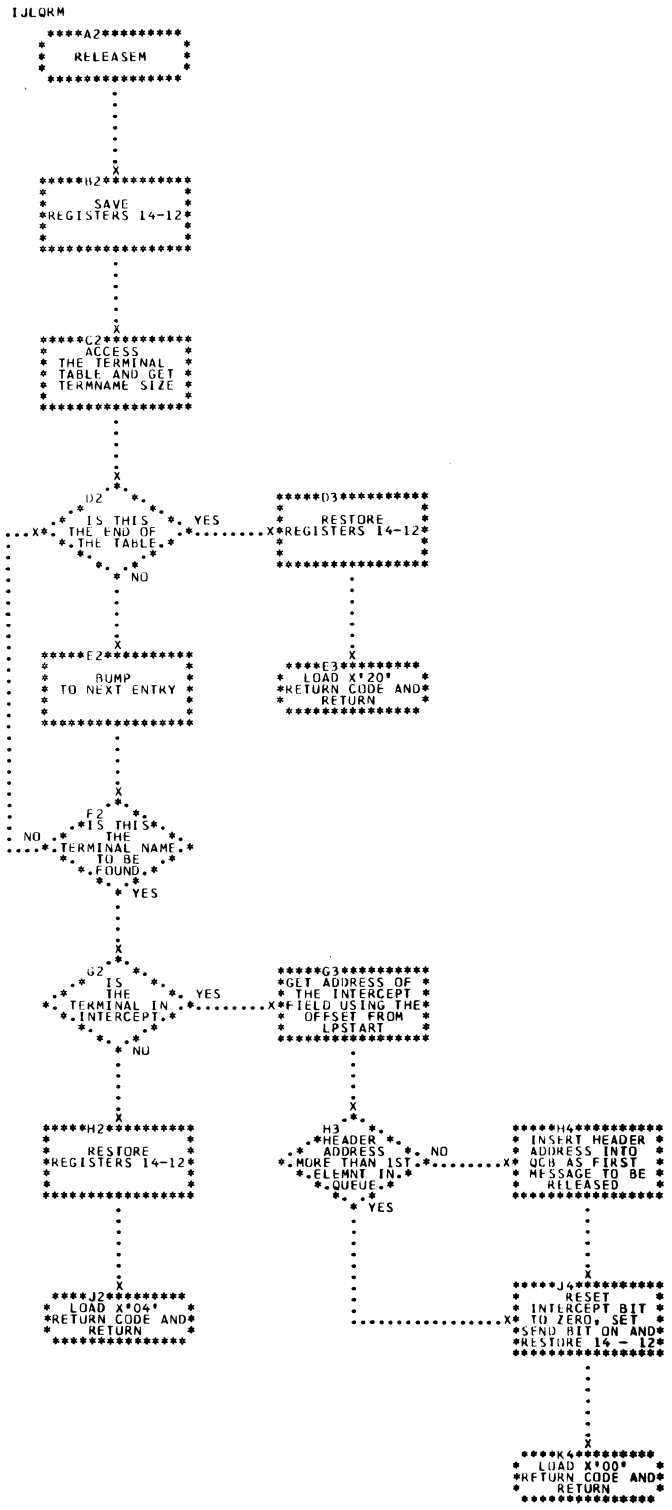


Chart RR. Reroute Message Routine

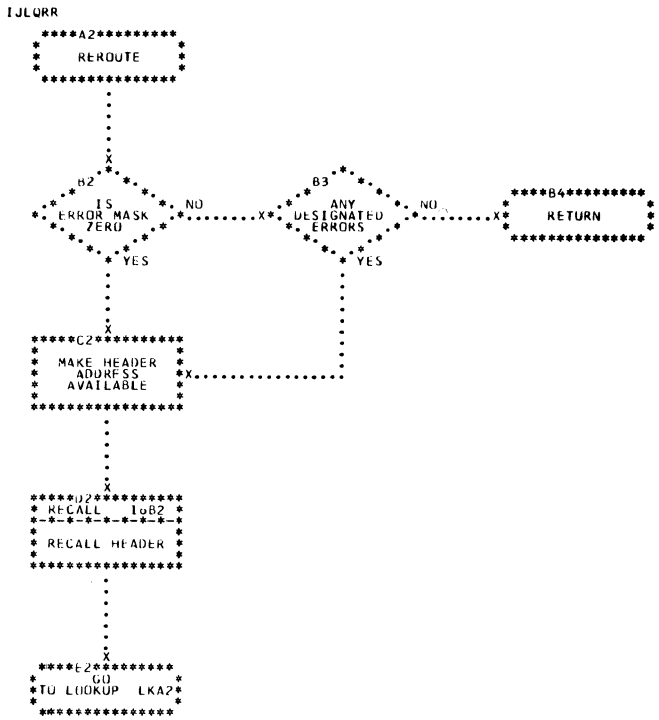


Chart RS. Retrieve by Sequence Number Routine

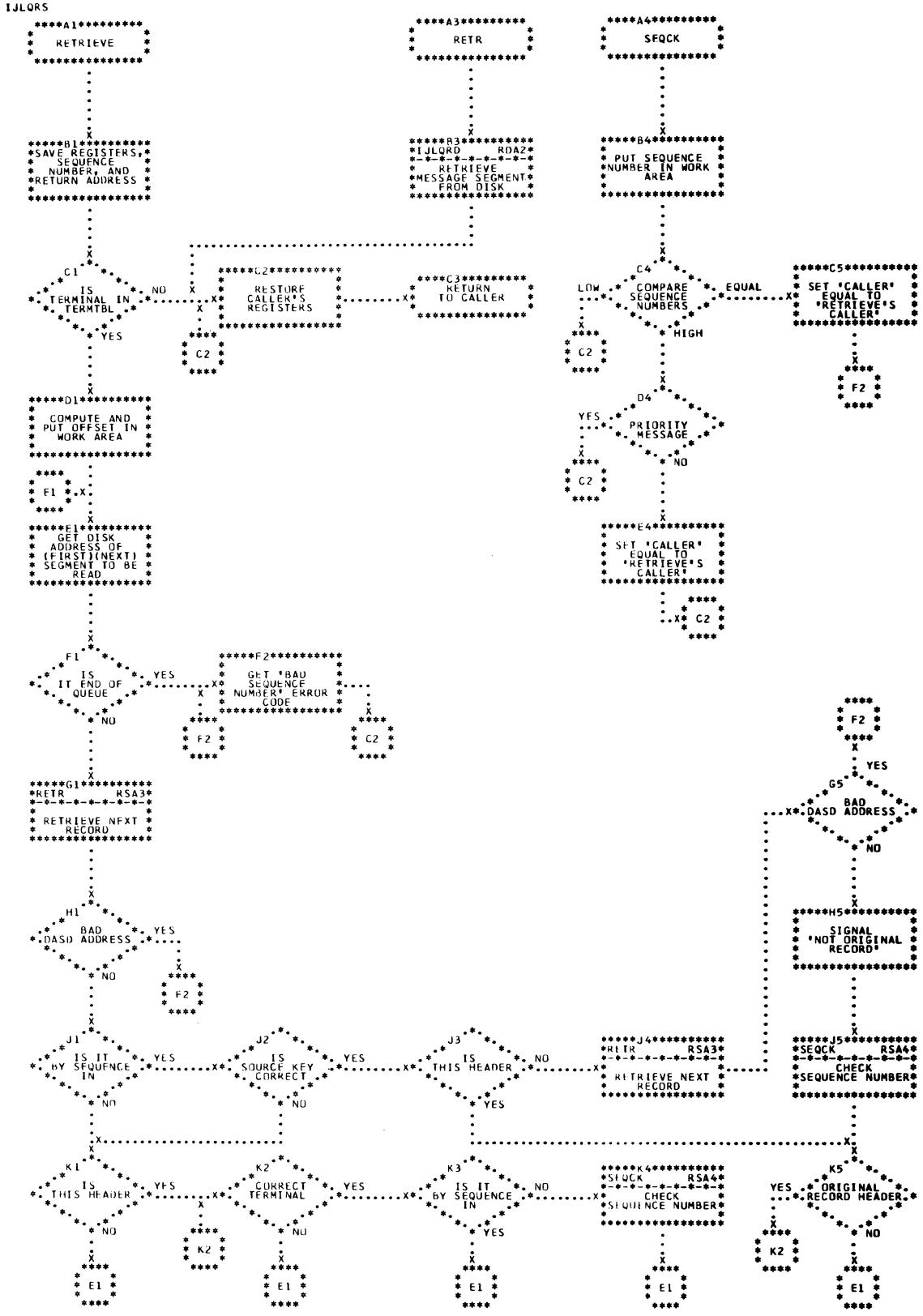


Chart RW. QTAM PIOCS--Activate Routine

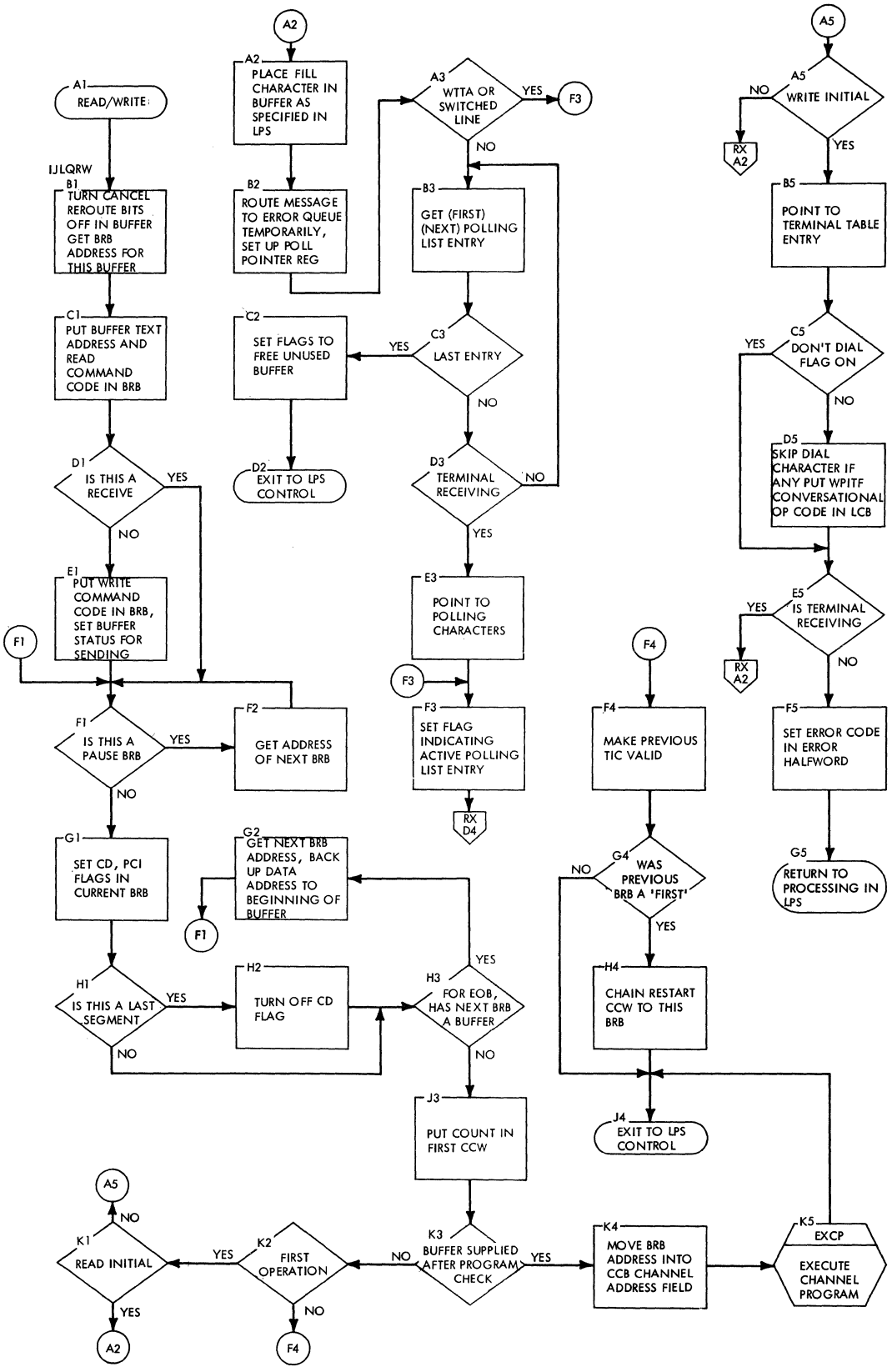


Chart RX. QTAM PIOCS--Channel Program Generator (Part 1 of 2)

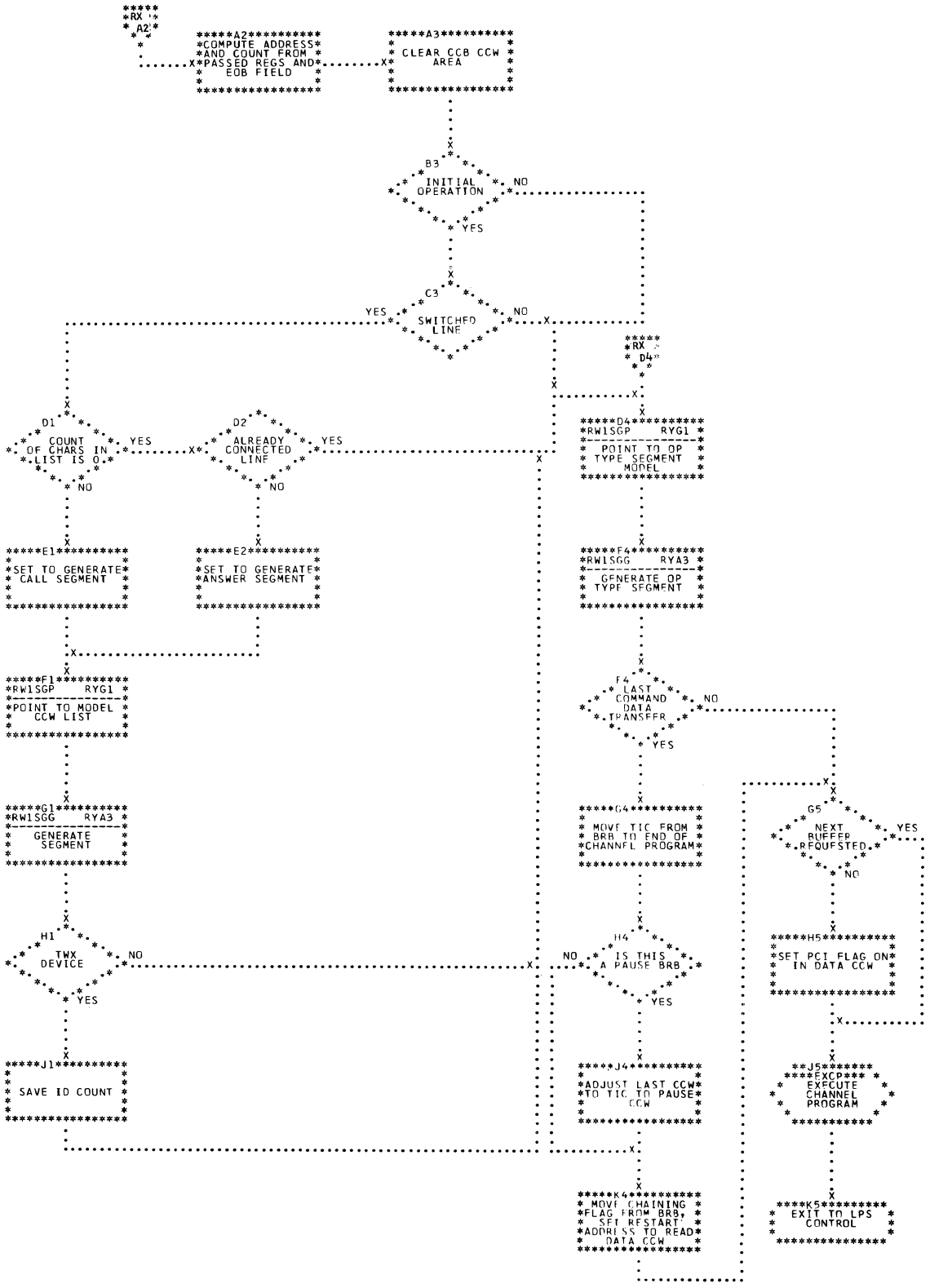


Chart RY. QTAM PIOCS--Channel Program Generator (Part 2 of 2)

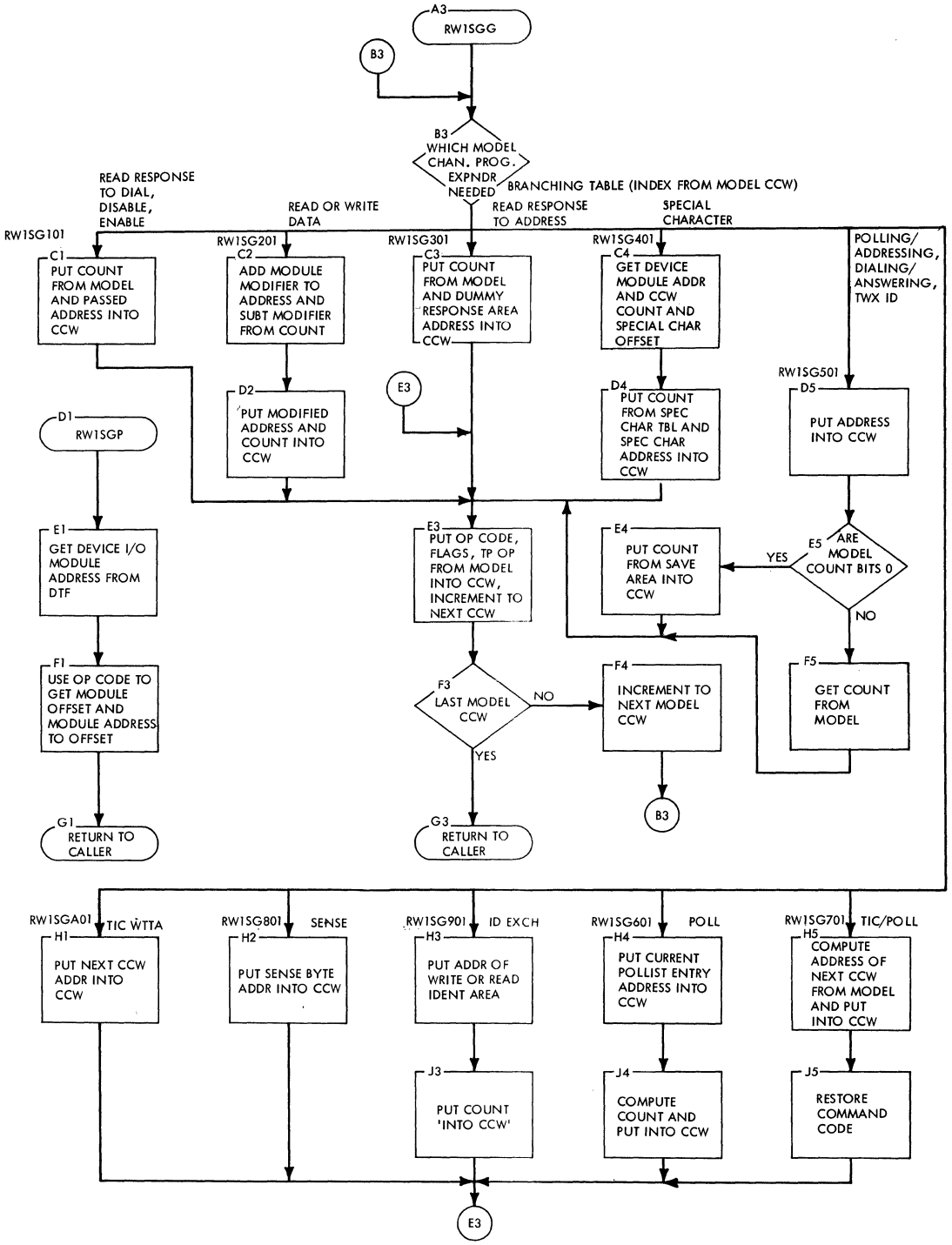


Chart SH. Scan Header Routine

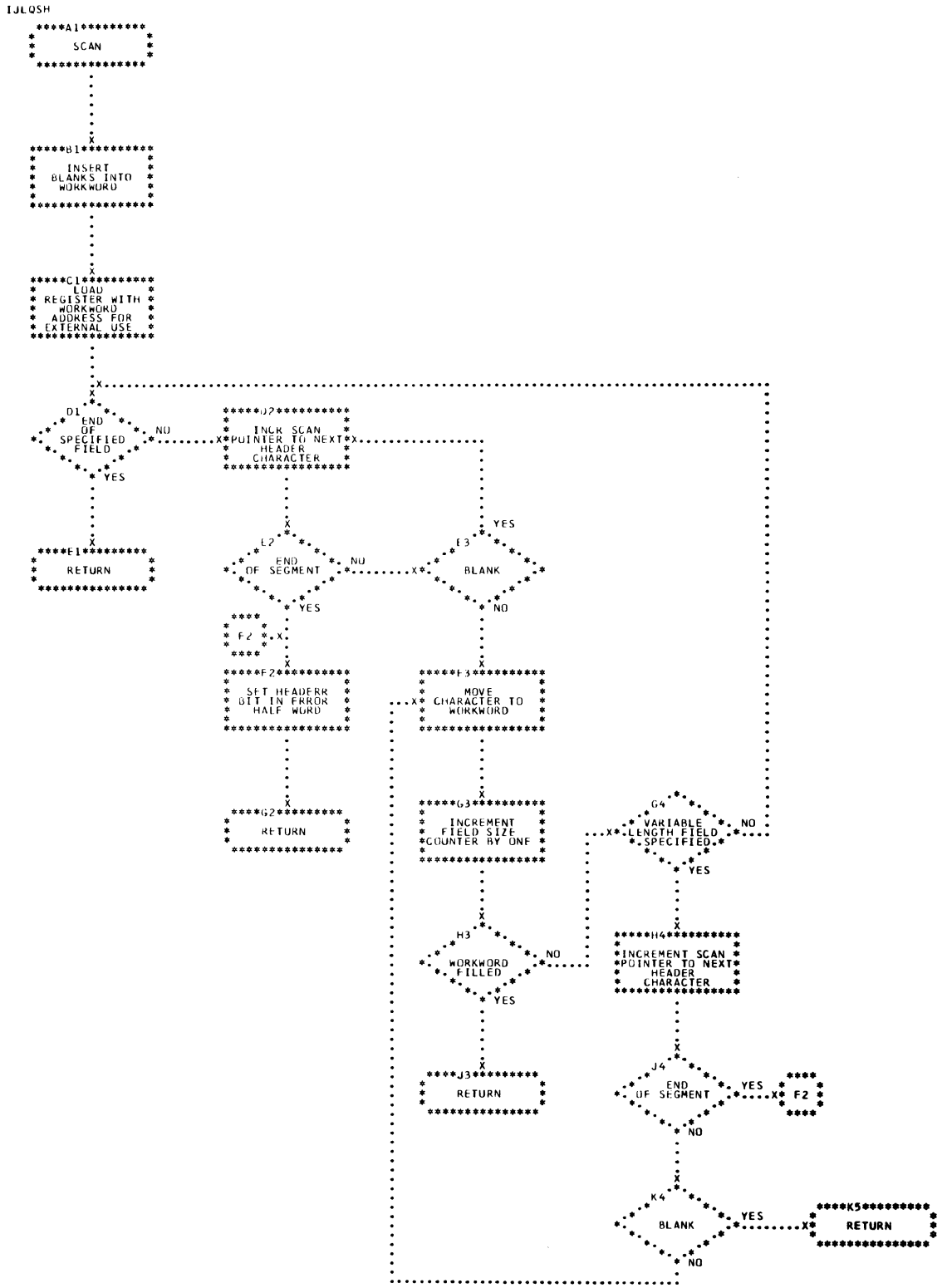


Chart SI. Sequence Number-In Routine

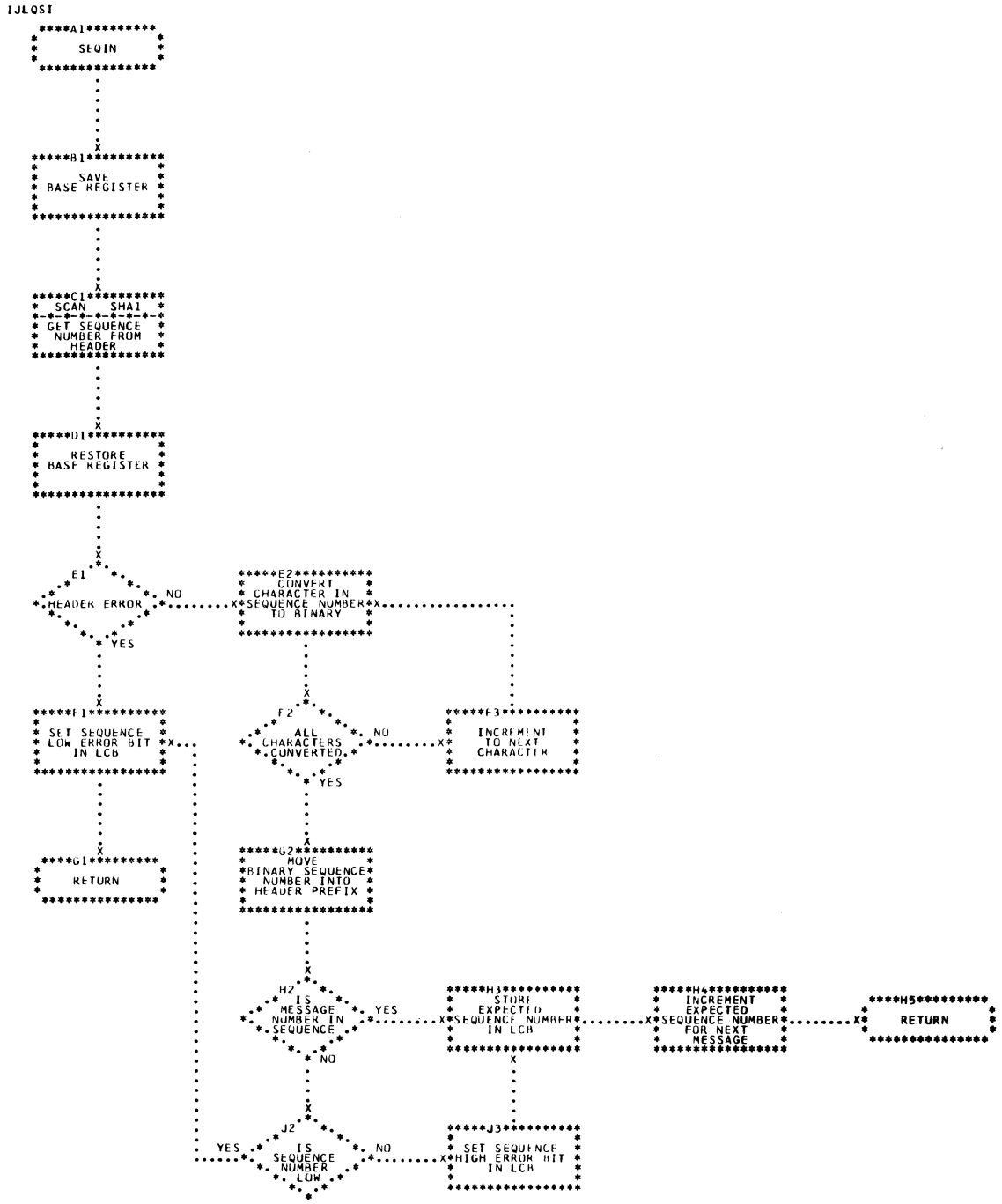


Chart SK. Skip Character Set Routine

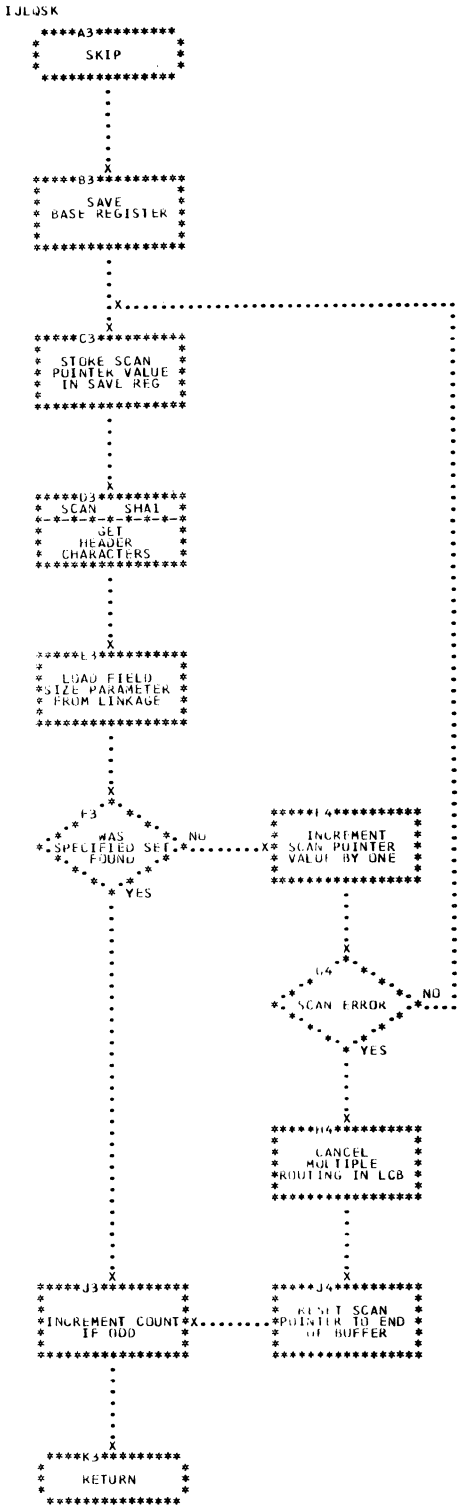


Chart SO. Sequence-Out Routine

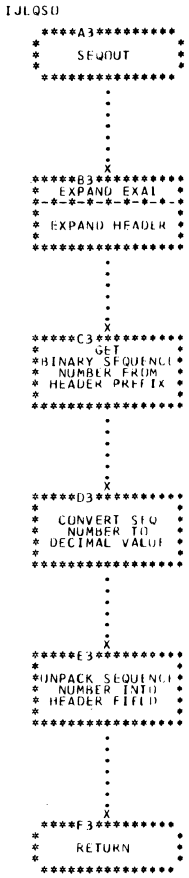


Chart SR. Source Terminal Verification Routine

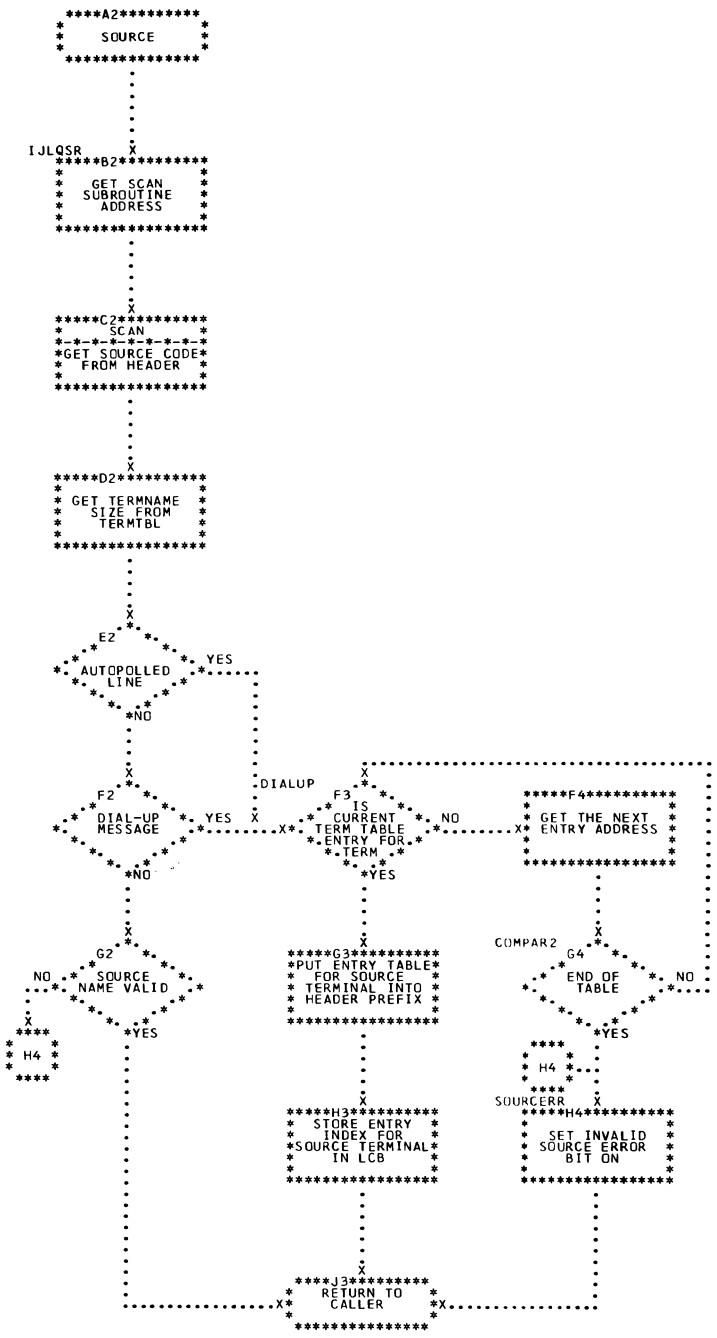


Chart SS. Start/Stop Audio Line Routine

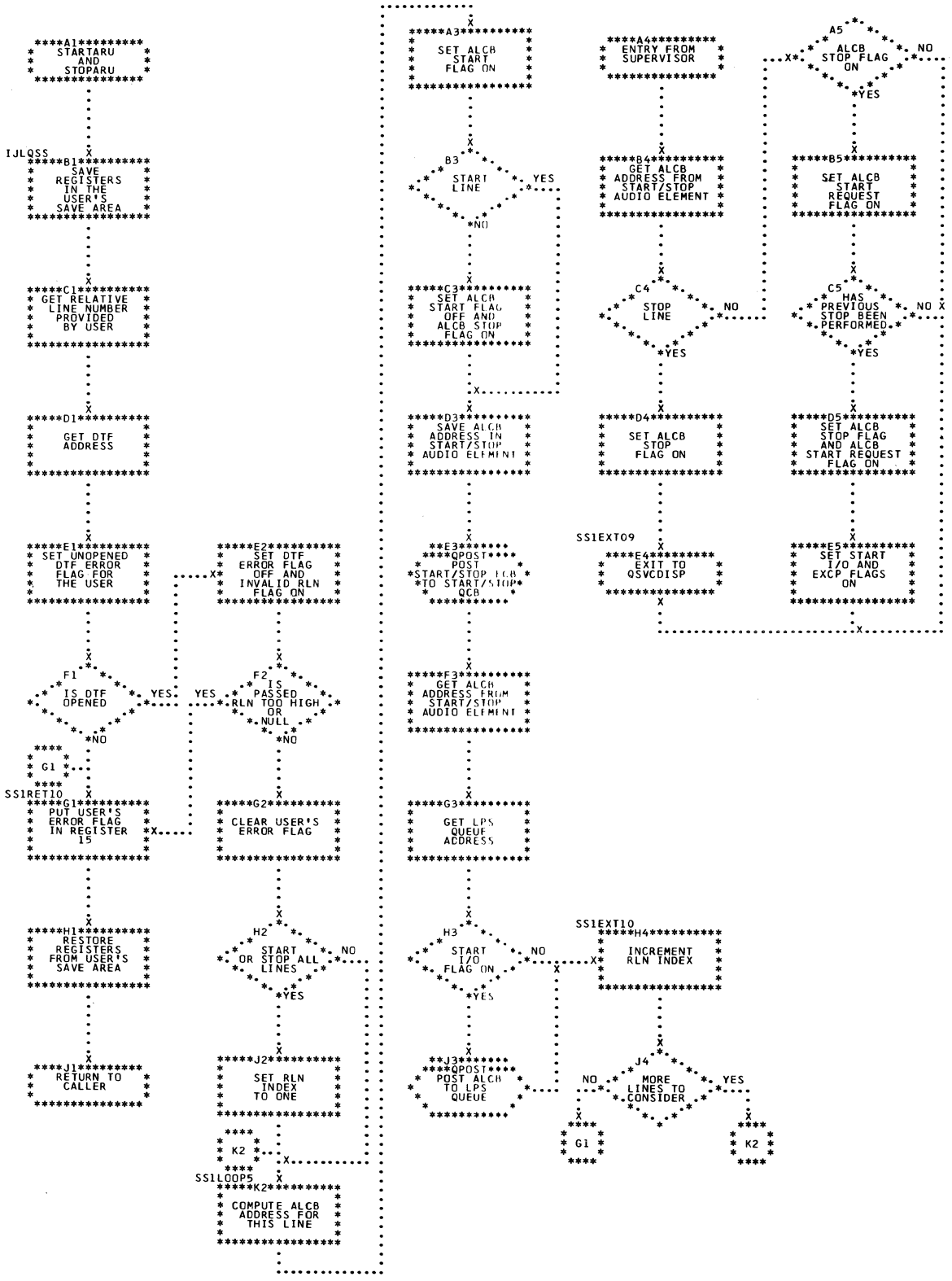


Chart ST. Skip-on-Count Routine

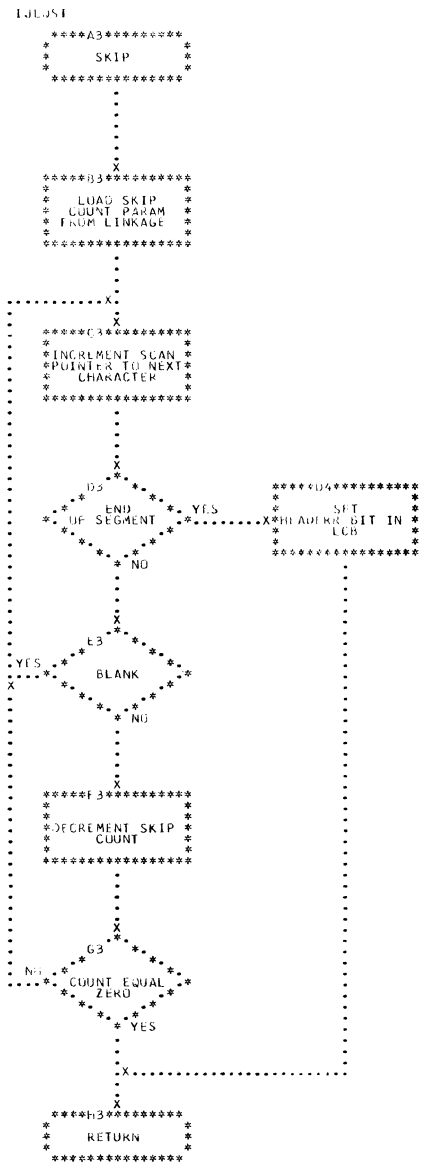


Chart TM. Terminal Test Header Analysis Routine

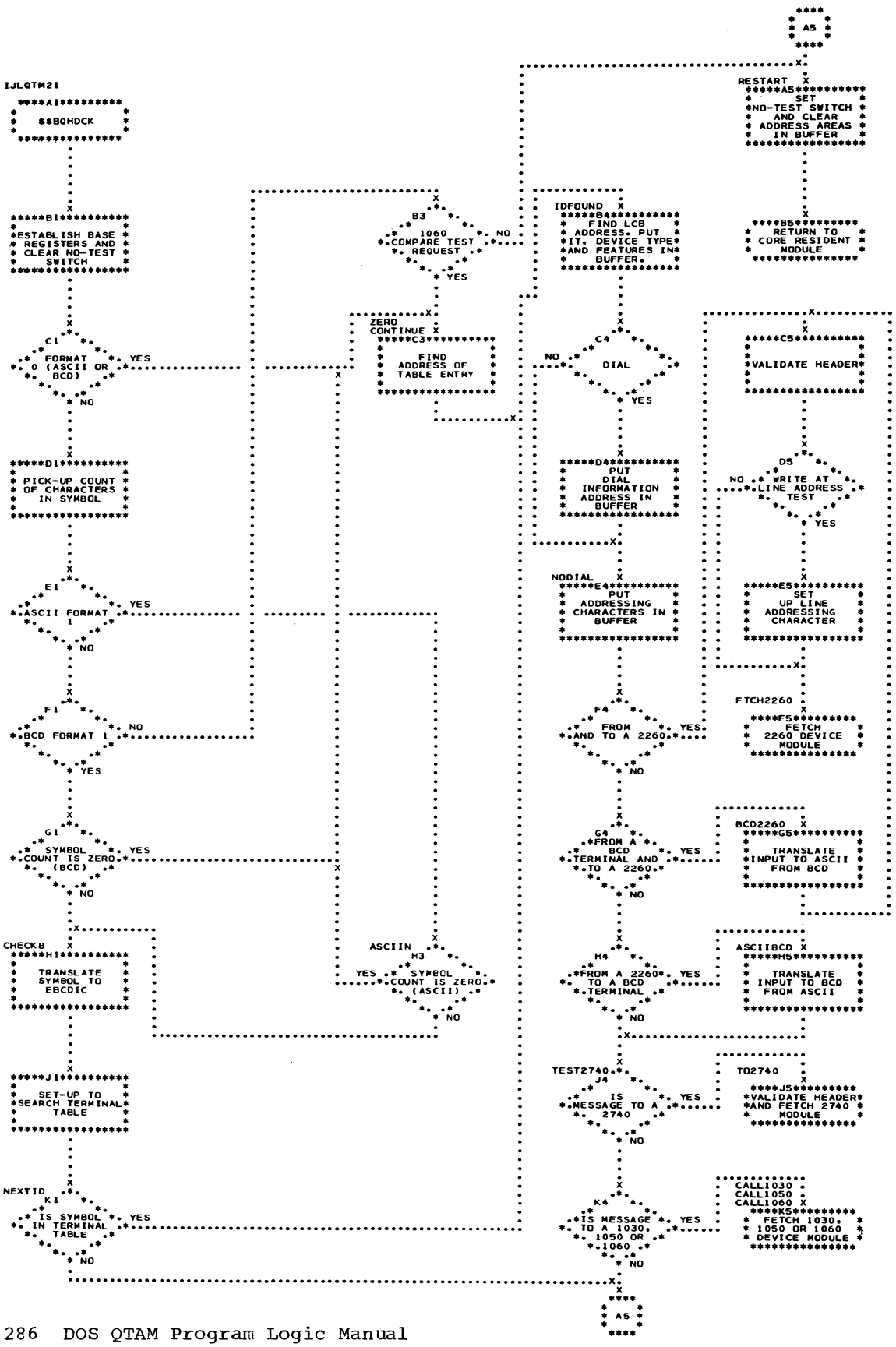


Chart TN. Terminal Test Subtasks

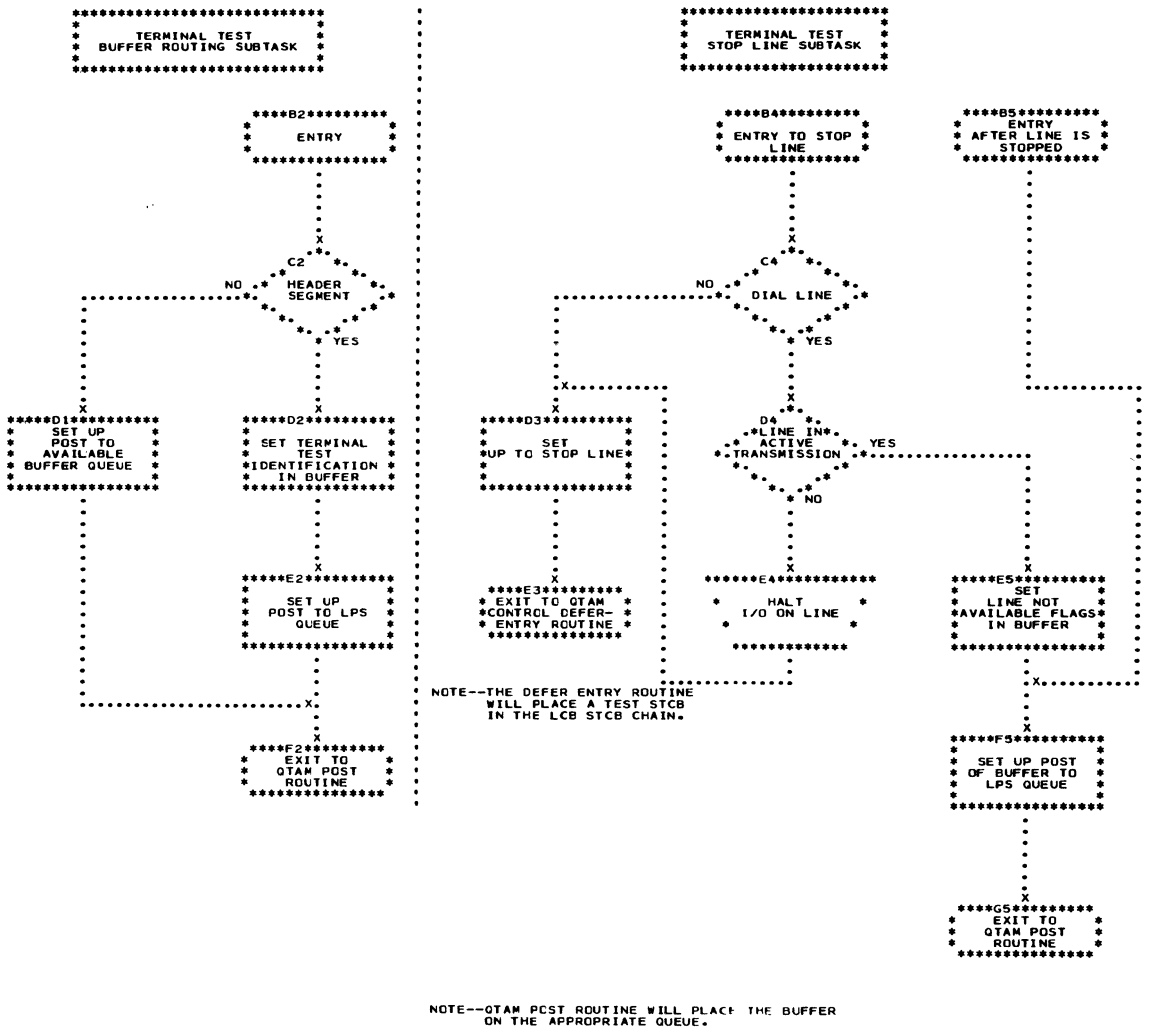


Chart TR. Translate Routine

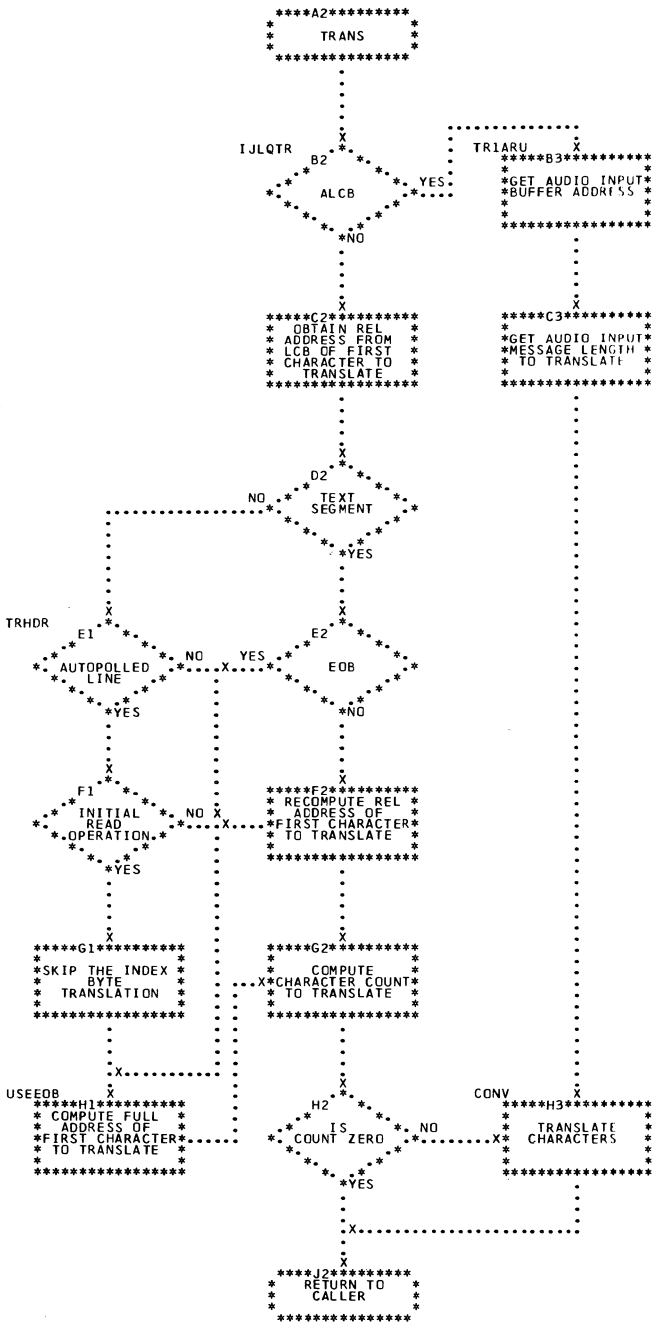


Chart TS. Time Stamp Routine

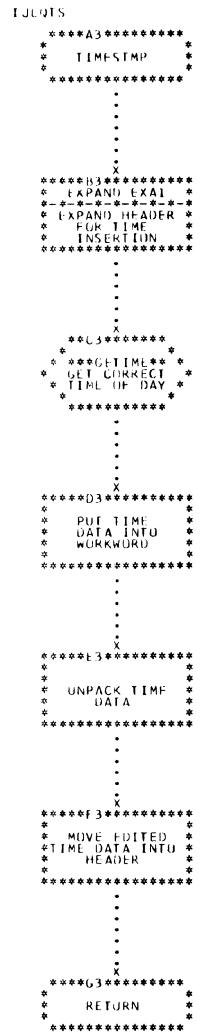
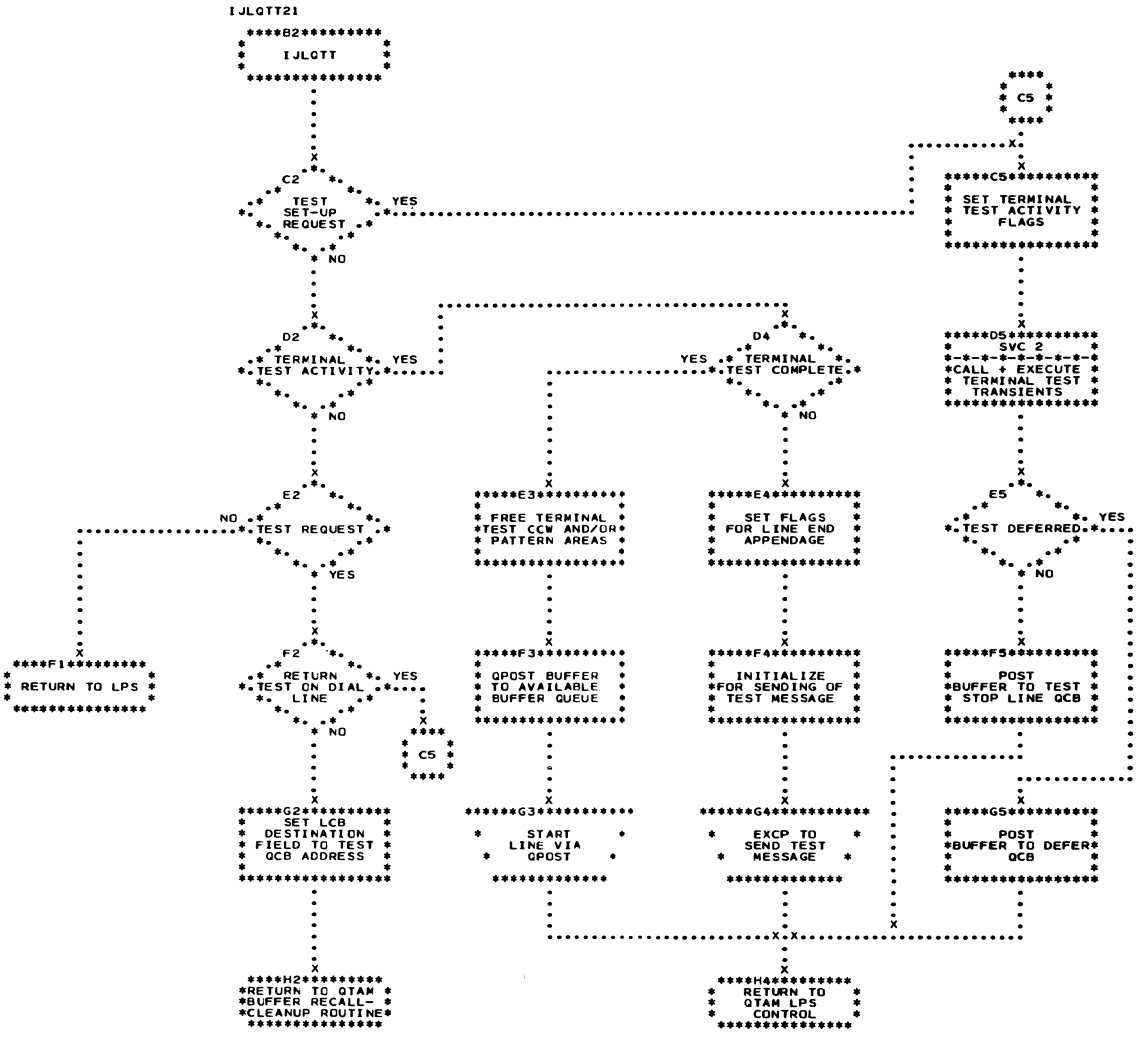


Chart TT. Terminal Test Recognition Routine



NOTE--THE QTAM BUFFER RECALL-CLEANUP ROUTINE POSTS THE BUFFER TO THE TEST BUFFER ROUTING QCB.

Chart T1. Terminal Test Module for IBM 1030

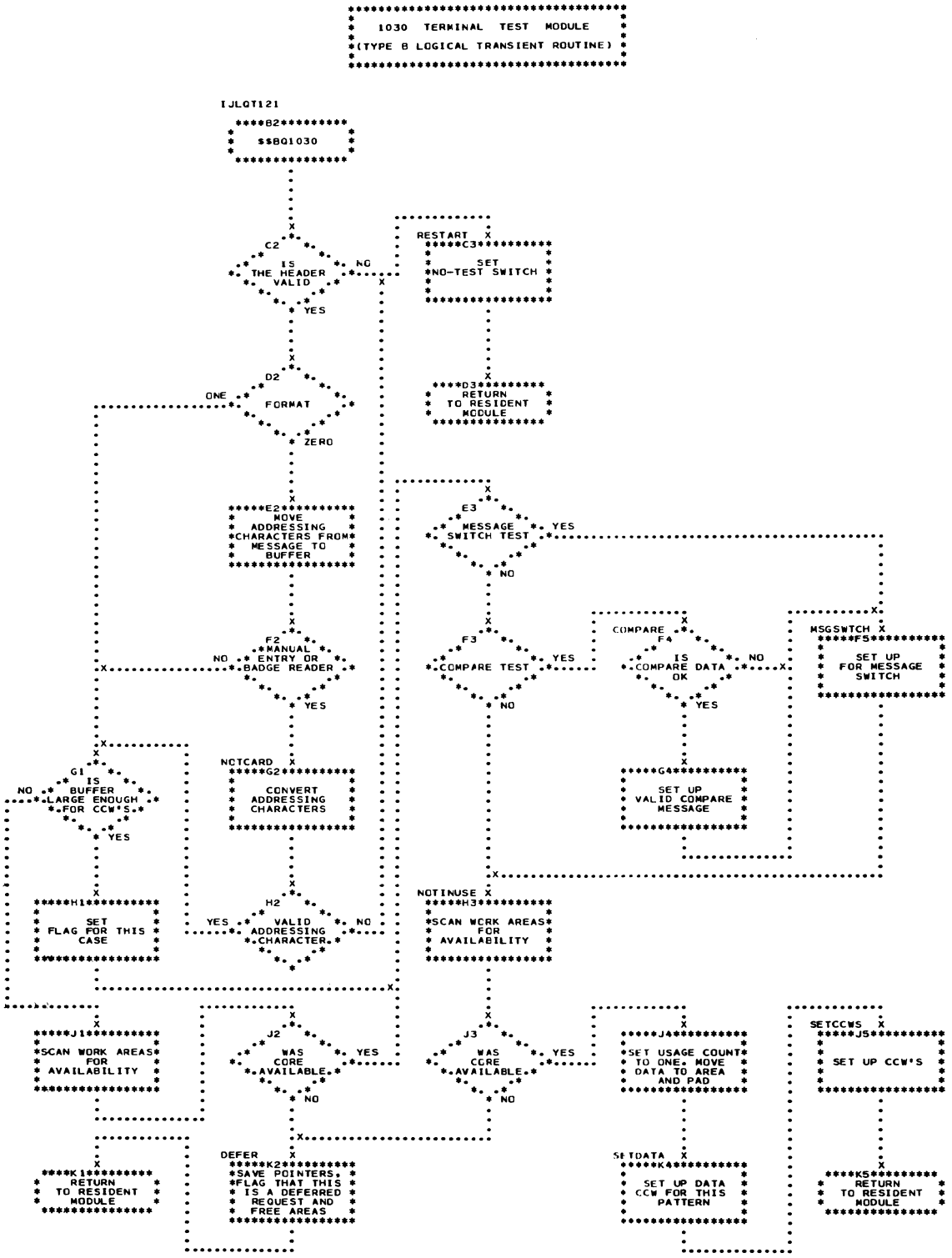


Chart T2. Terminal Test Module for IBM 1050

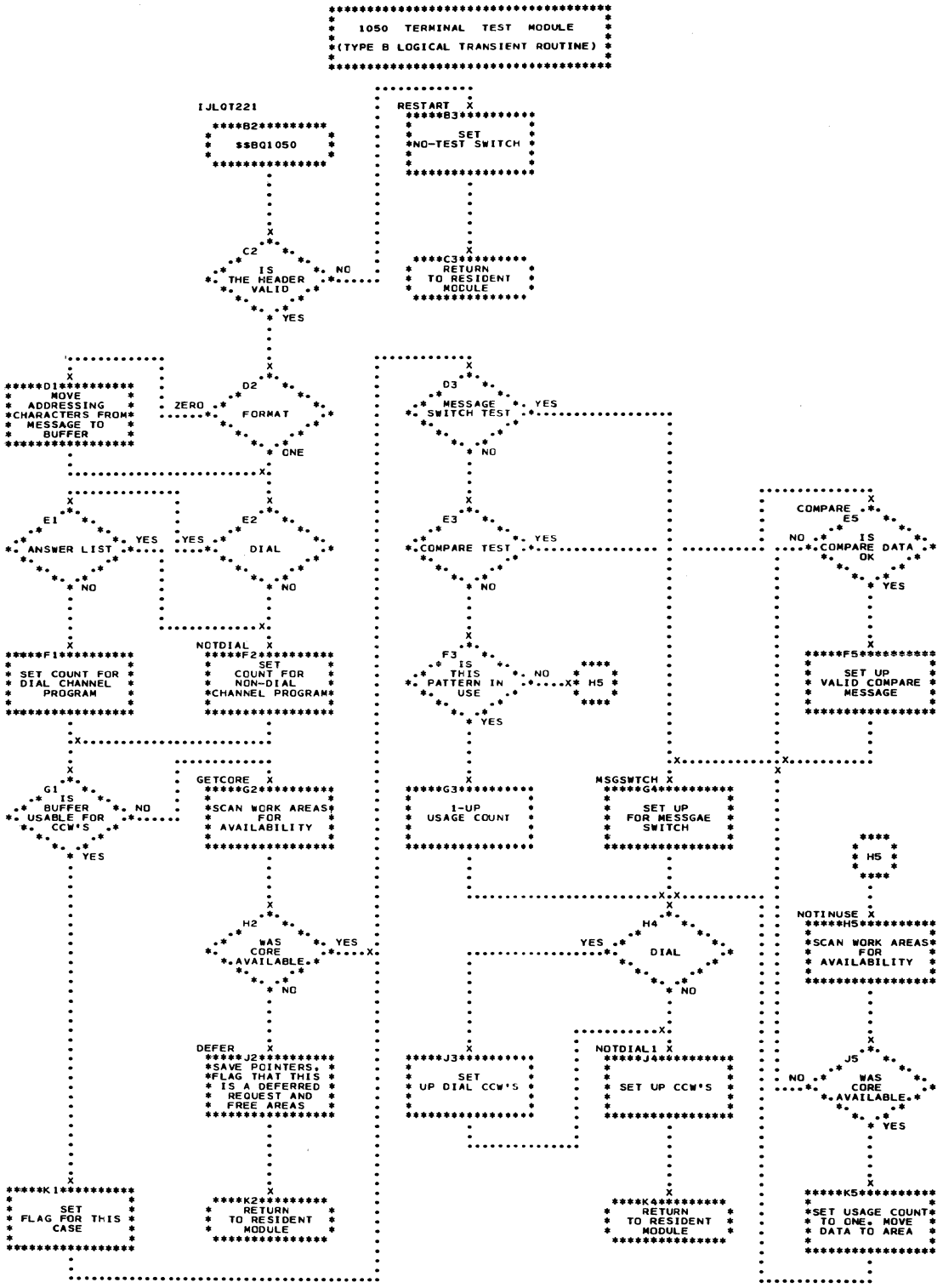


Chart T3. Terminal Test Module for IBM 1060

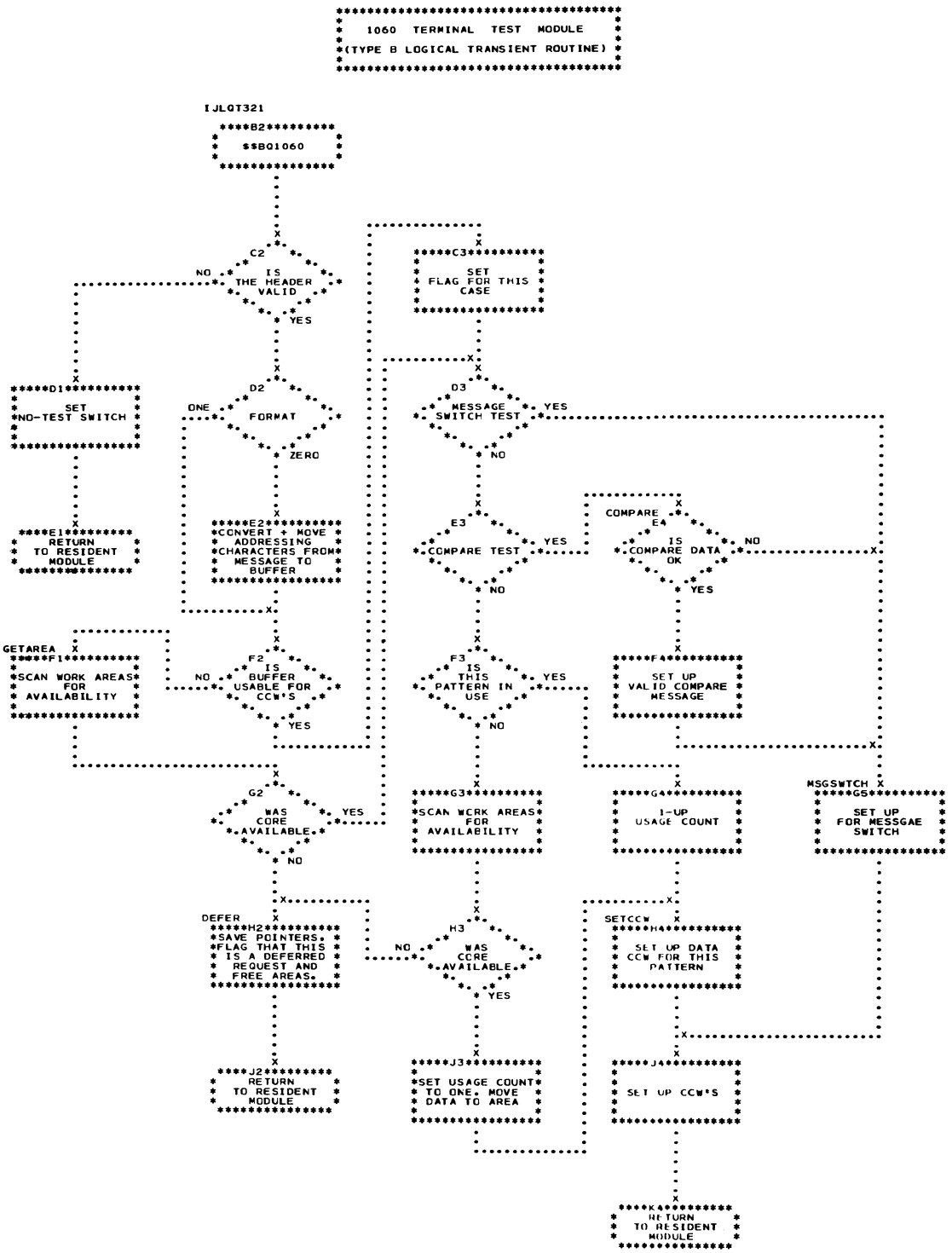


Chart T4. Terminal Test Module for IBM 2848/2260

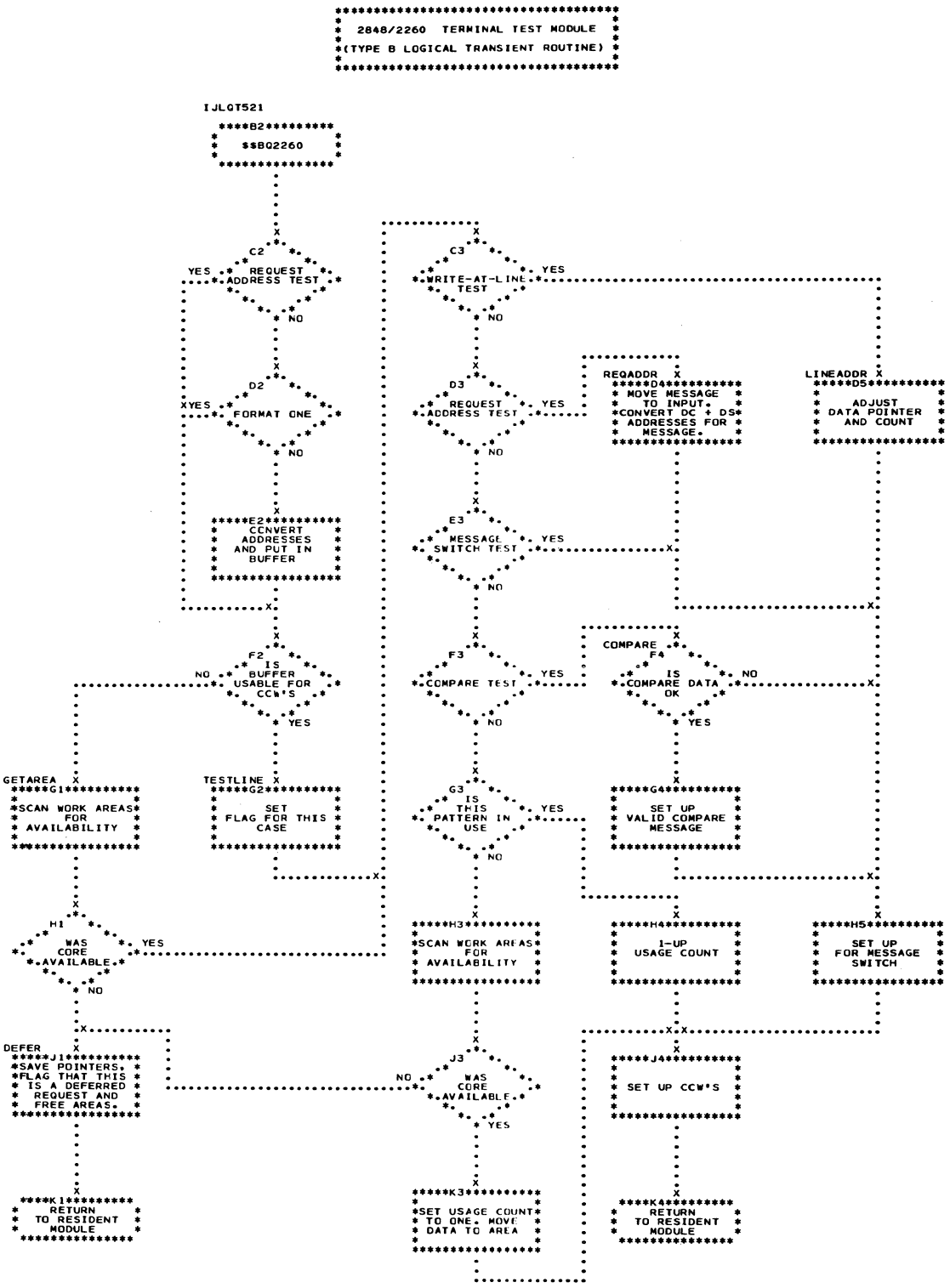


Chart T5. Terminal Test Module for IBM 2740

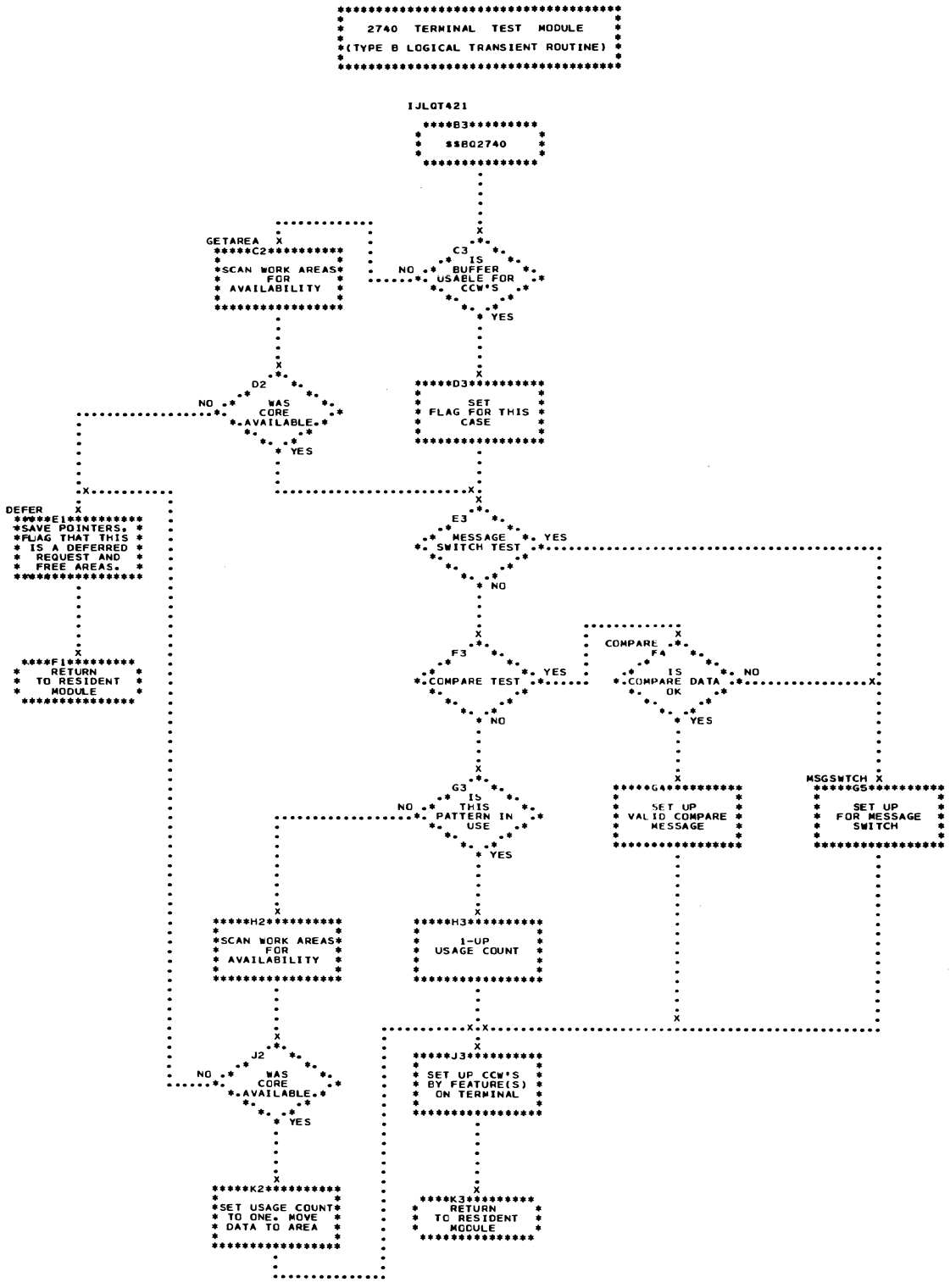


Chart WA. Audio ERP Message Writer Routine

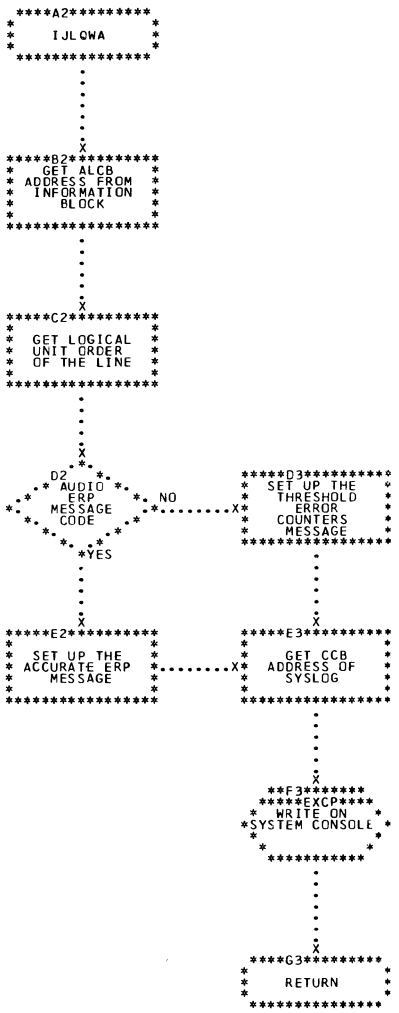


Chart W1. Message Write Routine (Transient) Phase 1

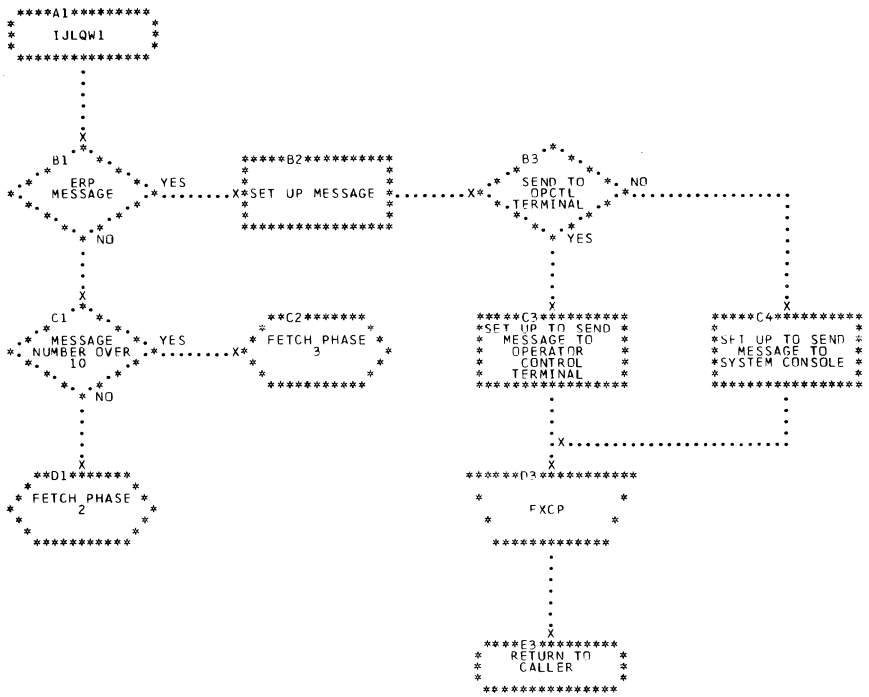


Chart W2. Message Write Routine (Transient) Phases 2 and 3

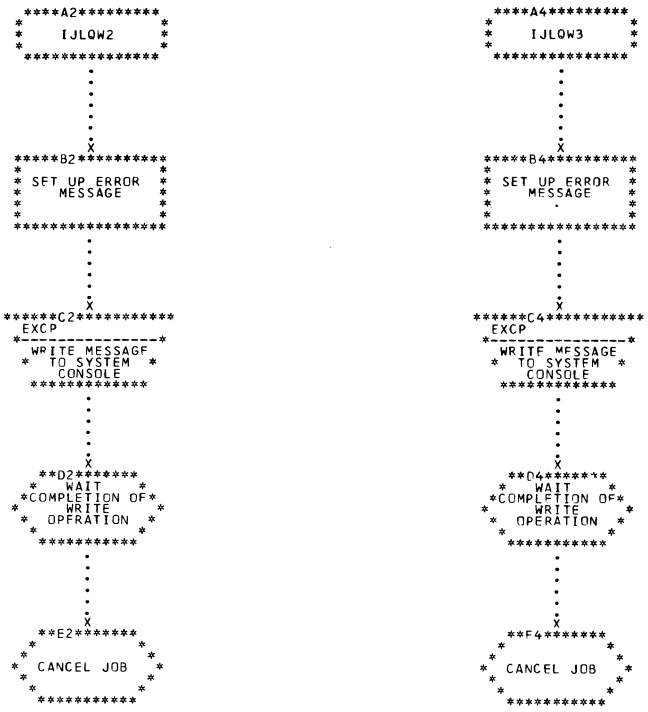


Chart XL. QTAM Cancel Routine (Phase 1)

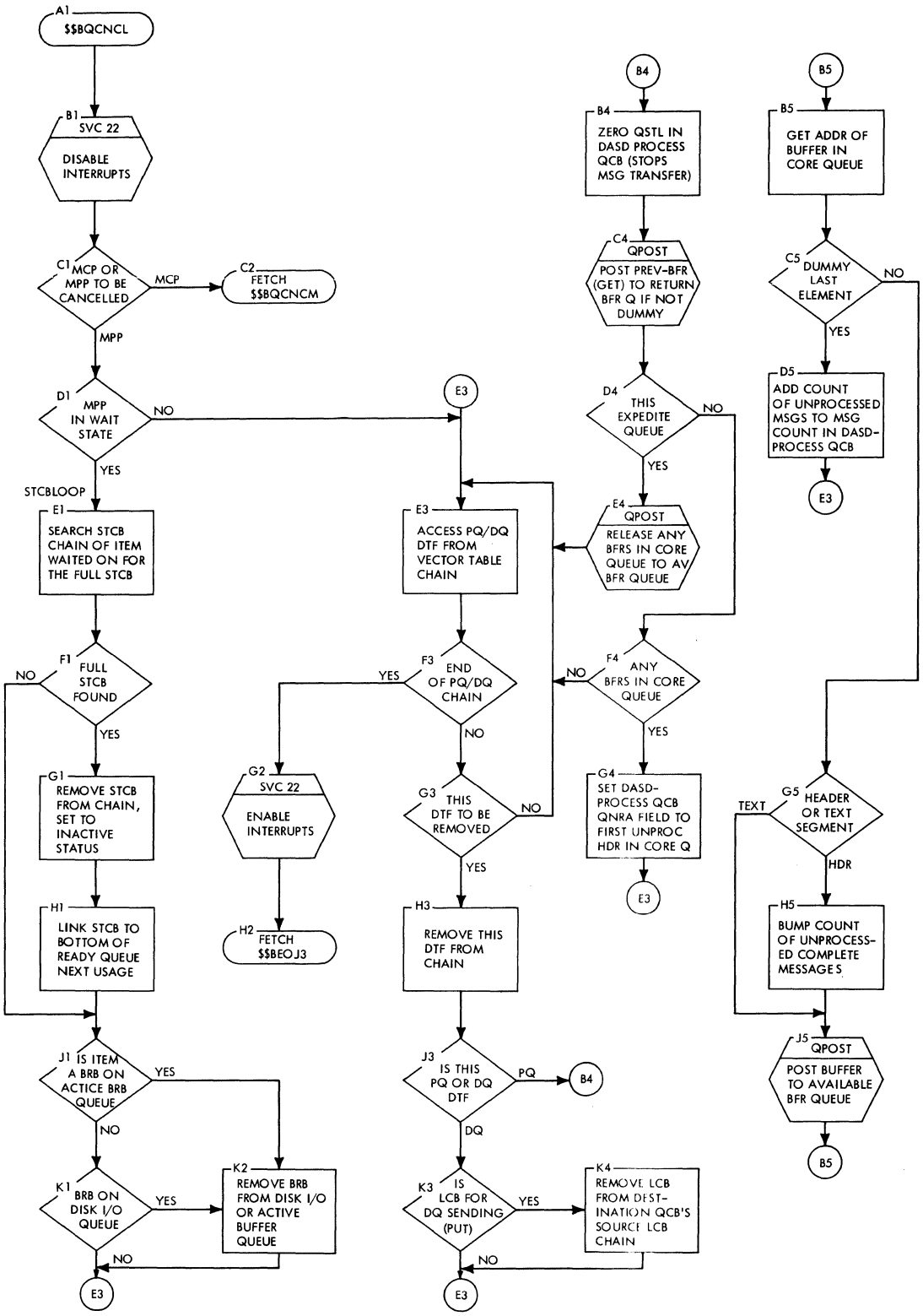


Chart XM. QTAM Cancel Routine (Phase 2)

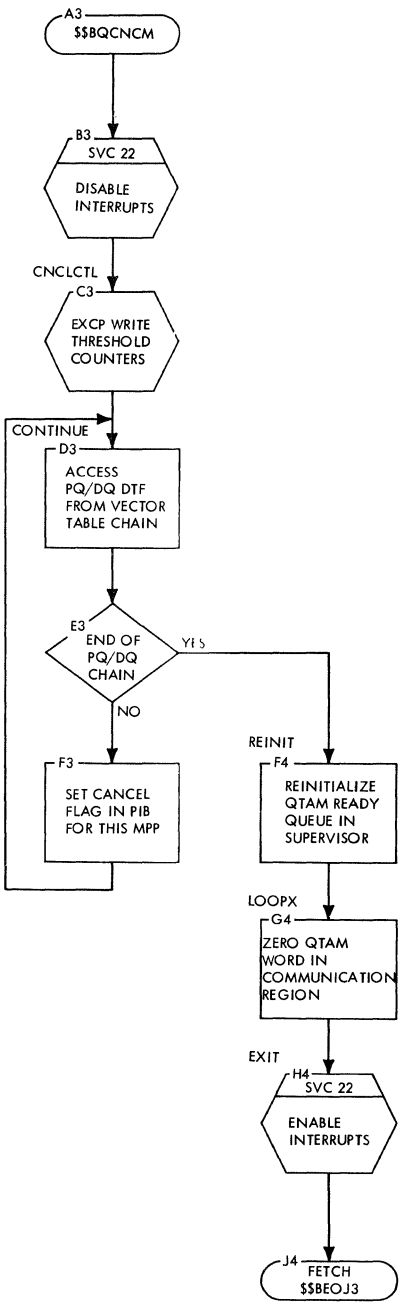


Chart Y1. WTTA Line Appendage (Part 1 of 3)

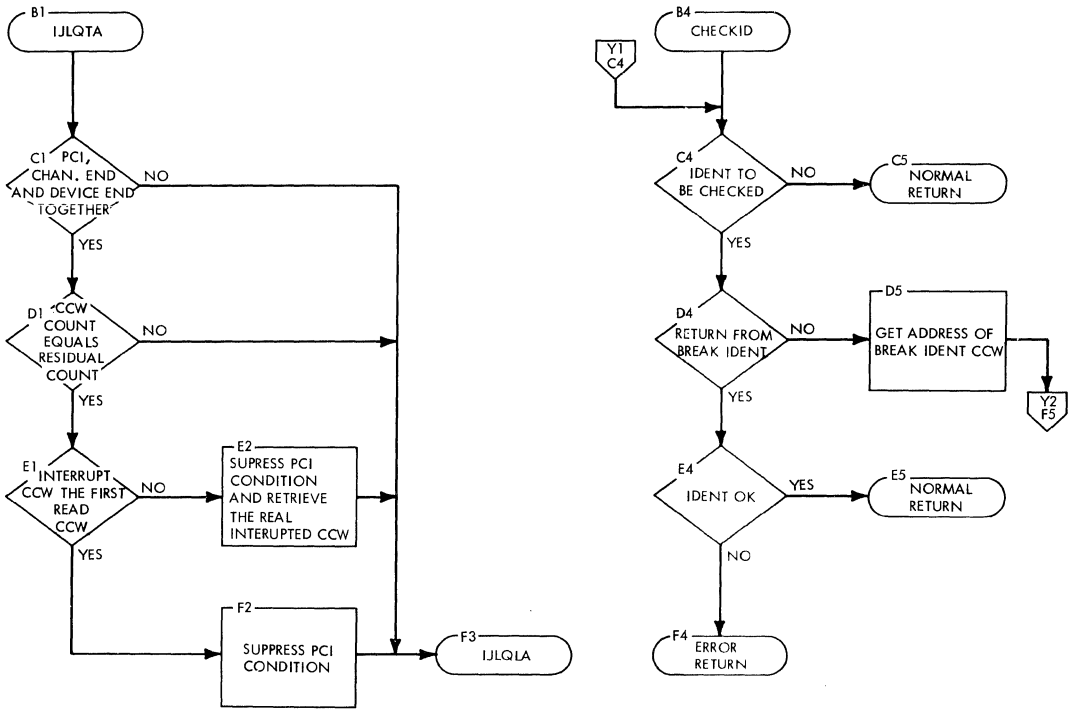


Chart Y2. WTTA Line Appendage (Part 2 of 3)

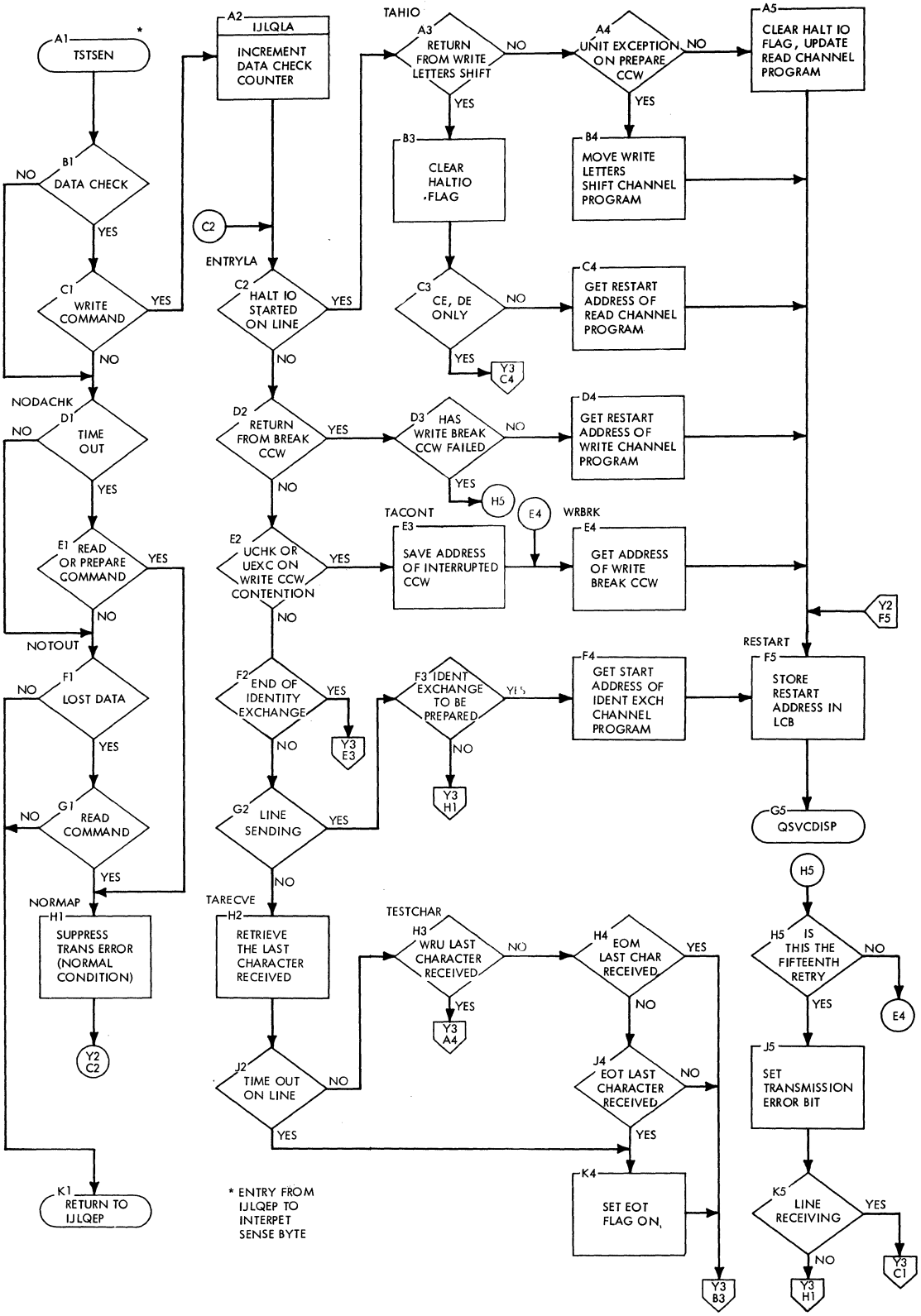


Chart Y3. WTTA Line Appendage (Part 3 of 3)

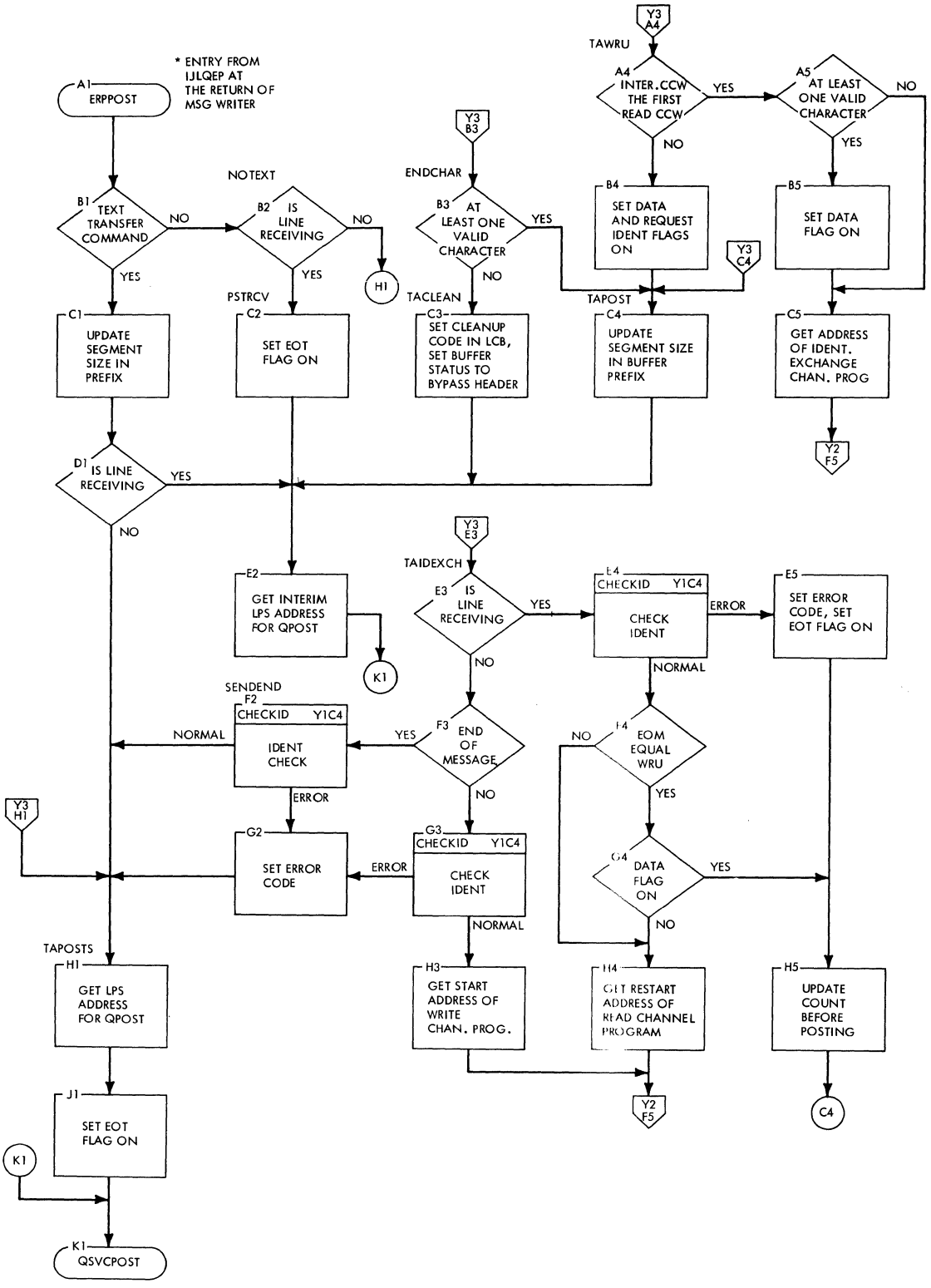


Chart 00. Receive Scheduler Routine

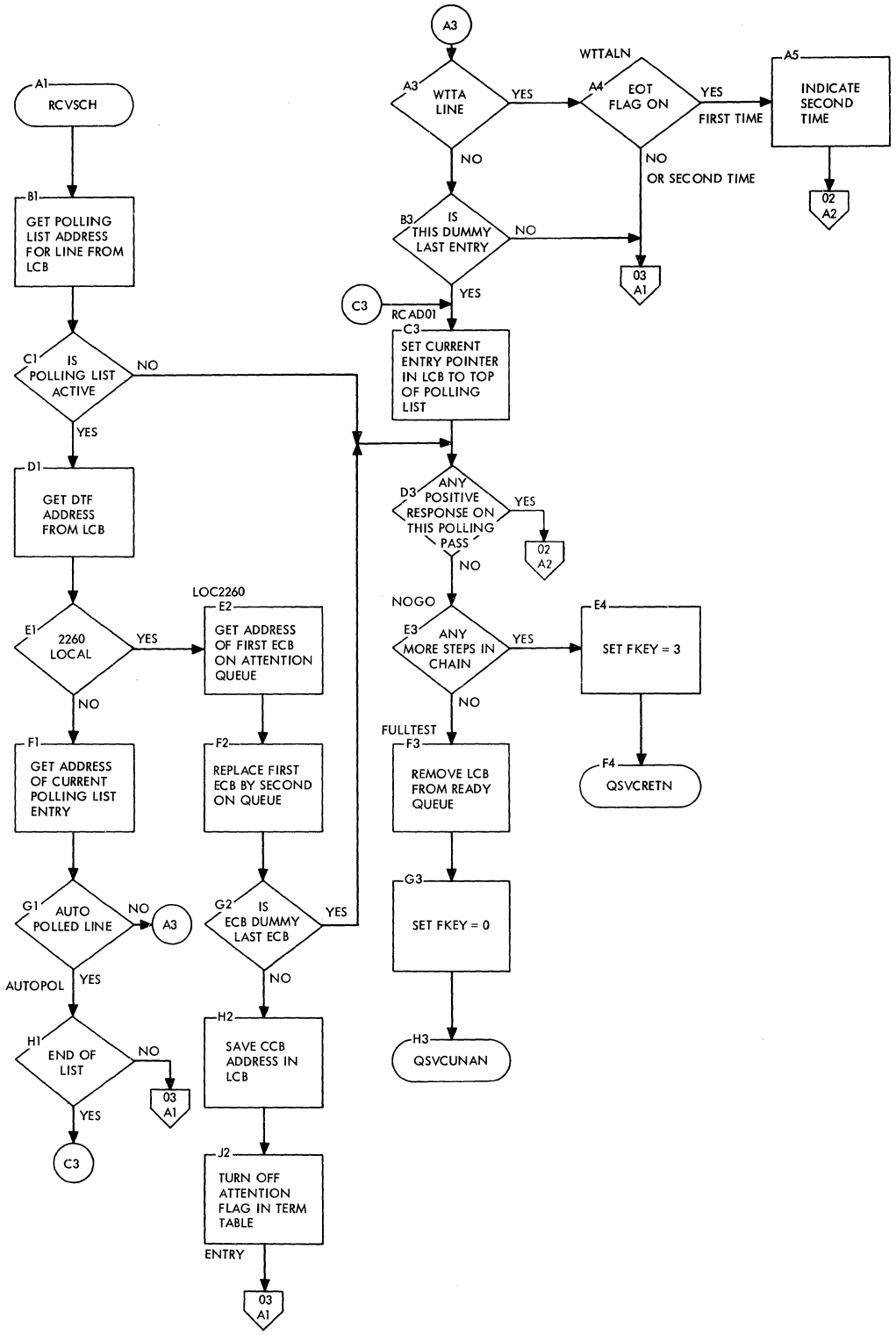


Chart 01. Send Scheduler Routine

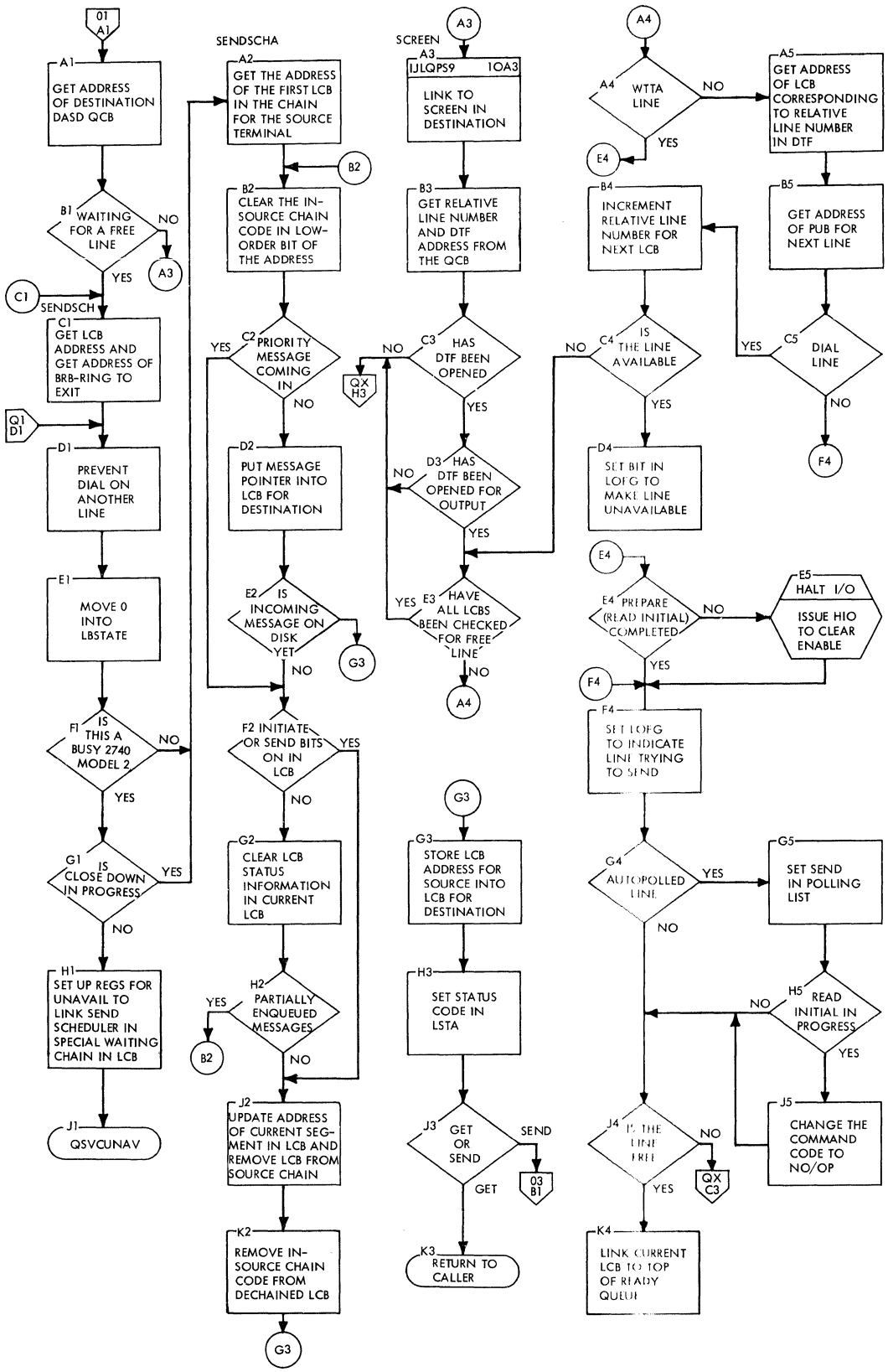


Chart 02. End-of-Poll Time-Delay Routine

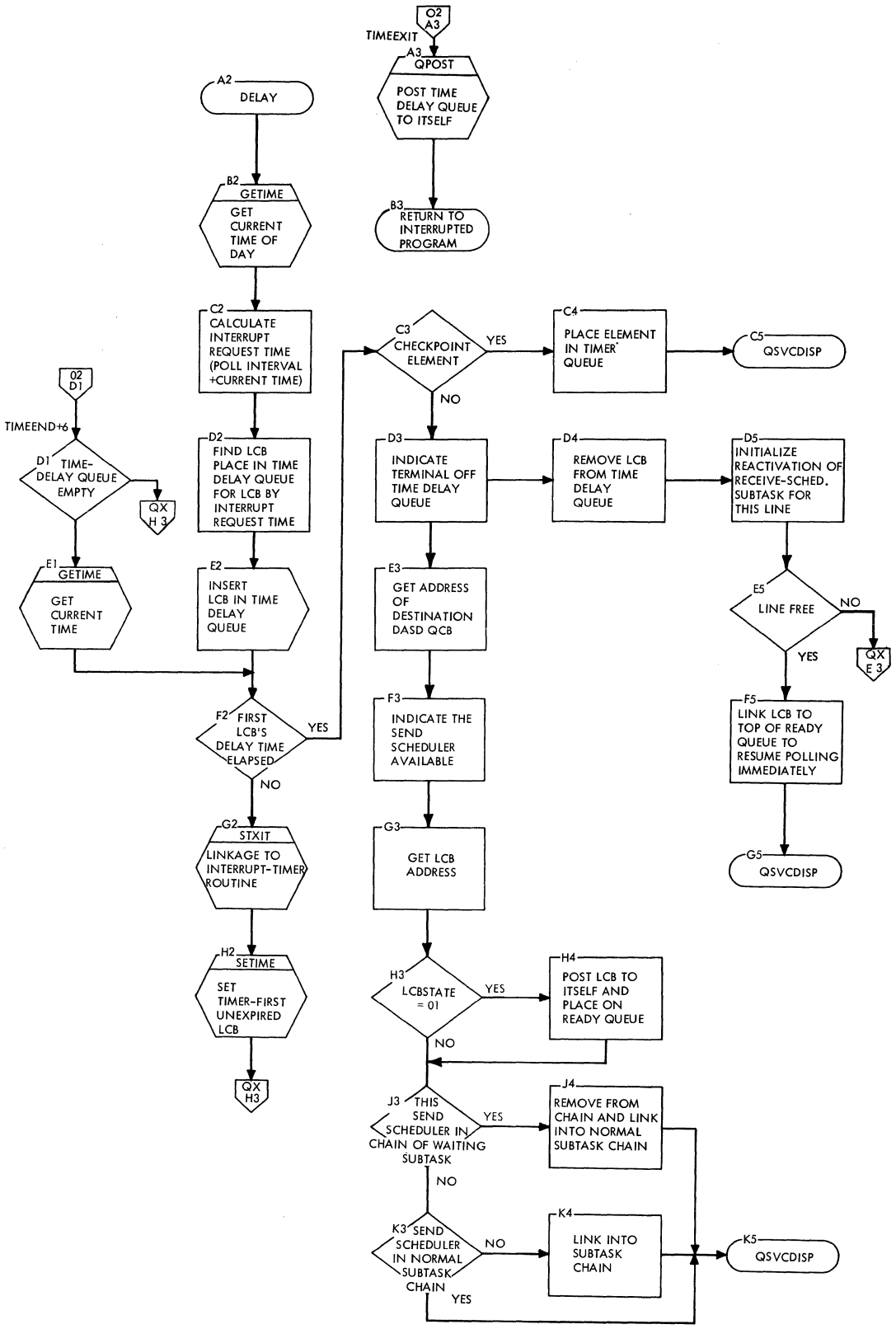


Chart 03. BRB Ring Routine

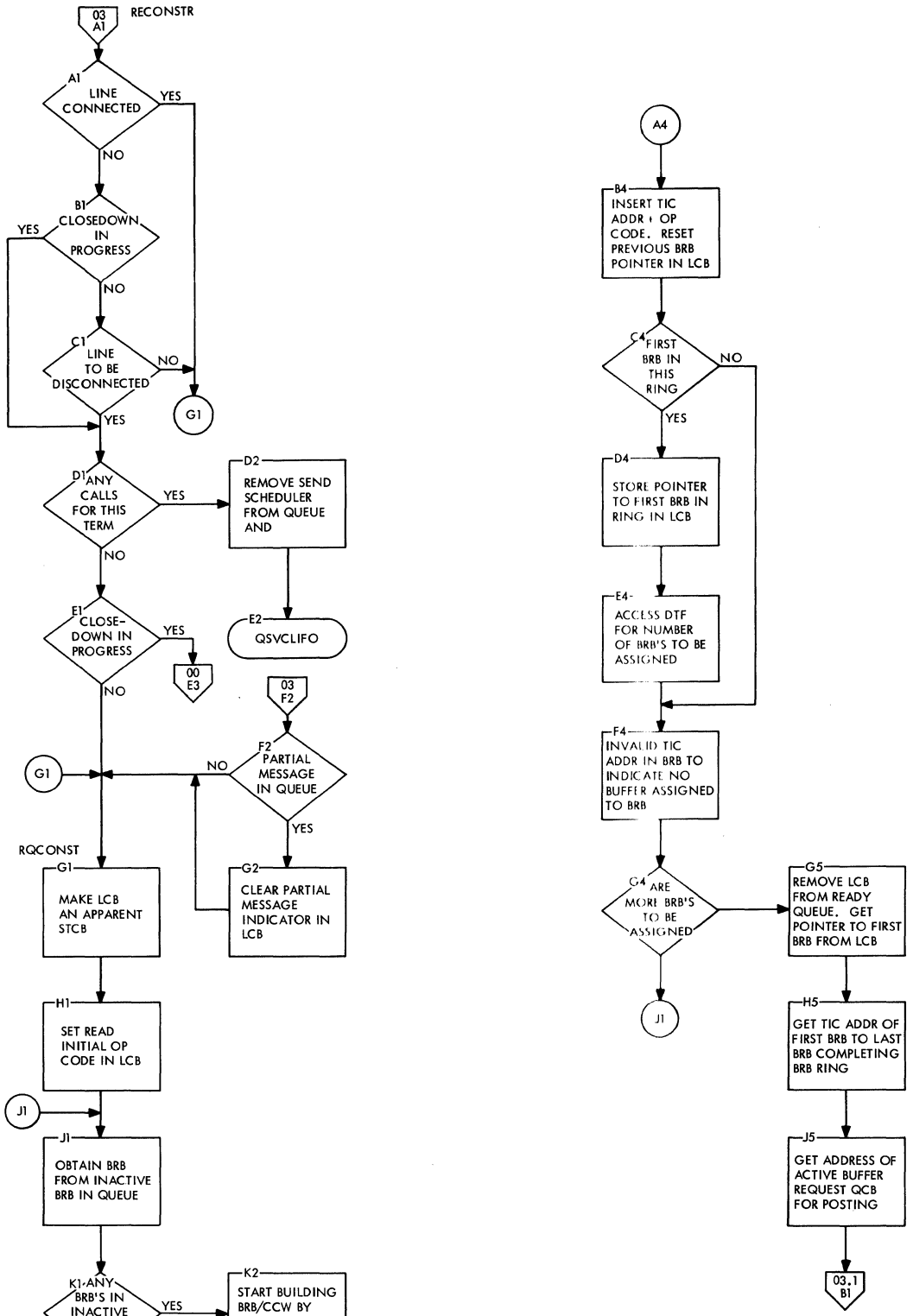


Chart 03.1. BRB Ring Routine

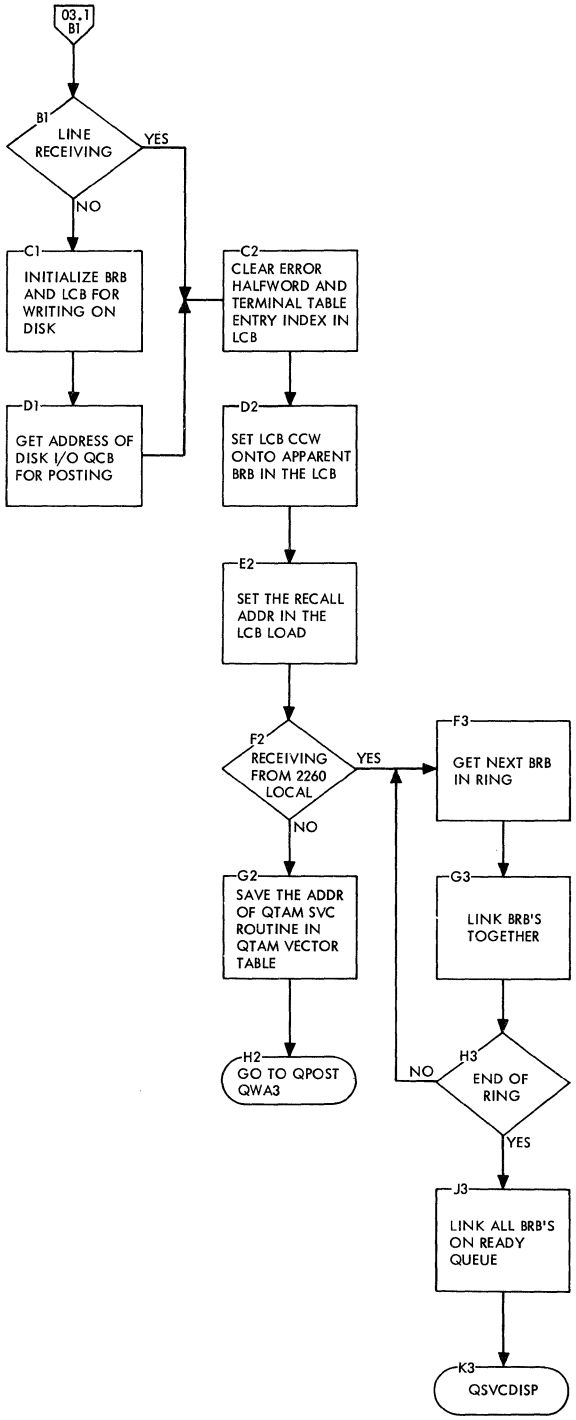
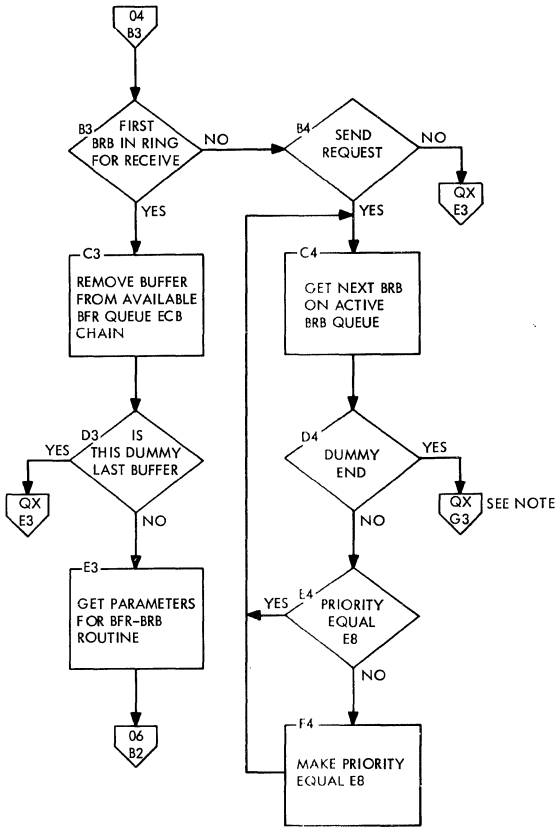


Chart 04. Active Buffer Request Routine



NOTE: BRB IS CODED TO LOOK LIKE A QCB. IN EFFECT, AN EXIT TO LIFO LINKS BRB'S TOGETHER

Chart 05. Available Buffer Routine

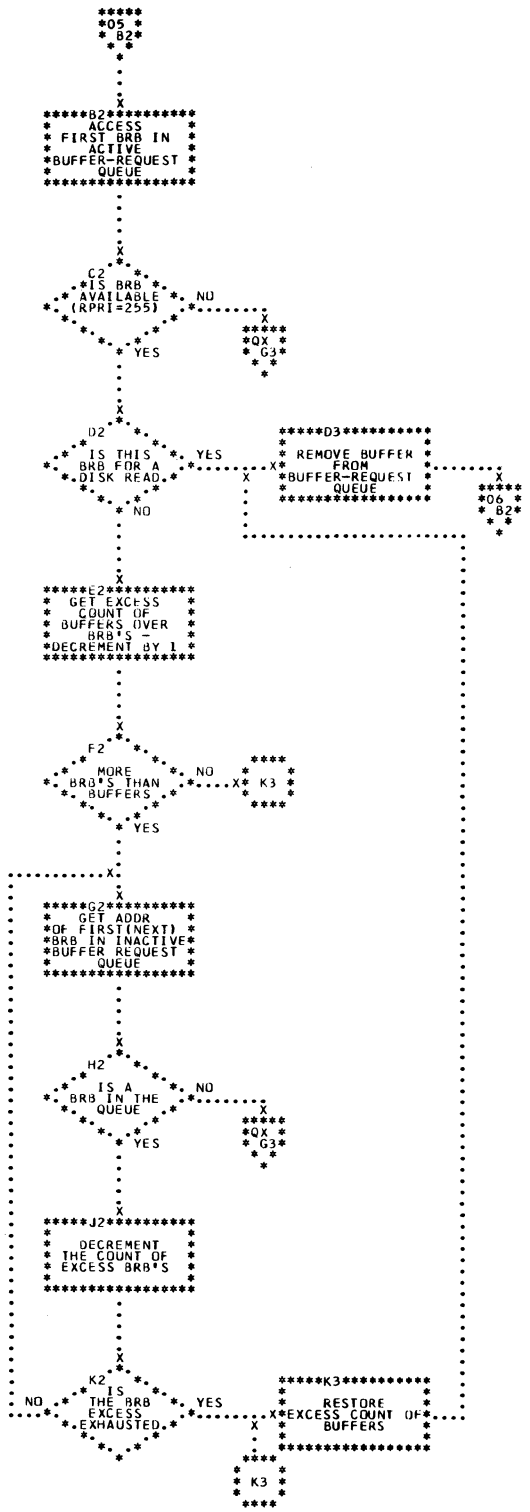


Chart 06. Buffer-BRB Routine

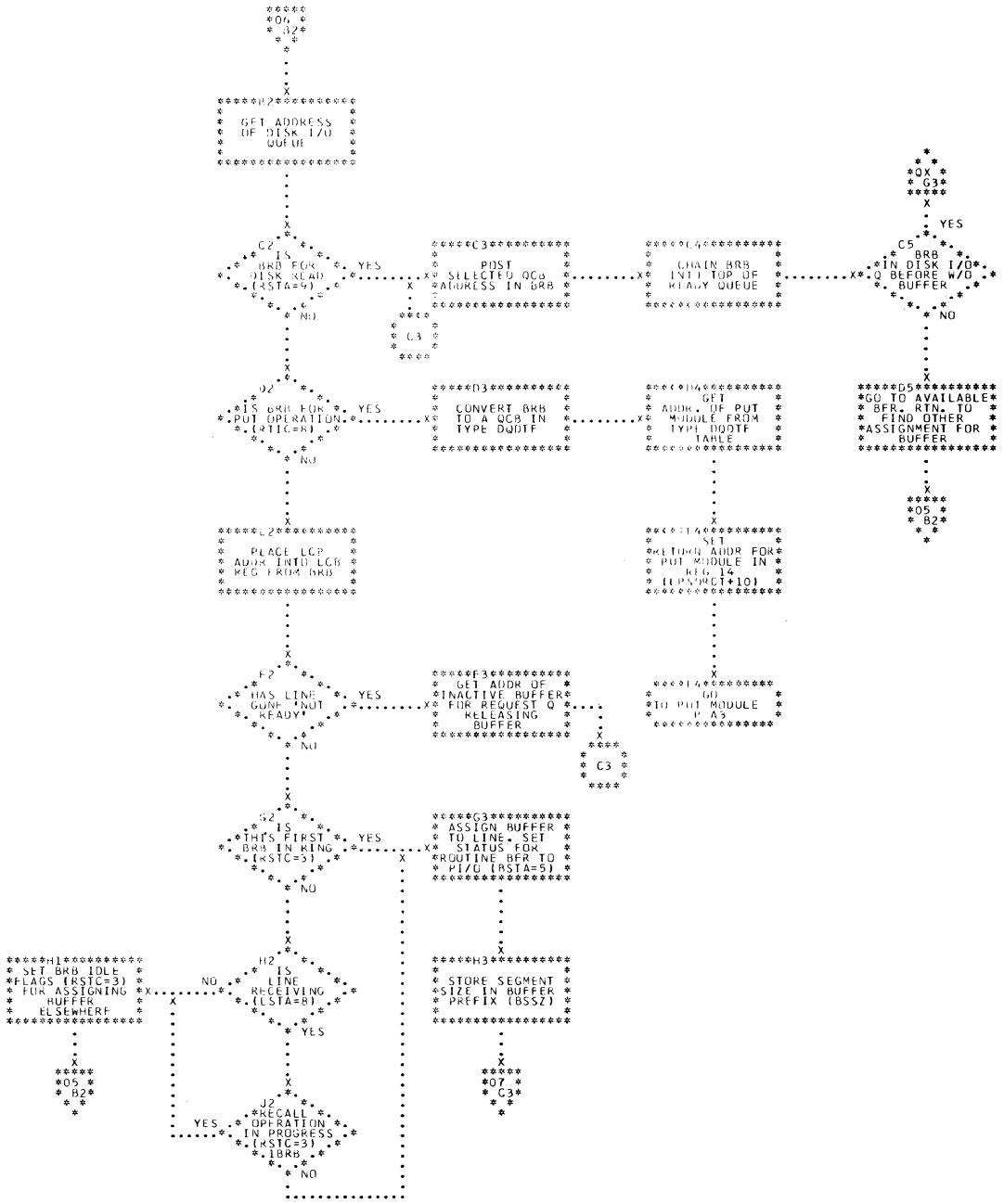


Chart 07. Interim LPS Routine

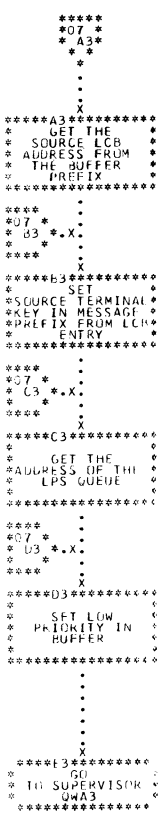


Chart 09. Qmover Routine

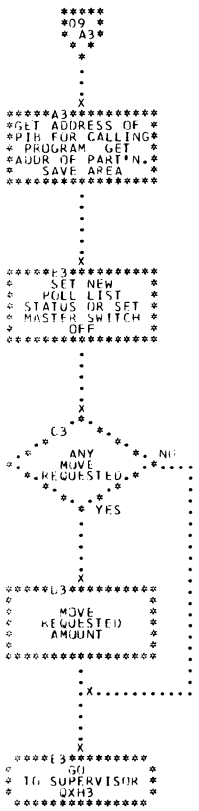


Chart 10. DASD Destination Routine

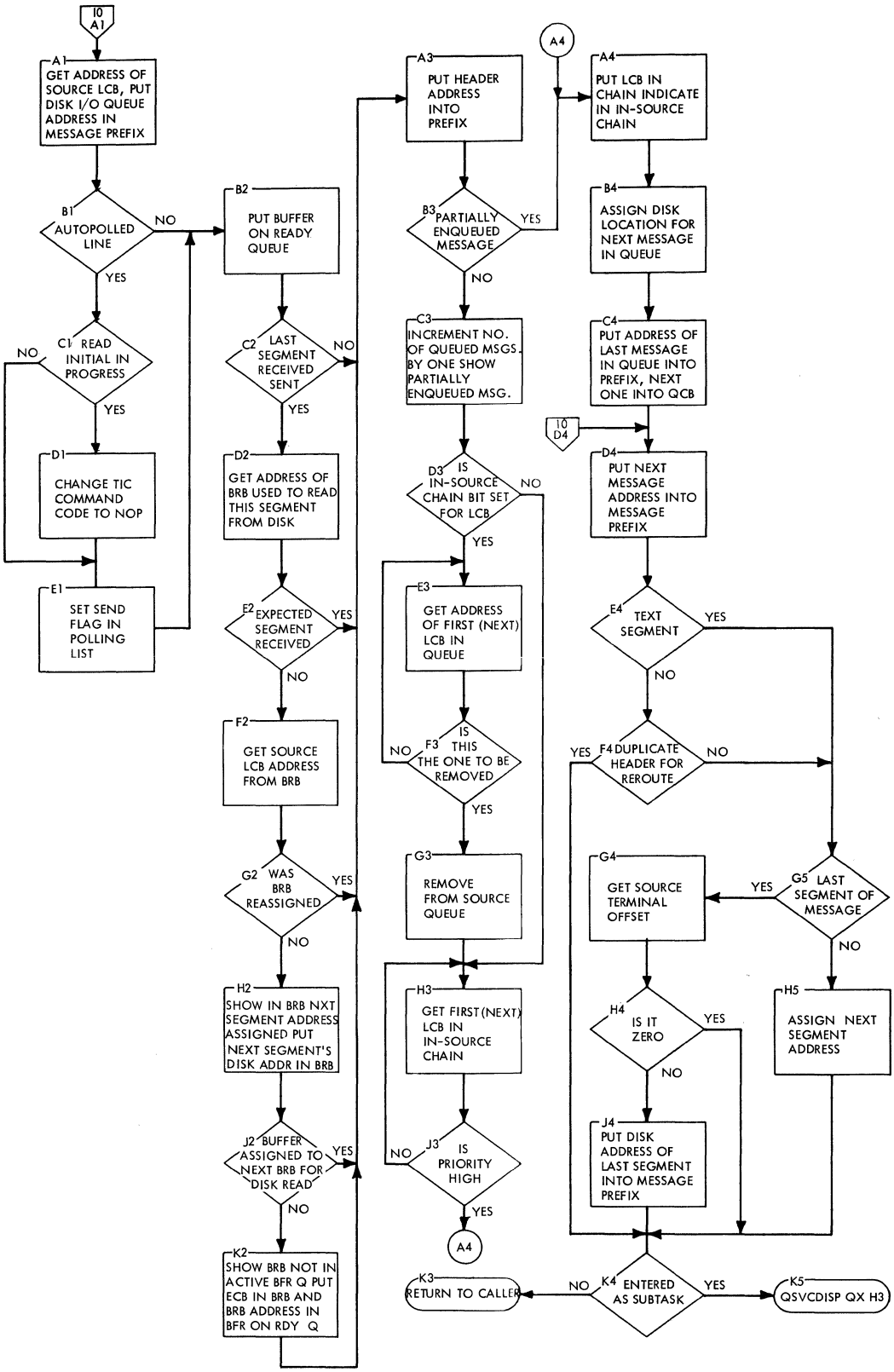


Chart 11. GET Scheduler Routine

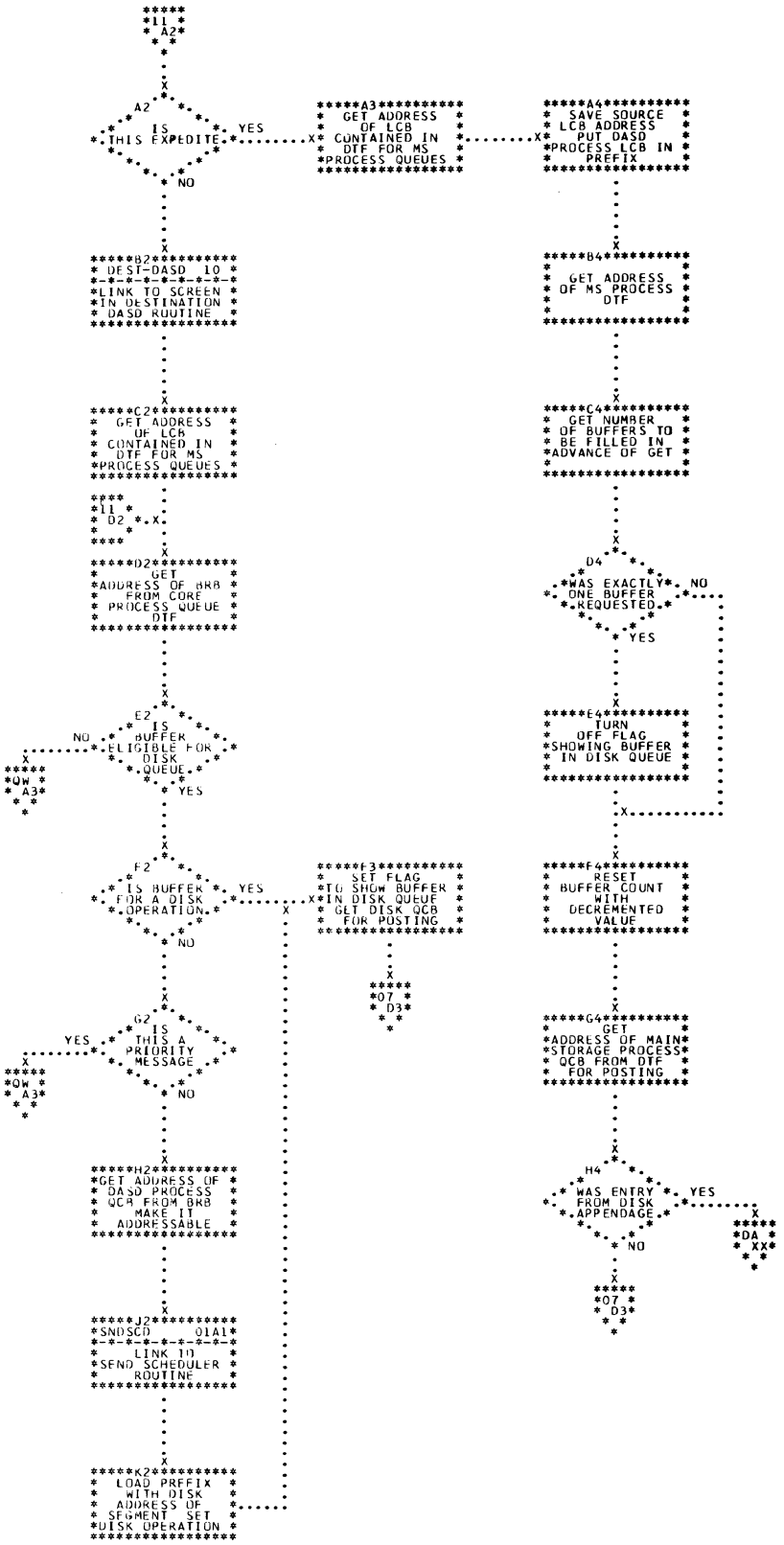


Chart 12. Return Buffer Routine

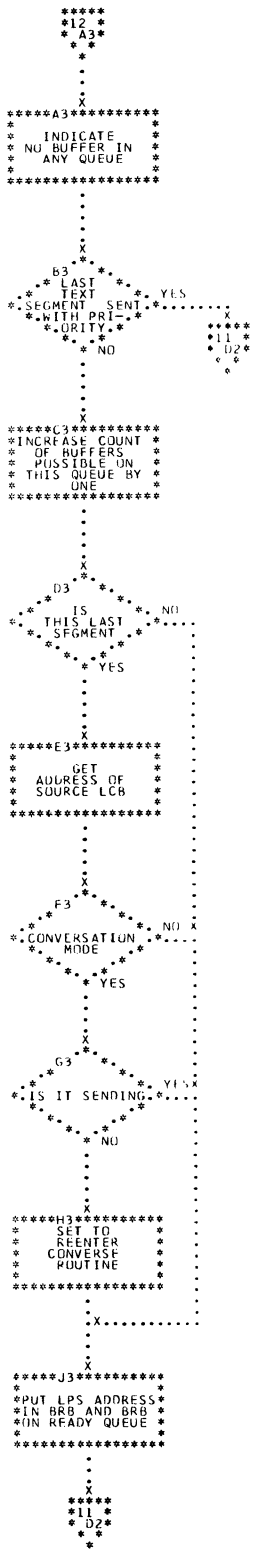


Chart 13. End Insert Routine

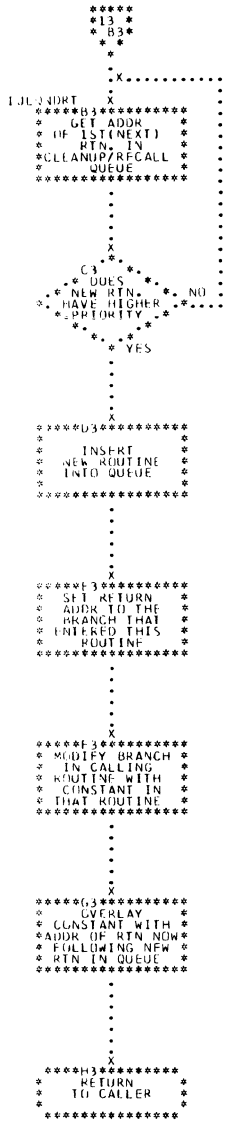


Chart 15. LPS Control Routine

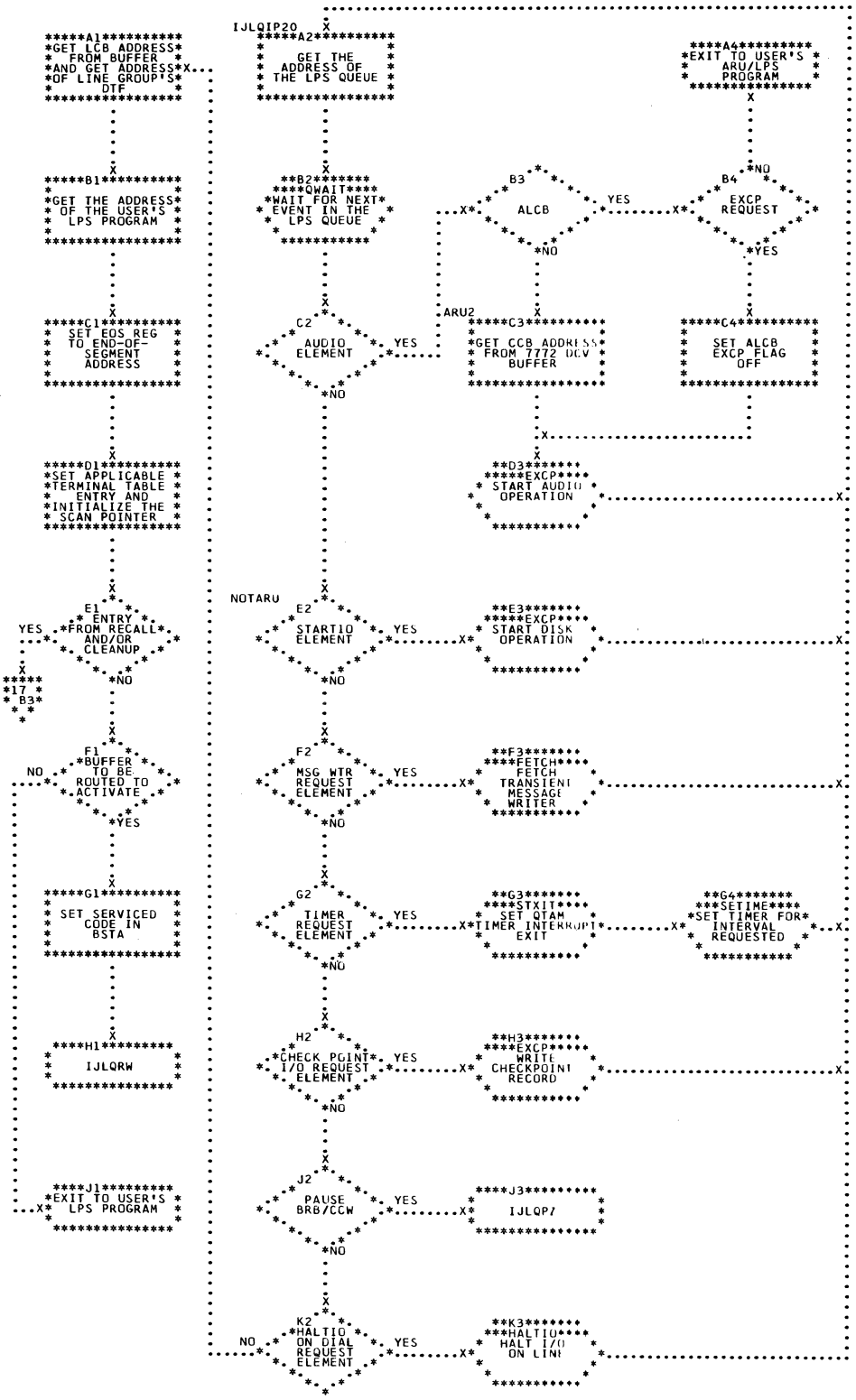


Chart 16. Buffer Recall/Cleanup Routine (Part 1 of 2)

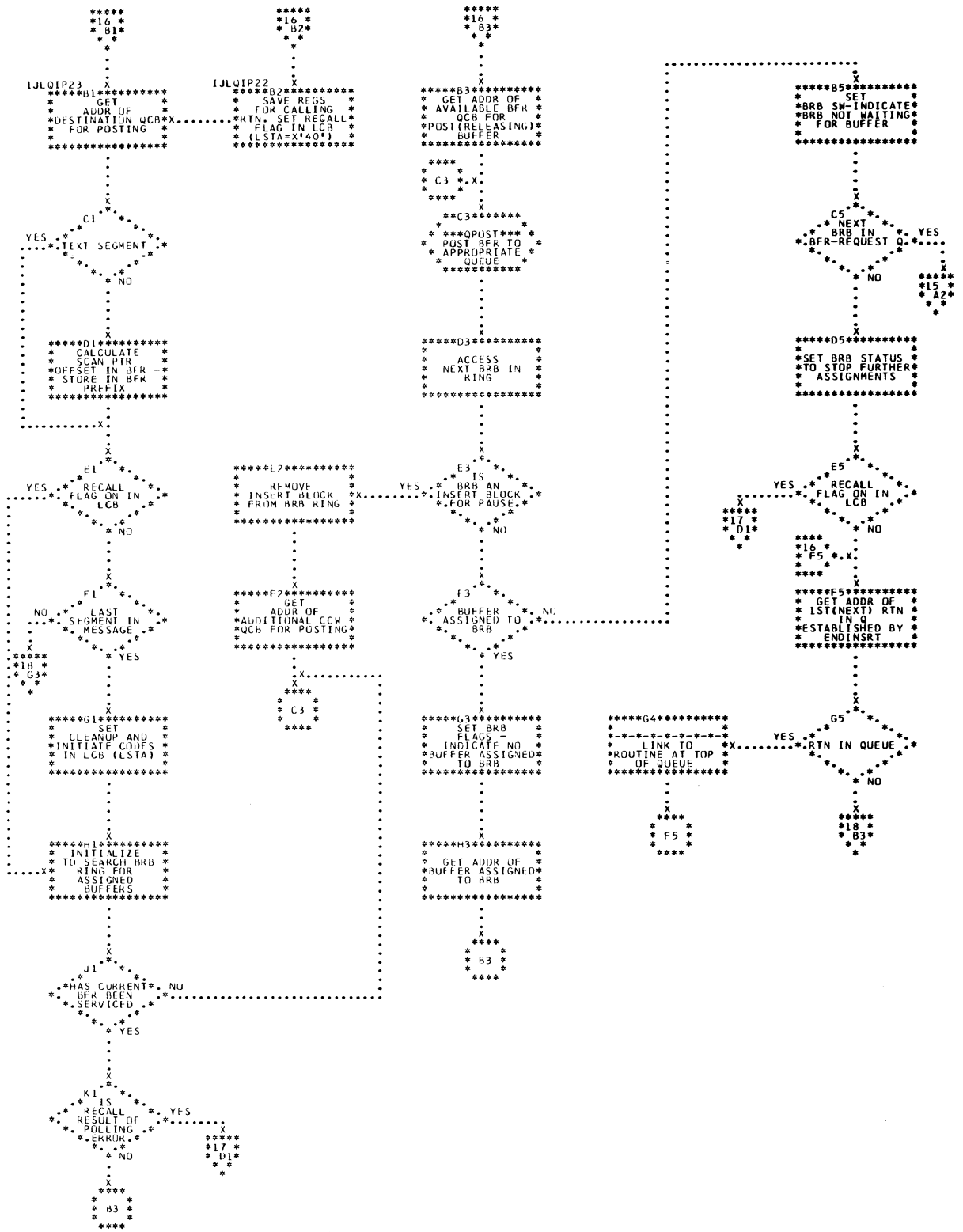


Chart 17. Buffer Recall/Cleanup Routine (Part 2 of 2)

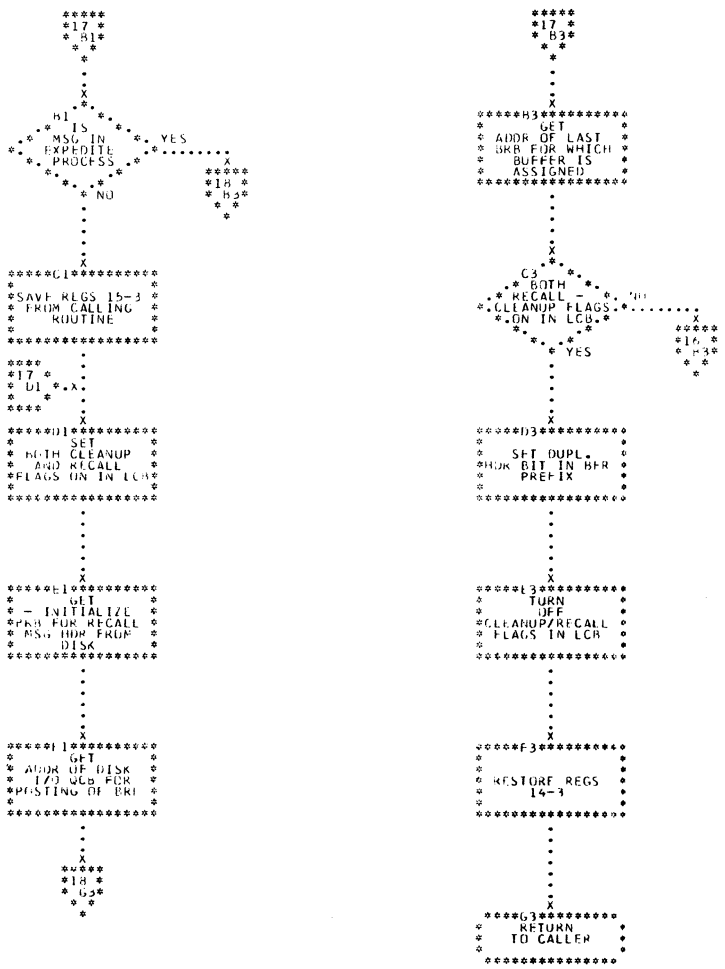
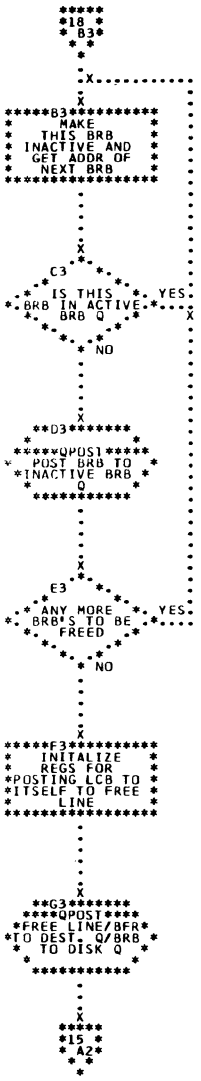


Chart 18. Free BRB Routine



APPENDIX A: QTAM QUEUES AND SUBTASKS

QUEUES

ACTIVE-BUFFER-REQUEST QUEUE

QCB: Preassembled in the Implementation module (IJLQIP); labeled IJLQIP71.

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. BRBs are posted to this chain by the BRB Ring, Disk I/O, Line PCI, and PUT routines.

STCB Chain: Limited to the STCB for the Active-Buffer-Request subtask.

ADDITIONAL-CCW QUEUE

QCB: Preassembled in the module (IJLQPZ) 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; the first BRB/CCW is labeled IJLQISRT. A chain of special-purpose BRB/CCWs used to schedule and contain channel commands for the transmission of idle characters (or any other user-specified characters). BRB/CCWs are requested from this queue by the Pause routine and are returned by the Line PCI and Cleanup routines.

STCB Chain: May contain the STCB for the LPS subtask. Always ends with the STCB for the Queue Insert subtask.

ARU-SEND QUEUE

QCB: Preassembled in the audio implementation routine in the Audio Line Appendage module (IJLQAA); labeled IJLQAA50.

Element Chain: Dynamically created. An element appearing on this chain is an Audio Line Control Block (ALCB) requesting an output operation. ALCBs are posted to this chain by the audio PUT, ARU-Internal, and Audio Line End routines.

STCB Chain: Limited to the STCB for the ARU-Send subtask.

ATTENTION QUEUE

QCB: A full word located in each LCB for an IBM 2260 Local line group; labeled IJLQLATN.

Element Chain: Limited to a CCB/ECB representing a read request (Attention Interrupt) from an IBM 2260 Local. The lastelement in the chain is always the dummy last element, IJLQIP5E.

STCB Chain: There is no STCB chain for this queue.

AVAILABLE-BUFFER QUEUE

QCB: Preassembled in the Implementation module (IJLQIP); labeled IJLQIP72.

Element Chain: Generated in the problem program on expansion of the BUFFER macro instruction; the first buffer is labeled IJLQIP92. A chain of operationally-empty buffers. Buffers are obtained from this queue as needed and are returned by the Cleanup, Return Buffer, and Line PCI routines and the Disk End appendage.

STCB Chain: Limited to the STCB for the Available-Buffer subtask.

CHECKPOINT QUEUE

QCB: Preassembled in the Checkpoint module (IJLQCK); labeled IJLQCQCB.

Element Chain: Limited to the checkpoint element labeled IJLQCECB.

STCB Chain: Limited to the STCB for the Checkpoint subtask.

COMMUNICATIONS-LINE QUEUE

QCB: Formed by the first 12 bytes of the Line Control Block (LCB) generated in the problem program on the expansion of the DTFQT macro for a line group file. There is one such QCB for each line defined in the system.

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 Qdispatch 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 and the preceding header segment through 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 actual DASD addresses. Buffers containing message segments are posted to these queues by the Cleanup and PUT routines.

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.

DCV BUFFER QUEUE

QCB: Generated in the problem program on expansion of the BUFARU macro instruction; labeled as this macro.

Element Chain: A chain of available DCV buffers generated on expansion of the BUFARU macro generating the QCB. These DCV buffers are obtained from this queue as needed, and returned by the Line End and Line PCI routines on completion of a 7772 line write operation.

STCB Chain: May contain the STCB for the 7772 DCV buffer subtask when no DCV buffer is available and one or more audio lines are waiting for an available DCV buffer. Generally contains the STCB for the Queue Insert subtask.

DISK INPUT/OUTPUT QUEUE

QCB: Preassembled in the Disk I/O module (IJLQDA); labeled IJLQIP73.

Element Chain: Dynamically created. A chain of BRBs (containing channel command words) for direct-access read operations, intermixed with full buffers to be written onto the DASD. BRBs are posted to this queue by the BRB Ring, Get Scheduler, and Available Buffer routines and the Disk End appendage. Buffers are posted to this queue by the DASD Destination routine.

STCB Chain: Limited to the STCB for the Disk Input/Output subtask.

DISTRIBUTION LIST QUEUE

QCB: Preassembled in the module (IJLQDL) introduced by the appearance of the LIST macro instruction in the problem program.

Element Chain: Limited to the dummy last element labeled IJLQIP5F. No element chain is developed. Elements (message-filled buffers) 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 (IJLQIP); labeled IJLQIP75.

Element Chain: Generated in the problem program on expansion of the BUFFER macro instruction; the first such BRB is labeled IJLQIP95. A chain of BRBs of which the third and fourth fullwords are effectively empty. BRBs are obtained from this queue by the BRB Ring routine and are returned by the Free BRB routine.

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 (IJLQIP); labeled IJLQIP76.

Element Chain: Limited to the dummy last element labeled IJLQIP5F. Buffers are posted to this queue by the Line PCI routine and are immediately transferred to the LPS queue upon activation of the Interim LPS subtask.

STCB Chain: Limited to the STCB for the Interim-LPS subtask.

LPS QUEUE

QCB: Preassembled in the Implementation module (IJLQIP); labeled IJLQIP77.

Element Chain: Dynamically created. A chain of empty buffers, to be used for messages incoming from terminals, interspersed with message-filled buffers to be processed by the LPS routines. Empty buffers are posted to this queue by the Buffer-BRB routine. Message-filled buffers are posted to this queue by the Interim LPS and Line End routines and by the Disk End appendage.

This element chain may also contain audio line control blocks requiring processing of their input messages by the ARU/LPS routines, or requiring issuance of an I/O operation, interspersed with 7772 DCV buffers (if any) requiring issuance of a disk read operation. These audio elements are posted to this queue by the Audio Line End routine.

STCB Chain: May contain the STCB for the LPS subtask. Always ends with the STCB for the Queue Insert by Priority subtask.

OPERATOR CONTROL CHNGT QUEUE

QCB: Preassembled in the Operator Control module (IJLQOC); labeled CHANGE.

Element Chain: Limited to a pointer to the Operator Control CHNGT QCB itself.

STCB Chain: Limited to the STCB for the Operator Control CHNGT 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 (IJLQIP); labeled IJLQIP78.

Element Chain: Limited to the dummy last element labeled IJLQIP5F. Buffers returned from a GET are immediately transferred to the Available Buffer queue.

STCB Chain: Limited to the STCB for the Return-Buffer subtask.

TERMINAL TEST BUFFER ROUTING QUEUE

QCB: Preassembled in the Resident Terminal Test routine (IJLQTT21); labeled TESTQCB.

Element Chain: Limited to the buffers containing the request for a terminal test. The buffers are posted to the queue by the Resident Terminal Test routine immediately after validation of the test activation code.

STCB: Limited to the STCB for the Terminal Test Buffer Routing subtask.

TERMINAL TEST STOPLINE QUEUE

QCB: Preassembled in the Resident Terminal Test routine (IJLQTT21); labeled ONLTTQCB.

Element Chain: Limited to the buffer containing the request for a terminal test.

STCB: Limited to the STCB for the Terminal Test Stopline Queue.

TIME DELAY QUEUE

QCB: Preassembled in the Implementation module (IJLQIP); labeled TIMEQ.

Element Chain: Limited to a pointer to the Time Delay QCB itself.

STCB Chain: Limited to the STCB for the Time Delay subtask.

QMOVE QUEUE

QCB: Preassembled in the Implementation module (IJLQIP); labeled IJLQIP70.

Element Chain: Limited to a pointer to the Qmove QCB itself.

STCB Chain: Limited to the STCB for the Qmover subtask.

SUBTASKS

ACTIVE-BUFFER-REQUEST SUBTASK

STCB: Preassembled in the Implementation module (IJLQIP); labeled IJLQIP81.

Program Entry: Enters the Active-Buffer-Request routine at IJLQIP81 + 6.

ARU-SEND SUBTASK

STCB: Preassembled in the Audio Line Appendage module (IJLQAA); labeled IJLQAA51.

Program Entry: Enters the ARU-Send routine at IJLQAA51+6.

AVAILABLE-BUFFER SUBTASK

STCB: Preassembled in the Implementation module (IJLQIP); labeled IJLQIP82.

Program Entry: Enters the Available-Buffer routine at IJLQIP82 + 6 (alternate label BFRREQ).

CHECKPOINT SUBTASK

STCB: Preassembled into the Checkpoint module (IJLQCK); labeled IJLQCSTB.

Program Entry: Enters the Checkpoint routine at IJLQCSTB+6.

DASD DESTINATION SUBTASK

STCB: Preassembled in the Implementation module (IJLQIP); labeled IJLQIP89.

Program Entry: Enters the DASD Destination routine at IJLQIP89 + 6.

DCV BUFFER SUBTASK

STCB: Preassembled in the 7772 Audio Disk Appendage module (IJLQAD); labeled IJLQAD40.

Program Entry: Enters the 7772 DCV buffer routine at IJLQAD40+6.

DISK INPUT/OUTPUT SUBTASK

STCB: Preassembled in the Disk I/O module (IJLQDA); labeled IJLQIP83.

Program Entry: Enters the Disk Input/Output routine at IJLQIP83 + 6.

DISTRIBUTION-LIST SUBTASK

STCB: Preassembled in the Distribution List (IJLQDL) module; located at IJLQDL + 8.

Program Entry: Enters the module at IJLQDL + 14.

GET-SCHEDULING SUBTASK

STCB: Preassembled in the Implementation module (IJLQIP); labeled IJLQIP8A.

Program Entry: Enters the GET-Scheduler routine at IJLQIP8A + 6.

INTERIM LPS SUBTASK

STCB: Preassembled in the Implementation module (IJLQIP), located at IJLQIP76 + 8.

Program Entry: Enters the Interim LPS routine at IJLQIP76 + 14.

LPS SUBTASK

STCB: Preassembled into the QTAM SVC/Subtask Control routine. A full STCB is initialized by this routine as a record of entry caused by issuance of a QTAM SVC in the problem program.

Program Entry: Activation of the LPS subtask causes the problem program to be re-entered at the instruction following the Supervisor call.

OPERATOR CONTROL CHNGT SUBTASK

STCB: Preassembled in the Operator Control module (IJLQOC); labeled CHANGE1.

Program Entry: Enters the Operator Control routine at CHANGE1+6.

QUEUE INSERT SUBTASK

STCB: Preassembled in the Implementation module (IJLQIP); labeled IJLQIP8B.

Program Entry: Enters the Implementation module at IJLQIP8B + 6 -- an unconditional branch to the Queue Insert subroutine (QSVCLIFO) in the QTAM SVC/Subtask Control routine.

QUEUE INSERT BY PRIORITY SUBTASK

STCB: Preassembled in the Implementation module (IJLQIP); labeled IJLQIP8C.

Program Entry: Enters the Implementation module at IJLQIP8C + 6 -- an unconditional branch to the Priority Search subroutine (QSVCPRI) in the QTAM SVC/Subtask Control routine.

QDISPATCH SUBTASK

STCB: Preassembled in the Implementation module (IJLQIP); labeled IJLQIP8D.

Program Entry: Enters the Implementation module at IJLQIP8D + 6 -- an unconditional branch to the Qdispatch subroutine (QSVCDISP) in the QTAM SVC/Subtask Control routine.

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 in the Implementation module (IJLQIP); labeled IJLQIP88.

Program Entry: Enters the Return-Buffer routine at IJLQIP88 + 6.

SEND-SCHEDULING SUBTASK

STCB: There is one Send-Scheduling subtask for each line or terminal, depending on the queuing technique selected 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-Scheduling subtasks enter the Send-Scheduler routine at ENQUEUE.

TERMINAL TEST BUFFER ROUTING SUBTASK

STCB: Preassembled in the Resident Terminal Test routine (IJLQTT); labeled TESTSTCB.

Program Entry: Enters the Resident Terminal Test routine at TESTSTCB+6.

TERMINAL TEST SINGLE STOPLINE SUBTASK

STCB: Preassembled in the Resident Terminal Test routine (IJLQTT); labeled STCBKEY.

Program Entry: Enters the Resident Terminal Test routine at STCBKEY+8.

TERMINAL TEST STOPLINE SUBTASK

STCB: Preassembled in the Resident Terminal Test routine (IJLQTT); labeled STPLSTCB.

Program Entry: Enters the Resident Terminal Test routine at POSTLPS.

TIME DELAY SUBTASK

STCB: Preassembled in the Implementation module (IJLQIP); labeled TIMEEND.

Program Entry: Enters the End of Poll Time Delay routine at TIMEEND + 6. This subtask is dispatched when the Time Delay queue appears at the top of the Ready Queue.

QMOVER SUBTASK

STCB: Preassembled in the Implementation module (IJLQIP); labeled IJLQIP80.

Program Entry: Enters the Qmover routine at IJLQIP80 + 6. This subtask is dispatched when the Qmove QCB appears at the top of the Ready Queue.

THE QUEUE CONTROL BLOCK (QCB)

The QCB, as used in QTAM, is a control block containing information on the status and contents of the queue it represents. There is one QCB for each queue. The QCB points to the head of a chain of element control blocks (ECB) and to the head of a chain of subtask control blocks (STCB).

The general form of a basic QCB is:

0	key	element chain pointer
+4	priority	link address
+8		STCB chain pointer

Typical DSECT (IJLQQCS1B0):

0	QKEY	QEBC
+4	QQPR	QLNK
+8		QSTC

Note: All DSECT names have the four character prefix IJLQ.

Contents:

Key (QKEY): A numeric value (1,2, or 3) indicating the status of the queue.

1--QCB is not on the Ready Queue.

2--QCB is on the Ready Queue and its highest priority subtask is ready for activation.

3--QCB is on the Ready Queue, but its highest priority subtask is waiting for an element and cannot be activated.

Element Chain Pointer (QEBC): The address of the first element control block in the element control block chain for the queue.

Priority (QQPR): Priority of the queue the QCB represents. This is the search argument for linking the QCB into its proper relative position (by priority) on the Ready Queue.

Link Address (QLNK): The address of the next item on the Ready Queue. This field is meaningful only when the QCB is on the Ready Queue.

STCB Chain Pointer (QSTC): The address of the first STCB on the STCB chain for the queue.

THE ELEMENT CONTROL BLOCK (ECB)

An element control block represents the condition of an element. As previously described, QTAM defines three types of elements:

1. Buffers
2. Communication lines (represented by LCBs or ALCBs)
3. Buffer requests (represented by BRBs)
4. 7772 DCV buffers

The first two fullwords of each type of element form an element control block which contains control information as to the current status of the element. The general form of an element control block is:

0	key	QCB address
+4	priority	link address

Key: Always zero when the ECB appears on the Ready Queue (to distinguish it from a QCB).

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.

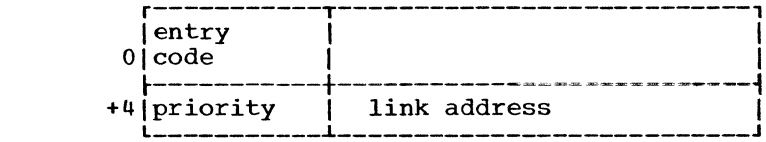
Priority: Priority of the element the control block represents. This field determines the relative position of the element when linked into the element chain of a QCB, or into the Ready queue. Priority 255 identifies the last element in a chain; this is a dummy element usable only as an indication that the end of the chain has been reached. QTAM controls 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

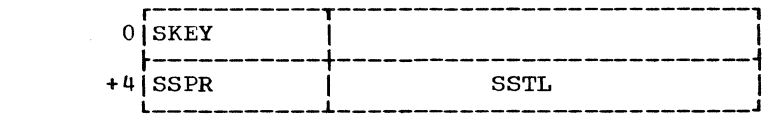
element control block is linked into the element chain of a Queue Control Block or into the Ready queue.

TRUNCATED SUBTASK CONTROL BLOCK (STCB)

General Form:



Typical DSECT (IJLQSTS1B0):



Note: All DSECT names have the four character prefix IJLQ.

Entry Code: Branch modifier; a numeric value (a multiple of 2 greater than zero) added, in effect, to the resolved address of storage location NRET in module IJLQIP 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.

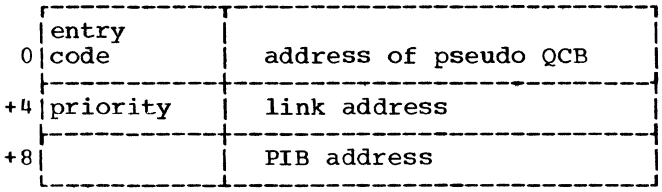
Priority: Priority of the subtask the STCB represents; determines the relative position of the STCB when linked into the STCB chain of a queue control block. Priority 255 identifies the last STCB in a chain.

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 DTFQT macro for the line group.

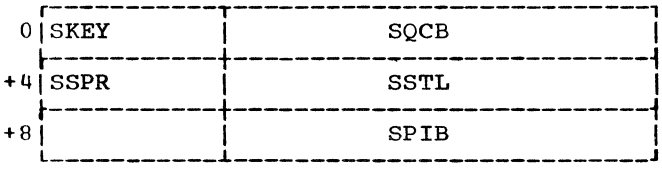
Link Address: A pointer to the next STCB in the STCB chain; the last STCB in a chain links to itself. This field is meaningful only when the STCB is linked into the STCB chain of a queue control block.

FULL SUBTASK CONTROL BLOCK (STCB)

General Form:



Typical DSECT (IJLQSTS1B0):



Note: All DSECT names have the four character prefix IJLQ.

Entry Code: Always zero; this enables the Exit Select subroutine in the QTAM SVC/Subtask Control routine to distinguish a full STCB (representing re-entry to a program that issued a QTAM SVC) from a truncated STCB.

Address of Pseudo QCB: This field always contains the address of the pseudo QCB labeled QSVCQCB (storage location QSVCRDYQ minus 8). This enables the Qdispatch subroutine to properly handle the case where it encounters the full STCB at the top of the Ready queue.

Priority: Used for the same purpose as in a truncated STCB. The Entry Interface subroutine assigns a priority to the full STCB initialized when the DOS Supervisor enters the QTAM SVC/Subtask Control routine in response to a QTAM SVC. When a full STCB is inactive, it always has a priority of 255.

Link Address: Initially contains the address of the next full STCB in the chain of full STCBs assembled into the SVC/Subtask Control routine. Thereafter, this field is the same as for a truncated STCB. The last full STCB in the initial chain of full STCB's is a dummy full STCB (labeled QSVCSTOP) that links to itself.

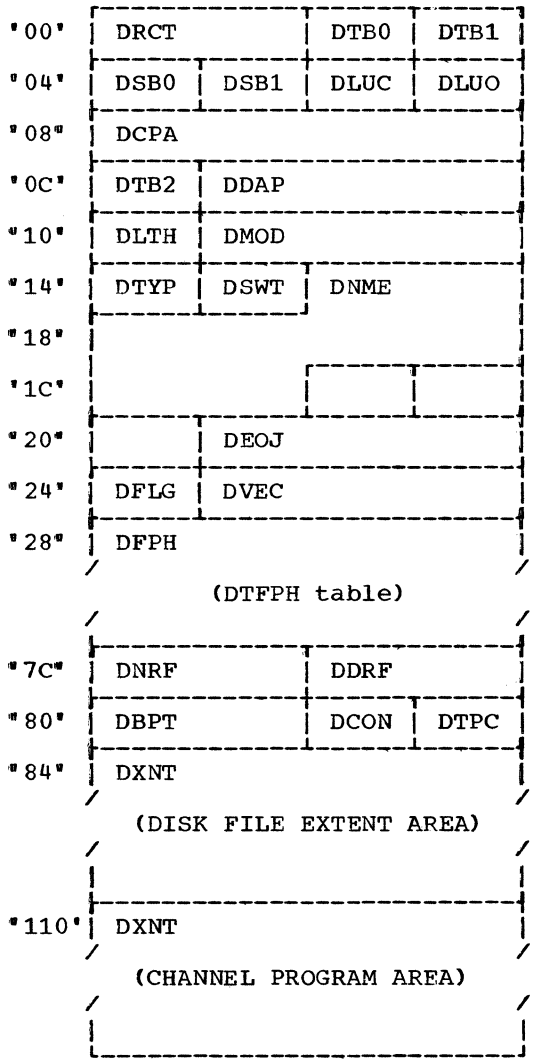
PIB Address: When the full STCB is active, that is, it has been initialized by the Entry Interface subroutine, this field contains the address of the Program Information Block (PIB) for the program interrupted by the QTAM SVC. At all other times, it contains the address of QTAM's dummy PIB (labeled QSVCDPIB).

There are eight different types of files in QTAM:

- Direct Access (DA)
- Checkpoint Records (CK)
- 7772 DCV Vocabulary (AV)
- Line Group (LG)
- Audio Line Group (LG)
- Process Queue (PQ)
- Destination Queue (DQ)
- Audio Output Queue (AQ)

The format of the DTF table for each type is different and is discussed separately.

DTF TYPE DA



Note: All DSECT names have the four-character prefix IJLQ.

Figure 10. DTF Table Format for Type DA

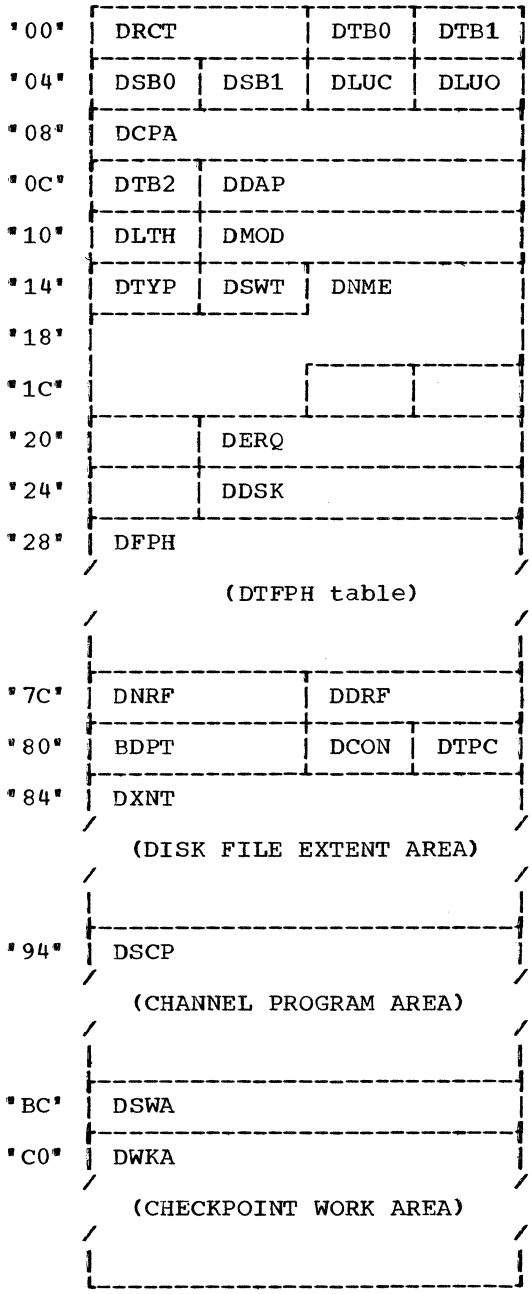
Contents:

The format of the DA type of DTF table is shown in Figure 10.

Field	Length in Bytes	Description
		The first sixteen bytes are the CCB data incorporated in the DTF table
DRCT '00'	2	Residual count.
DTB0 '02'	1	Transmission byte 0.
DTB1 '03'	1	Transmission byte 1.
DSB0 '04'	1	CSW status byte 0.
DSB1 '05'	1	CSW status byte 1.
DLUC '06'	1	Logical unit class. This byte is always set to X'01'.
DLUO '07'	1	Logical unit order (X'nnn').
DCPA '08'	4	Address of the first CCW in the channel program area.
DTB2 '0C'	1	Set to X'40' for DA type to inticate that the next three bytes contain an appendage address.
DDAP '0D'	3	Address of the Disk Appendage routine (IJLQDA).
		The next fourteen bytes are the System Interface Area and are common to all DTF tables (with the exception of the DNME field).
DLTH '10'	1	The number of double words in this DTF table.
DMOD '11'	3	Address of the Implementation module(IJLQIP).
DTYP '14'	1	DTF type code. Codes for different DTF types are: X'50'=DA, CK, or AV X'51'=LG X'52'=PQ X'53'=DQ X'54'=LG (Audio) X'55'=AQ
DSWT '15'	1	DOS system switches. Bit 5 (X'08) is set on to indicate that this is a DOS Version 3 DTF.
DNME '16'	7	The name of the DTFQT macro that caused generation of this table.
DNME+7 '1D'	1	DASD device code. Codes for different DASD devices are: X'00'=IBM 2311 X'01'=IBM 2314. End of System Interface Area.
DNME+8 '1E'	1	DASD type. X'00'=DASD Message Queues file
DEOJ '21'	3	Address of the user's end-of-job routine
DFLG '24'	1	QTAM flags. Bits 6 and 7 (X'03') are used by the GET modules. Other flags are used as follows: X'01'=Auto Polled lines X'04'=Input file X'08'=Output file X'10=DLBL information has been initially processed X'0C'=Input and output file X'20'=Checkpoint flag bit. In a type LG DTF table, this bit is flipped when the LCB data has been collected from the LCBs. X'40'=Separate assembly X'80'=Switched lines X'80'=Audio flag for PQ type if WU=A,MA,RA, or SA is specified.

Field	Length in Bytes	Description
DVEC '25'	3	Address of the QTAM Vector Table (IJLQIP5E).
DFPH '28'	84	This is a DTFPH table wholly included within the DTF table for type DA.
DNRF thru DTPC	8	These five fields contain DASD device type constants.
DXNT '84'	140	Save area for the extents of the file on disk
DDCP '110'	32	Channel program area

DTF TYPE CK



Note: All DSECT names have the four-character prefix IJLQ.

Figure 11. DTF Table Format for Type CK

Contents:

The format of the CK type of the DTF table is shown in Figure 11. The first 30 bytes of the CK type of DTF table are the same as those for the DA type of DTF table, with the exception of the DMOD field.

Field	Length in Bytes	Description
DMOD '11'	3	Address of the Checkpoint routine (IJLQCK).
DNME+7 '1D'	1	DASD device code. Codes for different DASD devices are: X'00'=IBM 2311 X'01'=IBM 2314
DNME+8 '1E'	1	DASD type. X'02'=DASD Checkpoint Records file.
DERQ '21'	3	Address of the error queue.
DDSK '25'	3	Address of IJLQDA60 in the Disk Appendage module (IJLQDA).
DFPH '28'	84	This is a DTFPH table wholly included with this DTF table.
DNRF thru DTPC	8	These five fields contain DASD device type constants.
DXNT '84'	16	Save area for the extent information
DSCP '94'	40	Channel program area.
DSWA 'BC'	4	Size of the checkpoint work area.
DWKA 'C0'	n	Checkpoint work area.

DTF TYPE AV

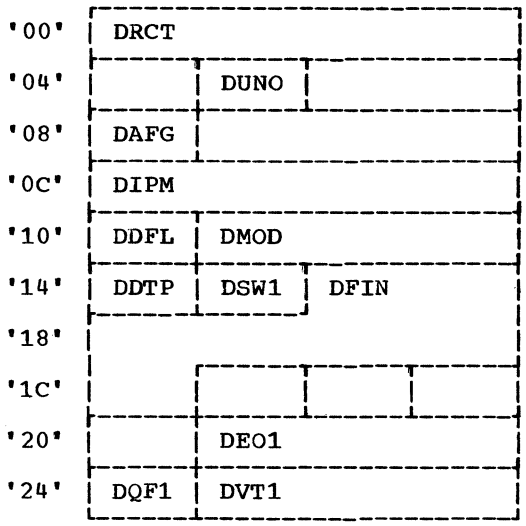


Figure 12. DTF Table Format for Type AV.

Contents:

The format of the AV type of the DTF table is shown in Figure 12.

Field	Length in Bytes	Description
DRCT '00'	4	Address of the 7772 word table (IJLQWTBL).
DUNO '05'	1	Logical unit order (X'nnn')
DAFG '08'	1	Audio flags. Bits 6 and 7 (X'03') are not used. Other flags are used: X'04'=DCV buffer pool list specified. X'08'=Word table specified. X'10'=Logging with time stamping specified. X'20'=Initial write (on) or read (off) operation. X'40'=Mode conversation (on) or invitation (off). X'80'=Mode information (on).
	3	Address of the Audio Line Appendage module (IJLQAA).
DIPM '0C'	4	Address of the Implementation module (IJLQIP).
DDFL '10'	1	Number of double words in this DTF table.
DMOD '11'	3	Address of the 7772 Disk Appendage module (IJLQAD).
DDTP '14'	1	DTF type code (X'50').
DSW1 '15'	1	DOS System Switches. Refer to DA type (DSWT field).
DFIN '16'	7	The name of the DTFQT macro that causes generation of this table.
DFIN+7 '1D'	1	DASD device code. Codes for different DASD devices are: X'00'=IBM 2311 X'01'=IBM 2314
DFIN+8 '1E'	1	DASD type. X'04'= DASD DCV Vocabulary file.
DEO1 '21'	3	Address of the last previous DTFQT file opened in the message control program or address of the user's end-of-job routine.
DQF1 '24'	1	QTAM flags. Refer to DA type (DFLG field).
DVT1 '25'	3	Address of the QTAM Vector Table (IJLQIP5E).

DTF NONAUDIO TYPE LG

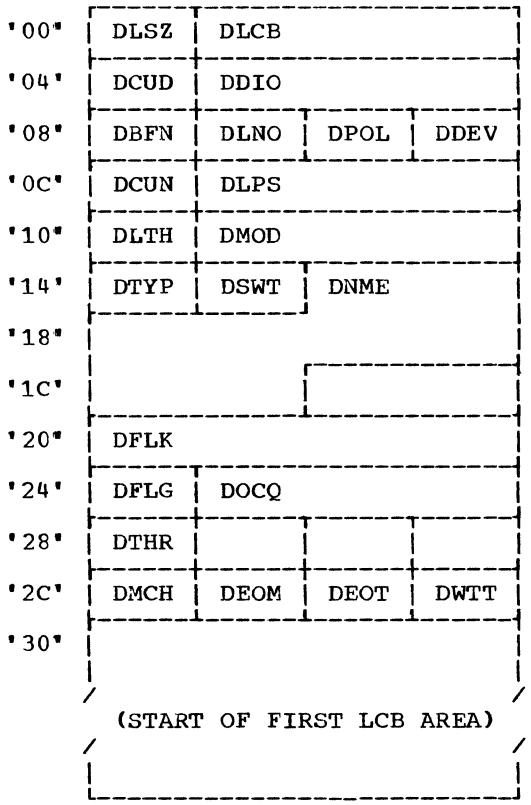


Figure 13. DTF Table Format for Nonaudio Type LG

Contents:

The format of the nonaudio LG type of DTF table is shown in Figure 13. The System Interface Area is the same as that for the DA type of DTF table with the exception of the DMOD field.

Field	Length in Bytes	Description
DLSZ '00'	1	Size of the LCB.
DLCB '01'	3	Address of the first LCB minus the size of the LCB
DCUD '04'	1	Device type code. Codes are: X'01'=IBM 1030 X'02'=AT&T 83B3 X'03'=WU Plan 115A X'04'=IBM 1060 X'05'=IBM 2260 remote X'06'=IBM 1050 X'07'=AT&T TWX X'08'=IBM 2260 Local X'09'=IBM 2740 (274A) X'0A'=IBM 2740 (274C) X'0B'=IBM 2740 (274D) X'0C'=IBM 2740 (274F) X'0D'=IBM 2740 (274B) X'0E'=IBM 2740 (274E) X'0F'=IBM 2740 (274G) X'10'=IBM 2740 (274H) X'11'=WTTA
DDIO '05'	3	Address of the Device I/O module for the line group.
DBFN '08'	1	Number of buffers needed for each transmission.
DLNO '09'	1	Number of lines in the line group.
DPOL '0A'	1	Polling interval in seconds.
DDEV '0B'	1	Device-access field offset for each terminal table entry.
DCUN '0C'	1	Control unit type code. Codes are: X'01'=2701 X'02'=2702 X'03'=2703
DLPS '0D'	3	Address of the user's LPS logic for this line group.
DLTH '10'	1	The number of double words in this DTF table (Bytes 16 through 29 form the System Interface Area. See DTFQT TYPE DA).
DMOD '11'	3	Address of the Physical IOCS module (IJLQRW).

Field	Length in Bytes	Description
DFLK '20'	4	Address of the last previous DTFQT file opened in the message control program.
DFLG '24'	1	QTAM flags. Same as in DA type of DTF file.
DOCQ '25'	3	Dial out call queue.
DTHR '28'	4	Threshold values for the error recovery procedures. First byte=number of transmissions. Second byte=number of data check errors. Third byte=number of intervention-required errors. Fourth byte=number of timeout errors.
DMCH '2C'	1	Number of Mark characters required before sending an output message to a World Trade terminal not equipped with the Motor-On optional feature.
DEOM '2D'	1	EOM signal for World Trade terminals.
DEOT '2E'	1	EOT signal for World Trade terminals
DWTT '2F'	1	Flags for World Trade terminals: X'01'=WRU macro instruction present in the Send Header subgroup of the LPS X'02'=WRU macro instruction present in the End Send subgroup of the LPS. X'04' X'08' Not used X'10' X'20'=IAM=YES specified in the DTFQT macro instruction X'20'=IAM=YES specified in the DTFQT macro instruction X'40'=WRU=YES specified in the DTFQT macro instruction X'80'=WTTA line indicator

DTF AUDIO TYPE LG

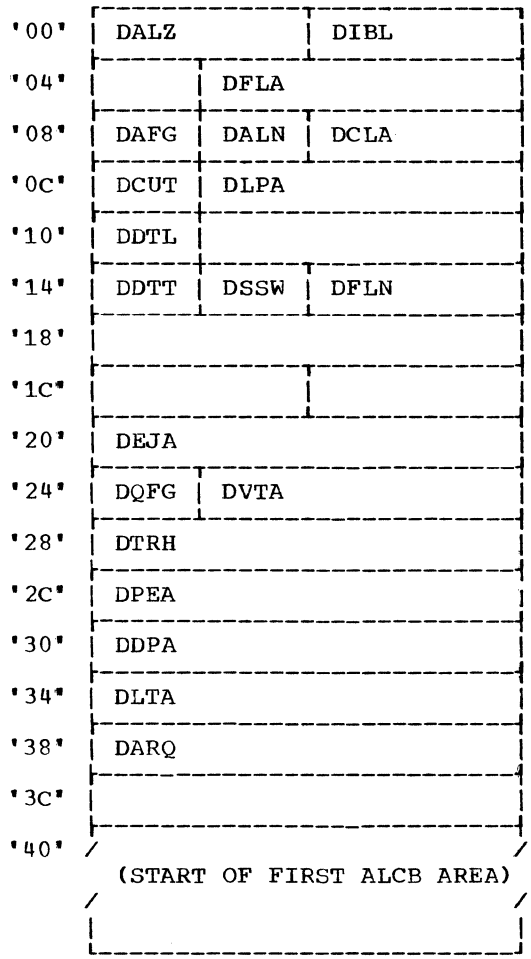


Figure 14. DTF Table Format for Audio Type LG.

Contents:

The format of the audio LG type of DTF table is shown in Figure 14.

Field	Length In Bytes	Description
DALZ '00'	2	Size of the ALCB.
DIBL '02'	2	Length of the input buffer.
DFLA '05'	3	Address of the first ALCB.
DAFG '08'	1	Audio flags. Refer to AV. type (DAFL field).
DALN '09'	1	Number of lines in the line group
DCLA '0A'	2	Length of the address chain buffer or length of the informational message.
DCUT '0C'	1	Control unit code X'70'=IBM 7770 X'72'=IBM 7772
DLPA '0D'	3	Address of the user's ARU/LPS logic for this line group.
DDTL '10'	1	Number of double words in this DTF table.
'11'	3	Address of the informational address chain.
DDTT '14'	1	DTF type code (X'54').
DSSW '15'	1	DOS System switches. Refer to DA type (DSWT field).
DFLN '16'	8	The name of the DTFQT macro that causes generation of this table.
DEJA '20'	4	Address of the last previous DTFQT file opened in the message control program or address of the user's end-of-job routine.
DQFG '24'	1	QTAM flags. Refer to DA type (DFLG field).
DVTA '25'	3	Address of the QTAM Vector Table (IJLQIP5E).

Field	Length in Bytes	Description
DTRH '28'	4	Threshold values for the error recovery procedures: First byte=number of transmissions. Second byte=number of hang-up operations. Third byte=number of errors on read operations. Fourth byte=number of errors on write operations.
DPEA '2C'	4	Address of the process program entry.
DDPA '30'	4	Address of the DCV buffer pool (7772) or address of the Implementation module (7770).
DLTA '34'	4	Address of the line table IJLQLTBL).
DARQ '38'	4	Address of the output Queue IJLQAA50).

DTF TYPE PQ

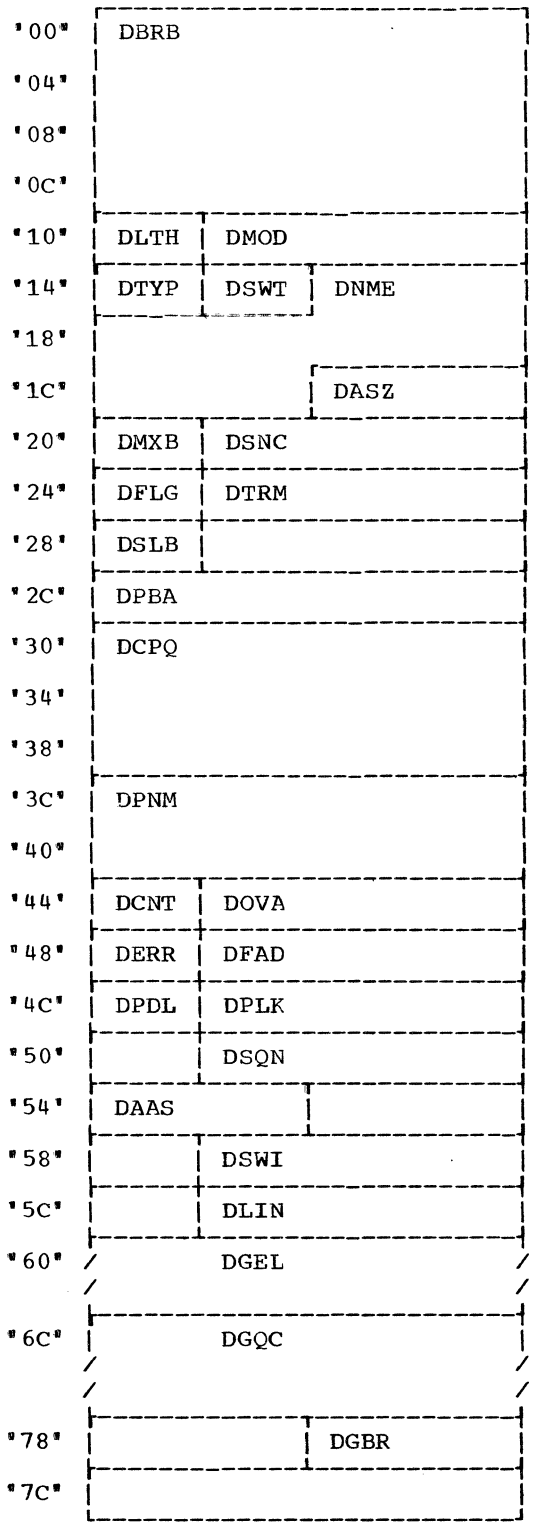


Figure 15. DTF Table Format for Type PQ

Contents:

The format of the PQ type of DTF table is shown in Figure 15. The System Interface Area is the same as that for the DA type of DTF table with the exception of the DMOD field.

Field	Length in Bytes	Description
DBRB '00'	16	Buffer request block used by GET. This field is subdivided as follows: Bytes 1-4=BRB word 1 Bytes 5-8=BRB word 2 Byte 9=TIC command code X'08' Bytes 10-12=Address + 2 of the DASD Process QCB Byte 13=Dummy buffer code X'07' Bytes 14-16=Address of the MS Process Queue LCB
DLTH '10'	1	The number of doublewords in this DTF table.
DMOD '11'	3	The address of the appropriate GET module (IJLQGR, IJLQGS, IJLQGM, IJLQGA, IJLQGB, IJLQGC, or IJLQGD). (SYSTEM INTERFACE AREA)
DASZ '1E'	2	Size of the user-specified work area. SOWA or SOWARU(if WU=A is specified).
DMXB '20'	1	Maximum number of buffers to be chained into this MS Process Queue (DCPQ).
DSNC '21'	3	Address of the user routine to be entered when no messages are available for this queue.
DFLG '24'	1	QTAM flags. Same as in DA type of DTF file.
DTRM '25'	3	Address of the user area that is to receive the source terminal name (IJLQTTID). TRMAD or LINAD (if WU=A is specified).

Field	Length in Bytes	Description
DSLB '28'	1	LCB status.
DSLB+1 '29'	3	Address of the source LCB.
DPBA '2C'	4	Address of the previous buffer used. This field is initially set to the address of DBRB.
DCPQ '30'	12	MS Process QCB.
DPNM '3C'	8	Name of the process program entry in the terminal table in the message control program.
DCNT '44'	1	Count of the characters remaining in the buffer from a previous GET.
DOVA '45'	3	Address of the user routine to be entered when a message unit is larger than the provided work area.
DERR '48'	1	Return code for GET. X'00'=No error X'01'=DTF not opened
DFAD '49'	3	Address of the user routine to be entered when no message is available and the system is in a closedown procedure.
DPDL '4C'	1	Partition Interrupt Key (PIK) for the partition containing the last previous PQ, DQ, or AQ DTF table opened.
DPLK '4D'	3	Address of the last previous PQ, DQ, or AQ DTF table opened.
DSQN '51'	3	Address of sequence number work area. The following fields are generated only if WU= MA, RA, or SA is specified.
DAAS '54'	2	Size of user's audio area (SOWARU).
DSWI '59'	3	Address of the user-provided switch byte (SWITCH).
DLIN '5D'	3	Address of the user area that is to receive the source line name (LINAD).
DGEL '60'	12	High priority element for immediate dispatching.
DGQC '6C'	14	Combined QCB/STCB.
DGBR '78'	4	Entry point to implementation subtask.

DTF TYPE DQ

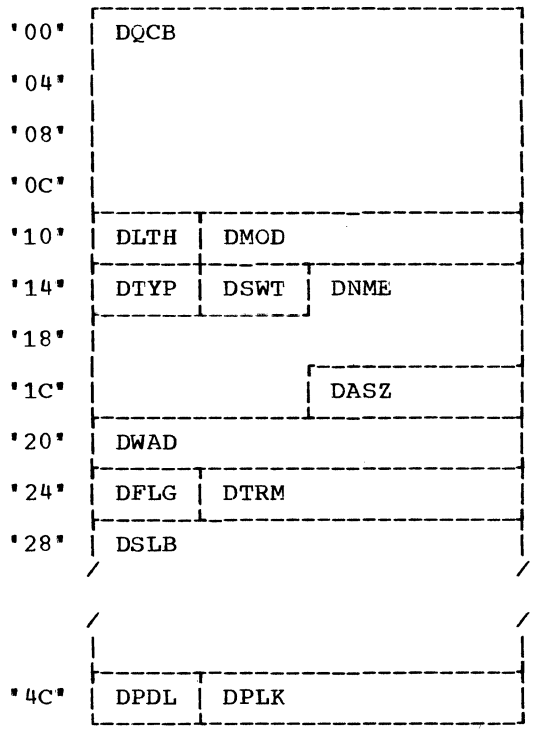


Figure 16. DTF Table Format for Type DQ

Contents:

The format of the DQ type of DTF table is shown in Figure 16. The System Interface Area is the same as that for the DA type of DTF table with the exception of the DMOD field.

Field	Length in Bytes	Description
DQCB '00'	16	The MS Destination QCB. This field is subdivided as follows: Bytes 1-4=QCB word 1 Bytes 5-8=QCB word 2 Byte 9=Always X'00' Bytes 10-12=Address of the first STCB. This field is originally set to IJLQIP8C. Bytes 13-16=QCB word 4
DLTH '10'	1	The number of doublewords in this DTF table.
DMOD '11'	3	Address of the appropriate PUT module (IJLQPR, IJLQPS, or IJLQPM). (SYSTEM INTERFACE AREA)
DASZ '1E'	2	Save area for count of characters to be moved from work area.
DWAD '20'	4	Save area for address of input work area.
DFLG '24'	1	QTAM flags. Same as in DA type of DTF file.
DTRM '25'	3	Address of the user area which contains the destination terminal name.
DSLB '28'	36	MS Destination Queue LCB. The last three bytes of this area contain the address of the DASD Destination or Process QCB for the current destination terminal.
DPDL '4C'	1	Partition Interrupt Key (PIK) for the partition containing the last previous PQ, DQ, or AQ DTF opened.
DPLK '4D'	3	Address of the last previous PQ, DQ, or AQ DTF opened.

DTF TYPE AQ

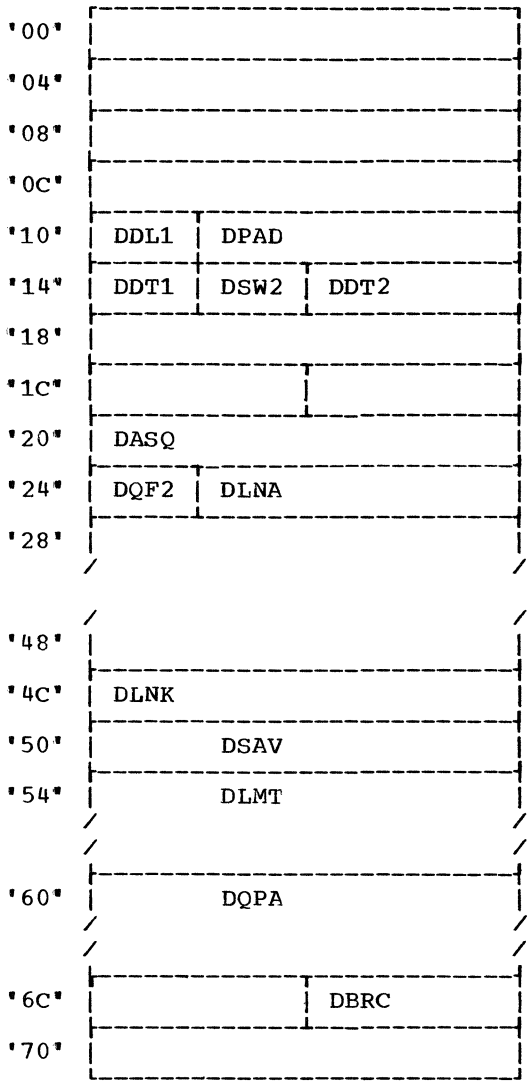


Figure 17. DTF Table Format for Type AQ

Contents:

The format of the AQ type DTF table is shown in Figure 17.

Field	Length in Bytes	Description
DDL1 '10'	1	Number of doublewords in this DTF table.
DPAD '11'	3	Address of the audio PUT module (IJLQPA).
DDT1 '14'	1	DTF type code (X'55').
DSW2 '15'	1	DOS System switches. Refer to DA type (DSWT field).
DDT2 '16'	8	The name of the DTFQT macro that causes generation of this table.
DASQ '20'	4	Address of the Output queue (IJLQAA50).
DQF2 '24'	1	QTAM flags. Refer to DA type (DFLG field).
DLNA '25'	3	Address of the user area that is to receive the destination line name (LINAD).
DLNK '4C'	4	Address of the last PQ, DQ, or AQ type DTF opened in a message processing program
DSAV '50'	4	Parameter for truncated subtask.
DLMT '54'	12	High priority element for immediate posting.
DQPA '60'	14	Combined QCB/STCB.
DBRC '6C'	4	Entry point to implementation subtask.

LINE CONTROL BLOCK (LCB)

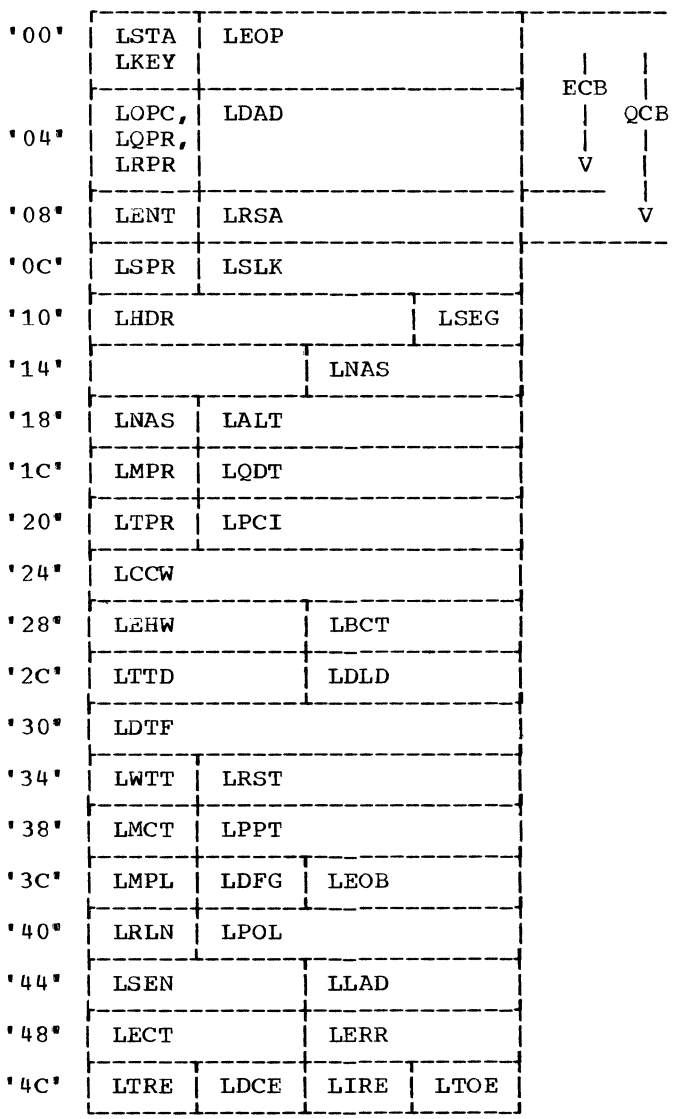
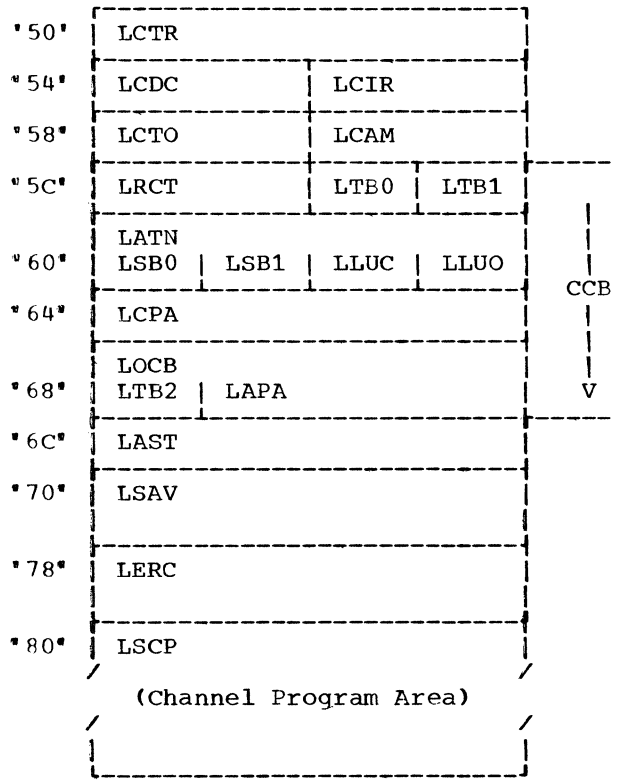


Figure 18. Typical LCB DSECT (IJLQLCB0) (Part 1 of 2)



Note: ALL DSECT names have the four character prefix IJLQ.

Figure 18. Typical LCB DSECT (IJLQLCB0) (Part 2 of 2)

Contents:

Field	Length in Bytes	Description
LSTA '00'	1	The status of the line: X'00'=inactive X'01'=line free X'02'=partial message in queue X'04'=send X'08'=receive X'10'=initiate X'20'=converse X'40'=recall
LKEY '00'	1	Status of communication line queue represented by this LCB. Key values have the same meaning as for other QCB's.
LEOP '01'	3	If receiving, the return address from the Route macro to the next in-line instruction in the user's LPS. If sending, address of the LCB for the originating line.
LOPC '04'	1	Read/Write operation code for the current segment of current message.
LQPR '04'	1	QCB's Ready Queue priority.
LRPR '04'	1	Response message priority.
LDAD '05'	3	Disk relative record number of last correctly transmitted segment of current message. When LCB appears on the Ready Queue, this field contains the address of the next item on the Ready Queue.
LENT '08'	1	Entry code (X'12') for Receive scheduler subtask.
LRSA '09'	3	Address of the first STCB in the STCB chain of the communication line

Field	Length in Bytes	Description
		queue. If the line is to be used for input, this field initially contains the address of the Receive Scheduler STCB (address of this field minus 1). If the line is for output only, it contains the address of the Qdispatch STCB (IJLQIP8D).
LSPR '0C'	1	Priority of the Receive Scheduler subtask: 1 = lower than Send Scheduler priority 2 = equal to Send Scheduler priority 4 = higher than Send Scheduler priority
LSLK '0D'	3	Link address field in Receive Scheduler STCB.
LHDR '10'	3	Disk relative record number of the current message header.
LSEG '13'	3	Disk relative record number of the current message segment.
LNAS '16'	3	Address of the first segment of the last message received on this line which is to be transmitted.
LALT '19'	3	Address of the LCB for another source line currently sending to the same destination.
LMPR '19'	1	Priority of the current message.
LQDT '1C'	3	Address of the QCB for the destination terminals. Valid only when receiving.
LTPR '20'	1	Temporary storage for priority of outgoing message.
LPCI '21'	3	Address of last CCW for which a PCI was received.
LCCW '24'	4	Address of last BRB/CCW for which a buffer was assigned.

Field	Length in Bytes	Description
LEHW '28'	2	Communication line error halfword.
LBCT '2A'	2	If receiving, contains the next expected sequence-in number (IJLQTSIN field in terminal table entry). If not receiving, this field contains the time of the requested interrupt as established and used by the End of Poll Time Delay routine.
LTTD '2C'	2	Offset from beginning of terminal table to the terminal table entry for the source terminal (if receiving) or the destination terminal (if sending).
LDLD '2E'	2	Offset (less 18) for current entry in terminal list field of distribution-list entry in terminal table.
LDTF '30'	4	Address of the DTF table for this LCB.
LWTT '34'	1	Flags for World Trade terminals: X'01' X'02' X'04' Not used X'08' X'10'=WRU flag X'20'=Working flag X'40'=EOT flag X'80'=Halt I/O flag
LRST '35'	3	Address of restart CCW.
LMCT '38'	1	Count of the messages received from terminal.
LPPT '39'	3	Address of current active entry in the polling list for the line.
LMPL '3C'	1	Scan pointer offset of the next destination code in message header.

Field	Length in Bytes	Description
LDFG '3D'	1	Flag field for dial line: X'01'=destination terminal is not currently connected to a line and an outgoing message is waiting in queue. X'02'=line has been assigned for dial or line is connected to terminal. X'04'=polling or addressing error has occurred; this is a signal to disconnect the line.
LEOB '3E'	2	Address of EOB character relative to the address of the last correctly transmitted segment of current message.
LRLN '40'	1	Relative line number for this LCB.
LPOL '41'	3	Address of the polling list for this line.
LSEN '44'	2	Sense data from sense command.
LLAD '46'	2	Line address (used for HIO in Send Scheduler).
LECT '48'	2	Character count in current incoming message, used byBREAKOFF routine (IJLQBO). The next 18 bytes are used by the Error Recovery Procedures in the Line Appendage module (IJLQLA)
LERR '4A'	2	Retry count of CCW error for ERP routine in Line Appendage ERP routine in Line Appendage module.
LTRE '4C'	1	Threshold line error counter=number of transmissions.
LDCE '4D'	1	Threshold line error counter=number of data checks.

Field	Length in Bytes	Description
LIRE '4E'	1	Threshold line error counter=number of intervention required errors.
LTOE '4F'	1	Threshold line error counter=number of nontext timeouts.
LCTR '50'	4	Cumulative line error counter=total number of transmissions.
LCDC '54'	2	Cumulative line error counter=total number of data checks.
LCIR '56'	2	Cumulative line error counter=total number of intervention required errors.
LCTO '58'	2	Cumulative line error counter=total number of nontext timeouts.
LCAM '5A'	2	Flag field for ERP routines in Line Appendage module.
		The next 16 bytes are the CCB data incorporated in the LCB
LRCT '5C'	2	Residual count
LTB0 '5E'	1	Transmission byte 0.
LTB1 '5F'	1	Transmission byte 1.
LATN '60'	4	A full word containing the Attention QCB for IBM 2260 Local line group. This field is initially set to the address of the dummy last element, IJLQIP5F.
LSB0 '60'	1	CSW status byte 0.
LSB1 '61'	1	CSW status byte 1.
LLUC '62'	1	Logical unit class; always X'01'
LLUO '63'	1	Logical unit order (X'nnn')
LCPA '64'	4	Address of channel program area (first CCW).
LOCB '68'	4	A full word containing the address of the CCB for the IBM 2260 Local terminal.
LTB2 '68'	1	Appendage indicator; set to X'40'.
LAPA '69'	3	Address of Line Appendage routine (IJLQIP40).
LAST '6C'	4	Address of last CCW executed.
LSAV '70'	8	Save area for failing CCW
LERC '78'	8	Error recovery CCW
LSCP '80'	8n	Start of channel program area; size of area is dependent on the terminal type on the line. The last three bytes of this area are used by OBR/SDR when this option is included in the system.

AUDIO LINE CONTROL BLOCK (ALCB)

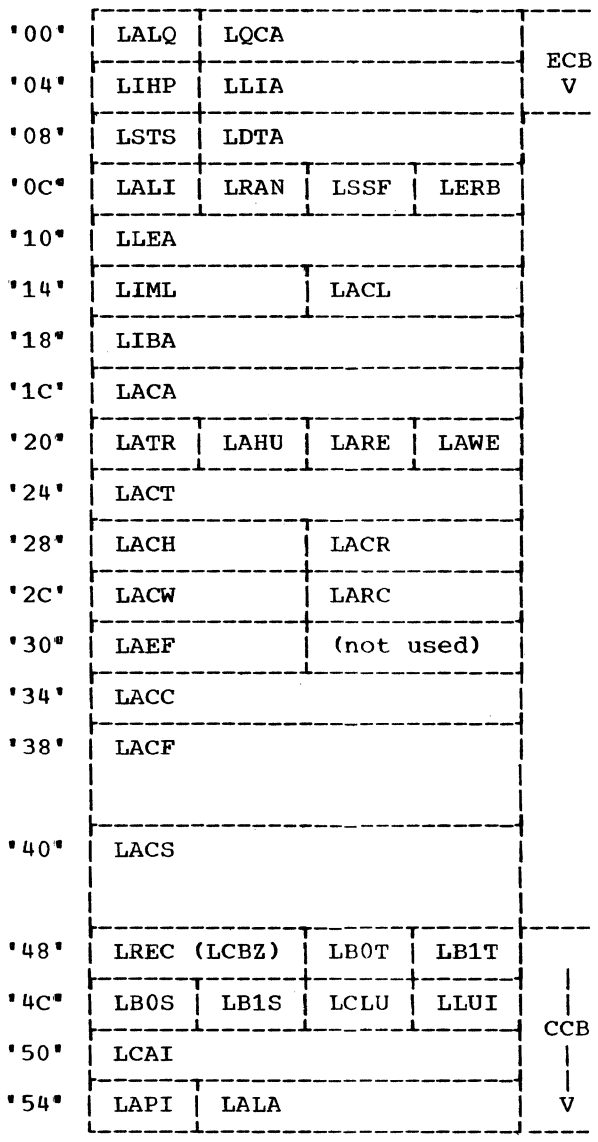
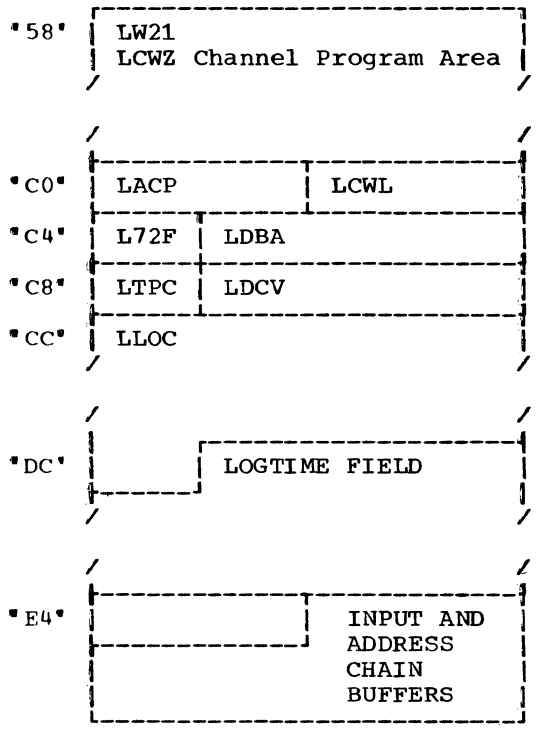


Figure 19. Typical ALCB DSECT (IJLQLAB0) (Part 1 of 2)

7772 ALCB



7770 ALCB

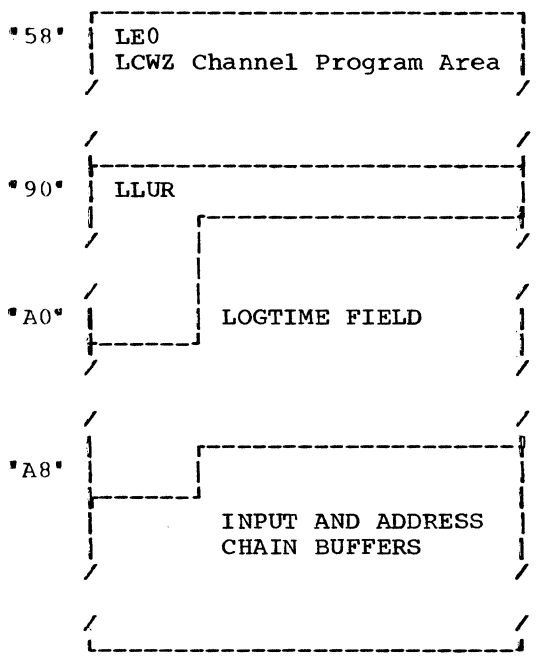


Figure 19. Typical ALCB DSECT (IJLQLAB0) (Part 2 of 2)

Contents:

Field	Length in Bytes	Description
LALQ '00'	1	ECB key always X'00'
LQCA '01'	3	Address of the QCB associated with the ALCB by the QPOST function.
LIHP '04'	1	ALCB priority. X'FD' if the ALCB is processed in the message control program. X'FC' if the ALCB is processed in the message processing program.
LLIA '05'	3	When ALCB appears on the Ready Queue, this field contains the address of the next item on the Ready Queue.
LSTS '08'	1	ALCB status, used by the audio programming to determine the next action to be taken on the audio line. X'01'=Enable status. The ENABLE CCW was issued or is ready to be issued. X'02'=Read status. A read operation was completed. X'04'=Disable status. A disable operation was completed. X'08'=Write status. A write operation is in progress. X"10"=7772 initial write status. An initial write is in progress. Only used by the 7772 ALCB. X'20'=Stop status. A STOPARU macro instruction was issued for this line. X"40"=PUT status. A PUT was issued for this line and the repeat function may take place until the line is disabled. X'80'=GET status. A GET was issued for this line, and one and only one PUT must be issued on this line.
LDTA '09'	3	Address of the DTF table for this ALCB.
LALI '0C'	1	ALCB indicator. Always X'FF'. This indicator distinguishes audio and nonaudio elements.
LRAN "0D"	1	Relative line number within the line group.
LSSF '0E'	1	Start/Stop flags. X"01"=The channel program is ready to be executed and an EXCP must be issued. X'20'=The halt function is requested by a CLOSE or CLOSEMC macro instruction. X"40"=The start function is requested by a STARTARU macro instruction. X"80"=The stop function requested by a STOPARU macro instruction was performed by the audio line appendage.
LERB "0F"	1	Audio error byte.
LLEA '10'	4	Address of the line table entry. The high-order byte is used to store the sense data from the sense command.
LIML '14'	2	Input message length, not including the EOT character if any.

Field	Length in Bytes	Description
LACL '16'	2	Address chain length.
LIBA '18'	4	Address of the input area located in the ALCB.
LACA '1C'	4	Address of the address chain area located in the ALCB after the input area.
LATR '20'	1	Number of transmissions.
LAHU '21'	1	Number of hang-up operations.
LARE '22'	1	Number of data-check errors on read operations.
LAWE '23'	1	Number of data-check errors on write operations.
LACT '24'	4	Cumulative counter for the total number of transmissions on the line.
LACH '28'	2	Total number of hang-up operations.
LACR '2A'	2	Total number of data-check errors on read operations.
LACW '2C'	2	Total number of data-check errors on write operations.
LARC '2E'	2	Retry counter of CCW errors for ERPs.
LAEF '30'	2	Flag field used by ERP's.
LACC '34'	4	Address of the last-executed CCW.
LACF '38'	8	Save area used to store the failing CCW.
LACS '40'	8	Error recovery CCW used to issue the Sense Command.
		The next 16 bytes are the CCB data incorporated in the ALCB.
LREC '48'	2	Residual count.
LB0T '4A'	1	Transmission byte 0.
LBIT '4B'	1	Transmission byte 1.
LB0S '4C'	1	CSW status byte 0.
LB1S '4D'	1	CSW status byte 1.
LCLU '4E'	1	Logical unit class; always X'01'.
LLUI '4F'	1	Logical unit order in class.
LCAI '50'	4	Address of the first CCW to be executed in the channel program area.
LAPI '54'	1	Appendage indicator; set to X'40'
LALA '55'	3	Address of the Audio Line Appendage routine (IJLQAA).
		The following bytes end a 7772 ALCB:
LCWZ '58'	104	Channel program area for 13 CCWs.
LACP 'C0'	2	Pointer in the 7772 address chain.
LCWL 'C2'	2	Length of the current DCV word to be sent.
L72F 'C4'	1	Send flags, used during the sending of a 7772 audio answer. X'01'=The first DCV word is to be sent. X'02'=The last DCV word is to be sent. X'04'=The previous DCV word may be followed by a programmed pause. X'08'=The DCV word representation has been found. X'10'=The first DCV word is searched.

Field	Length in Bytes	Description
		X'20'=The address chain ends abnormally, and a DISABLE command must be issued. X'40'=Used to determine which sequence of write CCWs is to be updated for the sending of the next DCV word. X'80'=Used to determine which part of the DCV buffer must contain the next DCV word retrieved from the 7772 DCV vocabulary file.
LDBA 'C5'	3	Address of the DCV buffer associated with the ALCB to send the audio answer.
LTPC 'C8'	1	Temporary pause counter.
LDCV 'C9'	3	Address of the current DCV word to be sent.
LLOC 'CC'	26	Optional field, used to store the logical order in class, the input message length, the date and time stamping in front of the input message. The 7772 ALCB ends with the input area immediately followed by the address chain area. The following bytes end a 7770 ALCB:
LCWZ '58'	56	Channel program area for 7 CCWs.
LLUR '90'	26	Optional field, used to store the logical order in class, the input message length, the date and time stamping in front of the input message (included in the LOGTIME field. The 7770 ALCB ends with the input area immediately followed by the address chain area.

QTAM VECTOR TABLE

The QTAM Vector Table is a 22-word table of addresses. The table contains the addresses of all areas containing control information necessary to effect the cross-partition communication required in DOS/QTAM. A four-byte field is reserved for QTAM in the DOS Communication Region (displacement X'80' from the beginning of the Communication Region). The address of the QTAM Vector Table is placed into this field at Open time so that QTAM routines requiring information in the Vector Table can access it. The fields in the QTAM Vector Table follow.

Word	Field Name	Description
1	VECTTBL	Address of the terminal table
2	VECIP1AD	Address of the Implementation module
3	VECSVCAD	Address of the QTAM SVC/Subtask Control routine in the supervisor.
4	VECLDTF	Address of the DTF table for the last QTAM file opened in the message control program.
5	VECIP78	Address of the Return Buffer QCB
6	VECEXTNT	Address of disk extent information (IP1EXTNT)
7	VECBUFSZ	Address of the halfword containing the buffer size minus 8
8	VECDADTF	Address of the DTF table for the first opened message control file (DA, AV, or LG).
9	VECIP71	Address of the Active-BRB QCB
10	VECIP5F	Address of the dummy end of chain ECB (IJLQIP5F)
11	VECIP8C	Address of the Queue Insert by Priority STCB
12	VECPDQLK	Address of the DTF table for the last QTAM file opened in a message processing program
13	VECQMOVR	Address of the Qmove QCB

14	VECMRSW	Address (minus one) of the master receive switch
15	VECCPFLG	Flags for the checkpoint records file
15 + 1 byte	VECCPDTF	Address of the DTF table for the checkpoint records file
16	VECIP72	Address of the Available Buffer QCB
17	VECIP73	Address of the Disk I/O QCB.
18	VECIP77	Address of the LPS QCB
19	VECDRRN	Next relative record number to be used on the direct access file.
20	VECQIP26	Address of address for Post Send and Post Receive.
21	VECLTBL	Address of the audio line table.
22	VECARUSQ	Address of the ARU-Send QCB.

SPECIAL CONTROL BLOCK FORMS

COMBINED QCB/STCB

The pattern of unused bytes in the QCB and the truncated STCB is such that these blocks are capable of being combined as may be seen from the general forms.

QCB

'00'	Key	element chain pointer
'04'	priority	link address
'08'		STCB chain pointer

STCB (truncated)

'00'	entry code	
'04'	priority	link address

To conserve storage, the QCB and truncated STCB frequently are combined. The general form of these control blocks when combined is:

Combined QCB/STCB

'00'	Key	element chain pointer
'04'	QCB Priority	QCB link address
'08'	Entry code	STCB chain pointer*
'0C'	STCB Priority	STCB link address

* address of this field minus 1.

QCB FOR DASD DESTINATION QUEUE

An extended QCB of 32 bytes is generated in the problem program on expansion of a TERM macro instruction. A QCB is generated either for each line or for each terminal on the line, depending on the type of queueing specified in the TERM macro instruction. Typical DSECT:

'00'	QKEY	QNRA
'04'		QNNT
'08'	QENT	QSTC
'0C'	QSPR	QSTL
'10'	QRLN	QDTF
'14'	QMCT	QNWA
'18'	(QNWA)	QLCB
'1C'	QLQF	QBAK

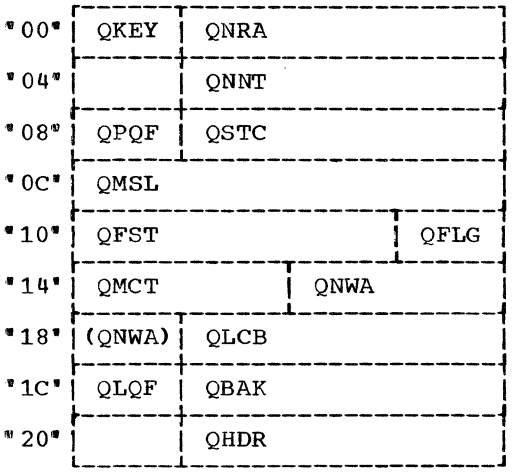
Note: All DSECT names have the IJLQ prefix.

Contents:

Field	Length in Bytes	Description
QKEY '00'	1	A dummy queue status byte, set so that the QCB appears to be on the Ready Queue and is therefore never actually put on it.
QRNA '01'	3	The relative record number of the next message header to be read from this DASD queue.
QNNT '05'	3	The relative record number of the first segment to be normally transmitted after a restart. The preceding segments are transmitted even though the 'previously serviced' flag is set in the buffer prefix.
QENT '08'	1	The entry code (X'84') for the Send Scheduler subtask.
QSTC '09'	3	The address of the first STCB on the STCB chain for the queue. This field initially contains the address of the STCB for the Send Scheduler subtask (address of this field minus 1).
QSPR '0C'	1	The priority (FL1'2') for the Send Scheduler subtask.
QSTL '0D'	3	The address of the next STCB in the STCB chain. This field is initially set to the address of the STCB for the DASD Destination subtask.
QRLN '10'	1	The relative line number of the first line on which sending is attempted for messages in this queue.
QDTF '11'	3	The address of the DTF table associated with this queue.
QMCT '14'	2	The number of unprocessed header segments in this queue.
QNWA '16'	3	The disk relative record number for writing the next message segment.
QLCB '19'	3	The address plus 1 of the first LCB on the chain of LCBs currently sending to this destination. The dummy last element in the chain is this QCB.
QLQF '1C'	1	Always set to X'00', this field distinguishes this QCB from an LCB
QBAK '1D'	3	The disk relative record number of the last previous header placed into this queue. If there are none in the queue, this field contains zero.

QCB FOR DASD PROCESS QUEUE

An extended QCB of 32 bytes is generated in the problem program on expansion of each PROCESS macro instruction. Typical DSECT:



Note: All DSECT names have the IJLQ prefix.

Contents:

Field	Length in Bytes	Description
QKEY '00'	1	A dummy queue status byte, set so that the QLB appears to be on the Ready Queue and is therefore never actually put on it.
QNRA '01'	3	The relative record number of the next message header to be read from this DASD queue.
QNNT '05'	3	The relative record number of the first segment to be normally transmitted after a restart. The preceding segments are transmitted even though the 'previously serviced' flag is set in the buffer prefix.
QPQF '08'	1	Always set to X'00', this field distinguishes this QCB from the DASD Destination QCB (X'84') and from distribution list (X'02').
QSTC '09'	3	The address of the Get Scheduler STCB.
QMSL '0C'	4	The address of the LCB for the Main Storage Process Queue when the Main Storage Process Queue is open. If it is not open, this field contains zeros.
QFST '10'	3	The disk relative record number of the first header placed into the queue.
QFLG '13'	1	DASD Process Queue flags. Bit 7 (X'02') is set on if the Expedite option is specified for this queue; otherwise it is set off.
QMCT '14'	2	The number of unprocessed header segments in this queue.
QNWA '16'	3	The disk relative record number for writing the next message segment.
QLCB '19'	3	The address plus 1 of the first LCB on the chain of LCBs with messages to be placed on this DASD queue. The dummy last element in the chain is this QCB.
QLQF '1C'	1	Always set to X'00', this field distinguishes this QCB from an LCB.
QBAK '1D'	3	The disk relative record number of the last previous header placed into this queue. If there are none in the queue, this field contains zero.
QHDR '21'	3	The disk relative record number of the next header to be placed into this queue. If there are none to be placed into the queue, this field contains zero.

QCB FOR 7772 DCV BUFFER QUEUE

An extended QCB is generated in the problem program on expansion of a BUFARU macro instruction. The typical DSECT (IJLQBAB0) also includes one DCV buffer element.

"00"	BKQC	BAFE
"04"	BNIR	
"08"	always X'00'	BIFO
"0C"	BBUN	BDEL
"10"	BBUS	BBDL
"14"	not used	

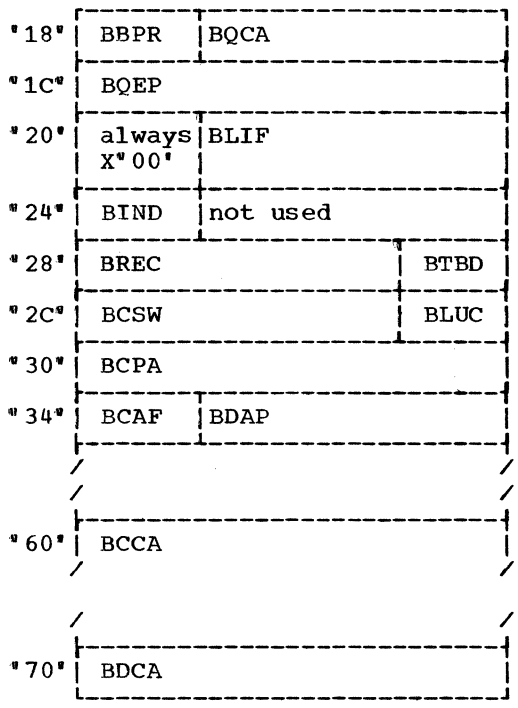
Note: All DSECT names have the IJLQ prefix.

Contents:

Field	Length in Bytes	Description
BKQC '00'	1	QCB key. Same as for other QCBs.
BAFE '01'	3	Address of the first available DCV buffer. When no more buffers are available, this field contains the address of the dummy element (IJLQIP5F).
BNIR '04'	4	The high-order byte contains the QCB priority always set to X'FD'. It is followed by the address of the next item on the Ready Queue.
BIFO '09'	3	The address of the Queue Insert STCB(IJLQIP8B). When at least one ALCB is waiting for an available DCV buffer, this field contains the address of the DCV buffer STCB (IJLQAD41).
BBUN '0C'	1	The number of DCV buffers in the DCV buffer pool defined by the BUFARU macro instruction.
BDEL '0D'	3	Address of the first ALCB waiting for an available DCV buffer. These ALCBs constitute an element chain ended with the dummy element (IJLQIP5F).
BBUS '10'	2	Size of a DCV buffer including the control information.
BBDL '12'	2	Length of the data area required by the user for each DCV buffer.

IBM 7772 DCV BUFFER ELEMENT

Each DCV buffer generated in the problem program on expansion of a BUFARU macro instruction is basically an element control block. Such an element is shown in the IJLQBAB0 DSECT following the 7772 DCV buffer QCB.



Contents:

Field	Length in Bytes	Description
BBPR '18'	1	Element key, always X'00'.
BQCA '19'	3	If the DCV buffer requests a disk read operation, this field contains the address of the LPS QCB (IJLQIP77). If the DCV buffer is free, this field contains the address of the corresponding DCV buffer QCB.
BQEP '1C'	4	The high-order byte contains the ECB priority always set to X'FD'. It is followed by the address of the next item on the Ready Queue.
UNUSED '20'	1	
BLIF '21'	3	The address of the ALCB associated with this DCV buffer.
BIND '24'	1	DCV buffer indicator, always X'FE'

Field	Length in Bytes	Description
		The next seven fields represent a CCB incorporated in the DCV buffer:
BREC '28'	2	Residual count.
BTBD '2A'	2	Transmission bytes.
BCSW '2C'	2	CSW status bytes.
BLUC '2E'	2	Logical unit class, always X'01', and logical unit order in class
BCPA '30'	4	Address of the channel program area (first CCW).
BCAF '34'	1	Appendage indicator, set to X'40'.
BDAP '35'	3	Address of the 7772 Disk Appendage routine (IJLQAD).
BSEE '38'	40	Channel program area for 5 CCWs.
BCCA '60'	16	Areas for disk address conversion and count.
BDCA '70'		DCV buffer area where the DCV words are to be read from the 7772 DCV vocabulary

BUFFER REQUEST BLOCK

The Buffer Request Block is basically an element control block with the element control block characteristics as previously outlined. The BRB, however, takes several different forms during its processing cycle. The significant fields of the BRB when it is in the element control block chain of the Inactive-BRB queue control block are shown here. This is also the form in which the BRB is generated on expansion of the BUFFER macro instruction.

BRB on Inactive-BRB Queue

'00'	Key (0)	QCB address
'04'	Priority	Link address
'08'		
'0C'		

The BRB Ring routine removes BRBs from the Inactive-BRB queue and forms the BRB ring. The following illustration shows the BRB after it has been processed by the BRB Ring Routine.

BRB before assignment of next-segment address (in BRB ring)

'00'	Key (0)	QCB address
'04'	Priority	Link address
'08'	TIC command	Address of next BRB in ring
'0C'	status (0)	LCB address

The key field is zero. All elements have a key of zero.

The QCB address is variable. For the first BRB in the 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, this field is insignificant.

The priority field is set to a value of X'EC' for the first BRB in the ring to be used for a Receive operation. This field is normally zero for all other BRBs.

The TIC command is a transfer-in-channel command, and the following three bytes point to the next BRB in the ring. For the first BRB in the ring only, the TIC address will point to the actual address of the next BRB which begins on a doubleword boundary. For all other BRBs in the ring, the last two bits will be set to one (see BRB Status Codes) and represents the BRB address (always on a double-word boundary) plus the BRB-idle flag.

The status field is zero, indicating no next-segment address has been assigned to this BRB.

The LCB address is the address of the Line Control Block over which the send or receive operation will occur.

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 this BRB is now associated with. The assigning of the next-segment address is indicated by changing the value of the status field to 9.

BRB after assignment of next-segment address has changed starting at the twelfth byte to:

'0C'	Status (9)	relative record number for next segment

The next illustration 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 CCW, and are followed (in the third fullword) by the previously initialized TIC. This means 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.

BRB/CCW initialized for Direct Access Read or Write

'00'	command code	data address	Stand-ard
'04'	flags	0 data count	CCW
'08'	TIC command	address of next BRB in ring	
'0C'	status	LCB address or relative record number for next segment	

In the BRB/CCW, as in any other forms of the BRB except that appropriate to the Inactive-BRB queue, the fourth fullword may contain either status = 0 and an LCB address, or status = 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 replaced is selected according to its position in the ring, without reference to the queue (if any) upon which it appears.

The typical DSECT for a BRB is:

RKEY RCC	RQCB or RCDA	
RPRI RFLG	RLKA or RCT	
RTIC	RCH	RSTC
RSTA	RLCB or RNSA	

Note: All DSECT names have the prefix IJLQ.

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 the RSTC field, 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 receive the last segment of a message, makes the address portion of a CCW containing a TIC command valid. (Refer to Line Appendage routine for additional information on invalid TIC address.) This code typically is set when a buffer has been assigned to the next BRB in the ring.

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; that is, it is not in any queue. This code commonly indicates that no buffer has been assigned, or that this is the first BRB in the ring and no buffer has been assigned to the next BRB. This code is also 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.

PAUSE BRB/CCW

Write Command Code	Data Address
Flags	Count of characters up to and including the special character
Write Code	Address of the Idles address ha
Flags	byte count of idles
TIC Command Code	Address of the next BRB in the ring
0	Address of queue for PAUSE BRB/CCWs (Additional-CCW queue)

The PAUSE BRB/CCW is a special-purpose BRB used to schedule and contain channel commands for the transmission of idle characters. It is inserted or linked into the ring of BRBs by the Pause routine.

ELEMENT CONTROL BLOCK--IJLQIP5F

Form:

IJLQIP5F	key = 0	QCB address
	pri = 255	link address = IJLQIP5F

The IJLQIP5F element control block is used as a dummy last element on the element chain of several queues and as the last item on the Ready Queue. It signals the end of a chain. It is never used as an "available" element.

COMBINED CCB/ECB FOR IBM 2260 LOCAL

This CCB/ECB is generated in the device access area in the terminal table entry for an IBM 2260 LOCAL OR 1053 Local. The value specified in the ACLOC operand in the DTFQT macro instruction determines the offset to the beginning of this control block. (See Figure 20.)

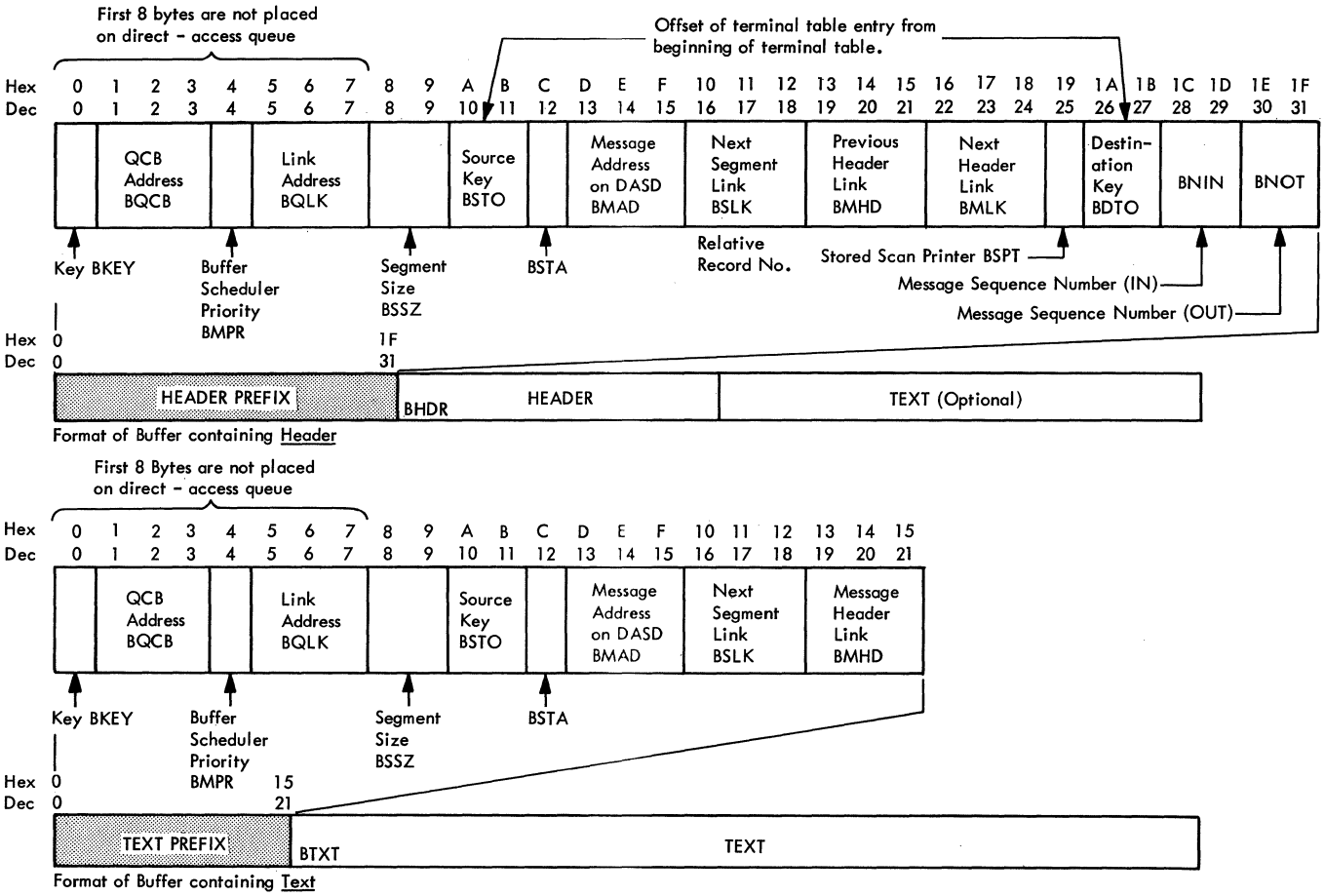
For an IBM 2260 Local, transmission byte 0 is assembled as X'04' (device end posting bit). This bit is used in conjunction with channel end (alone) in CSW status, to cause the Supervisor to leave the CCB on the channel queue. (This does not apply to the IBM 1053 Local.)

The typical DSECT for a CCB/ECB is shown at the right.

'00'	MRCT		MTBO	MTB1
'04'	MSB0	MSB1	MLUC	MLUO
'08'	MCPA			
'0C'	MTB2	MALA		
'10'	MKEY	MTTA		
'14'	MPRI	MNXT		

'00'	Residual Count		Transmission Byte 0	Transmission Byte 1
'04'	CSW Status Byte 0	CSW Status Byte 1	Logical Unit Class (always 1)	Logical Unit Order (nnn from SYSnnn)
'08	Address of first CCW in channel program			
'0C'	Appendage Indicator	Address of IBM 2260 Local Appendage		
'10'	(Reserved)	Address of Terminal Table Entry		
'14'	ECB Priority for Linking into Attention Queue	Address of Next CCB/ECB in Attention Queue		

Figure 20. Combined CCB/ECB for the IBM 2260 Local



Note: All DSECT names have the prefix IJLQ.

Figure 21. Buffer Prefix Formats (Part 1 of 3)

Field	Description	Initialized by:
BKEY '00'	The ECB key when the buffer appears on the Ready queue; always zero	Assembler
BQCB '01'	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
BMPR '04'	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
BQLK '05'	A pointer to the next item in the chain in which the buffer appears.	Numerous routines
BSSZ '08'	Segment size (includes buffer prefix minus 8).	Buffer-BRB, Line appendage, Put
BSTO '0A'	Relative address in terminal table of entry for source terminal.	Source, Interim LPS, Line appendage
BSTA '0C'	Used to indicate the status of the message segment contained in the buffer. The significance of the bits in this field is as follows:	
8	Bit 0: If 1, do not send or process message.	Cancel Message
4	Bit 1: If 1, duplicate copy of header.	Recall
2	Bit 2: If 1, an EOB character is present in some position in the buffer other than the last.	Line Appendage
1	Bit 3: If 1, the message was previously serviced or sent.	Disk-End appendage
8	Bit 4: Not used.	
4	Bit 5: If 1, this message was sent with priority.	Disk-End appendage
2-1	Bit 6-7 00 = header segment (not last segment) 01 = text segment (not last segment) 10 = header segment (last segment) 11 = text segment (last segment)	Activate, Line appendage, LPS Control, Put
BMAD '0D'	When in main storage, the address of the LCB to which the buffer is assigned. When on the disk, the relative record number of the segment.	Buffer-BRB DASD Destination

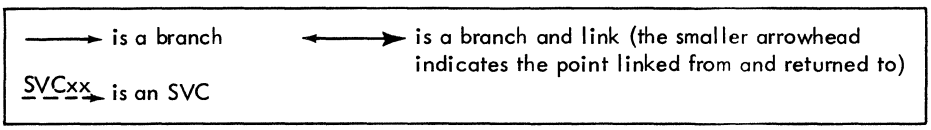
Figure 21. Buffer Prefix Formats (Part 2 of 3)

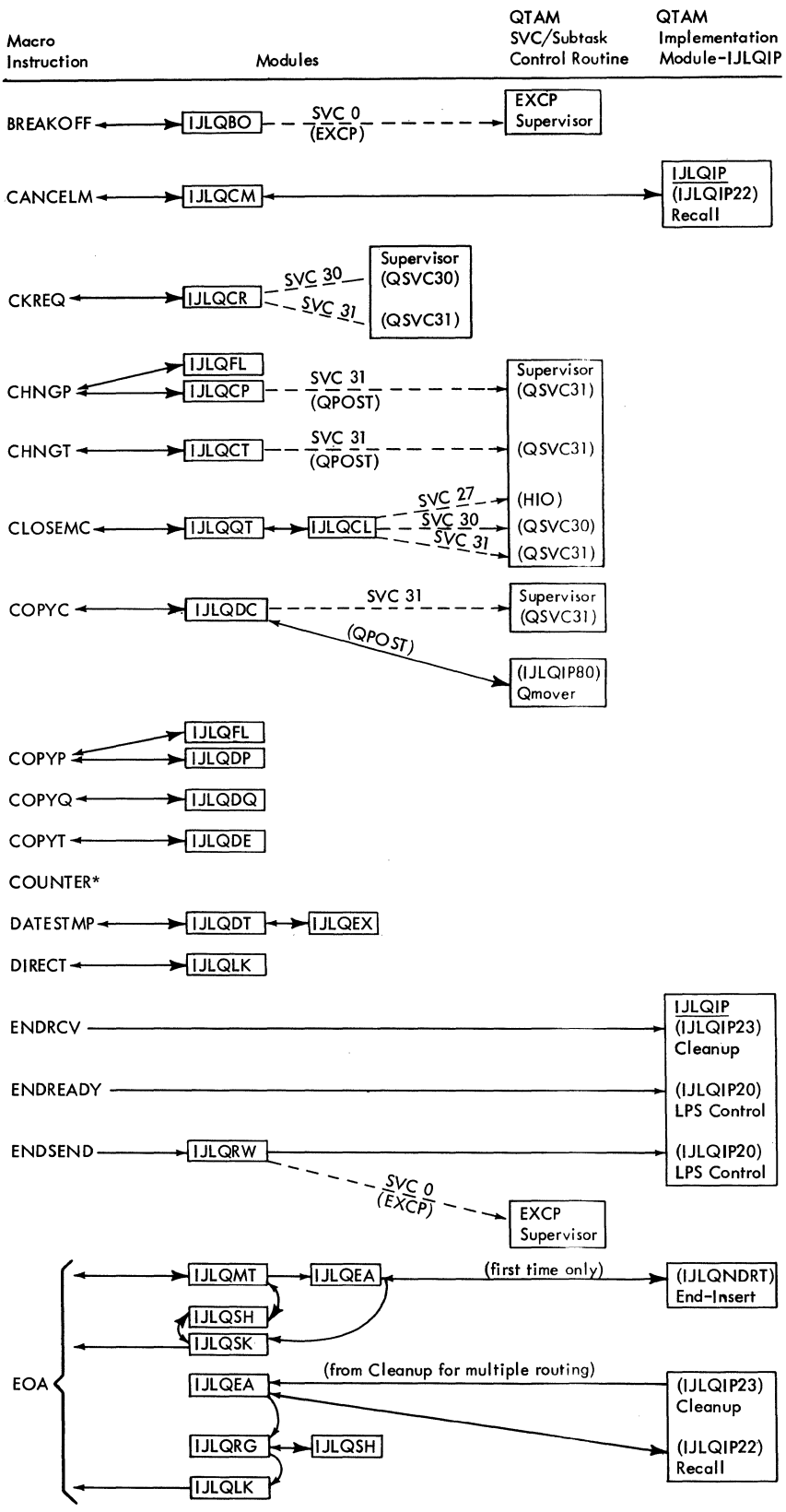
Field	Description	Initialized by:
BSLK '10'	Relative record number of the next segment in this message.	DASD Destination
BMHD '13'	For a header segment, the relative record number of the previous header segment in this queue. For a text segment, the relative record number of the header segment of this message.	DASD Destination
BTXT '15'	Start of message data for a text segment. The remaining fields in the prefix apply only to a header segment.	
BMLK '16'	The relative record number of the next header segment in this queue.	DASD Destination
BSPT '19'	Stored scan pointer; indicates the relative position within the buffer where scanning is to begin or resume.	Activate, Put, Cleanup, EOB, OR EOBLC
BDTC '1A'	Relative address in terminal table of entry for destination terminal.	Lookup or Put
BNIN '1C'	Sequence-in number assigned to message.	Sequence Number In
BNOT '1E'	Sequence-out number assigned to message.	Disk-End appendage
BHDR '1F'	Start of message data for header segment.	

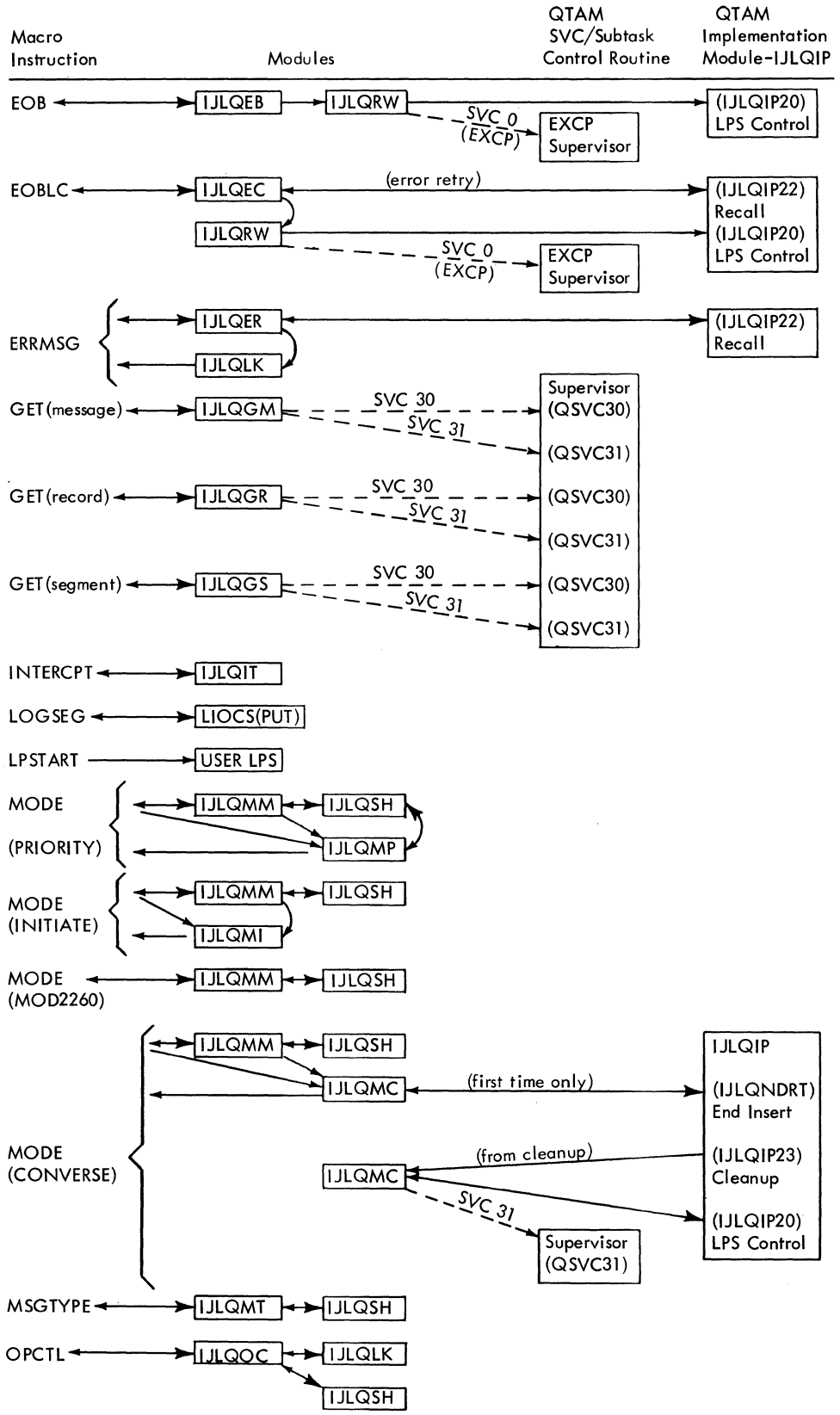
Figure 21. Buffer Prefix Formats (Part 3 of 3)

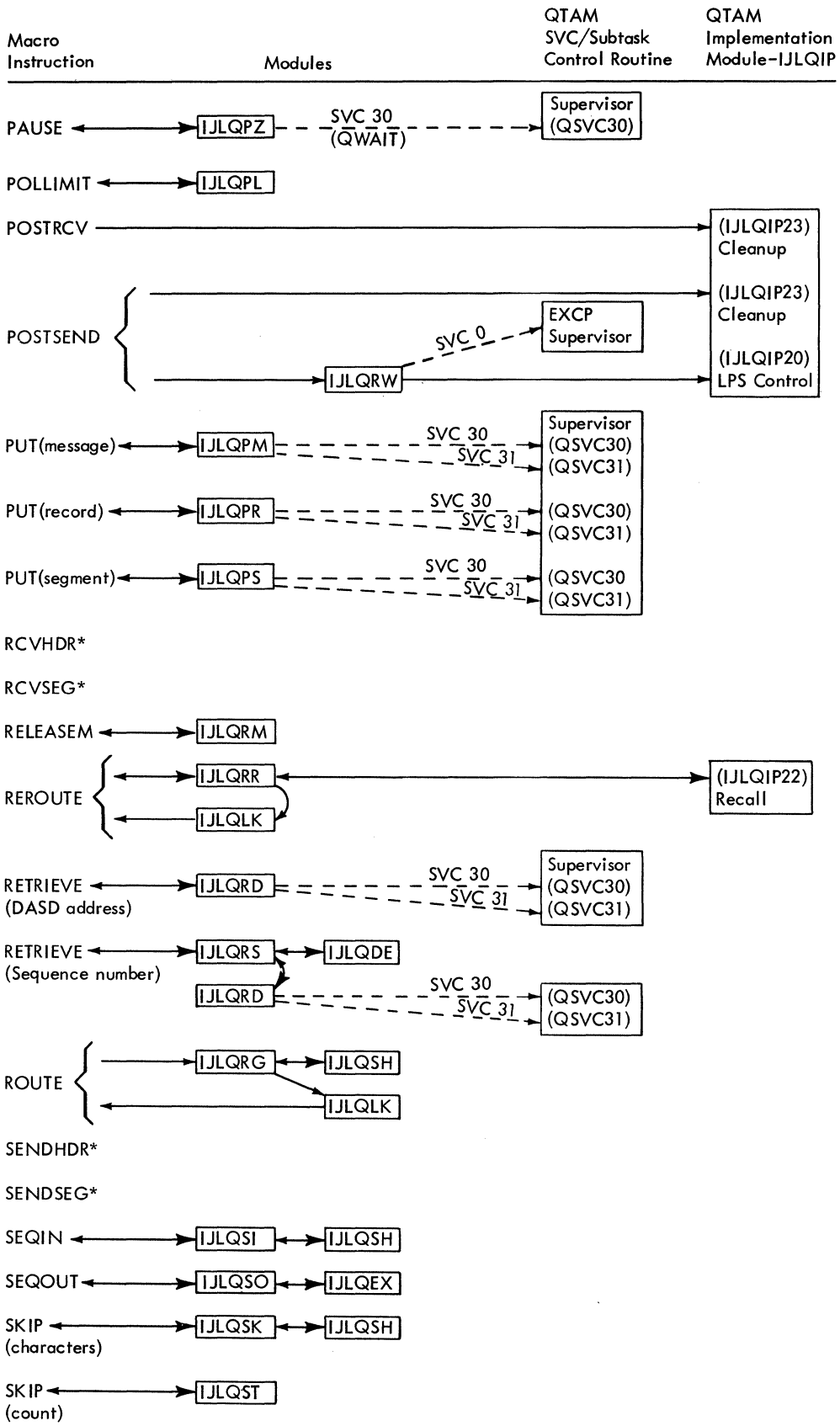
This chart depicts the linkages between macro expansions and the modules they call, for each of the LPS, system-status modifying, GET, and PUT macro instructions, with the exception of five of the LPS delimiter macro instructions. These five branch directly to the QTAM Implementation module, rather than to macro-called modules.

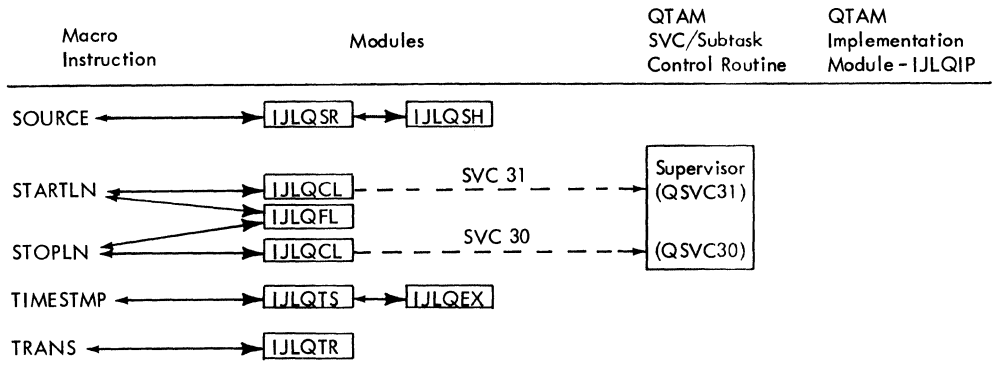
The entry point of each module is the same as the module name except where it is shown in parentheses below the module name. Types of linkages are as follows:




Macro Instruction
Modules
QTAM SVC/Subtask Control Routine
QTAM Implementation Module-IJLQIP
BREAKOFF
IJLQBO
SVC 0 (EXCP)
EXCP Supervisor
CANCELM
IJLQCM
IJLQIP (IJLQIP22) Recall
CKREQ
IJLQCR
SVC 30
SVC 31
Supervisor (QSVC30)
(QSVC31)
CHNGP
IJLQFL
IJLQCP
SVC 31 (QPOST)
Supervisor (QSVC31)
CHNGT
IJLQCT
SVC 31 (QPOST)
(QSVC31)
CLOSEMC
IJLQQT
IJLQCL
SVC 27
SVC 30
SVC 31
(HIO)
(QSVC30)
(QSVC31)
COPYC
IJLQDC
SVC 31
(QPOST)
Supervisor (QSVC31)
(IJLQIP80) Qmover
COPYP
IJLQFL
IJLQDP
COPYQ
IJLQDQ
COPYT
IJLQDE
COUNTER*
DATESTMP
IJLQDT
IJLQEX
DIRECT
IJLQLK
ENDRCV
IJLQIP (IJLQIP23) Cleanup
ENDREADY
(IJLQIP20) LPS Control
ENDSEND
IJLQRW
(IJLQIP20) LPS Control
SVC 0 (EXCP)
EXCP Supervisor
EOA
IJLQMT
IJLQEA
(first time only)
(IJLQNDRT) End-Insert
IJLQSH
IJLQSK
(from Cleanup for multiple routing)
IJLQEA
(IJLQIP23) Cleanup
IJLQRG
IJLQSH
(IJLQIP22) Recall
IJLQLK

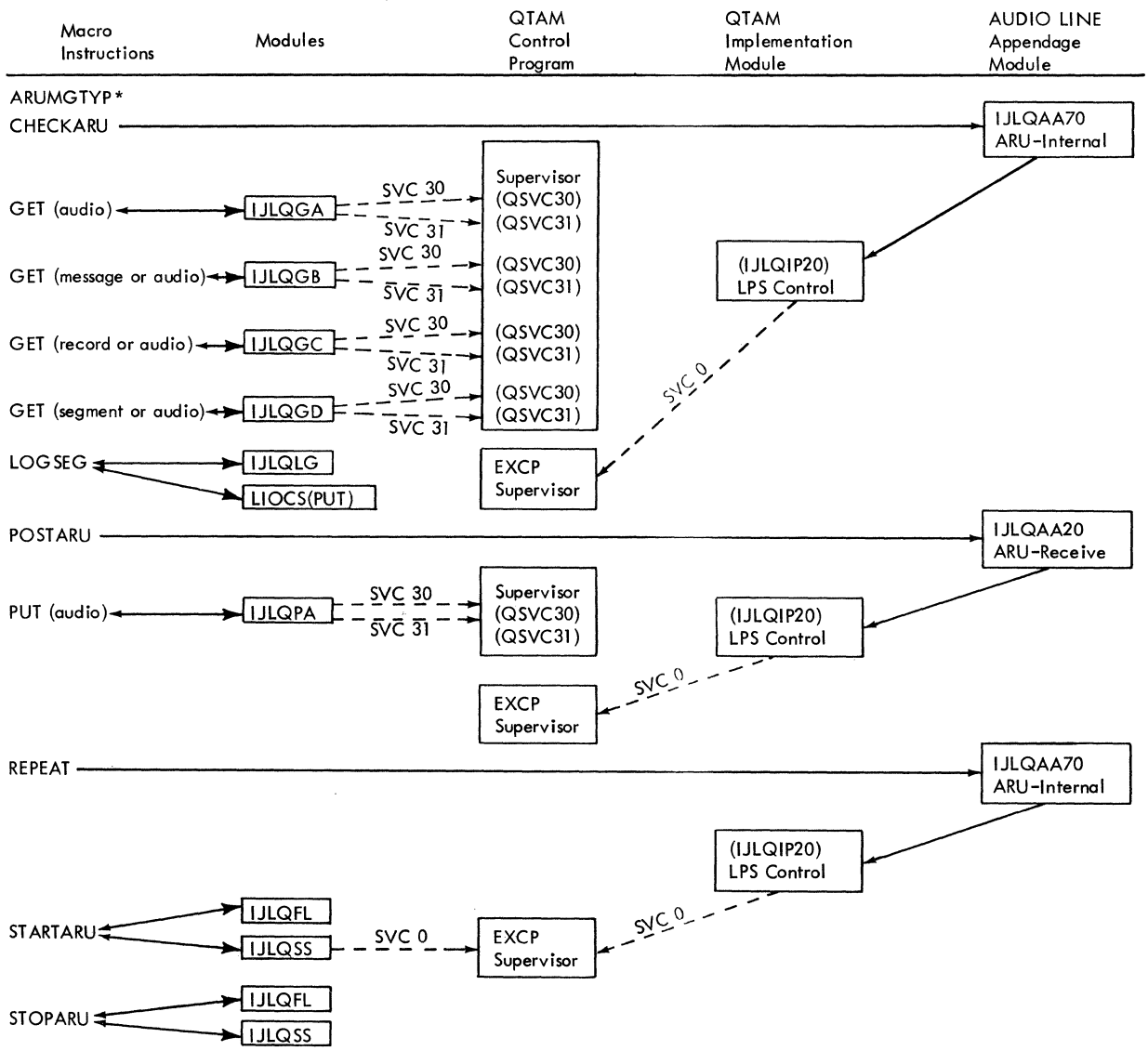

Macro Instruction
Modules
QTAM SVC/Subtask Control Routine
QTAM Implementation Module-IJLQIP
EOB
IJLQEB
IJLQRW
SVC 0 (EXCP)
EXCP Supervisor
(IJLQIP20) LPS Control
EOBLC
IJLQEC
(error retry)
IJLQRW
SVC 0 (EXCP)
EXCP Supervisor
(IJLQIP22) Recall
(IJLQIP20) LPS Control
ERRMSG
IJLQER
IJLQLK
(IJLQIP22) Recall
GET(message)
IJLQGM
SVC 30
SVC 31
Supervisor (QSVC30)
(QSVC31)
GET(record)
IJLQGR
SVC 30
SVC 31
(QSVC30)
(QSVC31)
GET(segment)
IJLQGS
SVC 30
SVC 31
(QSVC30)
(QSVC31)
INTERCPT
IJLQIT
LOGSEG
LIOCS(PUT)
LPSTART
USER LPS
MODE (PRIORITY)
IJLQMM
IJLQSH
IJLQMP
MODE (INITIATE)
IJLQMM
IJLQSH
IJLQMI
MODE (MOD2260)
IJLQMM
IJLQSH
MODE (CONVERSE)
IJLQMM
IJLQSH
IJLQMC
(first time only)
IJLQIP
(IJLQNDRT) End Insert
(from cleanup)
IJLQMC
(IJLQIP23) Cleanup
SVC 31
Supervisor (QSVC31)
(IJLQIP20) LPS Control
MSGTYPE
IJLQMT
IJLQSH
OPCTL
IJLQOC
IJLQLK
IJLQSH


Macro Instruction
Modules
QTAM SVC/Subtask Control Routine
QTAM Implementation Module-IJLQIP
PAUSE
IJLQPZ
SVC 30 (QWAIT)
Supervisor (QSVC30)
POLLIMIT
IJLQPL
POSTRCV
(IJLQIP23) Cleanup
POSTSEND
(IJLQIP23) Cleanup
EXCP Supervisor
SVC 0
IJLQRW
(IJLQIP20) LPS Control
PUT(message)
IJLQPM
SVC 30
SVC 31
Supervisor (QSVC30) (QSVC31)
PUT(record)
IJLQPR
SVC 30
SVC 31
(QSVC30) (QSVC31)
PUT(segment)
IJLQPS
SVC 30
SVC 31
(QSVC30 (QSVC31)
RCVHDR*
RCVSEG*
RELEASEM
IJLQRM
REROUTE
IJLQRR
IJLQLK
(IJLQIP22) Recall
RETRIEVE (DASD address)
IJLQRD
SVC 30
SVC 31
Supervisor (QSVC30) (QSVC31)
RETRIEVE (Sequence number)
IJLQRS
IJLQDE
IJLQRD
SVC 30
SVC 31
(QSVC30) (QSVC31)
ROUTE
IJLQRG
IJLQSH
IJLQLK
SENDHDR*
SENDSEG*
SEQIN
IJLQSI
IJLQSH
SEQOUT
IJLQSO
IJLQEX
SKIP (characters)
IJLQSK
IJLQSH
SKIP (count)
IJLQST



* This macro instruction generates no code or generates in-line code; it links to no module.

QTAM AUDIO LINKAGES



APPENDIX D: ALPHABETICAL LIST OF QTAM MODULES

IJLQAA -- Audio line appendage

IJLQAD -- IBM 7772 disk end appendage

IJLQBO -- Breakoff (BREAKOFF)

IJLQCK -- Checkpoint

IJLQCL -- Change line (STARTLN and STOPLN)

IJLQCM -- Cancel message (CANCELM)

IJLQCP -- Change polling list entry (CHNGP)

IJLQCR -- Checkpoint Request (CKREQ)

IJLQCT -- Change terminal table entry (CHNGT)

IJLQC1 -- QTAM Close Routine (CLOSE) Phase 1

IJLQC2 -- QTAM Close Routine (CLOSE) Phase 2

IJLQC3 -- QTAM Close Routine (CLOSE) Phase 3

IJLQDA -- Disk end appendage

IJLQDC -- Copy Line Error Counters (COPYC)

IJLQDE -- Copy terminal table entry (COPYT)

IJLQDL -- Distribution list

IJLQDP -- Copy polling list entry (COPYP)

IJLQDQ -- Copy queue control block status (COPYQ)

IJLQDT -- Insert date in message header (DATESTMP)

IJLQEA -- End-of-address (EOA)

IJLQEB -- End-of-block (EOB)

IJLQEC -- End-of-block and line correction (EOBLC)

IJLQER -- Error message (ERRMSG)

IJLQEX -- Expand message header

IJLQFL -- DTF locator

IJLQGA -- Get audio message (GET)

IJLQGB -- Get audio or nonaudio message (GET)

IJLQGC -- Get audio message or record (GET)

IJLQGD -- Get audio message or message segment (GET)

IJLQGM -- Get complete message (GET)

IJLQGR -- Get message record (GET)

IJLQGS -- Get message segment (GET)

IJLQIP -- QTAM Implementation

IJLQIT -- Intercept message (INTERCPT)

IJLQLA -- Line appendage

IJLQLC -- Line PCI - Program Check Module

IJLQLG -- Audio input message logging (LOGSEG)

IJLQLK -- Look-up terminal table entry (DIRECT)

IJLQLO -- IBM 2260 Local Appendage

IJLQMC -- Conversational mode (MODE)

IJLQMI -- Initiate mode (MODE)

IJLQMM -- Message-mode interface (MODE)

IJLQMP -- Priority mode (MODE)

IJLQMT -- Compare message type (MSGTYPE)

IJLQMW -- Message writer initiator

IJLQM0 -- Model channel program for IBM 1030 terminals

IJLQM1 -- Model channel program for IBM 1060 terminals

IJLQM2 -- Model channel program for IBM 2260 terminals

IJLQM3 -- Model channel program for AT&T 83B3 terminals

IJLQM4 -- Model channel program for Western Union Plan 115A terminals

IJLQM5 -- Model channel program for IBM 1050 switched and nonswitched terminals

IJLQM6 -- Model channel program for IBM 1050 nonswitched terminals; not included if the installation includes 1050 switched as well as 1050 nonswitched terminals

IJLQM8 -- Model channel program for AT&T TWX terminals (Models 33 and 35)

IJLQM9 -- Module channel program for IBM 2260 Local terminals

IJLQN0 -- Model channel program for IBM 2740 Terminals, Type 274A

IJLQN1 -- Model channel program for IBM 2740 Terminals, Type 274B

IJLQN2 -- Model channel program for IBM 2740 Terminals, Type 274C

IJLQN3 -- Model channel program for IBM 2740 Terminals, Type 274D

IJLQN4 -- Model channel program for IBM 2740 Terminals, Type 274E

IJLQN5 -- Model channel program for IBM 2740 Terminals, Type 274F

IJLQN6 -- Model channel program for IBM 2740 Terminals, Type 274G

IJLQN7 -- Model channel program for IBM 2740 Terminals, Type 274H

IJLQN8 -- Model channel program for WTTA

IJLQO1 -- Open Monitor/Open DASD Message Queues File (OPEN)

IJLQO2 -- Open Nonaudio Line/MS Queues

IJLQO3 -- Open Checkpoint/Restart (Phase 1)

IJLQO4 -- Open Checkpoint/Restart (Phase 2)

IJLQO7 -- Open IBM 7772 DCV Vocabulary File

IJLQO8 -- Open Audio Line Group/Output Queue Files

IJLQOA -- Operator Awareness

IJLQOB -- OBR/SDR

IJLQOC -- Operator Control (OPCTL)

IJLQPA -- Put audio message (PUT)

IJLQPL -- Polling limit control (POLLIMIT)

IJLQPM -- Put complete message (PUT)

IJLQPR -- Put message record (PUT)

IJLQPS -- Put message segment (PUT)

IJLQPZ -- Pause-transmit idle characters (PAUSE)

IJLQQT -- Close message control (CLOSEMC)

IJLQRA -- Translate table RCVARU: ARU code to EBCDIC

IJLQRB -- Translate table RCVITA2: ITA2 code to EBCDIC

IJLQRC -- Translate table RCVZSC3: ZSC3 code to EBCDIC

IJLQRD -- Retrieve message segment by DASD address (RETRIEVE)

IJLQRG -- Route message (ROUTE)

IJLQRM -- Release message (RELEASEM)

IJLQRR -- Re-route message (REROUTE)

IJLQRS -- Retrieve message header by sequence number (RETRIEVE)

IJLQRW -- Physical input/output control

IJLQR1 -- Translate table RCV1030: 1030 to EBCDIC

IJLQR2 -- Translate table RCV1050: 1050 to EBCDIC

IJLQR3 -- Translate table RCV1050F: 1050 to monocase EBCDIC

IJLQR4 -- Translate table RCV1060: 1060 to EBCDIC

IJLQR5 -- Translate table RCV2260: 2260 to EBCDIC

IJLQR6 -- Translate tables RCV83B3 or RCV115A: AT&T 83B3 or WU Plan 115A to EBCDIC

IJLQR7 -- Translate table RCVTWX: AT&T Models 33/35 (TWX) to EBCDIC

IJLQR8 -- Translate table RCV2740: 2740 to EBCDIC

IJLQR9 -- Translate table RCV2740F: 2740 to monocase EBCDIC

IJLQSB -- Translate table SNDITA2: EBCDIC to ITA2

IJLQSC -- Translate table SNDZSC3: EBCDIC to ZSC3

IJLQSH -- Scan message header

IJLQSI -- Sequence-in number verification (SEQIN)

IJLQSK -- Skip-through-character (SKIP)

IJLQSO -- Insert sequence-out number in message header (SEQOUT)

IJLQSR -- Source terminal name verification (SOURCE)

IJLQSS -- Start/Stop audio line (STARTARU and STOPARU)

IJLQST -- Skip-to-character (SKIP)

IJLQS1 -- Translate table SND1030: EBCDIC to 1030

IJLQS2 -- Translate table SND1050: EBCDIC to 1050

IJLQS4 -- Translate table SND1060: EBCDIC to 1060

IJLQS5 -- Translate table SND2260: EBCDIC to 2260

IJLQS6 -- Translate tables SND83B3 or SND115A: EBCDIC to AT&T 83B3 or WU Plan 115A

IJLQS7 -- Translate table SNDTWXE: EBCDIC to AT&T Models 33/35 with even parity (TWX)

IJLQS8 -- Translate table SND2740: EBCDIC to 2740

IJLQS9 -- Translate table SNDTWXO: EBCDIC to AT&T Models 33/35 with no

IJLQTA -- WTTA Line Appendage Module

IJLQTM -- Terminal Test Header Analysis

IJLQTR -- Code translation; used in conjunction with a QTAM or user-provided translate table

IJLQTS -- Insert time-of-day in message header (TIMESTMP)

IJLQTT -- On-Line Terminal Test

IJLQT1 -- Terminal Test Module, IBM 1030

IJLQT2 -- Terminal Test Module, IBM 1050

IJLQT3 -- Terminal Test Module, IBM 1060

IJLQT4 -- Terminal Test Module, IBM 2740

IJLQT5 -- Terminal Test Module, IBM 2260

IJLQWA -- Audio message Writer

IJLQW1 -- Message Writer for ERP

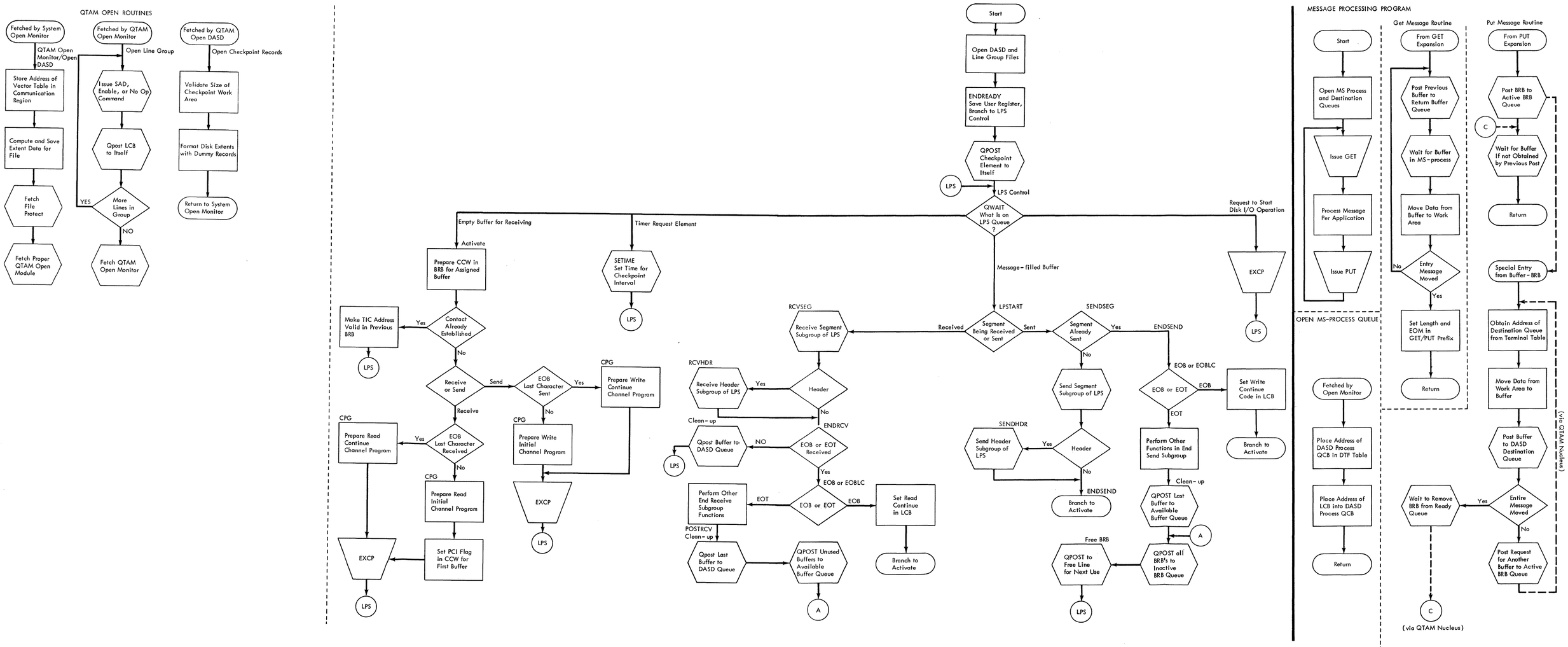
IJLQW2 -- Message Writer for Open/Close

IJLQW3 -- Message writer for Multivolume Facility

IJLQXL -- Cancel Routine (Phase 1)

IJLQXM -- Cancel Routine (Phase 2)

APPENDIX E: GENERAL FLOW OF QTAM OPERATIONS



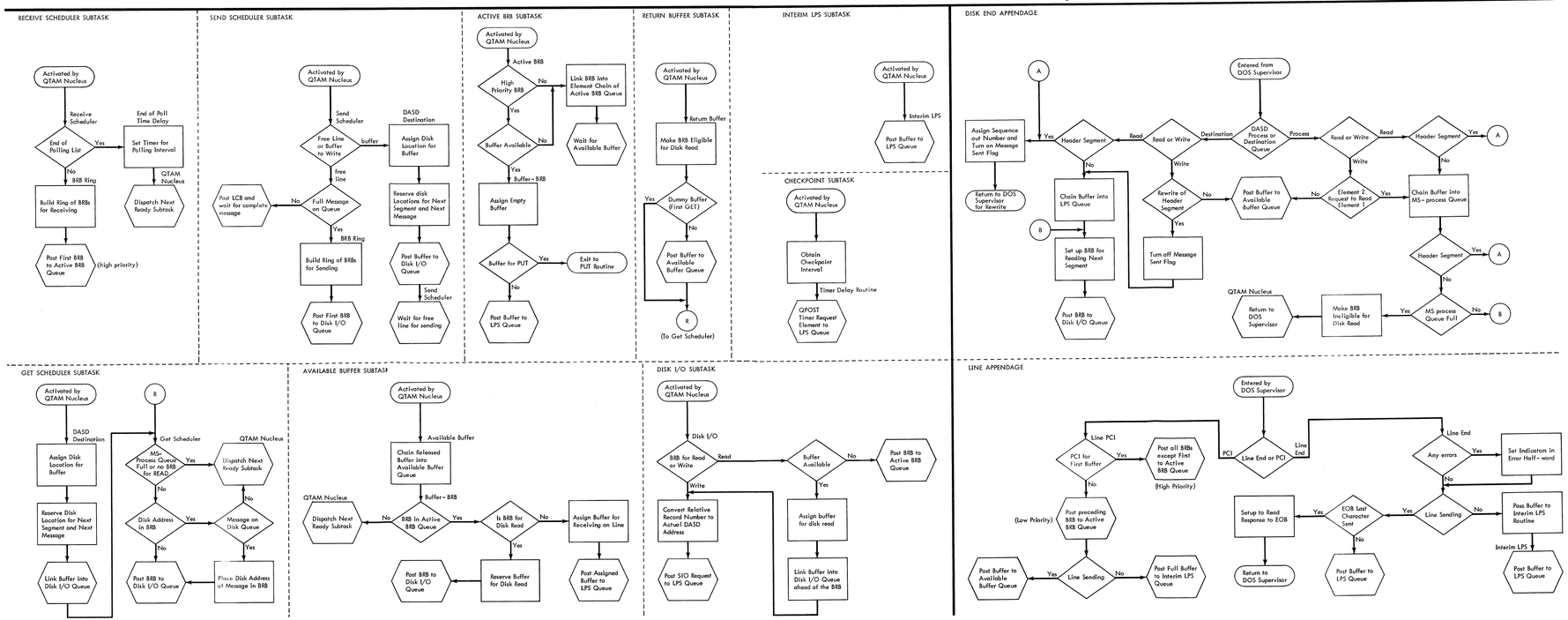

RECEIVE SCHEDULER SUBTASK
Activated by QTAM Nucleus
Receive Scheduler
End of Polling List
Yes
No
End of Poll Time Delay
Set Timer for Polling Interval
QTAM Nucleus
Dispatch Next Ready Subtask
BRB Ring
Build Ring of BRBs for Receiving
Post First BRB to Active BRB Queue
(high priority)
SEND SCHEDULER SUBTASK
Activated by QTAM Nucleus
Send Scheduler
Free Line or Buffer to Write
buffer
DASD Destination
Assign Disk Location for Buffer
free line
Reserve disk Locations for Next Segment and Next Message
Full Message on Queue
No
Post LCB and wait for complete message
Yes
BRB Ring
Build Ring of BRBs for Sending
Post First BRB to Disk I/O Queue
Post Buffer to Disk I/O Queue
Send Scheduler
Wait for free line for sending
ACTIVE BRB SUBTASK
Activated by QTAM Nucleus
Active BRB
High Priority BRB
No
Yes
Link BRB into Element Chain of Active BRB Queue
Buffer Available
No
Yes
Wait for Available Buffer
Buffer=BRB
Assign Empty Buffer
Buffer for PUT
Yes
Exit to PUT Routine
No
Post Buffer to LPS Queue
RETURN BUFFER SUBTASK
Activated by QTAM Nucleus
Return Buffer
Make BRB Eligible for Disk Read
Dummy Buffer (First GET)
Yes
No
Post Buffer to Available Buffer Queue
R
(To Get Scheduler)
INTERIM LPS SUBTASK
Activated by QTAM Nucleus
Interim LPS
Post Buffer to LPS Queue
CHECKPOINT SUBTASK
Activated by QTAM Nucleus
Obtain Checkpoint Interval
Timer Delay Routine
QPOST Timer Request Element to LPS Queue
DISK END APPENDAGE
A
Entered from DOS Supervisor
Assign Sequence out Number and Turn on Message Sent Flag
Yes
Header Segment
Read
Read or Write
Destination
DASD Process or Destination Queue
Process
Read or Write
Read
Header Segment
Yes
A
No
Write
Write
Return to DOS Supervisor for Rewrite
Chain Buffer into LPS Queue
Rewrite of Header Segment
No
Post Buffer to Available Buffer Queue
No
Element 2 Request to Read Element 1
Yes
Chain Buffer into MS-process Queue
B
Yes
Set up BRB for Reading Next Segment
Turn off Message Sent Flag
Header Segment
Yes
A
No
QTAM Nucleus
Post BRB to Disk I/O Queue
Return to DOS Supervisor
Make BRB Ineligible for Disk Read
Yes
MS process Queue Full
No
B
GET SCHEDULER SUBTASK
Activated by QTAM Nucleus
R
DASD Destination
Get Scheduler
QTAM Nucleus
Assign Disk Location for Buffer
MS-Process Queue Full or no BRB for READ
Yes
Dispatch Next Ready Subtask
No
Reserve Disk Location for Next Segment and Next Message
Disk Address in BRB
Yes
Message on Disk Queue
No
No
Link Buffer into Disk I/O Queue
Post BRB to Disk I/O Queue
Place Disk Address of Message in BRB
Yes
AVAILABLE BUFFER SUBTASK
Activated by QTAM Nucleus
Available Buffer
Chain Released Buffer into Available Buffer Queue
Buffer=BRB
QTAM Nucleus
Dispatch Next Ready Subtask
No
BRB in Active BRB Queue
Yes
Is BRB for Disk Read
No
Assign Buffer for Receiving on Line
Post BRB to Disk I/O Queue
Reserve Buffer for Disk Read
Yes
Post Assigned Buffer to LPS Queue
DISK I/O SUBTASK
Activated by QTAM Nucleus
Disk I/O
BRB for Read or Write
Read
Buffer Available
No
Post BRB to Active BRB Queue
Write
Yes
Convert Relative Record Number to Actual DASD Address
Assign buffer for disk read
Post SIO Request to LPS Queue
Link Buffer into Disk I/O Queue ahead of the BRB
LINE APPENDAGE
Entered by DOS Supervisor
Line PCI
PCI for First Buffer
Yes
Post all BRBs except First to Active BRB Queue
(High Priority)
PCI
Line End or PCI
Line End
Line End
Any errors
Yes
Set Indicators in Error Half-word
No
No
Post preceding BRB to Active BRB Queue
(Low Priority)
Setup to Read Response to EOB
Yes
EOB Last Character Sent
Yes
Line Sending
No
Pass Buffer to Interim LPS Routine
Interim LPS
No
Yes
Line Sending
No
Post Full Buffer to Interim LPS Queue
Post Buffer to Available Buffer Queue
Return to DOS Supervisor
Post Buffer to LPS Queue
Post Buffer to LPS Queue

Figure 22 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 DASD message queues file 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 (that 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 the current segment is the last segment of the message, no area is reserved for the 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 the:

1. First text segment of the same message,
2. Previous header segment, and
3. Next header segment.

Each text prefix contains the relative record numbers of the areas occupied by the:

1. Next text segment of the same message, and
2. 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 (BSLK) 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 BSLK field. Figure 17 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 DASD message queues file 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 22. 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).

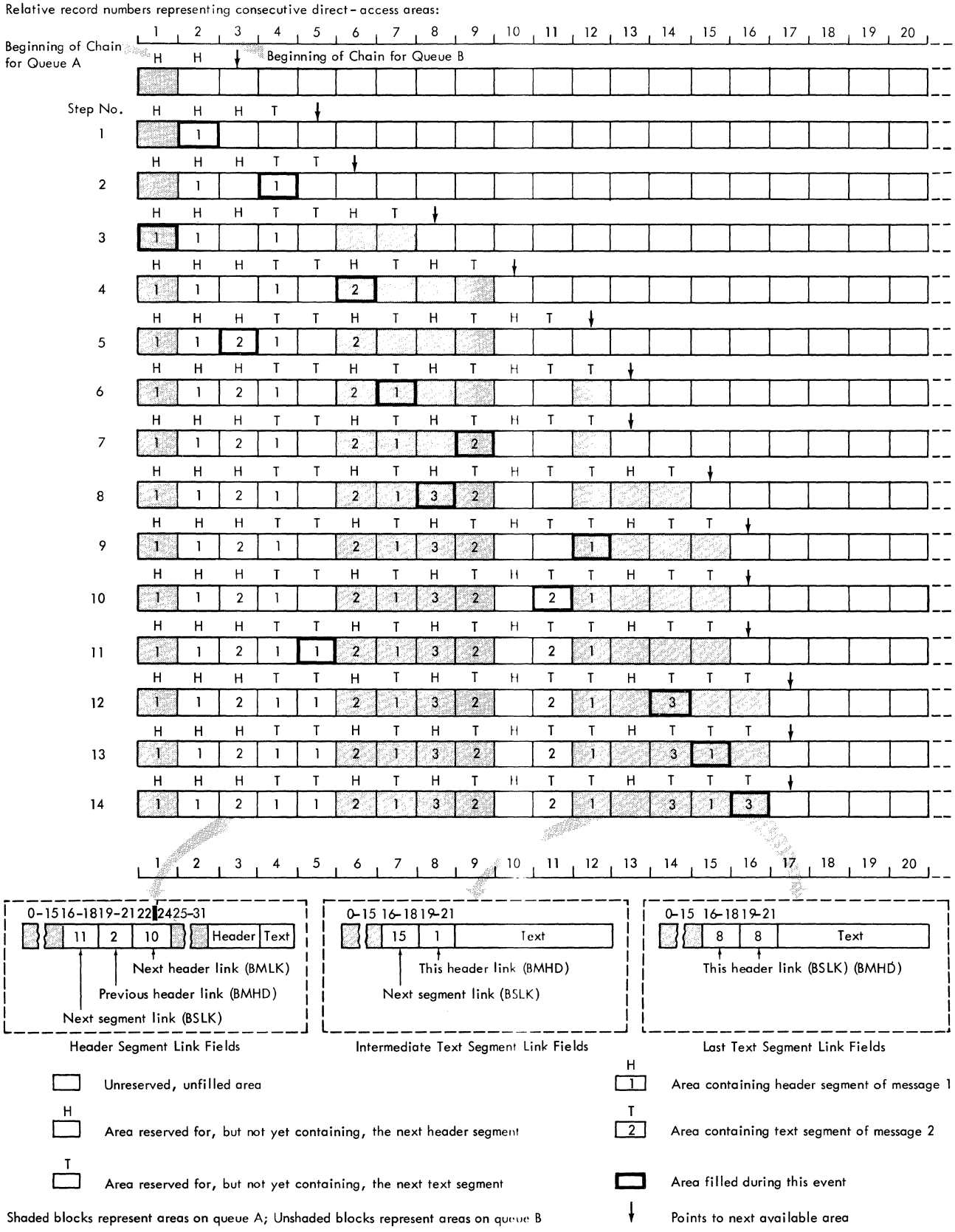
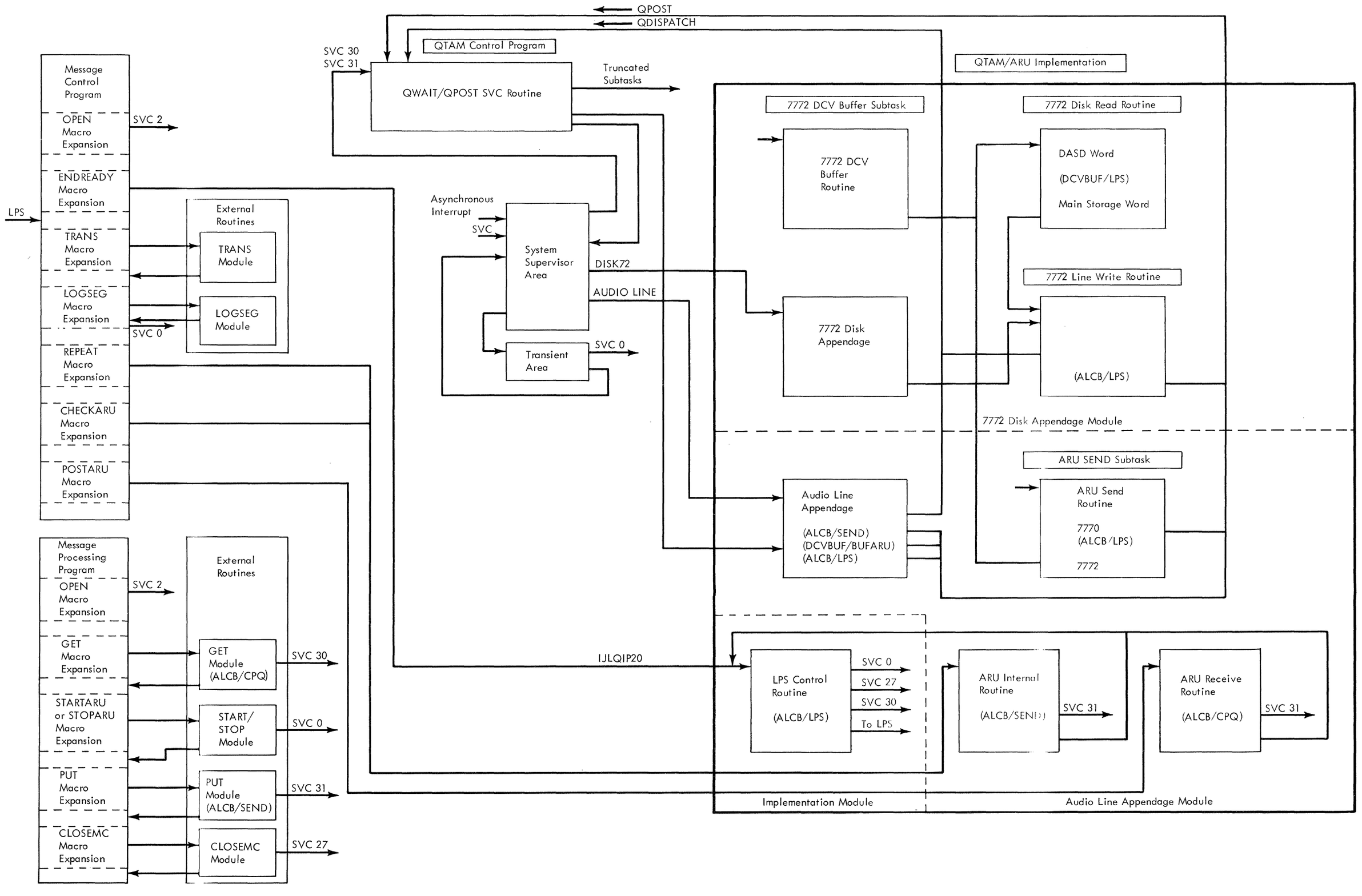


Figure 22. Example of Message Header and Text Relationships in Direct Access Destination and Process Queue


QPOST
QDISPATCH
QTAM Control Program
SVC 30
SVC 31
QWAIT/QPOST SVC Routine
Truncated Subtasks
QTAM/ARU Implementation
Message Control Program
OPEN Macro Expansion
SVC 2
ENDREADY Macro Expansion
LPS
TRANS Macro Expansion
LOGSEG Macro Expansion
SVC 0
REPEAT Macro Expansion
CHECKARU Macro Expansion
POSTARU Macro Expansion
External Routines
TRANS Module
LOGSEG Module
Asynchronous Interrupt
SVC
System Supervisor Area
Transient Area
SVC 0
DISK72
AUDIO LINE
7772 DCV Buffer Subtask
7772 DCV Buffer Routine
7772 Disk Read Routine
DASD Word
(DCVBUF/LPS)
Main Storage Word
7772 Line Write Routine
7772 Disk Appendage
(ALCB/LPS)
7772 Disk Appendage Module
ARU SEND Subtask
ARU Send Routine
7770
(ALCB/LPS)
7772
Audio Line Appendage
(ALCB/SEND)
(DCVBUF/BUFARU)
(ALCB/LPS)
Message Processing Program
OPEN Macro Expansion
SVC 2
External Routines
GET Macro Expansion
GET Module (ALCB/CPQ)
SVC 30
STARTARU or STOPARU Macro Expansion
START/ STOP Module
SVC 0
PUT Macro Expansion
PUT Module (ALCB/SEND)
SVC 31
CLOSEMC Macro Expansion
CLOSEMC Module
SVC 27
IJLQIP20
LPS Control Routine
(ALCB/LPS)
SVC 0
SVC 27
SVC 30
To LPS
ARU Internal Routine
(ALCB/SEND)
SVC 31
ARU Receive Routine
(ALCB/CPQ)
SVC 31
Implementation Module
Audio Line Appendage Module

Where more than one page reference is given, the major reference appears first.

READER'S COMMENT FORM

IBM System/360
Disk Operating System
Queued Telecommunication Access Method
Program Logic Manual

Order No. GY30-5002-4

- 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.

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.

Cut Along Line

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: Publications Center, Dept. E01

Fold Fold

IBM S/360 DOS QTAM Program Logic Manual (S360-30) Printed in U.S.A. GY30-5002-4

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]

IBM S/360 DOS QT ogram Logic Manual (S360-30) Printed in U.S.A. GY30-5002-4

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]