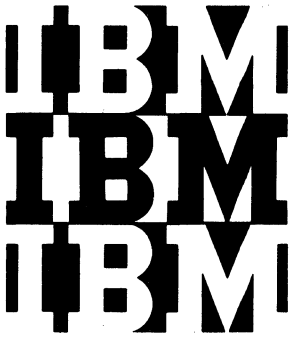


Y27-7128-0

Program Logic

IBM System/360 Operating System
Control Program With Option 2

Program Numbers 360S-CI-505, 360S-DM-508

Multiprogramming with a fixed number of tasks is Option 2 of the control program for the IBM System/360 Operating System. This publication describes the internal logic of the control program to the extent that it is modified for Option 2. These modifications affect the job management, task management, and data management routines of the control program.

The Program Logic Manual is to be used with the program assembly listings and is primarily a guide to those listings. It is intended for personnel involved in program maintenance and system programmers who are altering the system design. Program logic information is not necessary for the use and operation of the control program; therefore, distribution of this document is limited to those with the aforementioned requirements.

PREFACE

This publication describes the differences in internal logic that are introduced by the expansion of the control program to include Option 2: multiprogramming with a fixed number of tasks. It is assumed that the reader of this publication is thoroughly familiar with the basic operation of the control program. Only areas of difference are discussed in detail; however, information on sequential scheduling systems in general is included where necessary to assist the reader in relating new topics to known characteristics of the system.

The manual is divided into four major sections. The first section, the Introduction, outlines the function and organization of the entire control program and provides references to sources of information on various control program elements. The Theory of Operation section describes control program flow, with emphasis on job management operations, which is the aspect of the control program most significantly different under Option 2. The Program Organization section provides detailed descriptions of added or significantly changed routines. The Load Modules and Assembly Modules section contains a directory to the contents of the nucleus, the SVC library, and the link library.

Information in this document is directed to the customer engineer who maintains and services IBM System/360 Computing Systems and who is responsible for field maintenance and updating of IBM System/360 Operating System. This information may also be used by the programming systems maintenance programmer and the development programmer who will expand the system.

This publication may be used to locate those areas of the system to be analyzed or modified. The information is presented to enable the reader to relate Option 2 functions to the program listings (coding) for those functions. The comments in the listings provide information for thorough analysis and understanding of the coding.

PREREQUISITE PUBLICATIONS

Knowledge of the information in the following publications is required for a full understanding of the manual.

IBM System/360: Principles of Operation, Form A22-6821
IBM System/360: Operating System: Concepts and Facilities, Form C28-6535
IBM System/360 Operating System: Control Program Services, Form C28-6541
IBM System/360 Operating System: Linkage Editor, Form C28-6538
IBM System/360 Operating System: System Programmer's Guide, Form C28-6550
IBM System/360 Operating System: System Generation, Form C28-6554
IBM System/360 Operating System: Introduction to Control Program Logic, Program Logic Manual, Form Y28-6605
IBM System/360 Operating System: Fixed-Task Supervisor, Program Logic Manual, Form Y28-6612
IBM System/360 Operating System: Job Management, Program Logic Manual, Form Y28-6613

This publication was prepared for production using an IBM computer to update the text and to control the page and line format. Page impressions for photo-offset printing were obtained from an IBM 1403 Printer using a special print chain.

A form for reader's comments appears at the back of this publication. Address any additional comments concerning the contents of this publication to: IBM Corporation, Programming Publications, Department 637, Neighborhood Road, Kingston, New York 12401

CONTENTS

ILLUSTRATIONS

FIGURES

CHARTS

In a single-task environment main storage is divided into two areas: the fixed or system area, and the processing program area. In a multiprogramming environment with a fixed number of tasks, the processing program area is further divided into from one to four partitions. Figure 1 shows the division of main storage for a four-partition system. One task uses each partition, and all tasks operate concurrently.

The system area is used for system routines that perform control functions during the execution of a processing program, and for control blocks and tables used by the system for the performance of those control functions. Each partition is used for a processing program and its data, control blocks, and tables.

Option 2 of the control program provides for the concurrent execution of up to four jobs, each in its own fixed partition of main storage. Each job consists of a single task. The Option 2 system provides for task switching between the user tasks operating in the partitions, and between those tasks and the communication task (master scheduler) in the system area.

Jobs are sequentially scheduled in the Option 2 system. The job scheduling function is unchanged, except that the capability for performing that function in different partitions at different times is added.

With the Option 2 system, task dispatching differs primarily in that task switching is required, and that certain system functions such as abnormal termination must be carried out in such a way that other, unrelated tasks are not affected.

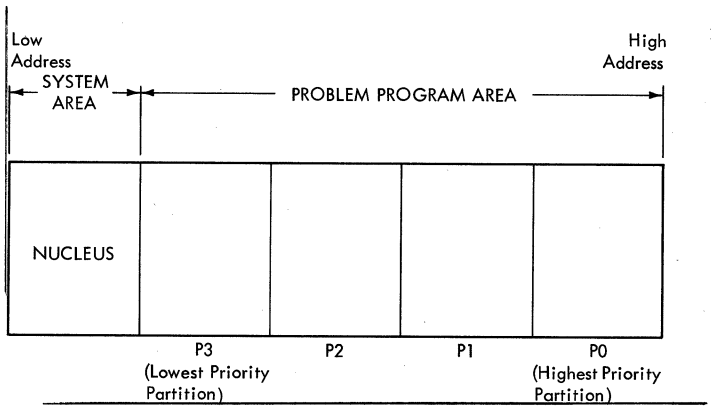


Figure 1. Storage Allocation for a Four-Partition System

Job and task management functions are performed under control program Option 2 through modified or expanded versions of the corresponding routines described in the publications IBM System/360 Operating System: Job Management, Program Logic Manual, and IBM System/360 Operating System: Fixed-Task Supervisor, Program Logic Manual. General information on those modifications and expansions is provided in this publication.

FUNCTIONS OF THE CONTROL PROGRAM WITH OPTION 2

Control program routines are grouped into three functional areas:

- Job Management routines.
- Task Management routines.
- Data Management routines.

JOB MANAGEMENT ROUTINES

Job management routines provide communication between the user and the operating system by:

- Analyzing the input job stream and collecting the information needed to prepare a job for execution.
- Analyzing operator commands, and transmitting messages from a program to the operator.

There are four major job management routines:

- Master scheduler, referred to for the Option 2 system as the communication task, which analyzes commands from the console and transmits messages to the operator.
- Reader/interpreter, which reads the input job stream and constructs control blocks and tables from information in the control statements.
- Initiator/terminator, which collects the information and resources needed to execute a job step and performs the operations required to terminate a job step.

- Scheduler controller, which governs the sequence in which operation of the reader/interpreter and the initiator/terminator occurs in the system's problem program partitions; this function is added for the Option 2 system.

The operation of these routines, to the extent that operational differences exist, is described in this publication. Operations of these routines that are not significantly different in either environment are described in the publication IBM System/360 Operating System: Job Management, Program Logic Manual.

TASK MANAGEMENT ROUTINES

Task management routines monitor and control the entire operating system, and are used throughout the operation of both the control and processing programs.

There are six functions performed by these routines:

- Interruption handling
- Task supervision
- Main storage supervision
- Contents supervision (and program fetch)
- Overlay supervision
- Time supervision

The task management routines are collectively referred to as the "supervisor." Of these functions, all are identical for either environment except for task supervision, changes to which are discussed in this publication. A description of all task management routines is given in the publication IBM System/360 Operating System: Fixed-Task Supervisor, Program Logic Manual.

DATA MANAGEMENT ROUTINES

Data management routines control all operations associated with input/output devices: allocation of space on volumes, channel scheduling, storing, naming, and cataloging of data sets, movement of data between main and auxiliary storage, and handling of errors that occur during I/O operations. Data management routines are used both by problem programs and by control program routines that require data movement. Problem programs use data management routines primarily to read and write data. The control program uses data management routines not only to read and write required data, but also to locate input data sets and to reserve auxiliary storage space for output data sets of the problem programs.

There are five categories of data management routines:

- Input/output (I/O) supervisor, which performs I/O operations and processes I/O interruptions.
- Access methods, which communicate with the I/O supervisor.
- Catalog management, which maintains the catalog and locates data sets on auxiliary storage.
- Direct-access device space management (DADSM), which allocates auxiliary storage space.
- Open/Close/End-of-Volume, which performs required initialization for I/O operations and handles end-of-volume conditions.

Of these routines, the only category affected by the selection of control program Option 2 is DADSM. All other data management routines operate identically with or without Option 2. The differences in DADSM operation are summarized in the "Program Organization" section of this publication. The operation of all data management routines is described in the following publications:

IBM System/360 Operating System: Input/Output Supervisor, Program Logic Manual; Form Y28-6616
IBM System/360 Operating System: Sequential Access Methods, Program Logic Manual; Form Y28-6604
IBM System/360 Operating System: Basic Direct Access Method, Program Logic Manual; Form Y28-6617
IBM System/360 Operating System: Catalog Management, Program Logic Manual; Form Y28-6606
IBM System/360 Operating System: Direct Access Device Space Management, Program Logic Manual; Form Y28-6607
IBM System/360 Operating System: Input/Output Support (OPEN/CLOSE/EOV), Program Logic Manual, Form Y28-6609

ORGANIZATION OF THE CONTROL PROGRAM

Different portions of the control program operate from different areas of main storage. The fixed (system) area of main storage is the lower portion of main storage; its size is determined by the control program configuration. The system area

contains those control program routines that perform a system function during the execution of a processing program.

The problem program area is the upper portion of main storage. It is defined at system generation as containing from two to four partitions; the number of partitions may be reduced and the size of each may be redefined at nucleus initialization, but is fixed thereafter until another initial program loading (IPL) is performed. Each partition may be occupied by a processing program, or by control program routines that either prepare job steps for execution (i.e., job management routines), or handle data for a processing program (i.e., the access methods).

On auxiliary storage, the control program resides in three partitioned data sets that are created when the operating system is generated. These data sets are:

- The NUCLEUS partitioned data set (SYS1.NUCLEUS), which contains the resident portion of the control program and the nucleus initialization program.
- The SVCLIB partitioned data set (SYS1.SVCLIB), which contains the nonresident SVC routines, nonresident error handling routines, and the access method routines.
- The LINKLIB partitioned data set (SYS1.LINKLIB), which contains the other nonresident control program routines and the IBM-supplied processing programs.

Figure 2 shows the main storage areas into which the routines from each partitioned data set are loaded.

RESIDENT PORTION OF THE CONTROL PROGRAM

The resident portion (nucleus) of the control program resides in the NUCLEUS partitioned data set. This portion of the control program is made up of those routines, control blocks, and tables that are brought into main storage at IPL and that are never overlaid by another part of the operating system. The nucleus is loaded into the system area of main storage.

The resident task management routines are: all of the routines that perform interruption handling, main storage supervision, and time supervision; and some of the routines that perform task supervision, contents supervision, and overlay supervision. These routines are described in this publication and in the publication IBM System/360 Operating System: Fixed-Task Supervisor, Program Logic Manual.

Resident for job management are those portions of the communication task that receive commands from the operator. The communication task is described in this publication.

The resident data management routines are the input/output supervisor and the BLDL routines, which are part of the partitioned access method. These routines are described in the publications IBM System/360 Operating System: Input/Output Supervisor, Program Logic Manual and IBM System/360 Operating System: Sequential Access Methods, Program Logic Manual.

NONRESIDENT PORTION OF THE CONTROL PROGRAM

The nonresident portion of the control program is made up of those routines that are loaded into main storage as they are needed, and can be overlaid after their completion. The nonresident routines operate from the partitions and from two sections of the system area called transient areas.

TRANSIENT AREAS: The transient areas are two blocks of main storage defined in the nucleus and embedded in the system area. The first, the SVC transient area, is reserved for nonresident SVC routines. The second, the I/O supervisor transient area, is used by nonresident I/O error handling routines that are brought in by the I/O supervisor. Each transient area contains only one routine at a time. When a nonresident SVC or error handling routine is required, it is read into the appropriate transient area. All routines read into the transient areas reside in SYS1.SVCLIB.

PARTITIONS: Each partition may be used for a processing program as well as for the access method routines and the nonresident job management routines of the control program. When the control program needs main storage to build control blocks or for a work area, it obtains this space from the partition in which the processing program that caused the requirement to arise was operating.

Access method routines are brought into each partition from SYS1.SVCLIB. Job management routines are brought in from SYS1.LINKLIB. Processing programs are brought in from either SYS1.LINKLIB, or a user-specified partitioned data set.

The program area is subdivided as shown in Figure 2. Job management routines, processing programs, and routines brought into storage via a LINK or XCTL macroinstruction are loaded into the lowest available portion of a partition. The

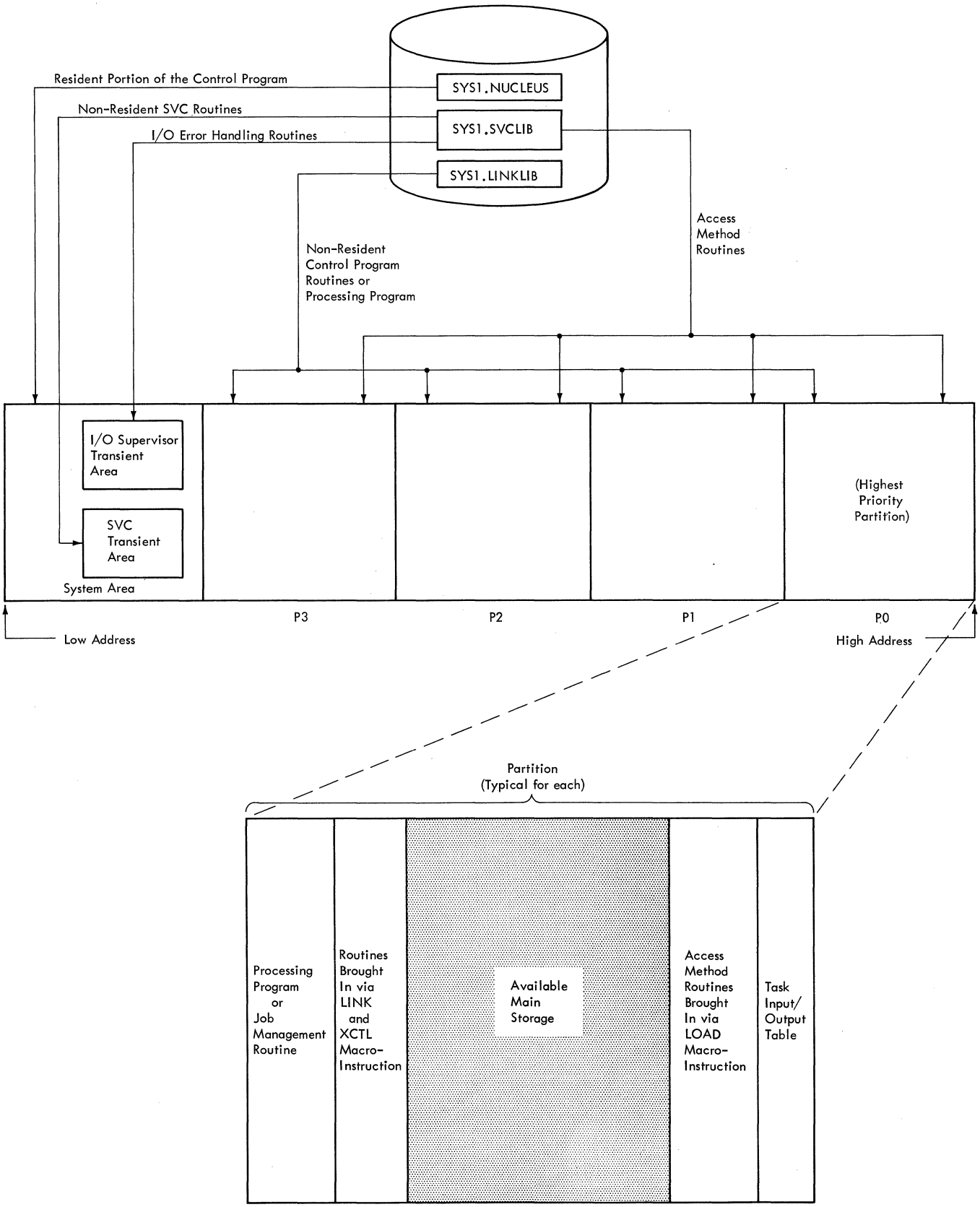


Figure 2. Division of Main Storage for the Operating System

highest portion of a partition is occupied by a table (the task input/output table) built by a job management routine. This table is used by data management routines and contains information about DD statements. It remains in storage for the whole job step. Access method routines and routines brought into storage via a LOAD macro-instruction are placed in the highest available locations in a partition.

SYSTEM ENVIRONMENT

MACHINE TYPES

The control program with Option 2 is designed for use with IBM System/360, Model 30 or higher. A two-partition system using the 18K scheduler (where K is equal to 1024 bytes) will operate in a configuration having a 64K byte main storage capacity; a system having more partitions and/or using the 44K scheduler requires additional main storage.

MINIMUM REQUIRED CONFIGURATION

Selection of Option 2 does not affect the minimum required configuration.

PROGRAM FLOW

The stages of program execution under the Option 2 system of the IBM System/360 Operating System are:

0. Loading the nucleus into main storage (IPL).
1. Reading control statements.
2. Initiating a job step.
3. Executing a job step, and (optionally) activating a lower-priority partition.
4. Terminating a job step, and (optionally) preparing for job scheduling in a higher-priority partition.

The operating system is given control of the computer when the control program nucleus is loaded. Thereafter, jobs may be processed without reloading the nucleus.

When the user introduces a job into the input stream, the initial processing required to prepare his job for execution is performed by job management routines. Control statements for a complete job are read during stage 1.

Stage 2 is the processing required to initiate the execution of a user's job step. Stage 3 occurs when CPU control is passed to that job step.

Up to this point, only one partition has been active. During stage 3 the problem program can cause another partition to become active; stages 1, 2, and 3 then proceed in that partition. This process can be repeated in each partition until all are active, with job step execution proceeding concurrently in each partition.

The Control Program with Option 2 is designed to operate with single-step, unending jobs in all partitions except the partition of lowest priority. In that configuration, step and job termination normally occur only in the lowest-priority partition. When a program enters stage 4, job management routines perform termination procedures for the step (and, when applicable, for the job).

Upon completion of a job, control passes back to stage 1. If further job step control statements had been read during stage 1, control passes to the initiation of the next job step (stage 2).

The user can, through a system command (SHIFT), reverse the process through which successive partitions are made active. When stage 4 is complete in a partition, stage 1 will normally proceed in the same partition; however, the user can cause the partition from which the terminating partition was originally activated, rather than the terminating partition itself, to be the next partition in which stage 1 is to proceed.

When termination is complete for all jobs in the system and there are no further jobs in the input job stream, the control program places the CPU in the wait state. As long as the nucleus remains intact in main storage, the user can introduce new jobs into the job stream without reloading the nucleus.

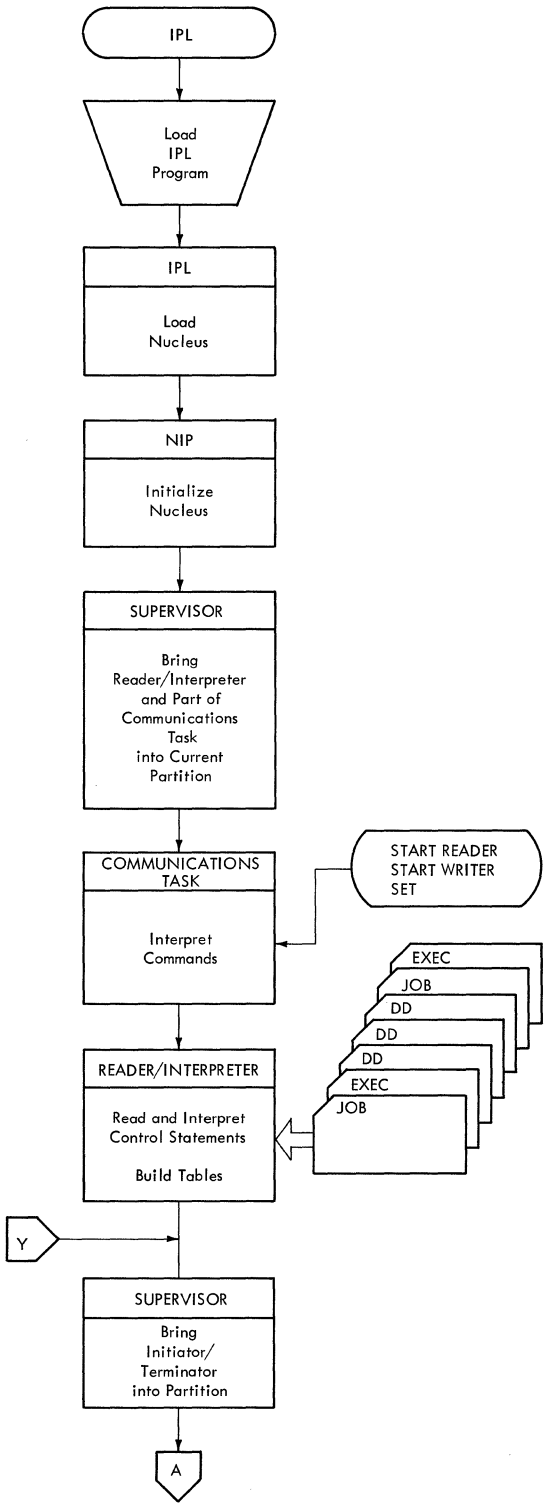
Reading control statements and initiating a job step are performed by the reader/interpreter and the initiator/terminator routines, respectively. Descriptions of these routines are given in the publication IBM System/360 Operating System: Job Management, Program Logic Manual.

A job step is performed by a user-written program (e.g., a payroll program), or an IBM-supplied processing program (e.g., linkage editor, COBOL).

Terminating a job step is performed by the initiator/terminator and the supervisor. Terminator functions peculiar to the Option 2 system are discussed in the "Job Management" section of this publication. Descriptions of these routines applicable to either environment are given in the publications IBM System/360 Operating System: Job Management, Program Logic Manual, and IBM System/360 Operating System: Fixed-Task Supervisor, Program Logic Manual.

The routines through which successive partitions are activated during problem program execution and relinquish control after termination are described in the "Job Management" section of this publication.

Figure 3 describes the overall flow of CPU control through the job processing cycle. These paragraphs describe the processing performed by various components of the control program as it loads the nucleus, reads control statements, initiates the job step, causes processing to begin or end in successive partitions, and terminates the job step. Control program processing performed during the execution of a job step, including control flow to the control program, control flow to a processing program, and input/output control, is unchanged under the Option 2 system. For a discussion of those topics, refer to the publication IBM System/360 Operating System: Introduction to Control Program Logic, Program Logic Manual.



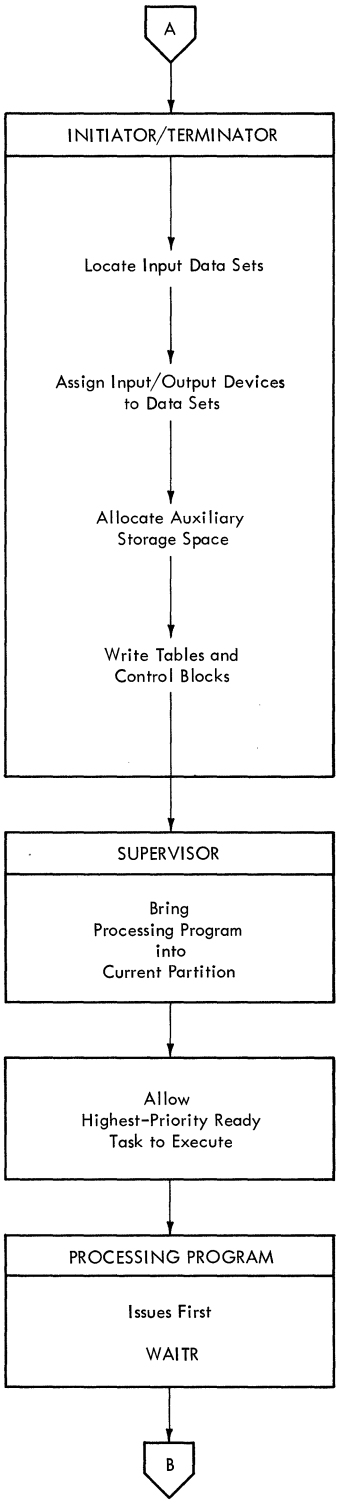
To load the nucleus, the operator specifies the device on which the system residence volume is mounted, and presses the load button on the console. This action causes an IPL record to be read and to be given CPU control. This record reads a second IPL record which, in turn, reads the rest of the IPL program into main storage.

The IPL program searches the volume label to locate the volume table of contents (VTOC) of the system residence volume. The volume table of contents is then searched for the SYS1.NUCLEUS. The nucleus is brought into the system area, and the nucleus initialization program (NIP) is brought into the dynamic area. NIP receives CPU control from the IPL program, and initializes the nucleus. Nucleus initialization includes initializing the control blocks that establish the absolute location and extent of each partition within the processing program area; communication between the operator and the system provides for the redefinition, if desired, of the partitions to be used. After completing its processing, NIP causes the reader/interpreter to be brought into the highest-priority partition in the problem program area. (NIP remains in main storage, but is not re-entered. It may or may not be overlayed as successive partitions

When the start reader (START RDR), start writer (START WTR), and set date (SET) commands are issued, the resulting interruption causes CPU control to be given to the master command routine. After processing the commands, this communication task routine passes CPU control to the reader/interpreter. The reader/interpreter is described in the publication IBM System/360 Operating System: Job Management, Program Logic Manual. Changes to the communication task that, in the Option 2 system, replaces the master scheduler are described below.

The reader/interpreter reads the control statements from the input job stream. Information from the JOB, EXEC, and DD statements is used to control the execution of job steps. This information is used to construct a job control table (JCT) for the job being read, a step control table (SCT) for the job step being read, and a job file control block (JFCB) and step input/output table (SIOT) for each data set being used or created by the job step. Information from these tables and control blocks is combined with information in the data control block (DCB) and data set control block (DSCB) or label when a data set is opened during step execution.

Figure 3. Example of CPU Control Flow for a Job Processing Cycle (Sheet 1 of 4)



The reader/interpreter is itself replaced in the highest-priority partition by the initiator/terminator routine.

After receiving CPU control, the initiator/terminator prepares to initiate the job step that has been read and interpreted. Using the data which the reader/interpreter extracted from the DD statements, the initiator/terminator:

Locates Input Data Sets: The initiator/terminator determines the volume containing a given input data set from the data definition (DD) statement, or from a search of the catalog. This search is performed by a catalog management routine that is entered from the initiator/terminator. A description of the routines that maintain and search the catalog is given in the publication IBM System/360 Operating System: Catalog Management, Program Logic Manual.

Assigns I/O Devices: A job step cannot be initiated unless there are enough I/O devices to fill its needs. The initiator/terminator determines whether the required devices are available, and makes specific assignments. If necessary, messages to the operator direct the mounting of volumes (tapes, etc.).

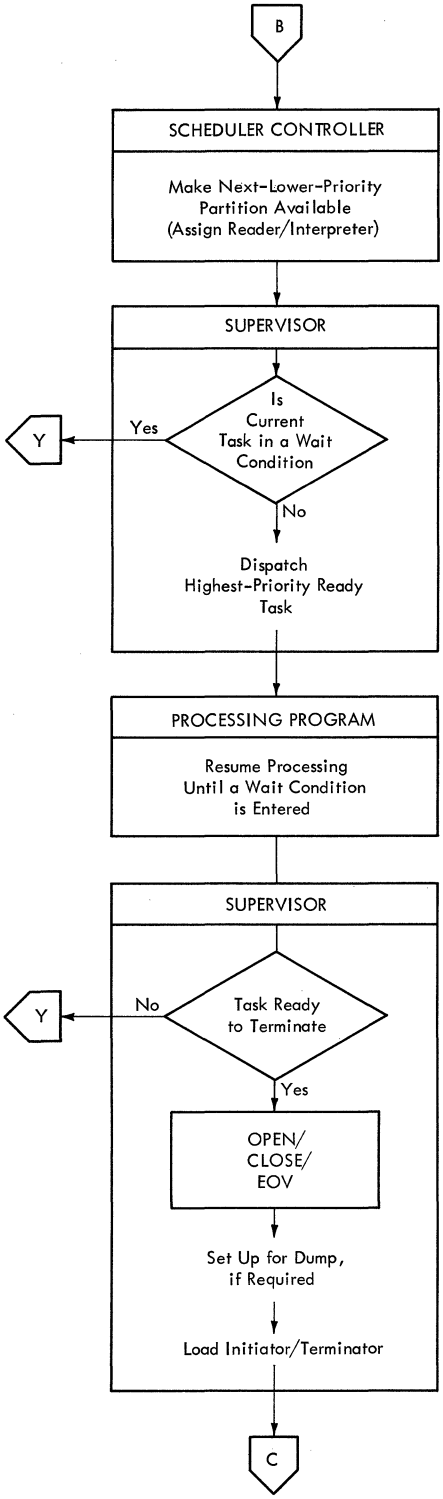
Allocates Auxiliary Storage Space: Direct access volume space required for output data sets of a job step is acquired by the initiator/terminator, which uses DADSM. A description of the operation of DADSM is given in the publication IBM System/360 Operating System: Direct Access Space Management, Program Logic Manual.

The JFCB, which contains information concerning the data sets to be used during step execution, is written on auxiliary storage. This data is used when a data set is opened, and when the job step is terminated (e.g., disposition).

The initiator/terminator causes itself to be replaced by the problem program to be executed.

The processing program can be one of the IBM-supplied processors (e.g., COBOL, linkage editor), or a user-written program. The processing program uses control program services for operations such as loading other programs and performing I/O operations.

Figure 3. Example of CPU Control Flow for a Job Processing Cycle (Sheet 2 of 4)



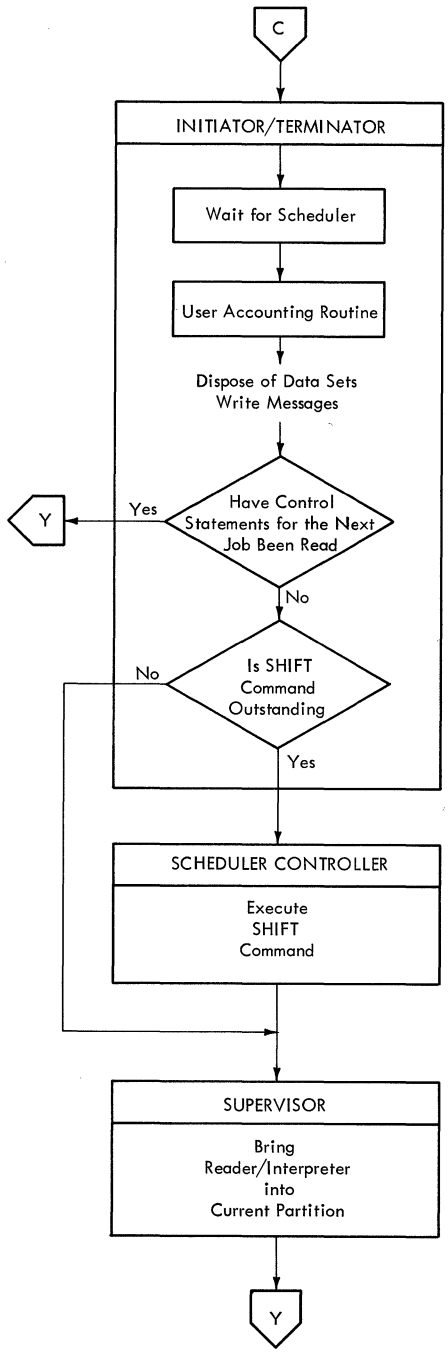
Initiation of operations in another partition begins when a WAITR macro-instruction is issued in the processing program. When a WAITR is encountered, control passes to the scheduler controller, which is resident in the nucleus. That routine restructures the system job queue in such a way that information related to the partition from which the WAITR was issued is saved, and a new area, to be used in connection with the partition now to be activated, is available to the reader/interpreter. Control then returns to the supervisor.

After a WAITR has been processed, both the partition in which the macro-instruction was issued and the next-lower-priority partition contain tasks that are potentially dispatchable. (The communication task is always potentially dispatchable, and is of higher priority than any partition-related task.) When the supervisor is re-entered, the task dispatcher dispatches the ready task of highest priority; therefore, processing continues in the partition from which the WAITR was issued until a Wait condition is entered in that partition. A Wait causes the next-lower-priority-partition's task to be dispatched. This causes the reader/interpreter to be brought into the lower-priority partition and to receive control. The reader/interpreter proceeds as described above, exactly as though the new job were the first to have been read after IPL or START; however, the reader/interpreter is now working with a different portion of the system job queue and in a different segment of main storage.

When a processing program terminates, the supervisor receives CPU control. The supervisor uses the OPEN/CLOSE/EOV routines to close any open data control blocks. These routines are described in the publication IBM System/360 Operating System: Input/Output Support (OPEN/CLOSE/EOV), Program Logic Manual.

Under abnormal termination conditions, the supervisor may also provide special termination procedures such as a storage dump.

Figure 3. Example of CPU Control Flow for a Job Processing Cycle (Sheet 3 of 4)



The supervisor passes control to the initiator/terminator, which is brought into the partition in which termination is to occur. The initiator/terminator determines whether the scheduler is currently associated with the partition; if not, the task in the terminating partition must WAIT until the scheduler has been re-associated with the partition.

When the scheduler is again available, the initiator/terminator performs the functions required to terminate individual job steps and complete jobs. It executes an installation accounting routine if one if provided.

The initiator/terminator releases the I/O devices, and disposes of data sets used and/or created during the job step. (This requires reading tables prepared during initiation. Some of these tables are part of the system job queue. It is for this reason that termination cannot proceed until the scheduler has again been associated with the terminating partition -- that is, until the portion of the job queue containing information for the terminating partition has again become the apparent "single" job queue for the system.)

If the control statements for the next job step were read and interpreted, the initiator/terminator initiates that step. If the statements were not read, the initiator/terminator determines whether a shift operation is pending. (A shift operation is pending when a SHIFT command has been entered by the operator or encountered in the job stream and has not been fully effected.) If no shift is outstanding, the initiator/terminator is replaced in the same partition with the reader/interpreter, which starts the read-initiate-execute-terminate cycle for the next job. If a shift is outstanding, the initiator transfers control to the scheduler controller, which reverses the previous restructuring of the job queue so that the effective job queue is associated with the next-higher-priority partition. The scheduler controller then causes the reader/interpreter to be brought not into the partition that just terminated, but into the next-higher-priority partition; the read-initiate-execute-terminate cycle then begins in that higher-priority partition.

Figure 3. Example of CPU Control Flow for a Job Processing Cycle (Sheet 4 of 4)

JOB MANAGEMENT

Job management (Chart 1) is the first and last portion of the control program that a job encounters. Its primary function is to prepare job steps for execution and, when they have been executed, to direct the disposition of data sets created during execution. Prior to step execution, job management:

- Reads control statements from the input job stream.
- Places information contained in the statements into a series of tables.
- Analyzes input/output (I/O) requirements.
- Assigns I/O devices.
- Passes control to the job step.

Following step execution, job management:

- Releases main storage space occupied by the tables.
- Frees I/O devices assigned to the step.
- Disposes of data sets referred to or created during execution.

Job management also performs all processing required for communication between the operator and the control program. Major components of job management are the job scheduler, which introduces each job step to System/360, and the communication task, which handles all operator-system communication.

JOB SCHEDULER FUNCTIONS

The job scheduler includes three programs: the reader/interpreter, the initiator/terminator and the scheduler controller. The functions of the reader/interpreter are unchanged from the sequential scheduling system; for further information, refer to the publication IBM System/360 Operating System: Job Management, Program Logic Manual.

After all control statements for a job have been processed, or when data is encountered in the input job stream, the reader/interpreter gives control to the initiator/terminator. The initiator portion of the initiator/terminator function is unchanged from the sequential scheduling system; for further information, refer to the publication IBM System/360 Operating System: Job Management, Program Logic Manual.

When the job step has been executed, control is again given to the initiator/terminator which, when the scheduler is assigned to the partition in which the job step has executed, performs data set dispositions and releases I/O resources. The shift count is interrogated, at job termination, to determine if the scheduler is to be shifted into a higher priority partition.

COMMUNICATION TASK FUNCTIONS

The routines of the communication task process the following types of communication between the operator and the system.

- Operator commands, whether they are issued through the console or through the input job stream.
- Write-to-operator (WTO) and write-to-operator with reply (WTOR) macro-instructions.
- Interruptions caused when the INTERRUPT key is pressed.

JOB PROCESSING

Figure 4 shows the major components of job management and illustrates the general flow of control.

Control is passed to job management whenever the supervisor finds that there are no program request blocks in the request block queue. This can occur for two reasons: either the initial program loading (IPL) procedure has just been completed or a job step has just been executed.

Entry to Job Management Following Initial Program Loading

Following IPL, certain actions must be taken by the operator before job processing can begin. Therefore, control passes to the communication task and a message is issued to the operator instructing him to enter commands. These "initialization" commands include a SET command, a start writer (START WTR) command, and a start reader (START RDR) command. When a START command with a blank operand is issued, control is passed to the reader/interpreter.

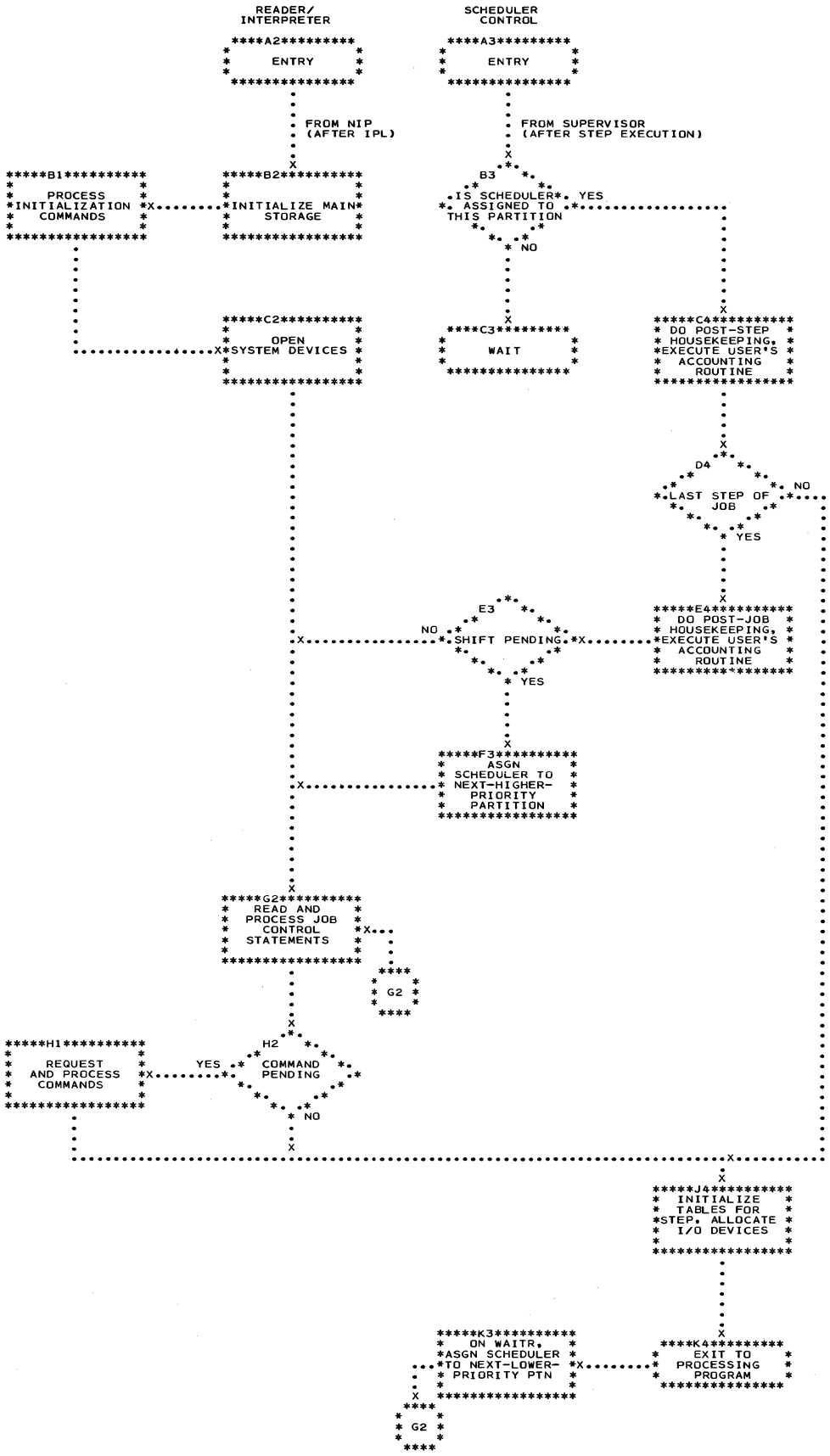


Figure 4. Job Management Logic

Entry to Job Management Following Step Execution

Following step execution, control is routed to the step termination routine of the initiator/terminator. If the job had been completed, control is also passed to the job termination routine of the initiator/terminator. Both routines are described under "Job and Step Termination."

Control Statement Processing

After completion of the processing that immediately follows IPL, or after termination of a job or of a step containing data in the input job stream, control is passed to the reader/interpreter. The reader/interpreter reads and processes control statements until one of the following conditions is encountered:

- A DD * or DD DATA statement.
- Another JOB statement.
- A null statement.
- An end-of-data set (EOF) on the system input device.

Meanwhile, if the operator has pressed the REQUEST key and has entered a request (REQ) command during execution of the job step or any of the above processing, the communication task routines set a command-pending indicator on during the ensuing interruption. The indicator is now checked and, if found to be on, control is passed to the communication task, which causes a message to be issued instructing the operator to enter commands, and then processes the commands.

Step Initiation

Control next passes to the initiator/terminator, which examines I/O device requirements, assigns (allocates) I/O devices to the job step, issues mounting instructions, and verifies that volumes have been mounted on the correct units. Finally, the initiator/terminator passes control to the job step.

Job and Step Termination

When processing program execution is completed, the supervisor, finding no program request blocks in its request block queue, passes control to the job management routines. Entry is first made to the step termination routine.

Step termination may occur only when the scheduler is attached to the terminating partition. If termination cannot occur, the pre-termination routine issues a 'PARTITION n WAITING TO TERMINATE' message and waits until the partition gains control of the scheduler. The step termination routine performs end-of-step housekeeping and passes control to the user's accounting routine, if one was provided. When the accounting routine has been executed, the supervisor returns control to the step termination routine. If the job termination indicator is on, control is then passed to the job termination routine; or to the reader/interpreter if the indicator is off and no more steps are ready for initiation; or to the step initiation routine.

The job termination routine performs end-of-job housekeeping. It exits to the user's accounting routine, if one was provided. After the accounting routine is executed, the supervisor returns control to the job termination routine which decrements the partition shift count by one if neither the partition number nor shift count is already zero. Control is then passed to the reader/interpreter.

OPERATOR-SYSTEM COMMUNICATION PROCESSING

The routines that handle operator-system communication are contained in the communication task. Communication may take one of two forms: commands, which allow the operator to change the status of the system or of a job or job step; and the WTO or WTOR macro-instructions, which allow processing programs or system components to issue messages to the operator. The communication task routines also switch functions from the primary console device to an alternate console device when the INTERRUPT key is pressed.

Command Processing

Commands may be issued by the operator in two ways: he may insert command statements between job steps in the input job stream, or he may issue commands through the console input device. Commands encountered in the input job stream cause control to be passed to the communication task, which processes them. Before entering commands through the console, however, the operator must press the REQUEST key to cause an attention interruption. Figure 5 shows the actions taken after the key is pressed.

WTO/WTOR Macro-Instruction Processing

Whenever the WTO or WTOR macro-instruction is issued, a supervisor interruption occurs. (See Figure 6.)

External Interruption Processing

When the operator presses the INTERRUPT key, an external interruption occurs. The communication task then switches functions from the primary to the alternate console I/O device. (See Figure 7.)

ENQ/DEQ PROCESSING

The enqueue and dequeue service routines, through which the ENQ and DEQ macro-instructions are implemented, provide for controlled, sequential access to serially reusable resources such as data sets, programs, or work areas in main storage. The routines service both problem program ENQ/DEQ requests, and requests from the system's job management and fixed-task supervision routines. The primary function of the enqueue and dequeue service routines is to test for the availability to the requesting task of a serially reusable resource, to enqueue the request if necessary, and to dequeue the request when use of the resource is complete.

In addition, the service routines permit system routines to set a system-must-complete flag before performing a critical operation, then to remove (reset) the flag when the operation has been successfully completed. This feature is available only to system routines; use of the system-must-complete feature in a problem program causes abnormal termination.

ENQ/DEQ Control Blocks

Resources are identified by the requester through a major name, specifying a set of resources, and a minor name, specifying a particular resource within that set. An enqueued request has associated with it three control blocks: a major queue control block (QCB), a minor QCB, and a queue element. (See the program listing for the structure and contents of these control blocks.)

The major QCB represents the set of resources specified by the major name parameter of the ENQ request. All major QCBs existing in the system at a given time are linked together; the head of the major QCB chain is a control field (IEAQQCB0) within the enqueue service routine.

Queued on each major QCB are the minor QCBs corresponding to the minor names of the specific resources for which requests have been issued. Queued on each minor QCB are queue elements representing the tasks under which the outstanding requests were issued.

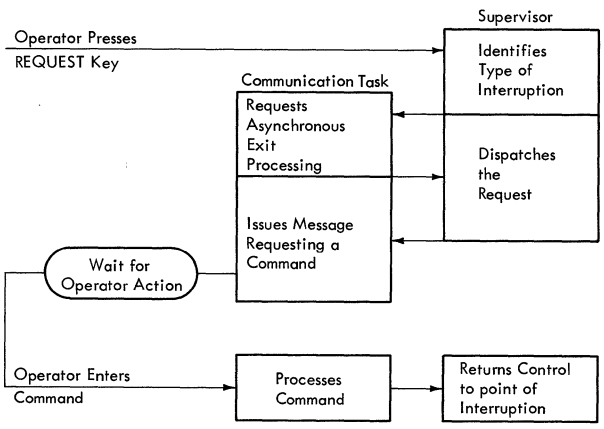
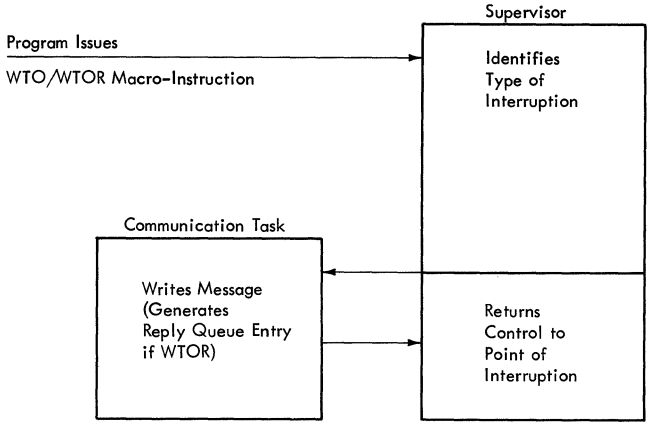
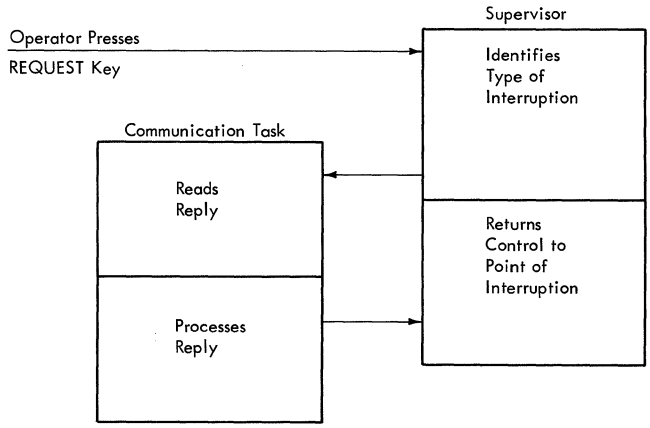


Figure 5. Attention Interruption Processing Flow



A. Message Processing



B. Reply Processing

Figure 6. WTO/WTOR Macro-Instruction Processing Flow

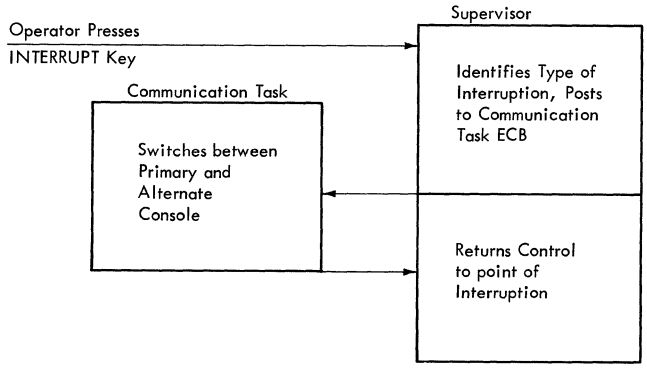


Figure 7. External Interruption Processing Flow

Note: If the STEP operand is included in an ENQ or DEQ macro-instruction, the protection key for the job step is treated as part of the minor name when the minor QCB queue is searched. If two requests specify the same major and minor name and if either request or both includes the STEP operand, both requests will be represented by the same major QCB but different minor QCBs. However, because the Option 2 system does not include the ATTACH and DETACH macro-instructions, the STEP operand has no effect.

If the SYSTEM operand is included in an ENQ or DEQ macro-instruction, the minor name is used as specified. Two requests specifying the same major and minor name and SYSTEM will be represented by the same major QCB and the same minor QCB.

All ENQ/DEQ control blocks are dynamically created and deleted, as ENQ and DEQ requests are processed and as other system functions, such as abnormal termination, are performed. The physical location of the major and minor QCBs, with respect to the partition in which the requesting task was operating, varies depending upon the circumstances of their creation and deletion. When an ENQ request is serviced, a GETMAIN is issued to obtain main storage for a major QCB, a minor QCB, and a queue element. The queue element is always developed and linked to the appropriate control block; queue elements remain in the requesting partition from their creation (on ENQ) until their deletion (normally on DEQ). The main storage obtained for the QCBs may or may not be used at the time that the queue element is created. Major and minor QCBs are copied from partition to partition as required by the sequence in which queue elements are dequeued. If the required major and/or minor QCB already exist in another partition, the corresponding area(s) in the requesting partition is reserved for use if it becomes necessary to copy the QCB(s) into the requesting partition.

For a summary of typical control block patterns during ENQ/DEQ, see Figure 8.

Sequence of Execution for Enqueued Tasks

The queue elements enqueued upon any one minor QCB represent tasks that have requested access to the corresponding resource. When control within a task passes to the enqueue service routine, the task may enter an effective wait until the request is serviced; that is, control is not returned from the enqueue service routine to the processing program until the resource has actually been made available to the task. The time at which a task proceeds (through re-entry to the calling routine) is determined by the relative position of shared and exclusive requests on the queue, and by the status of each, as described in the following paragraphs.

A queue element may be thought of as being ready or not-ready, where the condition ascribed to the queue element is actually the condition of the associated task. Then an ENQ specifying several resources is issued, the wait count in the SVC request block (SVRB) associated with the request is set to the number of resources requested by, but unavailable to, the task. Whenever the wait count in an SVRB is non-zero, the routine to which the SVRB points cannot proceed, although the the task with which the SVRB is associated may not be waiting. This condition is summarized by describing the queue element as not-ready. Conversely, a queue element may be described as "ready" when the wait count in the associated SVRB is zero.

If any queue element preceding the first exclusive request on the queue for a resource is shared and ready, the task associated with that queue element proceeds. Furthermore, the tasks represented by any subsequent shared and ready requests on the queue that precede the first exclusive request proceed concurrently. The first exclusive request, whether ready or not-ready, and all subsequent requests, whether exclusive or shared, are not serviced at this time.

If the queue element at the head of the queue is exclusive and ready, the task associated with that queue element proceeds. No other task represented on that queue proceeds until the exclusive request has been dequeued.

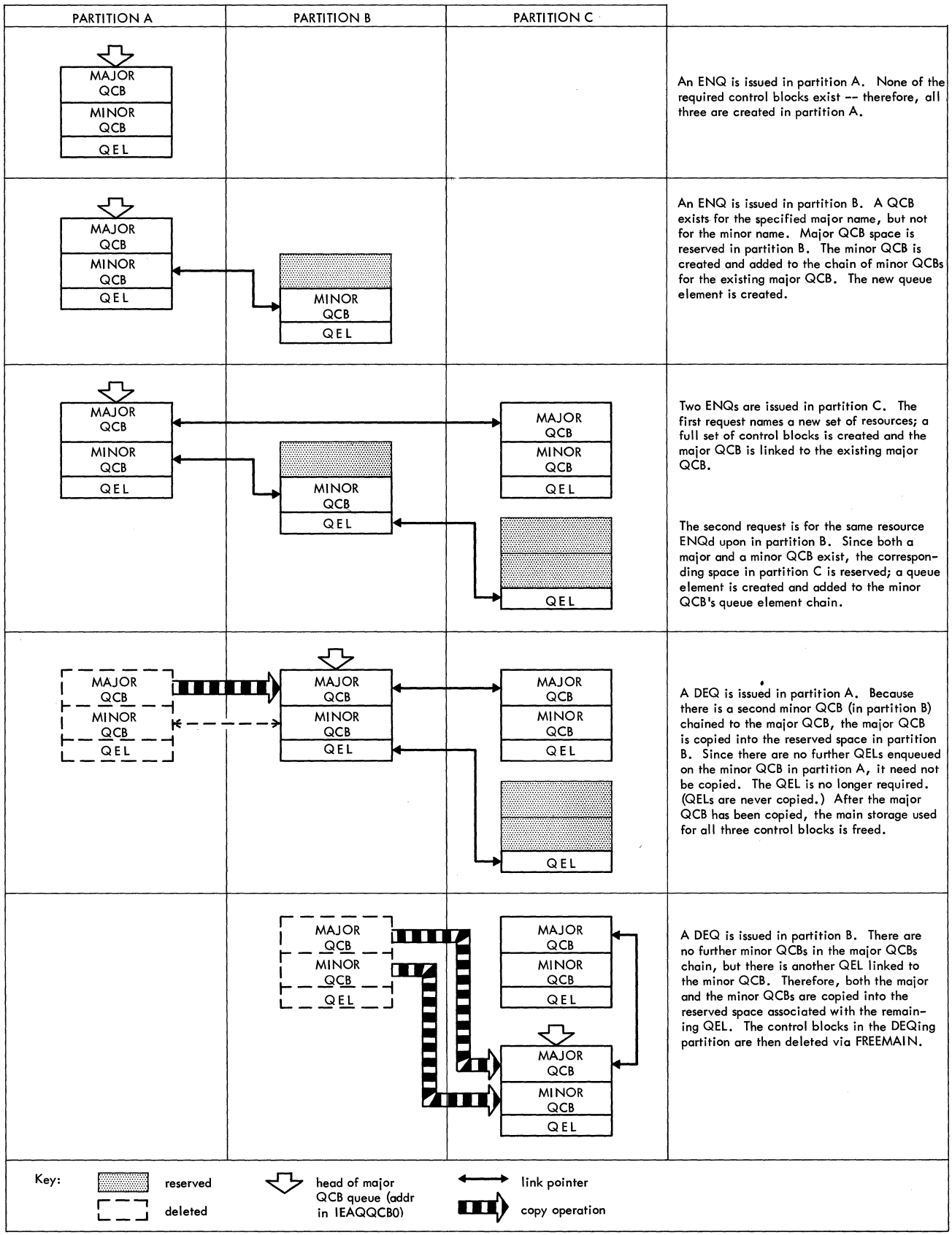


Figure 8. ENQ/DEQ Control Block Creation and Deletion

If the queue element at the head of the queue is exclusive and not-ready, no tasks represented on the queue can proceed.

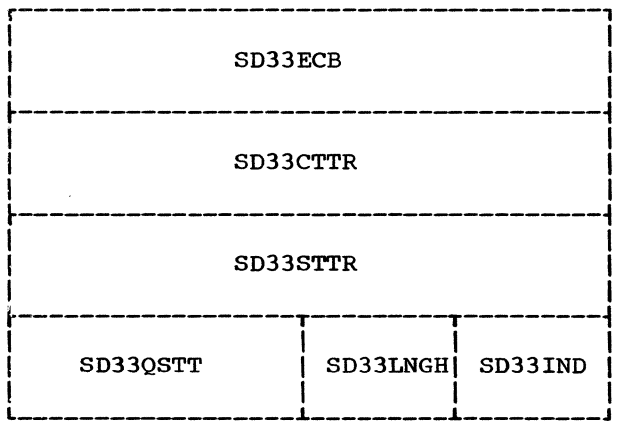
LOAD MODULES

Most job management routines exist as a series of load modules that reside on a permanently resident volume. The only exceptions are the posting routines of the communication task, which reside in the nucleus. The "Load Modules and Assembly Modules" section contains a list of the routines that make up each job management load module.

JOB SCHEDULER MODIFICATIONS

PARTITION-RELATED SCHEDULER CONTROL BLOCK

The partition-related scheduler control block (PRSCB) is the only new control block introduced into the system by Option 2 of the control program. One PRSCB is created for each partition at nucleus initialization. The PRSCBs reside in the nucleus, as module IEFSD032, and are defined by a DSECT, module IEFSD033. PRSCBs are contiguous and are arranged by priority order, beginning with highest priority (Partition 0). A pointer to the PRSCB for a given partition is contained in the three bytes immediately preceding the boundary box for that partition. The content and structure of the PRSCB are described below.



Field	Bytes	Contents
SD33ECB	4	Scheduler-controlling event control block. This ECB is posted complete whenever the scheduler is assigned to the partition through a WAITR issued in the next-higher-priority partition. The wait flag in this ECB is turned on when the scheduler is relinquished, either through a WAITR in this partition or through processing of a SHIFT command.
SD33CTTR	4	Current TTR save area. When a partition relinquishes scheduler control through a WAITR, the scheduler downshift routine stores in this field (in TTR form) the next location in the queue-manager's extent that would have been used by the queue-manager if further records were to have been written for the relinquishing partition. The system job queue variable area applicable to the next-lower-priority partition begins on the next full track following this location.
SD33STTR	4	Fixed-area table save area. When a partition relinquishes scheduler control through a WAITR, the relinquishing partition's JCT, SCT, and LCT are moved from the fixed area to this save area, following the variable information for the relinquishing partition.
SD33QSTT	2	Starting track location save area. This area contains (in TT form) the location of the track on which the variable area for the partition begins.
SD33LNGH	1	Offset to PRSCB for active partition. This byte is meaningful only in the PRSCB for Partition 0. Whenever a scheduler upshift or downshift is effected, the length of one PRSCB (16 bytes) is added to or subtracted from this field in the Partition 0 PRSCB. This value, added to the address of the PRSCB Partition 0, yields the address of the PRSCB for the partition to which the scheduler is currently assigned.
SD33IND	1	Partition identification; contains 00 for Partition 0, 01 for Partition 1, etc.

TERMINATION

The termination function of the initiator/terminator (Chart 12) performs post-step and post-job housekeeping. It is normally given control following step execution, but is also given control when a job management routine encounters an irrecoverable error while processing a job step. Termination routines:

- Release space occupied by tables.
- Free I/O devices.
- Dispose of data sets referred to or created during execution.

Major components of termination are:

- The pre-termination routine, which determines if the scheduler is currently associated with the terminating partition.
- The step termination routine, which performs post-step housekeeping functions.
- The job termination routine, which performs post-job housekeeping functions.
- The shift count interrogator, which determines whether a shift is to be performed.

The disposition and unallocation subroutine is used by both the step and job termination routines. Basically, this subroutine handles disposition of data sets and frees devices allocated to a step. The disposition and unallocation subroutine is described in the publication IBM System/360 Operating System: Job Management, Program Logic Manual.

PRE-TERMINATION ROUTINE: The pre-termination routine (Chart 13) is new for the Option 2 system. The routine is entered from the supervisor when the problem program has issued its highest-level return, causing the supervisor's ABEND routine to be entered; the second load module of the ABEND routine exits to the job management GO module.

Working through the communication vector table, the pre-termination routine obtains the address of the TCB for the current task (the task that is attempting to terminate), obtains from the TCB a pointer to the related boundary box, and obtains from the boundary box the address of the partition-related scheduler control block (PRSCB) for the partition in which the terminating task was operating (see Figure 9). The first fullword of the PRSCB is the scheduler-controlling ECB for that partition.

The ECB is posted complete if the terminating partition has never issued a first WAITR macro-instruction, and has therefore never relinquished control of the scheduler, or if the partition has relinquished control but has again been assigned scheduler control through SHIFT command processing. If the wait flag is on in the ECB, the partition has relinquished scheduler control through a WAITR and the scheduler is currently oriented toward some partition of lower priority; termination can proceed only after the scheduler has been re-associated with the terminating partition.

If the wait flag is on in the scheduler-controlling ECB for the terminating partition, the pre-termination routine issues a 'PARTITION n WAITING TO TERMINATE' message and waits on the ECB. (The ECB is posted complete when a SHIFT command causes the scheduler upshift routine to pass control of the scheduler from the next-lower-priority partition to this partition.) If the complete flag is on, the routine bypasses the message, issues a WAIT on the ECB to decrement the wait count, and continues processing.

When the wait for scheduler control is satisfied, the pre-termination routine examines the completion code in the ECB. A completion code of 4 indicates that scheduler control was relinquished by, and returned to, the terminating partition. Control was originally relinquished through a WAITR macro-instruction; when the WAITR was processed, the first-time WAITR switch for this partition was turned off. If this is the case, the pre-termination routine turns the switch back on, in preparation for the first WAITR macro-instruction in the next job (if any) to be scheduled into the terminating partition, and resets the completion code in the ECB to zeros.

If the completion code is not 4, the terminating partition has never relinquished control and its first-time WAITR switch is, therefore, still on. In this case, resetting the switch is bypassed.

When these actions are complete, the pre-termination routine enters the step termination routine through a branch.

STEP TERMINATION ROUTINE: The step termination routine performs its functions when a step has been terminated either normally due to successful completion of execution or abnormally due to an error condition. It uses five major routines:

- Step termination control routine.
- Step termination data set driver routine.
- Job statement condition code routine.
- Disposition and unallocation subroutine.
- User's accounting routine (if included in the configuration).

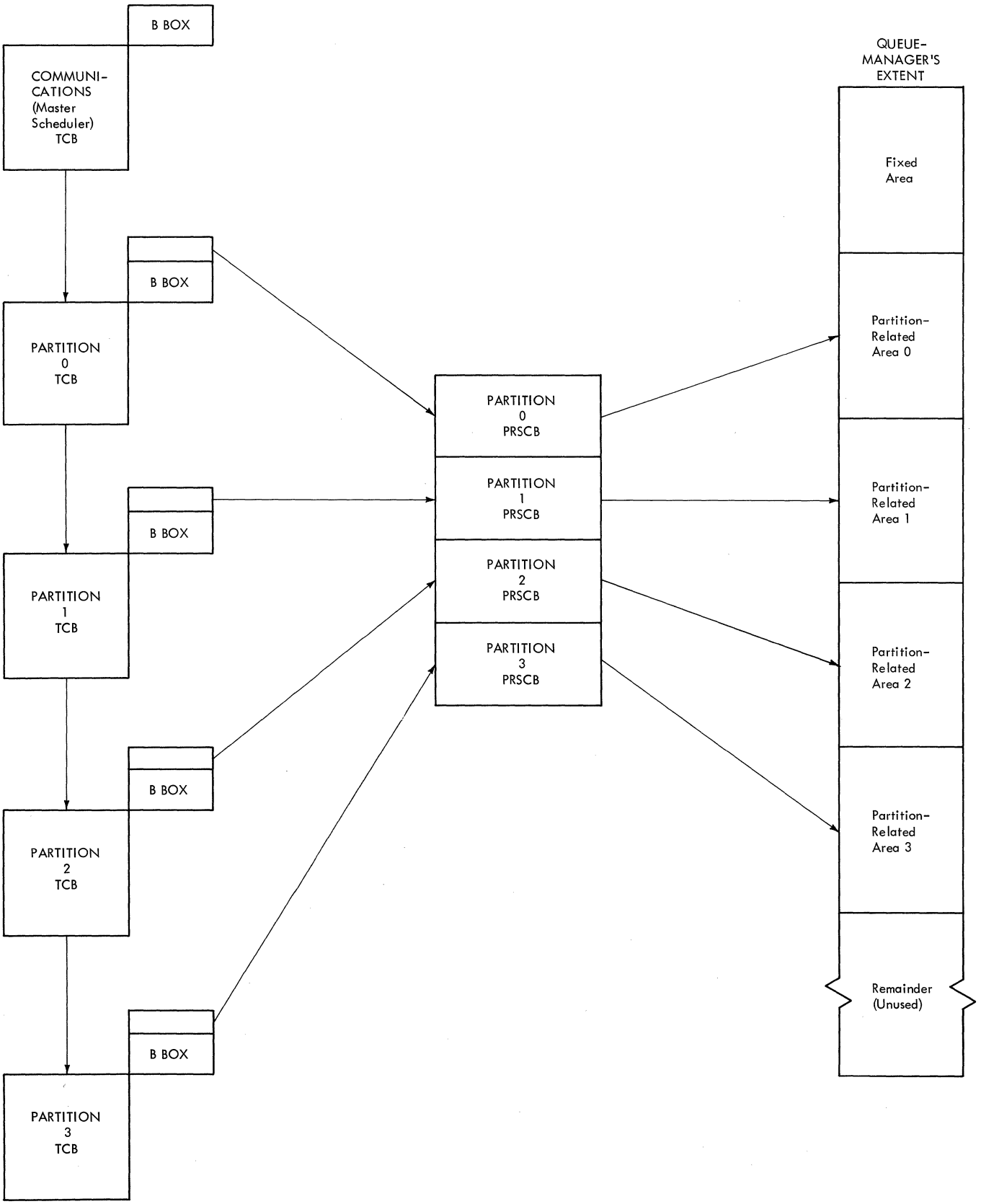


Figure 9. Control Block Relationships

Upon successful execution of a step or abnormal termination of execution, control is passed from the supervisor to the step termination control routine. In addition, when a job management routine encounters an irrecoverable error, it immediately passes control to the step termination control routine.

First, the initiator/terminator task input/output table (TIOT) and the linkage control table (LCT) are read into main storage. Next, the cancel ECB is set to zero in the selected job queue. The job control table (JCT) and the step control table (SCT) are then read into main storage (if they are not in main storage at the time), and a step status code is inserted into the SCT.

The step data set driver routine is then entered. It reads the step input/output table (SIOT) for each data set into main storage and branches to the disposition and unallocation subroutine. The loop through the data set driver routine and the disposition and unallocation subroutine is then repeated for each SIOT.

When all data sets have been processed by the disposition and unallocation subroutine, the updated SCT is returned to auxiliary storage. Control is then passed to the job statement condition code routine, unless it is known that there are no further steps for the job (the reader/interpreter had encountered a JOB or null statement). In the latter case the job statement condition code routine is bypassed.

The job statement condition code routine processes condition codes specified in the JOB statement.

If, upon entry into the job statement condition code routine, it is found that there were no condition codes specified in the JOB statement, control is returned to the step termination routine. Each condition code in the JCT for the job is in turn compared with the step completion code of the previous step, which appears in its SCT. Up to eight conditions are checked by this routine for each step. Any additional condition codes are ignored. If any of the condition operators are satisfied by the codes, the job-failed indicator in the JCT is updated to indicate that the job failed, the message subroutine is used to issue a message to the programmer, and control is returned to the step termination routine.

Upon return from the job statement condition code routine, or if it had been bypassed, the step termination routine exits to the user's accounting routine, if one is present. On return from the accounting routine, or if there was none, the step termination routine passes control to:

- The job termination routine, if the current step is known to be the last step of the job.
- The initiator/terminator system control routine, if additional steps have been interpreted and are ready to be initiated.
- The reader/interpreter control routine, which resumes processing the input job stream.

JOB TERMINATION ROUTINE: The job termination routine (Chart 15) performs its functions when an entire job has been executed and step termination for its last step has been completed. It consists of four major routines:

- Job termination control routine.
- Release job queue routine.
- Disposition and unallocation subroutine.
- User's accounting routine (if included in the configuration).

Control is passed to the job termination control routine from the step termination routine.

The job termination control routine determines if a passed data set queue exists and, if so, reads each block into main storage and tests for unreceived data sets. (An unreceived data set is a passed data set to which no reference is made after PASS is specified.) When an unreceived data set is found, entry is made to the disposition and unallocation subroutine. When all unreceived data sets have been processed, or if no passed data set queue exists, the job termination control routine passes control to the accounting routine, if there is one.

When the accounting routine returns, or if there is no accounting routine, the completed job's control tables are removed from the system by the release job queue routine. This routine releases the auxiliary storage space occupied by all control tables for the job. If the job notification switch is on, the message

IEF402I jobname ENDED

is written on the console device. Control is then passed to the shift count interrogation routine.

SHIFT COUNT INTERROGATION ROUTINE: For the Option 2 system, the shift count interrogator (Chart 16) is added as the final step of the job termination routine. If the scheduler is not already in partition 0 and the shift count is not zero, the count is decremented by one and control is passed to the scheduler upshift routine. Otherwise the shift count is zeroed out and control is passed to the reader/interpreter control routine.

SCHEDULER CONTROLLER

Acting in conjunction with the reader/interpreter and the initiator/terminator, the scheduler controller is the third element of the job scheduler. The function of the controller is to adjust the system job queue and monitor the operation of the reader/interpreter and initiator/terminator as required for multi-partition processing.

The system job queue is a data set containing control information produced by the reader/interpreter and used throughout job scheduling. The direct access area on which the data set resides is known as the queue-manager extent (see Figure 10). This extent is defined at system generation time and is initialized at nucleus initialization.

During initial reader/interpreter operations -- that is, up until the time when the first job in the input stream begins execution and issues a WAITR -- the contents of the queue-manager extent is organized as for the sequential scheduled system. The extent includes a fixed area (sometimes referred to as the "pre-empted track area") immediately followed by a variable area. Within the fixed area are, among other control fields, three key control tables: a link control table (LCT), a job control table (JCT), and a step control table (SCT). The variable area contains additional control fields and tables. (Each record in the variable area is fixed at 176 bytes; however, the number of records in the area can vary.)

Major scheduler control components are:

- The downshift routine, which reinitializes the scheduler for operation in the next-lower-priority partition.
- The upshift routine, which is entered when the scheduler is to be shifted to the next-higher-priority partition.

SCHEDULER DOWNSHIFT ROUTINE: The scheduler downshift routine (Chart 18) is entered as a result of WAITR issuance in the next-higher-priority partition. This routine reinitializes the scheduler for operation in the next-lower-priority partition, issues the message

PARTITION n STARTED

and exits to the reader/interpreter. The following paragraphs describe how preparation for scheduling in the second partition is performed. (Throughout the following discussion, 'Partition A' refers to the partition in which the WAITR was issued and which is relinquishing the scheduler. 'Partition B' refers to the next-lower-priority partition -- the partition to which the scheduler is being assigned.)

When the scheduler downshift routine is entered, the PRSCB for Partition B is cleared to zeroes, except for the complete flag in the scheduler-controlling ECB, which was just set on by the WAITR routine, and the partition identification byte which remains constant. The routine then gets main storage and reads in the LCT, JCT, and SCT from the queue-manager's extent on direct access. New job, link, and step control tables are constructed and read back into the fixed area; the tables that were read in from the fixed area are then written into the variable area associated with Partition A.

The variable area associated with Partition A now contains the scheduler information in the same state as when the scheduler was operating in that partition. The control information in the standard portion of the variable area is applicable only to Partition A and will not be affected by operation of the scheduler in another partition. The control information in those portions of the fixed area that are always referred to by the scheduler (the LCT, SCT, and JCT), regardless of what partition it is operating in, has been saved and the fixed area re-initialized for further use.

When this operation is complete, the pointers in the PRSCB for Partition A indicate (in TTR form) the location of:

SD33QSTT The beginning of Partition A's variable area.

SD33STTR The beginning of the LCT/SCT/JCT save area within that variable area.

SD33CTTR The next available TTR on the queue-manager's extent; i.e., the location beyond which the variable area for Partition B, if one is required, is to be built.

Control is then passed to the reader-interpreter.

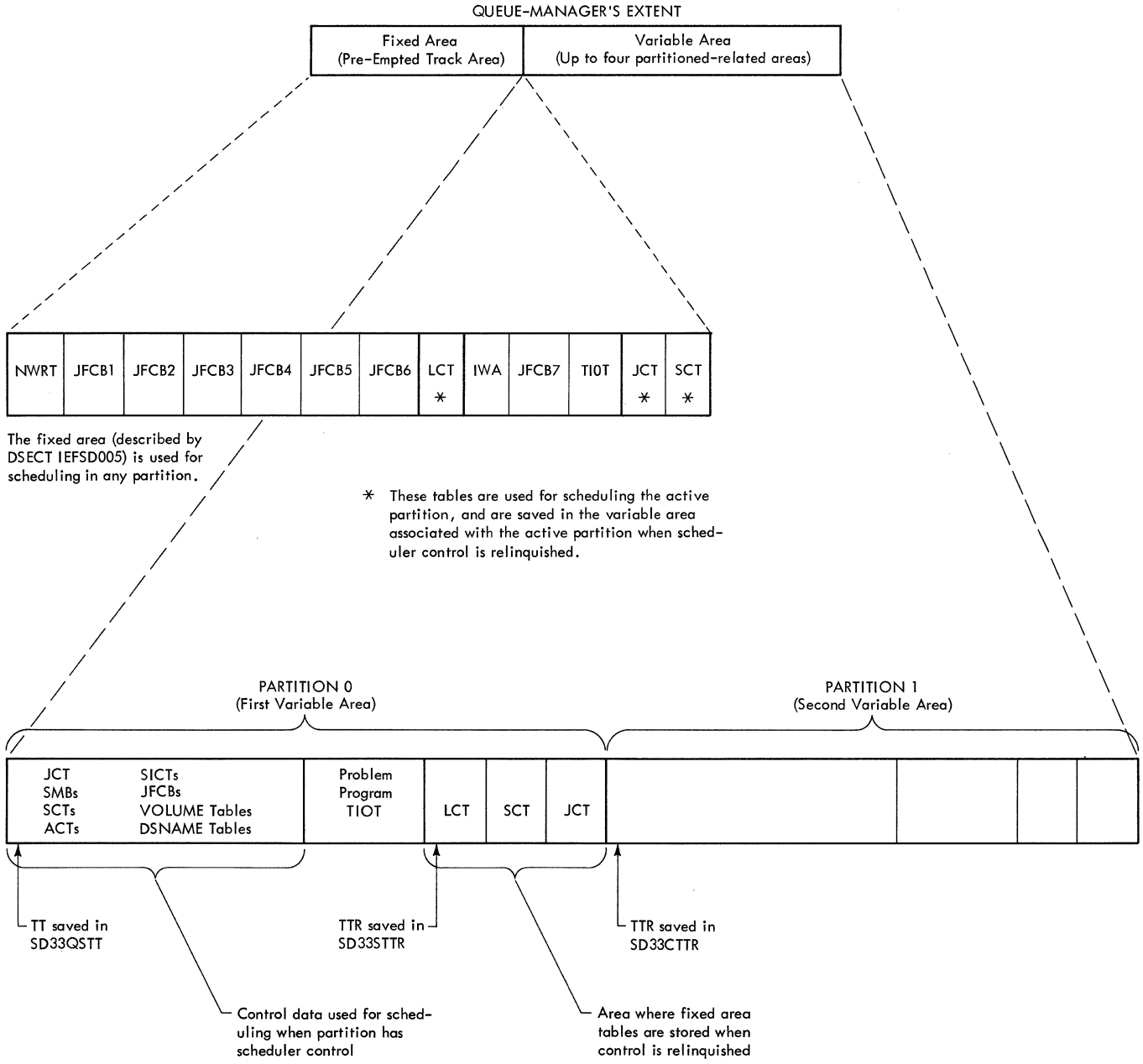


Figure 10. Queue-manager's Extent Layout

SCHEDULER UPSHIFT ROUTINE: The scheduler upshift routine (Chart 17) is entered from the shift count interrogator when job termination has been completed in a partition (Partition B, for purposes of discussion) and the scheduler is to be shifted to the next-higher-priority partition (Partition A) in response to a non-zero shift count.

When the scheduler upshift routine is entered, the last of the scheduler's termination routines has already issued a GETMAIN for main storage to be used by the reader/interpreter. This main storage is freed, and a GETMAIN is issued to obtain the main storage required by the pointer restore routine. Into this main storage is read the LCT, SCT, and JCT associated with Partition A.

After reading the required control tables into main storage, the routine writes them into the queue-manager's fixed area and resets the queue-manager's 'active area' pointer (SCATALLY) to the beginning of Partition A's variable area. The control information available to the queue

manager is now in exactly the same status as it was when scheduler control was initially relinquished.

With the scheduler switch complete, the routine posts the scheduler-controlling ECB for Partition A and issues a wait on the ECB for Partition B. This wait is satisfied if a subsequent job scheduled into Partition A issues a WAITR; if the wait is satisfied, the scheduler downshift routine is brought into Partition B and executed.

COMMUNICATION TASK

The communication task (Chart 02) processes all operator commands and messages directed to the operator through use of the WTO and WTOR macro-instructions. It also performs console switching when the secondary console is to be used in place of the primary console.

The eight major routines of the communication task are:

Console interrupt routine, which notifies the communication task wait routine that a console read has been requested.

Communication task wait routine, which waits for all WTO/WTOR requests and console interrupts and calls the communication task router routine.

Communication task router routine, which determines the type of request or interrupt that occurred and passes control to the appropriate processing routine.

Console device processor routine, which performs console read and write operations and error checking.

Master command processor routine, which processes all commands read from the console input device except SET, START RDR, and START WTR.

Master command routine, which analyzes command verbs and routes control to appropriate command execution routines.

Write-to-operator routine, which manages WTO buffers and requests console writes via the communication task wait routine.

External interrupt routine, which switches to the alternate console device when an external interruption occurs.

COMMUNICATION TASK CONTROL FLOW

Commands are issued through either the console I/O device or the input reader (see Figure 11). Before entering commands through the console I/O device, the operator must cause an I/O interruption. When he does, control is given to the supervisor which recognizes the interruption and passes control to the I/O supervisor. The I/O supervisor determines that the interruption is an attention signal and passes control to the master scheduler console interrupt routine.

The console interrupt routine resides in the nucleus. It posts the attention ECB in the unit control module (UCM) and sets the attention flag in the UCM list entry corresponding to the device from which the interrupt came. Posting of the attention ECB causes the communication task wait routine to be dispatched.

The communication task wait routine waits on all communication ECBs associated with WTO/WTOR. The wait module issues a multiple wait macro-instruction on a list of event control blocks contained in the UCM. When one of the event control blocks is posted, as by attention or external interrupts, the wait is satisfied and the communication task thus becomes ready. When it becomes the active task, it issues the SVC 72. This SVC includes the console communication service routines and the router.

Because the communication task serves a number of purposes, the first segment of SVC 72 is a routine that distinguishes among these purposes and establishes the order of response. This routine is called the router. The primary order of response is: external interruption, I/O completion, attention, and WTO(R).

When a posted ECB is found by the router, the router XCTLs to the specified processor module.

The console device processor routines perform reading and writing by using the EXCP macro-instruction. The processor routines consist of a routine to service external interruption and three device-oriented routines: 1052 operator console routine, card reader routine, and printer routine. With each of the three console I/O processor routines is associated an OPEN/CLOSE support routine, which provides Data Management and I/O Supervisor control blocks.

The specified processor routine reads the input message into a buffer area and calls the master command processor routine via an SVC.

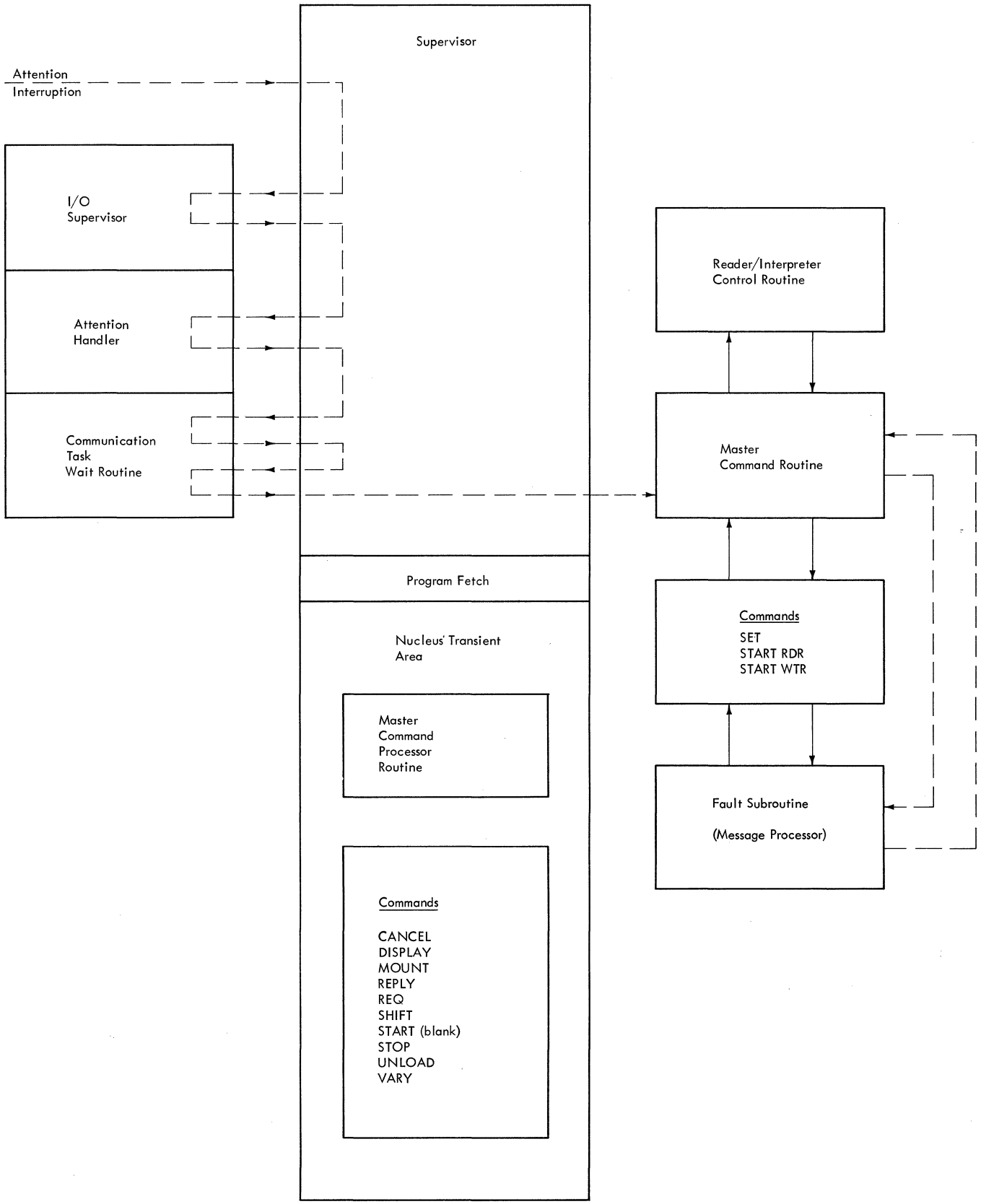


Figure 11. Communication Task Control Flow

The master command processor routine analyzes the command for validity. Ten commands (REQ, START (blank), CANCEL, DISPLAY, MOUNT, STOP, UNLOAD, VARY, SHIFT and REPLY) are always accepted and processed. All other commands are ignored (control is returned to the supervisor) if issued at any time other than in response to a message issued by the master command routine. If the command is acceptable, it is moved from the buffer into which it was read to a local buffer, and control is passed to the master command routine.

The master command routine analyzes commands and routes control to appropriate command execution routines. If a command is issued through the input job stream, control is passed directly to the master command routine by the reader/interpreter. When all commands have been entered and processed, control returns to the reader/interpreter.

The write-to-operator routine moves the text from the requesting program's area into a buffer area within the nucleus and posts the communication ECB for write-to-operator. If the request was a WTOR, a message ID is generated and a reply queue entry is created to allow handling of the reply by the operator.

The external interrupt routine assigns the functions performed by the primary console device to the alternate console device. When the operator presses the INTERRUPT key on the console, an external interruption occurs and control is given to the supervisor, which identifies the interruption and passes control to the external interrupt routine. The external interrupt routine then switches consoles and returns control to the supervisor. Console functions may later be reassigned to the primary console device if the operator causes another external interrupt.

CONSOLE INTERRUPT ROUTINE

The console attention interrupt routine (Chart 04) POSTs the communication task attention ECB to request reading of the console. The routine is logically part of IOS. It operates in privileged mode, I/O interrupt disabled, without destroying the registers, and without macro access to supervisor services. Using the pointer to the UCB found in register 7, the UCB address is matched to a UCM entry. The attention flag for the entry is turned on. A branch entry to POST pointing at the attention ECB in the UCM, is then taken. Register 14 is used to return to IOS.

COMMUNICATION TASK WAIT ROUTINE

The communication task wait routine (Chart 07) issues a WAIT to the list of ECB addresses contained in the Event Indication List (EIL). The communication task is thus able to respond to a variety of events since the POSTing of any one ECB satisfies the wait. The POST issued in the console attention interrupt routine satisfies the wait, and results in the placement of the TCB on the ready queue. When next dispatched, the wait routine issues an SVC 72 which results in: (1) the creation of an SVRB; and (2) the fetching of the first segment of the console processor routines into the system transient area.

COMMUNICATION TASK ROUTER

The router (Chart 08), IEECVCTR, is the first segment of SVC 72 brought into the transient area. Since the communication task serves a number of purposes, and since service requests may be simultaneously pending, the router establishes the order of response. The primary order of treatment is external interrupt, I/O completion, attention (console interrupt), and WTO(R). Multiple attentions are treated in order of appearance in the UCM. Multiple I/O completions are treated in order of first use of the device. The router responds to an attention by building a parameter list in the SVRB extended save area. It consists of a remote XCTL parameter list, a pointer to the appropriate UCM entry, and a pointer to the UCM (contents of CVTCUCB). The router then passes control to a processor routine by issuing an XCTL to the remote parameter list, using the name obtained from the UCB entry. The flag signifying the request to be serviced by the processor routine will be turned off by the processor routine. Consequently, processor routines return control to the router with XCTL to allow it to schedule service for other requests.

If no requests are pending, the router exits to the wait routine using the address in register 14.

In addition to distinguishing the output request from other requests, the router selects the particular device to which the message is to be sent. The router establishes the output device by interrogating UCB entry attribute indicators. The appropriate entry is the first active entry that supports WTO. As before, the router builds a remote interface for, and passes control via XCTL to, a processor routine.

CONSOLE DEVICE PROCESSOR ROUTINES

Control flow in a processor routine (Chart 10) is determined almost exclusively by the setting of flags in the router-selected UCM entry. The close flag is tested first. If this flag is on, any pending I/O activity is suspended by issuing a WAIT. An XCTL is then issued to an associated OPEN/CLOSE support routine for release of various control blocks. If the close flag is off, the busy flag is tested to determine I/O status. If there is outstanding I/O activity, error checking and buffer disposition occur if the activity has been POSTed complete. Otherwise, any attention request is temporarily abandoned (so are output requests), and an XCTL return to the router is taken. If the busy flag is off, the attention flag is tested, and if on, the status of the device is examined. If the device has not been opened, an XCTL to an associated OPEN/CLOSE support routine is issued for the purpose of obtaining core for a DCB and access-method dependent control blocks, and for execution of the OPEN macro.

When return is made from the OPEN/CLOSE support routine, a response to the attention flag is prepared. A fixed buffer in the UCB is reserved and an access-method dependent interface is constructed. I/O activity is initiated by issuing EXCP for a 1052, and by issuing a READ for a unit record device. In no case does the process routine await completion of this activity. Control is immediately returned to the router by issuing XCTL.

Control flow within the processor routine is as previously described up to the point at which the output request flag is tested. If on, the processor routine obtains the address of an output buffer from the UCM. The element is not removed from the queue at this time; this occurs only on successful completion of I/O. The reason is to preserve a natural method of having the message retried if an external interrupt intervenes before the message is successfully presented to the current device. Since output buffers are always selected from the top of the queue, the initiation of output to an alternate device would be unaffected by any previous attempts to present the message to the primary device.

Having selected a buffer, the processor routine establishes data management and IOS control block linkages; and issues EXCP for a 1052, or WRITE for a printer. Without awaiting completion of the I/O, the processor routine returns via XCTL to the router.

MASTER COMMAND PROCESSOR ROUTINE

The master command processor routine (Chart 05) processes the CANCEL, DISPLAY, MOUNT, REPLY, REQ, SHIFT, START (blank), STOP, UNLOAD, and VARY commands. It resides on the system residence device and is brought into the transient area of the nucleus by the supervisor when an SVC 34 instruction is issued by the communication task or the master command routine.

If the command is one of the ten previously mentioned commands, it is processed by the SVC 34 routine. SET, START RDR, and START WTR commands are ignored unless they were issued in response to a message from the master command routine. If so, control is passed to the master command routine, which processes them.

If entry to the master command processor routine was from the master command routine, the command is available in a buffer (placed there by the master command routine). The command is processed.

The master command processor routine returns control to the router.

MASTER COMMAND ROUTINE

The master command routine analyzes command verbs and routes control to appropriate command execution routines. It also issues a message to the operator, informing him that commands will be accepted from the console. The master command routine is brought into main storage and entered when any of the following occur:

- The reader/interpreter encounters a command in the input job stream.
- The reader/interpreter is performing the initialization procedures that follow IPL.
- The reader/interpreter finds the command pending switch on. (The command pending switch is turned on by the routine that processes the REQ command.)
- The reader/interpreter encounters an end-of-data set condition in the input job stream, indicating the end of a job step or job. Control is passed to the master command routine after the job step has been processed.

Upon entry to the master command routine, general register 0 is examined. If it contains zeros, entry was made because the reader/interpreter encountered a command in the input job stream. The command is moved to the master command routine

buffer and is written out on the console output device for the operator's records. The command verb is then analyzed: if it is a SET, START RDR, or START WTR command, control is passed to an appropriate command execution routine. Otherwise, an SVC 34 instruction is used to pass control to the master command EXCP routine.

If general register 0 does not contain zeros upon entry to the master command routine, the IPL pending, new reader pending, and new writer pending switches are checked. If any of these switches are on, the command pending switch is turned on and a message is issued requesting the operator to enter commands. Control is then passed to the initialization command routine, which provides certain commands, specified by the installation during system generation (SYSGEN), to relieve the operator of entering initialization commands. Each of the commands provided is moved to the master command routine buffer, written on the console output device for the operator's records, and executed.

If general register 0 does not contain zeros and none of the previously-mentioned pending switches are on, entry to this routine was made because the reader/interpreter found the command pending switch on, or encountered an end-of-data set condition in the input job stream. A message is issued requesting commands from the operator. After the operator has issued commands and they have been processed, control is returned to the reader/interpreter.

WRITE-TO-OPERATOR ROUTINE

The write-to-operator routine (Chart 06) writes operator messages on the console output device when a WTO or WTOR macro-instruction is issued. These macro-instructions may be issued by the system component programs and processing programs. Messages and replies are buffered; the period of time between the message and the reply is available for processing. Issuance of either macro-instruction causes an SVC interruption. When the interruption is handled, the supervisor has the routine read into the transient area of the nucleus and passes control to it.

There are two console queues: the buffer queue and the reply queue. Each WTO and WTOR results in the addition of a WTO Queue Element (WQE) to the buffer queue, and each WTOR results in the addition of a Reply Queue Element (RPQE) to the reply queue. WTO and WTOR represent requests to present a message to the operator. SVC 35 sets up the user's messages and, if WTOR, inserts the message identification (ID) which the operator must use for his reply. The same message ID is placed in the RPQE with other information to insure passing the reply, when received, to the proper area. WTO messages are invariably written out; a WTOR message may be purged (removed from the queue) if the issuing task terminates while the message is on the buffer queue. Therefore, an RPQE differs from a WQE in that it contains the address of the issuing task's TCB. The buffer queue is accessed through the entry UCMWTOQ in the UCM.

The reply queue contains RPQEs for operator replies to WTOR. Elements in this queue, like WTOR elements in the buffer queue, contain a TCB address to permit purging.

The extent of both queues is limited by specifying the number of buffers at system generation. An attempt to exceed a threshold value will result in an ENQ of the requesting task.

For a reply (to WTOR), the processor issues SVC 34 (command processing). The SVC routine determines that the incoming command is in fact a reply, processes the reply, POSTs the user's ECB and branches back to the processor.

EXTERNAL INTERRUPT ROUTINE

The external interrupt routine (Chart 04) switches to an alternate console device when the operator presses the INTERRUPT key on the console. This routine resides in the nucleus.

SUPERVISOR MODIFICATIONS

WAITR--SINGLE EVENT

For the Option 2 system, the WAIT service routine also processes WAITR macro-instructions issued by a processing program to cause job management to be initiated in the next-lower-priority partition. If, when the routine is entered, the wait count is negative -- i.e., has been complemented -- a WAITR has been issued. The routine determines whether the WAITR is the first that has been issued by the processing program. If the WAITR is not the first, or if the WAITR has been issued in the lowest-priority partition (from which no down shift is possible), the WAITR is treated as a WAIT with the same parameters.

When a first WAITR is encountered and there is a next-lower-priority partition, the routine makes the task associated with that partition dispatchable. When that task is dispatched, job management routines are entered to cause a job to be scheduled into the partition.

When a first WAITR is serviced, a switch is set so that any subsequent WAITR issued in the same partition is treated as a WAIT. This switch is reset only upon termination of the job.

NUCLEUS INITIALIZATION PROGRAM

The primary change in the operation of NIP under control program Option 2 is that the standard partition initialization functions are repeated for each partition in the system. For each partition, just as for the single partition that exists without Option 2, a boundary box, a free area queue element, a PRB, and the required XCTL code are established. For a full explanation of the nucleus initialization program, including partition initialization, refer to IBM System/360 Operating System: Fixed-Task Supervisor, Program Logic Manual.

ENQ/DEQ SUPPORT

Enqueue Service Routine--IEAQENQ0

This routine (see Charts 19 through 22) is entered through a branch from a system routine, or from the SVC second-level interrupt handler in response to an ENQ (SVC 56). When the routine is entered, the major and minor QCBs are searched for existing control blocks representing the requested resource. If the required major and/or minor QCB are not found, the routine takes the action appropriate to the RET=parameter, as follows:

- RET=TEST -- the routine sets a return code of 00 (resource is available).
- RET=USE or HAVE -- the routine sets a return code of 00. The routine issues a GETMAIN and creates a queue element. A minor QCB or a minor and a major QCB is created if required.
- RET=NONE (or parameter left blank) -- no return code is set by the routine; control blocks are constructed as for RET=USE.

When the required action is complete, the routine branches to the pre-exit subroutine described below, or begins again with the queue search if additional requests are to be processed.

Pre-Exit Subroutine: This subroutine (TESTEND1 and TESTEND2 in CSECT IGC048) is entered to determine if the calling task can proceed. The task can always proceed if the RET=TEST parameter was used. Register 15 is set and control is returned to the task. If the SVRB wait count is not zero, the registers are saved in the TCB, the resume PSW is set to the address of the SMC test, the new PSW is set to zero, and the routine then branches to the dispatcher. The return codes are set and control is returned to the calling task if the SVRB wait count is zero and must-complete is not requested. If must-complete is requested, the routine proceeds as described below.

If the specified major QCB is found, the routine searches the major QCB's queue of minor QCBs for the specified minor name. If the minor QCB is not found, a return code is set and/or control blocks are created as explained above. If the minor QCB is found the queue elements queued on the minor QCB are searched for another queue element for the enqueueing task chained to the same minor QCB. Such a duplicate queue element indicates that the task has attempted to enqueue twice on the same resource without an intervening dequeue. If a duplicate request is encountered, the routine causes the task to be abnormally terminated unless the new request is an inquiry (RET=HAVE, USE, or TEST). If the request is an inquiry, a return code of 08 is set and a subroutine is entered to determine whether the request includes a must-complete requirement.

For a non-duplicate request, the routine determines whether all queue elements already enqueued on the minor QCB are "shared" and whether this is also a "shared" request. If both conditions are true, a queue element is created, the count field in the TCB (TCBCT) is incremented by one for each resource enqueued upon, and a return code is set and/or QCBs are created as explained above.

If a queue element representing an "exclusive" request is already enqueued on the resource, the wait count in the SVRB associated with the new request is incremented by one. This wait count, which will be decremented by the dequeue service routine when the exclusive request is satisfied, causes the requesting task to wait in the enqueue routine but does not affect the dispatchability of the task as a whole. Asynchronous routines, called by IRBs added to the TCB's request block chain, can still operate under the task's control.

The wait count is not incremented if the RET=USE parameter was included. In that case, the routine sets the "resource in use" return code and processes any further requests or proceeds to the pre-exit subroutine.

If must-complete was specified and the requesting task is a system task (rather than a user task, which would be abnormally terminated if 'set-must-complete' were specified), the subroutine sets on the

must-complete flag in the queue element and waits until any preceding requests on the queue have been dequeued. During this period the must-complete condition is not in effect. The flag in the queue element is set on to indicate that the condition is to be imposed, but only after use of the resource has actually begun.

When the queue element containing the must-complete flag reaches the top of the resource queue -- that is, when the resource becomes available to the task that requested the resource and the must-complete restriction -- the step or system must-complete flag is set on in the task's TCB and all other TCBs in the system are made non-dispatchable. This ensures that the task that imposed the must-complete restriction will be the only task operating until the restriction is lifted, through issuance of a release-must-complete in that task.

An exception arises if the system interrupt request block (SIRB) has been placed in the RB chain of another task. In that case, the task under which the SIRB is running is not set non-dispatchable, but a flag is set on in the exit routine of the supervisor's exit and transient area handler. The two tasks operate concurrently until the restriction is lifted by the responsible task (upon DEQ), or the task under which the SIRB is being serviced exits. Exit from the SIRB causes the task for which non-dispatchability was deferred to be set non-dispatchable.

Dequeue Service Routine

The dequeue service routine (see Charts 23 through 25) is entered through the SVC second-level interrupt handler in response to a DEQ (SVC 48), or through a branch from a system routine. The function of the dequeue service routine is to remove from the list of pending requests a request that has been satisfied, and to cause the next request (if any) on the list to be serviced. In addition, the routine resets the must-complete condition when a reset is specified by a system task.

After performing initial validity checks (Chart 26), the routine searches the major and minor QCB queues for the control blocks corresponding to the major and minor names specified by the requester. If the required QCBs are not found, the action taken is determined by the value of the RET=parameter:

- If RET=HAVE, the request was conditional. A return code of 08 is set to indicate that the task in which the DEQ was issued was never enqueued upon the resource, and the routine proceeds to check for more parameter list entries to process.

- If RET=NONE or the parameter was omitted, the task in which the DEQ was issued is abnormally terminated with an error code of 130.

If the specified major QCB and minor QCB are found, the queue elements enqueued on the minor QCB are examined to determine whether a dequeue can be performed, and whether, if a dequeue cannot be performed, a return code is to be provided or the dequeueing task is to be abnormally terminated.

A dequeue can be performed if the queue element enqueued by the task issuing the dequeue request is:

- An exclusive request at the head of the queue, or

- A shared request in any position preceding the first exclusive request on the queue.

If either of those two conditions is met, the routine proceeds to dequeue the element.

If these conditions are not met, there are two possibilities: either the queue element being sought by the dequeueing task is not in the queue, or it is in the queue but has never been serviced. If the queue element is not in the queue, the routine sets a return code of 08 and continues if RET=HAVE was specified, or abnormally terminates the dequeueing task. If the element is in the queue but has never been serviced, the routine:

1. sets a return code of 04 and proceeds to the next item in the parameter list or

2. abnormally terminates the task

depending on the RET=parameter. The 04 return code in this case indicates that the request was not at the top of the queue and is exclusive, or is shared but is preceded on the queue by an exclusive request.

When a queue element to be dequeued is found, the count field in the TCB (TCBCT) is decremented by one and the queue element is removed from the queue. The TCBCT field is a record of the number of outstanding requests associated with the task. The count is incremented by 1 for each resource enqueued upon when the task issues an ENQ and decremented by 1 for each resource dequeued when a DEQ is issued by the task. This field is referred to, if the task is

abnormally terminated, to determine when all outstanding requests have been purged.

If there are no more requests remaining on the minor QCB's queue after the queue element is dequeued, the minor QCB itself is dequeued from the major QCB; similarly, if no additional minor QCBs remain, the major QCB is removed from the chain of major QCBs. A FREEMAIN is then issued, releasing the main storage formerly occupied by the removed control blocks.

The presence of another queue element after the element removed through the DEQ means that the resource is now to be made available to the next enqueued task(s). If the next queue element represents an exclusive request, the DECSVRB subroutine is entered (see below) to enable the requesting task to receive control. If the next queue element represents a shared request and the previous queue element was exclusive, the same function is performed not only for the task associated with the shared queue element, but for all subsequent shared tasks in the queue as well, until the end of the queue or an exclusive request is reached. After preparing for the receipt of control by the necessary task or tasks, the routine frees the main storage used for any removed control blocks and proceeds.

The DECSVRB subroutine (Chart 25) deals with a queue element that has just become the first element of the resource queue, or with a shared queue element not preceded by an exclusive request and therefore effectively at the top of the queue. The wait count in the SVRB associated with the queue element is examined. If the wait count is already zero (not the normal case), the subroutine exits. Otherwise, the wait count is decremented by one. If this does not reduce the wait count to zero, the enqueued task is still waiting for other resources and cannot, therefore, receive control; the subroutine exits. But if the wait count in the SVRB does reduce to zero, the enqueued task now has available to it all of the resources it requires and can receive control. A task switch is effected if the now-ready task is of higher priority than the task (pointed to by the NEW task control block address pointer in the communication vector table) last in control. If the enqueued task is of lower priority, no task switch occurs. In either case, however, the zero SVRB wait count makes it possible for the task to proceed when next dispatched.

After the FREEMAIN operation for removed control blocks is complete, the routine loops back to process any further elements on the parameter list, or proceeds to reset must-complete (if required), check for return codes, and exit. Exit takes one of two paths: either to the caller (the task in which the DEQ was issued), or to the newly ready task (the task in which the ENQ was originally issued). If the contents of NEW have been changed by the dequeue routine, the dispatcher performs the required task switch by giving control to the routine in which the ENQ had been issued.

Major and minor QCB's are moved to the partition represented by the next QEL when they reside in the partition that is issuing the DEQ. (See Figure 8.)

DADSM MODIFICATIONS

To provide volume table of contents (VTOC) integrity in a multi-task environment, the DADSM allocate, extend, scratch, and release routines use the ENQ and DEQ macro-instructions and the must-complete options thereof to ensure that no task other than the task performing a VTOC update will access the VTOC while the update is in progress. The manner in which the ENQ and DEQ macro-instructions are used is summarized below. For further information on those routines, refer to the publication IBM System/360 Operating System: Direct-Access Device Space Management, Program Logic Manual; Form Y28-6607.

Note: Except where specifically noted, the resource to which ENQ and DEQ requests relate is the VTOC for the volume upon which the DADSM routines are operating.

Allocate Routines--Non-Indexed Sequential Data Sets

On entry to the allocate routine, an ENQ is issued by the duplicate name search routine. The must-complete condition is subsequently set in (1) the sub-allocation routine, or (2) the DSCB creation routine. A DEQ is issued and the must-complete condition is reset in the VTOC updating routine.

Allocate Routines--Indexed Sequential (ISAM) Data Sets

An ENQ is issued by the duplicate name search routine. The must-complete condition is set in either (1) the DSCB build routine, (2) the duplicate format 1 action routine, or (3) the embedded index routine. A DEQ is issued and the must-complete condition is reset by the completion of processing routine.

If additional volumes are to be processed, an ENQ specifying the VTOC for the next volume is issued before the allocate routines are re-entered for that volume.

Extend Routines

An ENQ is issued and the must-complete condition is set by the duplicate name search routine. The VTOC updating issues a DEQ and resets the must-complete condition.

Scratch Routine

An ENQ is issued and the must-complete condition is set by the UCB search routine; DEQ is issued and the must-complete condition is reset by the VTOC updating routine. This process is repeated on each pass through the routine for a multi-volume data set.

Release Routine

An ENQ is issued and must-complete condition is set by the first module of the release routine. DEQ is issued and must-complete condition is reset by the close routine of I/O Support, to which the release routine transfers control when VTOC updating is complete.

This section lists job management load modules and indicates the assembly modules that are processed by the linkage editor into each load module during system generation. Included is a separate list that shows the load modules in which each assembly module is contained.

Job management routines for Option 2 of the control program are packaged in two configurations: 18K and 44K (where K is 1024 bytes of main storage). The numbers represent the maximum amount of main storage occupied by job management routines and work areas at any time. Both job management configurations function identically but differ in both the number of load modules and the number of assembly modules within each load module.

The configuration chosen at system generation determines the minimum partition size in the Option 2 system. Job management routines occupy each partition of main storage alternately with processing programs, therefore the scheduler size determines the minimum size permissible for each partition.

LOAD MODULES

In each configuration, all load modules are contained in three data sets: SYS1.NUCLEUS, SYS1.SVCLIB and SYS1.LINKLIB. These data sets also contain other parts of the control program. The load modules in the first two data sets remain the same for both job management configurations, but the SYS1.LINKLIB data set contains a different set of load modules for each configuration, depending on which one was selected at system generation time. In the 18K configuration, LINKLIB contains 36 load modules; in the 44K configuration, it contains 25 load modules.

Charts 27 and 28 show the control flow among load modules. The decision to transfer control (XCTL) to a particular succeeding load module is made in the previous load module. Each subsequent module loaded in response to an XCTL macro-instruction is read into main storage directly over the previous load module. Such load modules are read into the low-numbered end of the partition in which job scheduling is being performed.

Modules that are brought into storage with LINK macro-instructions and LOAD macro-instructions occupy separate storage areas within the partition; such modules are shown on the control-flow charts. Because storage is used in this manner, the load module lists may be used with chart 27 or 28 to determine the approximate layout of a partition at different times during the execution of job management routines. Other items present in the partition at the same time as the load modules are not shown on the control flow charts because, although these items are necessary, control is not passed among them. They are, generally, the tables and control blocks, work areas, access methods, buffers, and register save areas.

In the following load module lists, entry points are shown if a load module contains more than one assembly module. If only one assembly module is named, the entry point is the same as the assembly module's control section (CSECT) name given in the Assembly Modules and Control Sections table in this section.

LOAD MODULES CONTAINED IN THE SYS1.NUCLEUS DATA SET

The load modules and assembly modules in the following list are contained in the SYS1.NUCLEUS data set, and are always present in the nucleus, or system area of main storage, regardless of the job management configuration.

Load Module Name: SYS1.NUCLEUS
Assembly Modules:

IEEBC1PE	External interrupt routine.
IEECIR01	Console interrupt routine.
IEERSC01	Master scheduler buffers, switches, input/output block (IOB), event control block (ECB), channel control word (CCW), and device end block (DEB). This load module forms master scheduler resident main storage in the nucleus area when the primary or alternate console (1052) is used.
IEERSR01	Master scheduler buffers, switches, IOB, ECB, CCW, and DEB. This load module forms master scheduler resident main storage in the nucleus area when the composite console is used.

IEFDPOST Unsolicited-interrupt routine.
MCONRESA Table store subroutine work area.
IEECVCRX External Interrupt Routine (Option 2)
IEECVCRA Console Interrupt Routine (Option 2)
IEECVUCM Communication Task buffers, switches, input/output blocks (IOB), event control blocks (ECB), device end blocks (DEB), and data control blocks (DCB). This data area is used for operator communication in Option 2 systems.
IEECVPRG Operator communication reply queue purging routine (Option 2).
IEECVCTW Communication Task Wait Module.

LOAD MODULES CONTAINED IN THE SYS1.SVCLIB DATA SET

The load modules and assembly modules in the following list are contained in the SYS1.SVCLIB data set, and are called in response to SVC instructions.

Load Module Name: IGC0003D
Assembly Modules:
IEEMXC01 Master command EXCP routine (Part 1) -- primary/alternate console.
IEEMXR01 Master command EXCP routine (Part 1) -- composite console.
IEEMCP01 Option 2 Master Command EXCP routine (overlay module).

Load Module Name: IEE1203D
Assembly Module:
IEE1203D Option 2 Master Command Reply Processor (overlay module).

Load Module Name: IGC0007B
Assembly Module:
IEECVCTR Communication Task Router module.

Load Module Name: IGC0107B
Assembly Module:
IEECVPMX Option 2 Communication Task Process module -- access method EXCP (1052).

Load Module Name: IGC0113D
Assembly Module:
IEECVPMC Option 2 Communication Task Process module -- access method BSAM (2540).

Load Module Name: IGC2103D
Assembly Module:
IEECVPMP Option 2 Communication Task Process module -- access method BSAM (1443).

Load Module Name: IGC0I03D
Assembly Module:
IEECVOCX Option 2 Console unit initialization EXCP input/output.

Load Module Name: IGC1I03D
Assembly Module:
IEECVOCC Option 2 Console unit initialization BSAM input.

Load Module Name: IGC2I03D
Assembly Module:
IEECVOCP Option 2 Console unit initialization BSAM output.

Load Module Name: IGCX07B
Assembly Module:
IEECVCTX Option 2 Communication Task external interrupt processor.

Load Module Name: IGC0003E
Assembly Modules:
IEEWTC01 Write-to-operator (WTO) routine -- primary/alternate console.
IEEWTR01 Write-to-operator (WTO) routine -- composite console.
IEECVWTO Option 2 WTO/WTOR queueing routine.

Load Module Name: IGC0103D
Assembly Module:
IGC0103D Master command EXCP routine (Part 2), or command processing routine.
IGC0113D Option 2 Master Command EXCP routine (overlay module).

Load Module Name: IGC0003F
Assembly Module:
IEEBH1PE Not used in sequential scheduling system.

MODULES CONTAINED IN THE SYS1.LINKLIB DATA SET

The load modules and assembly modules in the following lists are contained in the SYS1.LINKLIB data set. A list is provided for both of the packaging configurations in which job management routines are available.

Load Module Name: IEECVCTI
Assembly Modules:

IEECVCTI	Option 2 Communication Task Initialization routine.
IEEVRFRX	Option 2 CVT, TCB, RB, TIOT, and UCB look-up module.

Load Module Name: IEFSTERM
Alias: IEFYN
Alias: GO
Entry Point: IEFSD034
Assembly Modules:

IEFSD011	Entry to job management from supervisor.
IEFW42SD	Passes control to IEFIDUMP (in IEFIDUMP Load Module) if necessary, or to IEFYNIMP (in this module).
IEFYNIMP	Step termination routine.
IEFYPJB3	Step data set driver routine.
IEFVJIMP	Job statement condition code routine.
IEFZGST1	Disposition and unallocation subroutine.
IEFACTLK	Linkage to user's accounting routine.
IEFACTRT	Dummy, to be replaced by user's accounting routine.

(The preceding two modules may be replaced by IEFACTFK assembly module if no accounting routine is specified as a system generation option.)

IEFSD017	Places logical track address (TTR) of first system message block (SMB) into job control table (JCT).
IEFW22SD	Passes control to IEFYNIMP (in this load module), then to IEFSD002 (in this load module) or to IEFZAJB3 (in IEFJTERM load module).
IEFSD002	Exit to IEF08FAK or IEF09FAK (both in this load module).
IEFSD006	Converts record number to logical track address (TTR).
IEFSD007	Call to table store subroutine.
IEFSD034	Pre-termination exits to IEFSD011
IEFYSSMB	Message enqueuing routine, enqueues SMBs.
IEFQMSSS	Table store subroutine.
IEFVJMSG	Contains initiator terminator messages.
IEFYNMSG	Contains initiator terminator messages.
IEFYPMSG	Contains initiator terminator messages.
IEFZGMSG	Contains initiator terminator messages.
IEFZHMSG	Contains initiator terminator messages.
IEFIDFAK	Linkage to IEFIDUMP (in IEFIDUMP load module).
IEFZAFAK	Linkage to IEFZAJB3 (in IEFJTERM load module).
IEF08FAK	Linkage to IEFSD008 (in IEFINTFC load module).
IEF09FAK	Linkage to IEFSD009 (in IEFSELCT load module).

Load Module Name: IEFSELCT
Alias: IEFSD009
Entry Point: IEFSD009
Assembly Modules:

IEFSD006	Converts record number to logical track address (TTR).
IEFSD009	Initializes initiator/terminator.
IEFW21SD	System control routine.
IEFVKIMP	Execute statement condition code routine.
IEFVMLS1	JFCB housekeeping control routine.
IEFVM2LS	JFCB housekeeping fetch DCB routine.
IEFVM3LS	JFCB housekeeping generation data group (GDG) single routine.
IEFVM4LS	JFCB housekeeping generation data group (GDG) all routine.
IEFVM5LS	JFCB housekeeping patterning data set control block (DSCB) routine.
IEFVM76	Processes passed, non-labeled tape data sets.
IEFWSTRT	Job started message routine.
IEFYSSMB	Message enqueuing routine, enqueues SMBs.
IEFQMSSS	Table store subroutine.
IEFWMAS1	Device name table.
IEFVKMSG	Contains initiator terminator messages.
IEFVMLK5	Linkage to IEFVMLS6 (in IEFERROR load module).
IEFXAFAK	Linkage to IEFXCSSS (in IEFALOC1 load module).
IEFYNFAK	Linkage to IEFYNIMP (in IEFSTERM load module).

Load Module Name: IEFALOC1
Alias: IEFXJ000
Alias: IEFXA
Entry Point: IEFXA
Assembly Modules:

IEFXCSS	Allocation control routine.
IEFXJIMP	Allocation error recovery routine.
IEFYSSMB	Message enqueuing routine.
IEFQMSSS	Table store subroutine.
IEFXAMSG	Contains initiator terminator messages.
IEFXJMSG	Contains initiator terminator messages.

IEFWAFAK Linkage to IEFWA000 (in IEFALOC2 load module).
IEFWCFAK Linkage to IEFWCIMP (in IEFALOC3 load module).
IEFYNFAK Linkage to IEFYNIMP (in IEFSTERM load module).

Load Module Name: IEFALOC2
Alias: IEFWA000
Entry Point: IEFWA000
Assembly Modules:
IEFWA000 Demand allocation routine.
IEFWSWIN Passes control to decision allocation or automatic volume recognition (AVR) routine.
IEFX5000 Decision allocation routine.
IEFX300A Device strikeout routine.
IEFXH000 Separation strikeout routine.
IEFWMSKA Device mask table.
IEFWCFAK Linkage to IEFWCIMP (in IEFALOC3 load module).
IEFXJFAK Linkage to IEFXCSSS (in IEFALOC1 load module).
IEFS15XL Check for duplicate allocation.

Load Module Name: IEFALOC3
Alias: IEFWC000
Entry Point: IEFWC000
Assembly Modules:
IEFWCIMP Task Input/Output Table construction routine.
IEFXH000 Separation strikeout routine.
IEFWDFAK Linkage to IEFWD000 (in IEFALOC4 module).
IEFXJFAK Linkage to IEFXCSSS (in IEFALOC1 module).

Load Module Name: IEFALOC4
Alias: IEFWD000
Entry Point: IEFWD000
Assembly Modules:
IEFWD000 External action routine.
IEFWD001 Message directory for external action routine.
IEFXT00D Space request routine.
IEFXKIMP Allocation error nonrecovery routine.
IEFXTDMY Queue overflow routine.
IEFYSSMB Message enqueuing routine, enqueues SMBs.
IEFQMSSS Table store subroutine.
IEFXKMSG Contains initiator terminator messages.
IEFXTMSG Contains initiator terminator messages.
IEFW41SD Exit to IEF04FAK (in this load module).
IEF04FAK Linkage to IEFSD004 (in IEFATACH load module).
IEFYNFAK Linkage to IEFYNIMP (in IEFSTERM load module).
IEFSD006 Converts record number to logical track address (TTR).

Load Module Name: IEFATACH
Alias: IEFSD004
Entry Point: IEFSD004
Assembly Modules:
IEFSD004 Step initiation routine, with exit to processing program.
IEFSD006 Converts record number to logical track address (TTR).
IEFSD007 Call to table store subroutine.
IEFSD010 Dequeues and writes out system message blocks (SMBs).
IEFQMSSS Table store subroutine.

Load Module Name: IEFCNTRL
Alias: IEF5DDHD
Alias: IEFMF
Alias: IEFMC
Alias: IEFKA
Entry Point: INDMRTN
Assembly Modules:
IEF7KAXX Reader/interpreter control routine.
IEF6DDHD DD routine.
IEF6BOCM Breakout routine.
IEF6MFXX Verb identification routine.
IEF6MCXX Scans job control language (JCL) statements.
IEF6STNM Scan stepname routine.
IEF6NAME Qualified name routine.
IEF6FRRS Resolves DD forward references.
IEF6DCB0 DCB refer-back routine.
IEF6MKXX Continuation routine.
IEF7MMCM Reader/interpreter message routine.
IEFSD006 Converts record number to logical track address (TTR).
IEF6MIXX Reader/interpreter call to IEFQMSSS.
IEFQMSSS Table store subroutine.
IEFK4DUM Linkage to IEFK4ENT (in IEFK4 load module).
IEF6DHX1 Linkage to IEF6SCAN (in IEFDD load module).
IEFKLDUM Linkage to IEF6KLXX (in IEF1STMT load module).
IEFJMDUM Linkage to IEF6NCJB (in IEFJOB load module).
IEFEMDUM Linkage to IEF6NJEX (in IEFEXEC load module).
IEF6OUT2 DD output routine, with exit to IEF7KAXX (in this load module).
IEFKGDUM Linkage to IEF7KGXX (in IEFINTFC load module).
IEFKPDUM Linkage to IEF7KPXX (in IEFCOMND load module).
IEFK3DUM Linkage to IEF7K3XX (in IEFEOF load module).

Load Module Name: IEFDD
Alias: IEF5SCAN
Entry Point: INDMON
Assembly Modules:
IEF6SCAN DD scan routine.
IEF6BOCM Breakout routine.
IEFSD012 DD* statement routine.
IEF6DDNM DD name routine.
IEF6DSNM DS name routine.
IEF6RFWD Processes DD forward references.
IEF6RTPR Right parenthesis routine.
IEF6LFPR Left parenthesis routine.
IEF6EQUL Equal sign routine.

IEF6LIST Subparameter list routine.
IEF6NLST Routine for no subparameter list.
IEF6NDDP DD parameter list table.
IEF6DCDP Data control block (DCB) DD parameter list table.
IEFSD013 Assigns unit to system output (SYSOUT).
IEF6ORDR Order subroutine.
IEF6INST Insert routine.
IEF6VALU Value subroutine.
IEF6CLNP Clean up after DD routine.
IEFSD006 Converts record number to logical track address (TTR).
IEF6ERR1 DD error-handling routine.
IEF7MMCM Reader/interpreter message routine.
IEF6MIXX Reader/interpreter call to IEFQMSSS
IEFQMSSS Table store subroutine.
IEF6CN17 Linkage to IEF6DDHD (in IEFCNTRL load module).

Load Module Name: IEFINTFC
Alias: IEFSD008
Alias: IEFSD001
Alias: IEFKG
Entry Point: IEFSD008
Assembly Modules:
IEFSD008 Initiator/terminator to reader/interpreter interface.
IEF7KGXX Output tables for step.
IEFSD006 Converts record number to logical track address (TTR).
IEFSD007 Call to table store subroutine.
IEEMCS01 Master command routine.
IEF7MMCM Reader/interpreter message routine.
IEF6MIXX Reader/interpreter call to IEFQMSSS.
IEFQMSSS Table store subroutine.
IEFSD001 Reader/interpreter entry to IEF09FAK or to IEFW23SD (both in this load module).
IEF09FAK Linkage to IEFSD009 (in IEFSELCT load module).
IEF23FAK Linkage to IEFW23SD (in IEFJTERM load module).
IEFMFDUM Linkage to IEF6MFXX (in IEFCNTRL load module).
IEFKADUM Linkage to IEF7KAXX (in IEFCNTRL load module).
IEFK3DUM Linkage to IEF7K3XX (in IEFEOF load module).

Load Module Name: IEFEXEC
Alias: IEFEM
Entry Point: IEFEM
Assembly Modules:
IEF6NJEX Execute (EXEC) statement routine.
IEF6BOCM Breakout routine.
IEF6STNM Scan stepname routine.
IEF6NAME Qualified name routine.
IEF6RFBK Refer-back routine.
IEF6PROC Procedure name routine.
IEF6TIME TIME keyword routine.
IEF6COND Condition (COND) keyword routine.
IEF6PARM Parameter (PARM) keyword routine.
IEF6NFCM Accounting information routine.
IEF7MMCM Reader/interpreter message routine.
IEF6MIXX Reader/interpreter call to IEFQMSSS.
IEFQMSSS Table store subroutine.
IEF8LINK Linkage to IEF6COND (in this module).
IEFMFDUM Linkage to IEF6MFXX (in IEFCNTRL load module).

Load Module Name: IEFJOB
Alias: IEFJM
Entry Point: IEFJM
Assembly Modules:
IEF6NCJB Job (JOB) statement routine.
IEF6BOCM Breakout routine.
IEF6STNM Scan stepname routine.
IEF6NAME Qualified name routine.
IEF6NFCM Accounting information routine.
IEF6NIJB TYPRUN keyword routine.
IEF6NYJB Priority (PRTY) keyword routine.
IEF6COND Condition (COND) keyword routine.
IEF6NXJB Message level (MSGLEVEL) keyword routine.
IEF6NZJB Message class (MSGCLASS) keyword routine.
IEF6NIJB Parenthesis routine.
IEF7MMCM Reader/interpreter message routine.
IEF6MIXX Reader/interpreter call to IEFQMSSS.
IEFQMSSS Table store subroutine.
IEFMFDUM Linkage to IEF6MFXX (in IEFCNTRL load module).

Load Module Name: IEFJTERM
Alias: IEFW23SD
Alias: IEFZA
Entry Point: IEFZA
Assembly Modules:
IEFW23SD Initializes for job termination, exits to IEFZAJB3 (in this load module).
IEFZAJB3 Job termination routine.
IEFWTERM Job ended message routine.
IEFZGJB1 Disposition and unallocation subroutine.
IEFACTLK Linkage to user's accounting routine.
IEFACTRT Dummy module to be replaced by user's accounting routine.
(The preceding two modules may be replaced by IEFACTFK assembly module if no accounting routine is specified as a system generation option.)
IEFSD006 Converts record number to logical track address (TTR).
IEFSD007 Call to table store subroutine.
IEFYSSMB Message enqueuing routine, enqueues SMBs.
IEFQMSSS Table store subroutine.
IEFZHFAK Call to ZPOQMGR1 subroutine.

IEFZGMSG Contains initiator terminator messages.
IEFZHMSG Contains initiator terminator messages.
IEFW31SD Exit to IEFSD003 (in this load module).
IEFSD003 Passes control to IEFSD010, then to IEF08FAK, (both in this load module).
IEFSD010 Dequeues and writes out system message blocks (SMBs).
IEFSD035 Check for downshift (exit to IEFSD031).

Load Module Name: IEFCOMND
Alias: IEFKP
Entry Point: IEFKP
Assembly Modules:
IEF7KPXX Processes commands in input stream.
IEEMCS01 Master command routine.
IEEILCDM Prevents unresolved IEEICCAN symbol after initialization.
IEFSD006 Converts record number to logical track address (TTR).
IEF7MMCM Reader/interpreter message routine.
IEF6MIXX Reader/interpreter call to IEFQMSSS.
IEFQMSSS Table store subroutine.
IEFKADUM Linkage to IEF7KAXX (in IEFCNTRL load module).

Load Module Name: IEF1STMT
Alias: IEFKL
Entry Point: IEFKL
Assembly Modules:
IEF6KLXX First statement routine.
IEF7MMCM Reader/interpreter message routine.
IEF6MIXX Reader/interpreter call to IEFQMSSS.
IEFQMSSS Table store subroutine.
IEFMCDUM Linkage to IEF6MCXX (in IEFCNTRL load module).
IEFKADUM Linkage to IEF7KAXX (in IEFCNTRL load module).

Load Module Name: IEFEOF
Alias: IEFK3
Entry Point: IEFK3
Assembly Modules:
IEF7K3XX Input stream end-of-file (EOF) routine.
IEF7K4XX Close devices routine.
IEF7K2XX Open devices routine.
IEFSD006 Converts record number to logical track address (TTR).
IEFSD007 Call to table store subroutine.
IEF6MIXX Reader/interpreter call to IEFQMSSS.
IEFQMSSS Table store subroutine.
IEEMCS01 Master command routine.
IEFKADUM Linkage to IEF7KAXX (in IEFCNTRL load module).
IEEILCDM Prevents unresolved IEEICCAN symbol after initialization (IPL).

Load Module Name: IEFK4
Entry Point: IEFK4DUM
Assembly Modules:
IEFK4ENT Switch input readers routine.
IEF7K4XX Close devices routine.
IEF7K2XX Open devices routine.
IEFSD006 Converts record number to logical track address (TTR).
IEFSD007 Call to table store subroutine.
IEF6MIXX Reader/interpreter call to IEFQMSSS.
IEFQMSSS Table store subroutine.

Load Module Name: IEFERROR
Alias: IEFVM6LS
Entry Point: IEFVMSGR
Assembly Modules:
IEFVMLS6 JFCB housekeeping error message processing routine.
IEFYSSMB Message enqueuing routine enqueues SMBs.
IEFQMSSS Table store subroutine.
IEFVMLS7 Contains initiator/terminator messages.
IEFYNFAK Linkage to IEFYNIMP (in IEFSTERM load module).

Load Module Name: IEFIDUMP
Entry Point: IEFIDUMP
Assembly Modules:
IEFIDUMP Indicative dump routine.
IEFYSSMB Message enqueuing routine, enqueues SMBs.
IEFQMSSS Table store subroutine.
IEFIDMPM Contains initiator/terminator messages.
IEFYNFAK Linkage to IEFYNIMP (in IEFSTERM load module).

Load Module Name: IEFDCB
Alias: IEF5DCDP
Assembly Module:
IEF6DCDP Data control block (DCB) DD parameter list table.

Load Module Name: IEFMSG01
Assembly Module:
IEF3MSG1 Contains reader/interpreter messages.

Load Module Name: IEFMSG02
Assembly Module:
IEF3MSG2 Contains reader/interpreter messages.

Load Module Name: IEFMSG03
Assembly Module:
IEF3MSG3 Contains reader/interpreter messages.

Load Module Name: IEFMSG04
Assembly Module:
IEF3MSG4 Contains reader/interpreter messages.

Load Module Name: IEFMSG05
Assembly Module:

IEF3MSG5 Contains reader/interpreter messages.

Load Module Name: IEFMSG06
Assembly Module:
IEF3MSG6 Contains reader/interpreter messages.

Load Module Name: IEFMSG07
Assembly Module:
IEF3MSG7 Contains reader/interpreter messages.

Load Module Name: IEFINITL
Alias: IEFK1
Assembly Modules:
IEF7K1XX Entry to job management from nucleus initialization program (NIP).
IEEMCS01 Master command routine.
IEEILC01 Automatic command routine.
IEF7K2XX Open devices routine.
IEFWSDIP Linkage control table (LCT) initialization routine.
IEFSD006 Converts record number to logical track address (TTR).
IEFSD007 Call to table store subroutine.
IEF6MIXX Reader/interpreter call to IEFQMSSS.
IEFQMSSS Table store subroutine.
IEFKADUM Linkage to IEF7KAXX (in IEFCNTRL load module).

Load Module Name: IEESET
Alias: IEEGESTO
Assembly Module:
IEEGES01 Master scheduler SET command routine.

Load Module Name: IEFJOBQE
Alias: IEFINTQS
Assembly Module:
IEFINTQA Initializes SYS1.SYSJOBQE data set.

Load Module Name: IEETIME
Alias: IEEQOT00
Assembly Module:
IEEQOT00 Sets time and date.

Load Module Name: IEEFAULT
Alias: IEEGK1GM
Assembly Module:
IEEGK1GM Fault routine -- issues master scheduler messages.

Load Module Name: IEESTART
Alias: IEEIC1PE
Entry Point: IEEIC1PE
Assembly Modules:
IEESTART START command routine.
IEEREADR Start reader routine.
IEEWRITR Start writer routine.

Load Module Name: IEEJFCB
Alias: IEEIC3JF
Assembly Module:
IEEIC3JF Contains preformatted JFCB for one START command.

Load Module Name: IEESJFCB
Alias: IEEIC2NQ
Entry Point: IEEIC2NQ
Assembly Modules:
IEEIC2NQ Saves START command JFCBs.
IEFQMSSS Table store subroutine.

Load Module Name: IEFSD030
Assembly Modules:
IEFSD030 Scheduler downshift routine.
IEFSD006 Converts record number to logical track address (TTR).
IEFSD007 Call to table store subroutine.
IEFSQMSS Table store subroutine.

Load Module Name: IEFSD031
Assembly Modules:
IEFSD031 Scheduler upshift routine.
IEFSD006 Converts record number to logical track address (TTR).
IEFSD007 Call to table store subroutine.
IEFSQMSS Table store subroutine.

Load Module Name: IEFPRINT
Alias: SPRINTER
Alias: IEFPRT
Assembly Module:
IEFPRTXX Tape SYSOUT to printer.

44K CONFIGURATION

Load Module Name: IEECVCTI
Assembly Modules:
IEECVCTI Option 2 Communication Task Initialization routine.
IEEVRFRX Option 2 CVT, TCB, RB, TIOT, and UCB look-up module.

Load Module Name: IEFSTERM
Alias: IEFYN
Alias: IEFSD009
Alias: GO
Entry Point: IEFSD034
Assembly Modules:
IEFSD011 Entry to job management from supervisor.
IEFW42SD Passes control to IEFIDUMP (in IEFIDUMP load module) if indicative dump is needed, or to IEFYNIMP (in this load module).
IEFYNIMP Step termination routine.
IEFYPJB3 Step data set driver routine.
IEFVJIMP JOB statement condition code routine.
IEFVZGST1 Disposition and unallocation subroutine.

IEFACTLK	Linkage to user's accounting routine.
IEFACTRT	Dummy user's accounting routine.
IEFSD017	Places logical track address (TTR) of first system message block (SMB) in job control table (JCT).
IEFW22SD	Passes control to IEFYNIMP (in this load module), and then to IEFSD002 (in this load module) or to IEFZAJB3 (in IEFCNTRL load module).
IEFSD002	Exit to IEF08FAK or IEFSD009 (both in this load module).
IEFSD009	Initiator/terminator initialization of output unit, passes control to IEFW21SD (in this load module).
IEFSD034	Pre-termination (exits to IEFSD011).
IEFW21SD	System control routine.
IEFVKIMP	EXEC statement condition code routine.
IEFVMLS1	JFCB housekeeping control routine.
IEFVM2LS	Fetch DCB routine.
IEFVM3LS	GDG single routine.
IEFVM4LS	GDG all routine.
IEFVM5LS	Patterning DSCB routine.
IEFVM76	Processes passed non-labeled tape data sets.
IEFWSTRT	Job started message routine.
IEFWMAS1	Device name table.
IEFSD006	Converts record number to logical track address (TTR).
IEFSD007	Call to table store subroutine.
IEFYSSMB	Message enqueuing routine, enqueues SMBs.
IEFQMSSS	Table store subroutine.
IEFVJMSG	Contains initiator/terminator messages.
IEFVKMSG	Contains initiator/terminator messages.
IEFYNMSG	Contains initiator/terminator messages.
IEFYPMSG	Contains initiator/terminator messages.
IEFZGMSG	Contains initiator/terminator messages.
IEFZHMSG	Contains initiator terminator messages.
IEFIDFAK	Linkage to IEFIDUMP (in IEFIDUMP load module).
IEFVMLK5	Linkage to IEFVMLS6 (in IEFERROR load module).
IEFXAFAK	Linkage to IEFXCSSS (in IEFALLOC load module).
IEFZAFAK	Linkage to IEFZAJB3 (in IEFCNTRL load module).
IEF08FAK	Linkage to IEFSD008 (in IEFCNTRL load module).

Load Module Name: IEFALLOC
Alias: IEFXA
Entry Point: IEFXA
Assembly Modules:

IEFXCSSS	Allocation control routine.
IEFXJIMP	Allocation error recovery routine.
IEFWA000	Demand allocation routine.
IEFWSWIN	Passes control to decision allocation or AVR routine.
IEFX5000	Decision allocation routine.
IEFX300A	Device strikeout routine.
IEFXH000	Separation strikeout routine.
IEFWMSKA	Device mask table.
IEFWCIMP	Task input/output table (TIOT) construction routine.
IEFWD000	External action routine.
IEFWD001	Message directory for external action routine.
IEFXT00D	Space request routine.
IEFXKIMP	Allocation error nonrecovery routine.
IEFW41SD	Exit to step initiation routine.
IEFSD004	Step initiation routine, with exit to processing program.
IEFSD006	Convert record number to logical track address (TTR).
IEFSD007	Call to table store subroutine.
IEFSD010	Dequeue and write out system message blocks (SMBs).
IEFXTDMY	Queue overflow routine.
IEFYSSMB	Message enqueuing routine.
IEFQMSSS	Table store subroutine.
IEFXAMSG	Contains initiator/terminator messages.
IEFXJMSG	Contains initiator/terminator messages.
IEFXKMSG	Contains initiator/terminator messages.
IEFXTMSG	Contains initiator/terminator messages.
IEFYNFAK	Linkage to IEFYNIMP (in IEFSTERM load module).
IEFS15XL	Check for duplicate allocation.

Load Module Name: IEFCNTRL
Alias: IEF5DDHD
Alias: IEFZA
Alias: IEFSD008
Alias: IEFMC
Alias: IEFKA
Entry Point: IEFKA
Assembly Modules:

IEF7KAXX	Reader/interpreter control routine.
IEF6MCXX	Scans job control language (JCL) statements.
IEF6BOCM	Breakout routine.
IEF6NAME	Qualified name routine.
IEF6STNM	Scan stepname routine.
IEF6MFXX	Verb identification routine.
IEF6MKXX	Continuation routine.
IEF6NCJB	JOB statement routine.
IEF6NFCM	Accounting information routine.
IEF6N1JB	TYPRUN keyword routine.
IEF6NYJB	Priority (PRTY) keyword routine.
IEF6COND	Condition (COND) keyword routine.
IEF6NXJB	MSGLEVEL keyword routine.
IEF6NZJB	MSGCLASS keyword routine.
IEF6NIJB	Parenthesis routine.
IEF6NJEX	EXEC statement routine.

IEF6RFBK Refer-back routine.
IEF6PROC Procedure name routine.
IEF6TIME TIME keyword routine.
IEF6PARM Parameter (PARM) keyword routine.
IEF6DDHD DD statement routine.
IEF6SCAN DD scan routine.
IEFSD012 DD* statement routine.
IEF6DDNM DD name routine.
IEF6FRRS Resolves DD forward references.
IEF6DSNM DS name routine.
IEF6RFWD Processes DD forward references.
IEF6LFPR Left parenthesis routine.
IEF6RTPR Right parenthesis routine.
IEF6EQUL Equal sign routine.
IEF6LIST Subparameter list routine.
IEF6NLST Routine for no subparameter list.
IEF6NDDP DD parameter list table.
IEF6DCB0 DCB refer-back routine.
IEF6DCDP DCB DD parameter list table.
IEF6ORDR Order subroutine.
IEF6INST Insert routine.
IEF6VALU Value subroutine.
IEF6CLNP Clean up after DD routine.
IEF6ERR1 DD error-handling routine.
IEF7KPXX Processes command in input stream.
IEEMCS01 Master command routine.
IEF7MMCM Reader/interpreter message routine.
IEF6MIXX Reader/interpreter call to table store subroutine.
IEFQMSSS Table store subroutine.
IEF7KGXX Output tables for step.
IEFSD013 Assigns unit for system output (SYSOUT).
IEFSD008 Initiator/terminator to reader/interpreter interface routine.
IEFSD001 Reader/interpreter entry to IEF09FAK or to IEFW23SD (both in this load module).
IEFW23SD Performs initialization for job termination routine and exits to IEFZAJB3 (in this load module).
IEFZAJB3 Job termination routine.
IEFWTERM Job ended message routine.
IEFZGJB1 Disposition and unallocation subroutine.
IEFACTLK Linkage to user accounting routine.
IEFACTRT Dummy routine to be replaced by user's accounting routine.

(The preceding two modules may be replaced by IEFACTFK assembly module if no accounting routine is specified as a system generation option.)

IEFYSSMB Message enqueuing routine enqueues SMBs.
IEFW31SD Job termination exit to IEFSD003 (in this load module).
IEFSD003 Passes control to IEFSD010, then goes to IEFSD008 (both in this load module).
IEFSD006 Converts record number to logical track address (TTR).
IEFSD007 Call to table store subroutine.
IEFSD010 Dequeue and write out system message blocks (SMBs).
IEFZHFAK Linkage to subroutine ZPOQMGR1.
IEFZGMSG Contains initiator/terminator messages.
IEFZHMSG Contains initiator/terminator messages.
IEEILCDM Prevents unresolved IEEICCAN symbol after IPL time.
IEFK4DUM Linkage to IEFK4ENT (in IEFNEWRD load module).
IEF09FAK Linkage to IEFSD009 (in IEFSTERM load module).
IEF8LINK Linkage to IEF6COND (in this load module).
IEFKLDUM Linkage to IEF6KLXX (in IEF1STMT load module).
IEF6OUT2 Linkage to IEF6SCAN (in this load module).
IEFK3DUM Linkage to IEF7K3XX (in IEFEOF load module).
IEFSD035 Check for downshift (exit to IEFSD031).

Load Module Name: IEF1STMT
Alias: IEFKL
Entry Point: IEFKL
Assembly Modules:
IEF6KLXX First statement routine.
IEF7MMCM Reader/interpreter message routine.
IEF6MIXX Reader/interpreter call to table store subroutine.
IEFQMSSS Table store subroutine.
IEFMCDUM Linkage to IEF6MCXX (in IEFCNTRL load module).
IEFKADUM Linkage to IEF7KAXX (in IEFCNTRL load module).

Load Module Name: IEFEOF
Alias: IEFK3
Entry Point: IEFK3
Assembly Modules:
IEF7K3XX Input stream end-of-file (EOF) routine.
IEF7K4XX Close devices routine.
IEF7K2XX Open devices routine.
IEFSD006 Converts record number to logical track address (TTR).
IEFSD007 Call to table store subroutine.
IEF6MIXX Reader/interpreter call to table store subroutine.
IEFQMSSS Table store subroutine.
IEEMCS01 Master command routine.
IEEILCDM Prevent unresolved IEEICCAN symbol after IPL.
IEFKADUM Linkage to IEF7KAXX (in IEFCNTRL load module).

Load Module Name: IEFK4
Entry Point: IEFK4DUM
Assembly Modules:
IEFK4ENT Switch input readers routine.
IEF7K4XX Close devices routine.
IEF7K2XX Open devices routine.
IEFSD006 Convert record number to logical track address (TTR).
IEFSD007 Call to table store subroutine.

IEF6MIXX Reader/interpreter call to table store subroutine.
IEFQMSSS Table store subroutine.

Load Module Name: IEFERROR
Alias: IEFVM6LS
Entry Point: IEFVMSGR
Assembly Modules:
IEFVMLS6 JFCB housekeeping error message processing routine.
IEFYSSMB Message enqueuing routine, enqueues SMBs.
IEFQMSSS Table store subroutine.
IEFVMLS7 Contains initiator/terminator messages
IEFYNFAK Linkage to IEFYNIMP (in IEFSTERM load module).

Load Module Name: IEFIDUMP
Entry Point: IEFIDUMP
Assembly Modules:
IEFIDUMP Indicative dump routine.
IEFYSSMB Message enqueuing routine, enqueues SMBs.
IEFQMSSS Table store subroutine.
IEFIDMPM Contains initiator/terminator messages.
IEFYNFAK Linkage to IEFYNIMP (in IEFSTERM load module).

Load Module Name: IEFDCB
Alias: IEF5DCDP
Assembly Module:
IEF6DCDP DCB DD parameter list table.

Load Module Name: IEFMSG01
Assembly Module:
IEF3MSG1 Contains reader/interpreter messages.

Load Module Name: IEFMSG02
Assembly Module:
IEF3MSG2 Contains reader/interpreter messages.

Load Module Name: IEFMSG03
Assembly Module:
IEF3MSG3 Contains reader/interpreter messages.

Load Module Name: IEFMSG04
Assembly Module:
IEF3MSG4 Contains reader/interpreter messages.

Load Module Name: IEFMSG05
Assembly Module:
IEF3MSG5 Contains reader/interpreter messages.

Load Module Name: IEFMSG06
Assembly Module:
IEF3MSG6 Contains reader/interpreter messages.

Load Module Name: IEFMSG07
Assembly Module:
IEF3MSG7 Contains reader/interpreter messages.

Load Module Name: IEFINITL
Alias: IEFK1
Entry Point: IEFK1
Assembly Modules:
IEF7K1XX Entry to job management from nucleus initialization program (NIP).
IEEMCS01 Master command routine.
IEEILC01 Automatic command routine.
IEF7K2XX Open devices routine.
IEFWSDIP Linkage control table (LCT) initialization.
IEFSD006 Converts record number to logical track address (TTR).
IEFSD007 Call to table store subroutine.
IEF6MIXX Reader/interpreter call to table store subroutine.
IEFQMSSS Table store subroutine.
IEFKADUM Linkage to IEF7KAXX (in IEFCNTRL load module).

Load Module Name: IEESET
Alias: IEEGESTO
Assembly Module:
IEEGES01 Master scheduler SET command routine.

Load Module Name: IEFJOBQE
Alias: IEFINTQS
Assembly Module:
IEFINTQA Initializes SYS1.SYSJOBQE data set.

Load Module Name: IEETIME
Alias: IEEQOT00
Assembly Module:
IEEQOT00 Sets time and date.

Load Module Name: IEEFAULT
Alias: IEEGK1GM
Assembly Module:
IEEGK1GM Fault routine, issues master scheduler messages.

Load Module Name: IEESTART
Alias: IEEIC1PE
Entry Point: IEEIC1PE
Assembly Modules:
IEESTART START command routine.
IEEREADR Start reader routine.
IEEWRITR Start writer routine.

Load Module Name: IEEJFCB
Alias: IEEIC3JF
Assembly Module:
IEEIC3JF Contains preformatted JFCB for one START command.

Load Module Name: IEESJFCB
Alias: IEEIC2NQ
Entry Point: IEEIC2NQ
Assembly Modules:
IEEIC2NQ Save JFCBs for START commands.
IEFQMSSS Table store subroutine.

Load Module Name: IEFSD030
Assembly Modules:
IEFSD030 Scheduler downshift routine.
IEFSD006 Converts record number to logical track address (TTR).
IEFSD007 Call to table store subroutine.
IEFSQMSS Table store subroutine.

Load Module Name: IEFSD031
Assembly Modules:
IEFSD031 Scheduler upshift routine.
IEFSD006 Converts record number to logical track address (TTR).
IEFSD007 Call to table store subroutine.
IEFSQMSS Table store subroutine.

Load Module Name: IEFPRINT
Alias: SPRINTER
Alias: IEFPRT
Assembly Module:
IEFPRTXX Tape SYSOUT to printer.

ASSEMBLY MODULES AND CONTROL SECTIONS

The following table shows in which load modules each assembly module is used in the three configurations of job management. The first column lists the assembly module names in alphameric order. Except for those indicated with asterisks, all assembly modules are contained in load modules in the SYS1.LINKLIB data set. As a system generation option, the assembly module IEFACTFK may replace both IEFACTLK and IEFACTRT. The second column lists the control section names that correspond to the assembly module names in the first column. The next two columns of the table indicate which load modules of each configuration contain each assembly module. The two right-hand columns of the table refer to the CHARTS section of the PLM. If a particular control section is shown as a subroutine block, the flowchart number is listed in the SUBR. BLOCK column; if the flow within a control section is given in a chart, the flowchart number is listed in the "Flow is Defined" column. Chart numbers which appear in parentheses refer to charts which can be found in the publication IBM System/360 Operating System: Job Management, Program Logic Manual, Form Y28-6613.

Assembly Modules and Control Sections

Assembly Module Name	Notes	Control Section Name	Load Modules in Which Assembly Modules are Used		Chart Number	
			18K	44K	Appears As Subr. Block	Flow is Defined
IEEBC1PE	*	IEEBC1PE			(02)	04,(07)
IEEBH1PE	Not Used	IEEBH1	IGC0003F	IGC0003F		
IEECIR01	*	IEEBA1			(02)	04,(03)
IEECVCTI		IEECVCTI	IEECVCTI	IEECVCTI		
IEEGES01		IEEGESTO	IEESET	IEESET	(05)	
IEEGK1GM		IEEGK1GM	IEEFAULT	IEEFAULT	(05,15)	
IEEIC2NQ		IEEIC2NQ	IEESJFCB	IEESJFCB		
IEEIC3JF	**	IEEIC3JF	IEEJFCB	IEEJFCB		
IEEILCDM		IEEICCAN	IEFINTFC	IEFCNTRL	(05)	
			IEFCOMND	IEFEOF		
			IEFEOF			
IEEILC01	**	IEEICCAN	IEFINITL	IEFINITL		
IEEMCR01		IEEBB1	IEFINTFC	IEFCNTRL	10,(02,09)	(05)
			IEFCOMND	IEFINITL		
			IEFINITL	IEFEOF		
			IEFEOF			
IEEMXC01		IGC03D	IGC0003D	IGC0003D	(02)	05,(04)
IEEMSR01	***	IGC03D	IGC0003D	IGC0003D	02(02)	(04)
IEEQOT00		IEEQOT00	IEETIME	IEETIME		
IEEREADR		IEEICRDR	IEESTART	IEESTART		
IEERSC01	*	IEEMSLT				
IEERSR01	*	IEEMSLT				
IEESTART		IEEIC1PE	IEESTART	IEESTART		
IEEVRFRX		IEEVRFRX	IEECVCTI	IEECVCTI		
IEEWRITR		IEECWTR	IEESTART	IEESTART		
IEEWTC01		IGC03E	IGC0003E	IGC0003E		(06)
IEEWTR01	***	IGC03E	IGC0003E	IGC0003E		(06)
IEFACTFK		IEFACTFK	IEFSTERM	IEFSTERM		
			IEFJTERM	IEFCNTRL		
IEFACTLK		IEFACTLK	IEFSTERM	IEFSTERM		
			IEFJTERM	IEFCNTRL		
IEFACTRT		IEFACTRT	IEFSTERM	IEFSTERM	15,(34)	
			IEFJTERM	IEFCNTRL		
IEFDPOST	*	IEFDPOST				
IEFEMDUM		IEFEM	IEFCNTRL			
IEFIDFAK		IEFIDUMP	IEFSTERM	IEFSTERM		
IEFIDMPM		IEFIDMPM	IEFIDUMP	IEFIDUMP		
IEFIDUMP		IEFIDUMP	IEFIDUMP	IEFIDUMP		
IEFINTQA		IEFINTQS	IEFJOBQE	IEFJOBQE		
IEFJMDUM		IEFJM	IEFCNTRL			
IEJKADUM		IEFKA	IEFINITL	IEFINITL		
			IEFEOF	IEFEOF		
			IEFINTFC	IEF1STMT		
			IEFCOMND			
IEFKGDUM		IEFKG	IEFCNTRL			
IEFKLDUM		IEFKL	IEFCNTRL	IEFCNTRL		
IEFKPDUM		IEFKP	IEFCNTRL			
IEFKRESA	*	IEFJOB				
IEFK3DUM		IEFK3	IEFCNTRL	IEFCNTRL		
			IEFINTFC			
IEFK4DUM		IEFK4	IEFCNTRL	IEFCNTRL		
IEFK4ENT		IEFK4DUM	IEFK4	IEFK4		
IEFMCDUM		IEFMC	IEF1STMT	IEF1STMT		
IEFMFDUM		IEFMF	IEFINTFC			
			IEFEXEC			
			IEFJOB			
IEFPRTXX		SPRINTER	IEFPRINT	IEFPRINT		

(Continued)

Assembly Modules and Control Sections (Continued)

Assembly Module Name	Notes	Control Section Name	Load Modules in Which Assembly Modules are Used		Chart Number	
			18K	44K	Appears As Subr. Block	Flow is Defined
IEFQMSSS		IEFQMSSS	IEFSTERM	IEFSTERM	17,(12,16)	
			IEFSELCT	IEFALLOC		
			IEFALOC1	IEFCNTRL		
			IEFALOC4	IEF1STMT		
			IEFATACH	IEFEOF		
			IEFCNTRL	IEFK4		
			IEFDD	IEFERROR		
			IEFINTFC	IEFIDUMP		
			IEFEXEC	IEFINITL		
			IEFJOB	IEESJFCB		
			IEFJTERM			
			IEFCOMND			
			IEF1STMT			
			IEFEOF			
			IEFK4			
			IEFERROR			
			IEFIDUMP			
			IEFINITL			
			IEESJFCB			
IEFSD001		IEFSD001	IEFINTFC	IEFCNTRL	(09)	
IEFSD002		IEFSD002	IEFSTERM	IEFSTERM		
IEFSD003		IEFSD003	IEFJTERM	IEFCNTRL		
IEFSD004		IEFSD004	IEFATACH	IEFALLOC		(32)
IEFSD006		IEFSD006	IEFSTERM	IEFSTERM		
			IEFALOC2	IEFALLOC		
			IEFALOC4	IEFCNTRL	(18)	
			IEFATACH	IEFEOF		
			IEFCNTRL	IEFK4		
			IEFDD	IEFINITL		
			IEFINTFC			
			IEFJTERM			
			IEFCOMND			
			IEFEOF			
			IEFK4			
			IEFINITL			
IEFSD007		IEFSD007	IEFSTERM	IEFSTERM		
			IEFATACH	IEFALLOC		
			IEFINTFC	IEFCNTRL	(18)	
			IEFJTERM	IEFEOF	(17)	
			IEFEOF	IEFK4		
			IEFK4	IEFINITL		
			IEFINITL			
IEFSD008		IEFSD008	IEFINTFC	IEFCNTRL		(09)
IEFSD009		IEFSD009	IEFSELCT	IEFSTERM		
IEFSD010		IEFSD010	IEFATACH	IEFALLOC		
			IEFJTERM	IEFCNTRL		
IEFSD011		IEFSD011	IEFSTERM	IEFSTERM	(14,33)	(34)
IEFSD012		IEFSD012	IEFDD	IEFCNTRL		
IEFSD013		IEFSD013	IEFDD	IEFCNTRL		
IEFSD017		IEFSD017	IEFSTERM	IEFSTERM		
IEFSD030		IEFSD030	IEFSD030	IEFSD030		18
IEFSD031		IEFSD031	IEFSD031	IEFSD031		17
IEFSD034		IEFSD034	IEFSTERM	IEFSTERM		13
IEFSD035		IEFSD035	IEFJTERM	IEFCNTRL		16
IEFVJIMP		IEFVJ	IEFSTERM	IEFSTERM	(34)	(35)
IEFVJMSG		IEFVJMSG	IEFSTERM	IEFSTERM		
IEFVKIMP		IEFVK	IEFSELCT	IEFSTERM	(14)	(16)
IEFVKMSG		IEFVKMSG	IEFSELCT	IEFSTERM		
IEFVMLK5		IEFVM6	IEFSELCT	IEFSTERM		

(Continued)

Assembly Modules and Control Sections (Continued)

Assembly Module Name	Notes	Control Section Name	Load Modules in Which Assembly Modules are Used		Chart Number	
			18K	44K	Appears As Subr. Block	Flow is Defined
IEFVMLS1		IEFVM1	IEFSELCT	IEFSTERM	(17)	(18)
IEFVMLS6		IEFVM6	IEFERROR	IEFERROR	(17,18)	(24)
IEFVMLS7		IEFVM7	IEFERROR	IEFERROR		
IEFVM2LS		IEFVM2	IEFSELCT	IEFSTERM	(17,18)	(20)
IEFVM3LS		IEFVM3	IEFSELCT	IEFSTERM	(17,18)	(21)
IEFVM4LS		IEFVM4	IEFSELCT	IEFSTERM	(17,18)	(22)
IEFVM5LS		IEFVM5	IEFSELCT	IEFSTERM	(17,18)	(23)
IEFVM76		IEFVM76	IEFSELCT	IEFSTERM		
IEFWAFAK		IEFWA000	IEFALOC1			
IEFWA000		IEFWA7	IEFALOC2	IEFALLOC	(25)	(27)
IEFWCFAK		IEFWC000	IEFALOC1			
			IEFALOC2			
IEFWCIMP		IEFWC000	IEFALOC3	IEFALLOC	(25)	(29)
IEFWDFAK		IEFWD000	IEFALOC3			
IEFWD000		IEFWD000	IEFALOC4	IEFALLOC	(25)	(30)
IEFWD001		IEFWD001	IEFALOC4	IEFALLOC		
IEFWMAS1	**	DEVNAMET	IEFSELCT	IEFSTERM		
IEFWMSKA	**	DEVMASKT	IEFALOC2	IEFALLOC		
IEFWSDIP		IEFWSDIP	IEFINITL	IEFINITL		
IEFWSTRT		IEFWSTRT	IEFSELCT	IEFSTERM		
IEFSWIN		IEFSWIT	IEFALOC2	IEFALLOC		
IEFWTERM		IEFWTERM	IEFJTERM	IEFCNTRL		
IEFW21SD		IEFW21SD	IEFSELCT	IEFSTERM	(14)	(15)
IEFW22SD		IEFW22SD	IEFSTERM	IEFSTERM	(34)	
IEFW23SD		IEFW23SD	IEFJTERM	IEFCNTRL		
IEFW31SD		IEFW31SD	IEFJTERM	IEFCNTRL		
IEFW41SD		IEFW41SD	IEFALOC4	IEFALLOC		
IEFW42SD		IEFW42SD	IEFSTERM	IEFSTERM		
IEFXAFAK		IEFXA	IEFSELCT	IEFSTERM		
IEFXAMSG		IEFXAMSG	IEFALOC1	IEFALLOC		
IEFXCSSS		IEFXA	IEFALOC1	IEFALLOC	(25)	(26)
IEFXH000		IEFXH000	IEFALOC2	IEFALLOC		
			IEFALOC3			
IEFXJFAK		IEFXJ000	IEFALOC2			
			IEFALOC3			
IEFXJIMP		IEFXJ000	IEFALOC1	IEFALLOC		
IEFXJMSG		IEFXJMSG	IEFALOC1	IEFALLOC		
IEFXKIMP		IEFXK000	IEFALOC4	IEFALLOC		
IEFXKMSG		IEFXKMSG	IEFALOC4	IEFALLOC		
IEFXTDMY		IEFXTDMY	IEFALOC4	IEFALLOC		
IEFXTMSG		IEFXTMSG	IEFALOC4	IEFALLOC		
IEFXT00D		IEFXT000	IEFALOC4	IEFALLOC	(25)	(31)
IEFX300A		IEFX3000	IEFALOC2	IEFALLOC		
IEFX5000		IEFX5000	IEFALOC2	IEFALLOC	(25)	(28)
IEFYNFAK		IEFYN	IEFSELCT	IEFALLOC		
			IEFALOC1	IEFERROR		
			IEFALOC4	IEFIDUMP		
			IEFERROR			
			IEFIDUMP			
IEFYNIMP		IEFYN	IEFSTERM	IEFSTERM		
IEFYNMSG		IEFYNMSG	IEFSTERM	IEFSTERM		
IEFYPJB3		IEFYP	IEFSTERM	IEFSTERM	(34)	(37)
IEFYPMSG		IEFYPMSG	IEFSTERM	IEFSTERM		
IEFYSSMB		IEFYS	IEFSTERM	IEFSTERM		
			IEFSELCT	IEFALLOC		
			IEFALOC1	IEFCNTRL		
			IEFALOC4	IEFERROR		
			IEFJTERM	IEFIDUMP		
			IEFERROR			

(Continued)

Assembly Modules and Control Sections (Continued)

Assembly Module Name	Notes	Control Section Name	Load Modules in Which Assembly Modules are Used		Chart Number	
			18K	44K	Appears As Subr. Block	Flow is Defined
			IEFIDUMP			
IEFZAFAK		IEFZA	IEFSTERM	IEFSTERM		
IEFZAJB3		IEFZA	IEFJTERM	IEFCNTRL	(33)	(36)
IEFZGJB1		IEFZG	IEFJTERM	IEFCNTRL	15,(36)	(38)
IEFZGMSG		IEFZGMSG	IEFSTERM	IEFSTERM		
			IEFJTERM	IEFCNTRL		
IEFZGST1		IEFZG	IEFSTERM	IEFSTERM		
IEFZHFAK		IEFZPOQM	IEFJTERM	IEFCNTRL		
IEFZHMSG		IEFZH	IEFSTERM	IEFSTERM		
			IEFJTERM	IEFCNTRL		
IEF04FAK		IEFSD004	IEFALOC4			
IEF08FAK		IEFSD008	IEFSTERM	IEFSTERM		
			IEFINTFC			
			IEFJTERM			
IEF09FAK		IEFSD009	IEFSTERM	IEFCNTRL		
			IEFINTFC			
IEF23FAK		IEFW23SD	IEFINTFC			
IEF3MSG1		IEFMSG1	IEFMSG01	IEFMSG01		
IEF3MSG2		IEFMSG2	IEFMSG02	IEFMSG02		
IEF3MSG3		IEFMSG3	IEFMSG03	IEFMSG03		
IEF3MSG4		IEFMSG4	IEFMSG04	IEFMSG04		
IEF3MSG5		IEFMSG5	IEFMSG05	IEFMSG05		
IEF3MSG6		IEFMSG6	IEFMSG06	IEFMSG06		
IEF3MSG7		IEFMSG7	IEFMSG07	IEFMSG07		
IEF6BOCM		IEFBM	IEFCNTRL	IEFCNTRL		
			IEFDD			
			IEFJOB			
			IEFEXEC			
IEF6CLNP		IEFMO2	IEFDD	IEFCNTRL		
IEF6CN17		INDMRTN	IEFDD			
IEF6COND		IEF6COND	IEFEXEC	IEFCNTRL		
			IEFJOB			
IEF6DCB0		IEF6DCB0	IEFCNTRL	IEFCNTRL		
IEF6DCDP		INDMB	IEFDD	IEFCNTRL		
			IEFDCB	IEFDCB		
IEF6DDHD		INDMRTN	IEFCNTRL	IEFCNTRL	(12)	
IEF6DDNM		INDMO1	IEFDD	IEFCNTRL		
IEF6DHX1		INDMON	IEFCNTRL			
IEF6DSNM		INDMO3	IEFDD	IEFCNTRL		
IEF6EQUL		INDMOP	IEFDD	IEFCNTRL		
IEF6ERR1		IEF6ERR1	IEFDD	IEFCNTRL		
IEF6FRRS		IEF6FRRS	IEFCNTRL	IEFCNTRL		
IEF6INST		INDMOZ	IEFDD	IEFCNTRL		
IEF6KLXX		IEFKL	IEF1STMT	IEF1STMT		
IEF6LFPR		INDMOS1	IEFDD	IEFCNTRL		
IEF6LIST		INDMOY	IEFDD	IEFCNTRL		
IEF6MCXX		IEFMC	IEFCNTRL	IEFCNTRL	(09)	
IEF6MFXX		IEFMF	IEFCNTRL	IEFCNTRL	(09)	
IEF6MIXX		IEFMI	IEFCNTRL	IEF1STMT		
			IEFDD	IEFNEWRD		
			IEFINTFC	IEFEOF		
			IEFEXEC	IEFINITL		
			IEFJOB	IEFCNTRL		
			IEFCOMND			
			IEF1STMT			
			IEFEOF			
			IEFK4			
			IEFINITL			
IEF6MKXX		IEFMK	IEFCNTRL	IEFCNTRL		

(Continued)

Assembly Modules and Control Sections (Continued)

Assembly Module Name	Notes	Control Section Name	Load Modules in Which Assembly Modules are Used		Chart Number	
			18K	44K	Appears As Subr. Block	Flow is Defined
IEF6NAME		INNAME	IEFCNTRL	IEFCNTRL		
			IEFEXEC			
			IEFJOB			
IEF6NCJB	**	IEFJM	IEFJOB	IEFCNTRL	(08)	(10)
IEF6NDDP		INDMA	IEFDD	IEFCNTRL		
IEF6NFCM		IEFAM	IEFEXEC	IEFCNTRL		
			IEFJOB			
IEF6NIJB		IEFNI	IEFJOB	IEFCNTRL		
IEF6NJEX		IEFEM	IEFEXEC	IEFCNTRL	(08)	(11)
IEF6NLST		INDMOX	IEFDD	IEFCNTRL		
IEF6NXJB		IEFNX	IEFJOB	IEFCNTRL		
IEF6NYJB		IEFNY	IEFJOB	IEFCNTRL		
IEF6NZJB		IEFNZ	IEFJOB	IEFCNTRL		
IEF6N1JB		IEFN1	IEFJOB	IEFCNTRL		
IEF6ORDR		INDMOV	IEFDD	IEFCNTRL		
IEF6OUT2		INDMOH	IEFCNTRL	IEFCNTRL	(12)	
IEF6PARM		IEFPARM	IEFEXEC	IEFCNTRL		
IEF6PROC		IEFPROC	IEFEXEC	IEFCNTRL		
IEF6RFBK		IEFRFBK	IEFEXEC	IEFCNTRL		
IEF6RFWD		IEF6RFWD	IEFDD	IEFCNTRL		
IEF6RTPR		INDMOR	IEFDD	IEFCNTRL		
IEF6SCAN		INDMON	IEFDD	IEFCNTRL	(08)	(12)
IEF6STNM		IEFSTNM	IEFCNTRL	IEFCNTRL		
			IEFEXEC			
			IEFJOB			
IEF6TIME		IEFTIME	IEFEXEC	IEFCNTRL		
IEF6VALU		INDMOT	IEFDD	IEFCNTRL		
IEF7KAXX		IEFKA	IEFCNTRL	IEFCNTRL	(09)	
IEF7KGXX		IEFKG	IEFINTFC	IEFCNTRL	(09)	
IEF7KPXX		IEFKP	IEFCOMND	IEFCNTRL	(09)	
IEF7K1XX		IEFK1	IEFINITL	IEFINITL	(08)	(09)
			IEFK4			
IEF7K2XX		IEFK2	IEFINITL	IEFINITL		
			IEFK4	IEFK4		
			IEFEOF	IEFEOF		
IEF7K3XX		IEFK3	IEFEOF	IEFEOF		
IEF7K4XX		IEFK4	IEFK4	IEFNEWRD		
			IEFEOF	IEFEOF		
IEF7MMCM		IEFWMSG	IEFCNTRL	IEFCNTRL		
			IEF1STMT	IEF1STMT		
			IEFDD			
			IEFINTFC			
			IEFEXEC			
			IEFJOB			
			IEFCOMND			
IEF8LINK		IEFLINK	IEFEXEC	IEFCNTRL		
IGC0103D	***	IGC0103D	IGC0103D	IGC0103D	(02,04)	

Notes:
*Assembly modules in SYS1.NUCLEUS data set.
**Modules are assembled at system generation time.
***Assembly modules in SYS1.SVCLIB data set.

CONTROL SECTIONS AND ASSEMBLY MODULES

The following list provides a cross-reference between job management control section (CSECT) names, which appear in alphameric order, and the corresponding assembly module names. Control section names are also listed in the preceding assembly module to load module cross reference table.

CSECT Name	Assembly Module Name
DEVMASKT	IEFWMSKA
DEVNAMET	IEFWMAS1
IEEBA1	IEECIR01
IEEBB1	IEEMCR01
IEEBC1PE	IEEBC1PE
IEEBH1	IEEBH1PE
IEECVCTI	IEECVCTI
IEEGESTO	IEEGES01
IEEGK1GM	IEEGK1GM
IEEICCAN	IEEILCDM
IEEICCAN	IEEILC01
IEEICRDR	IEEREADR
IEEICWTR	IEEWRITR
IEEIC1PE	IEESTART
IEEIC2NQ	IEEIC2NQ
IEEIC3JF	IEEIC3JF
IEEMSLT	IEERSC01
IEEMSLT	IEERSR01
IEEQOT00	IEEQOT00
IEEVRFRX	IEEVRFRX
IEFACTLK	IEFACTLK
IEFACTRT	IEFACTRT
IEFAM	IEF6NFCM
IEFBM	IEF6BOCM
IEFCOND	IEF6COND
IEFDPOST	IEFDPOST
IEFEM	IEFEMDUM
IEFEM	IEF6NJEX
IEFIDMPM	IEFIDMPM
IEFIDUMP	IEFIDFAK
IEFIDUMP	IEFIDUMP
IEFINTQS	IEFINTQA
IEFJM	IEFJMDUM
IEFJM	IEF6NCJB
IEFJOB	IEFKRESA
IEFKA	IEFKADUM
IEFKA	IEF7KAXX
IEFKG	IEFKGDUM
IEFKG	IEF7KGXX
IEFKL	IEFKLDUM
IEFKL	IEF6KLXX
IEFKP	IEFKPDUM
IEFKP	IEF7KPXX
IEFK1	IEF7K1XX
IEFK2	IEF7K2XX
IEFK3	IEFK3DUM
IEFK3	IEF7K3XX
IEFK4	IEFK4DUM
IEFK4	IEF7K4XX
IEFK4DUM	IEFK4ENT
IEFLINK	IEF8LINK
IEFMC	IEFMCDUM
IEFMC	IEF6MCXX
IEFMF	IEFMFDUM
IEFMF	IEF6MFXX
IEFMI	IEF6MIXX
IEFMK	IEF6MKXX
IEFMO2	IEF6CLNP
IEFMSG1	IEF3MSG1
IEFMSG2	IEF3MSG2
IEFMSG3	IEF3MSG3
IEFMSG4	IEF3MSG4
IEFMSG5	IEF3MSG5
IEFMSG6	IEF3MSG6
IEFMSG7	IEF3MSG7
IEFNI	IEF6NIJB
IEFNX	IEF6NXJB
CSECT	ASSEMBLY
IEFNY	IEF6NYJB
IEFNZ	IEF6NZJB
IEFN1	IEF6N1JB
IEFPARM	IEF6PARM
IEFPROC	IEF6PROC
IEFQMSSS	IEFQMSSS
IEFRFBK	IEF6RFBK
IEFSD001	IEFSD001
IEFSD002	IEFSD002
IEFSD003	IEFSD003
IEFSD004	IEFSD004
IEFSD004	IEF04FAK
IEFSD006	IEFSD006
IEFSD007	IEFSD007
IEFSD008	IEFSD008
IEFSD008	IEF08FAK
IEFSD009	IEFSD009
IEFSD009	IEF09FAK
IEFSD010	IEFSD010
IEFSD011	IEFSD011
IEFSD012	IEFSD012
IEFSD013	IEFSD013
IEFSD017	IEFSD017
IEFSD030	IEFSD030
IEFSD031	IEFSD031
IEFSD034	IEFSD034
IEFSD035	IEFSD035
IEFSTNM	IEF6STNM
IEFTIME	IEF6TIME
IEFVJMSG	IEFVJMSG
IEFVJ	IEFVJIMP
IEFVKMSG	IEFVKMSG
IEFVK	IEFVKIMP
IEFVM1	IEFVMLS1
IEFVM2	IEFVM2LS
IEFVM3	IEFVM3LS
IEFVM4	IEFVM4LS
IEFVM5	IEFVM5LS
IEFVM6	IEFVMLK5
IEFVM6	IEFVMLS6
IEFVM76	IEFVM76
IEFVM7	IEFVMLS7
IEFWA000	IEFWAFAK
IEFWA7	IEFWA000
IEFWC000	IEFWCFAK
IEFWC000	IEFWCIMP
IEFWD000	IEFWDFAK
IEFWD000	IEFWD000
IEFWD001	IEFWD001
IEFWMSG	IEF7MMCM
IEFWSDIP	IEFWSDIP
IEFWSTRT	IEFWSTRT

CSECT Name (Cont.)	Assembly Module Name (Cont.)
IEFWSWIT	IEFWSWIN
IEFWTERM	IEFWTERM
IEFW21SD	IEFW21SD
IEFW22SD	IEFW22SD
IEFW23SD	IEFW23SD
IEFW23SD	IEF23FAK
IEFW31SD	IEFW31SD
IEFW41SD	IEFW41SD
IEFW42SD	IEFW42SD
IEFXAMSG	IEFXAMSG
IEFXA	IEFXAFAK
IEFXA	IEFXCSSS
IEFXH000	IEFXH000
IEFXJMSG	IEFXJMSG
IEFXJ000	IEFXJFAK
IEFXJ000	IEFXJIMP
IEFXKMSG	IEFXKMSG
IEFXK000	IEFXKIMP
IEFXTDMY	IEFXTDMY
IEFXTMSG	IEFXTMSG
IEFXT000	IEFXT00D
IEFX3000	IEFX300A
IEFX5000	IEFX5000
IEFYNIMP	IEFYNIMP
IEFYNMSG	IEFYNMSG
IEFYN	IEFYNFAK
IEFYPMSG	IEFYPMSG
IEFYP	IEFYPJB3
IEFYS	IEFYSSMB
IEFZA	IEFZAPAK
IEFZA	IEFZAJB3

CSECT Name (Cont.)	Assembly Module Name (Cont.)
IEFZGMSG	IEFZGMSG
IEFZG	IEFZGJB1
IEFZG	IEFZGST1
IEFZH	IEFZHMSG
IEFZPOQM	IEFZHFAK
IEF6DCB0	IEF6DCB0
IEF6ERR1	IEF6ERR1
IEF6FRRS	IEF6FRRS
IEF6RFWD	IEF6RFWD
IGC0103D	IGC0103D
IGC03D	IEEMXC01
IGC03E	IEEWTC01
INDMA	IEF6NDDP
INDMB	IEF6DCDP
INDMOH	IEF6OUT2
INDMON	IEF6DHX1
INDMON	IEF6SCAN
INDMOP	IEF6EQUL
INDMOR	IEF6RTPR
INDMOS1	IEF6LFPR
INDMOT	IEF6VALU
INDMOV	IEF6ORDR
INDMOX	IEF6NLST
INDMOY	IEF6LIST
INDMOZ	IEF6INST
INDMO1	IEF6DDNM
INDMO3	IEF6DSNM
INDMRTN	IEF6CN17
INNAME	IEF6NAME
SPRINTER	IEFPRTXX

Chart 01. Job Management

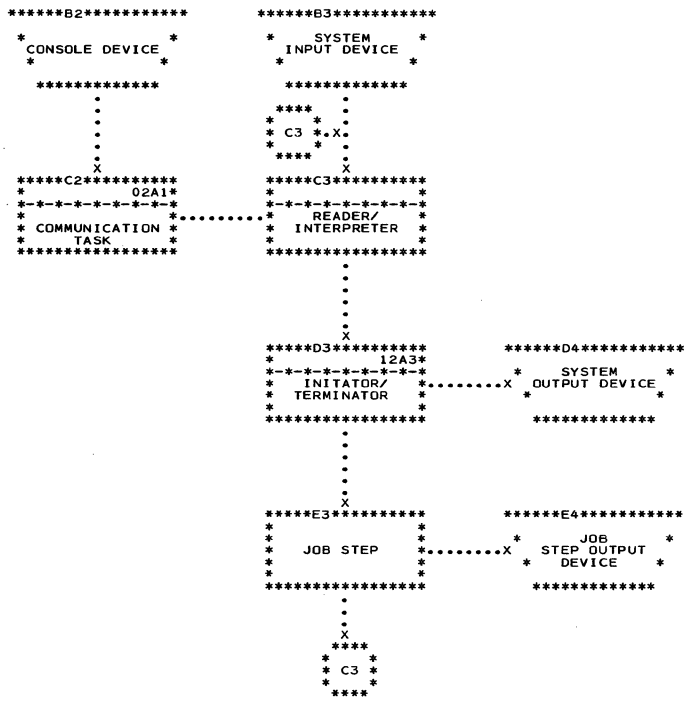


Chart 02. Communication Task Control Flow

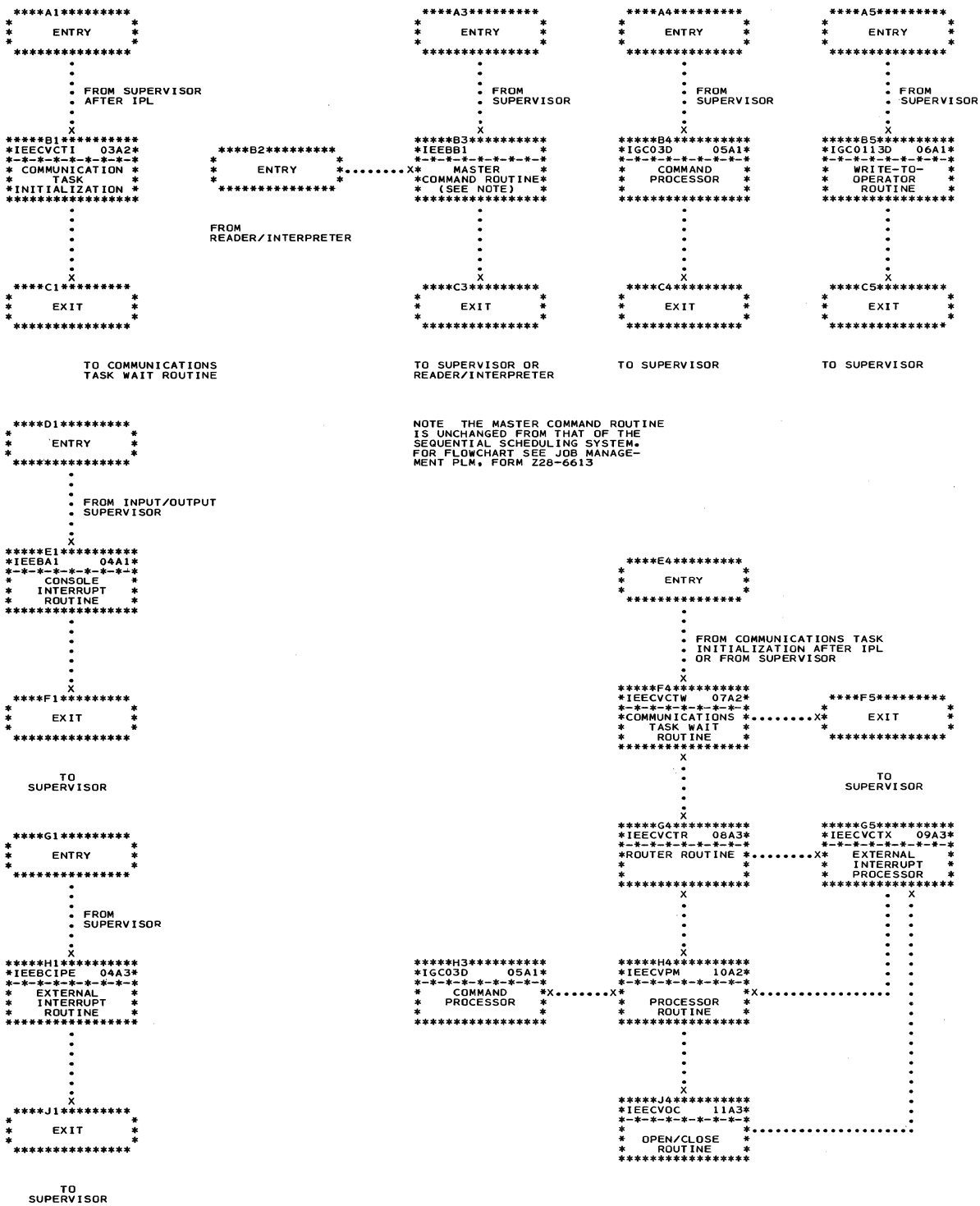


Chart 03. Communication Task Initialization Routine

IEECVCTI

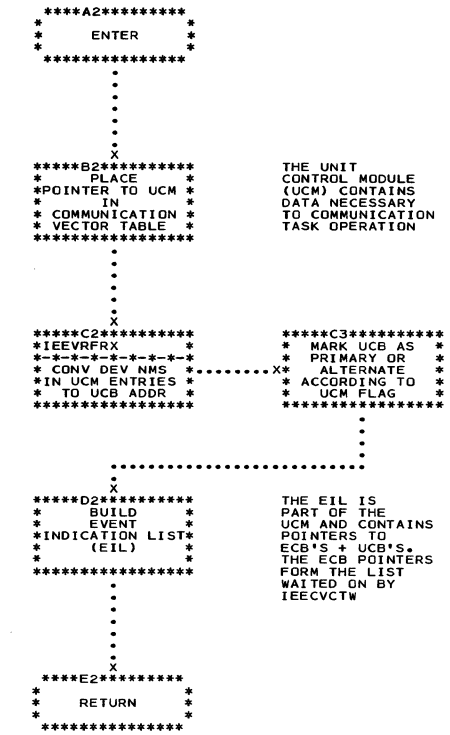


Chart 04. Console and External Interrupt Routines

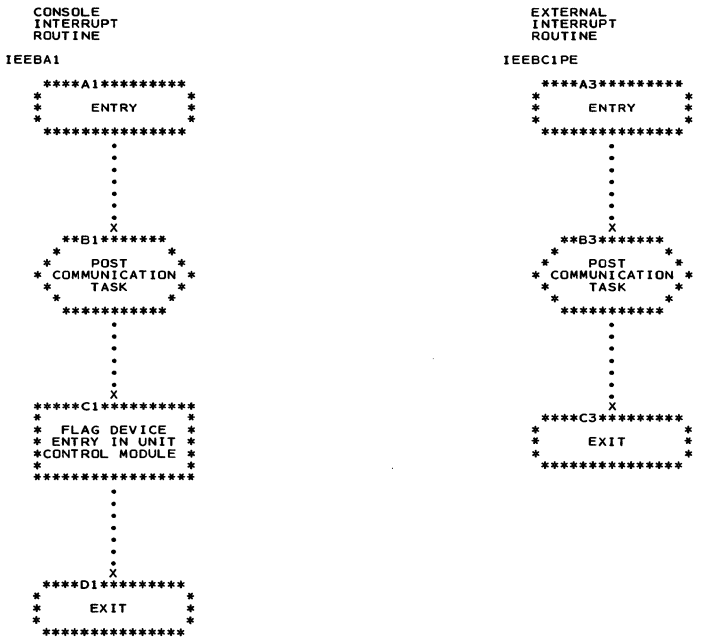


Chart 05. Master Command EXCP Routine

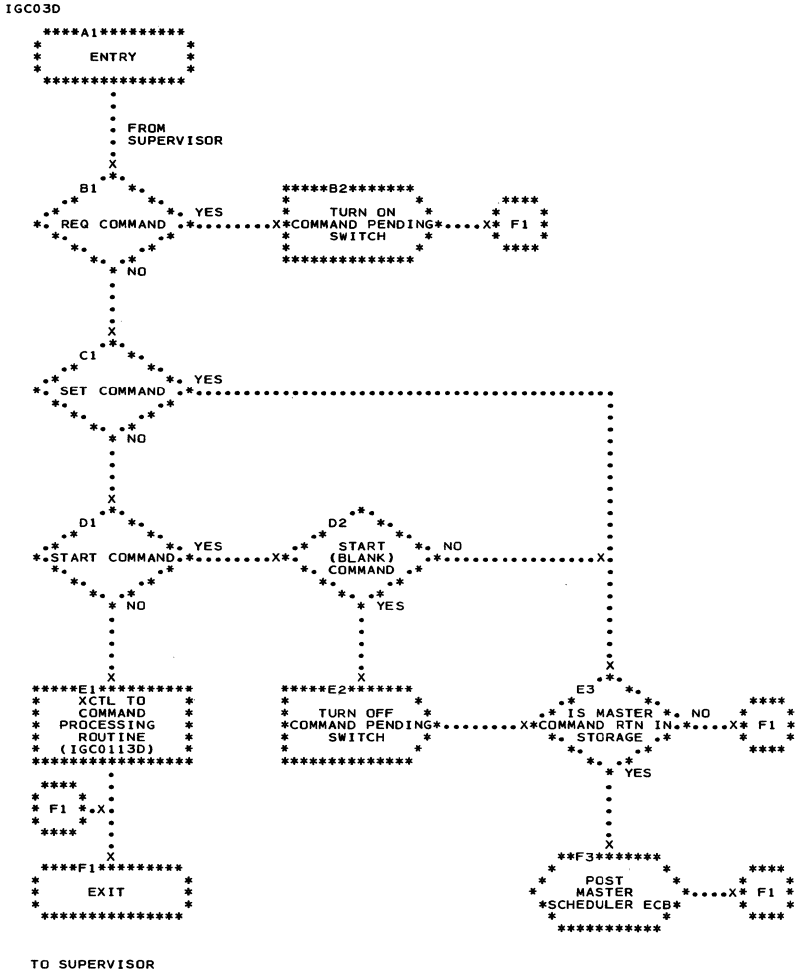


Chart 06. Write-To-Operator Routine

IGC0003E

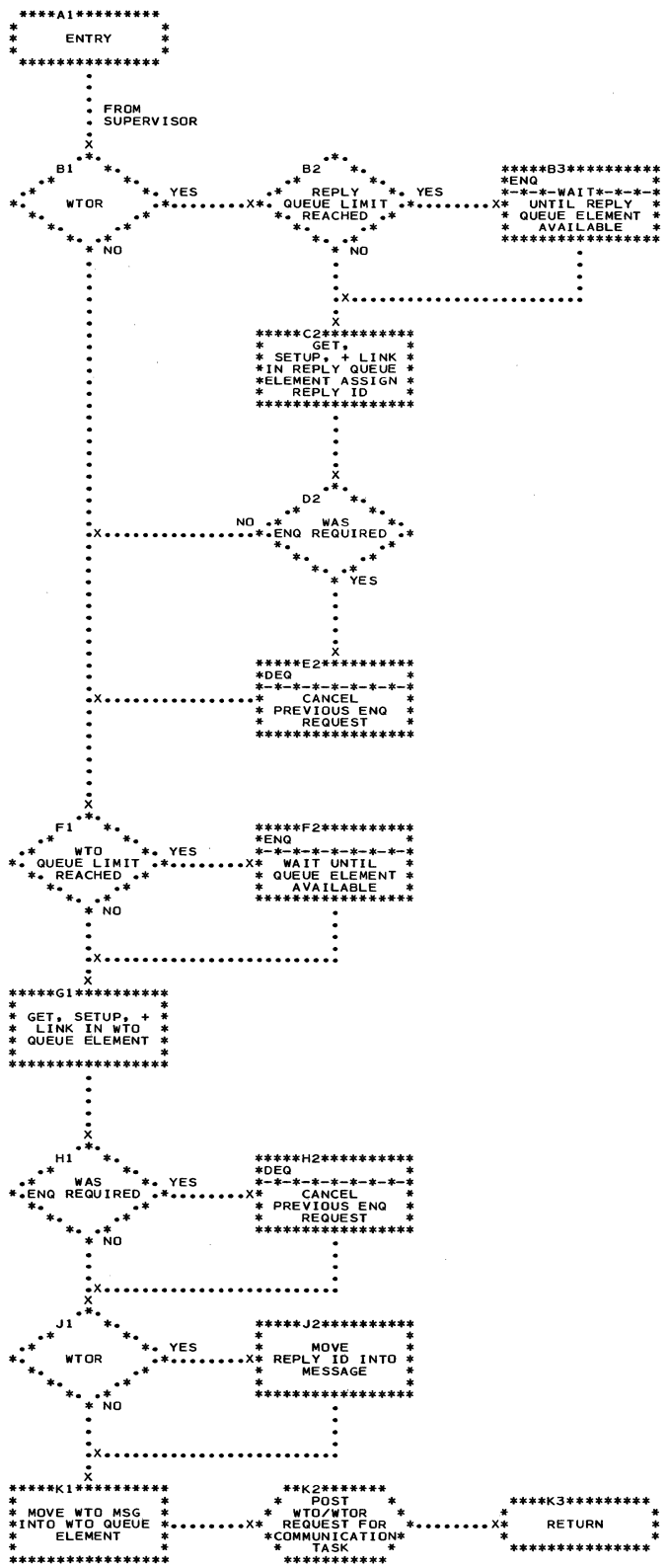


Chart 07. Communications Task Wait Routine

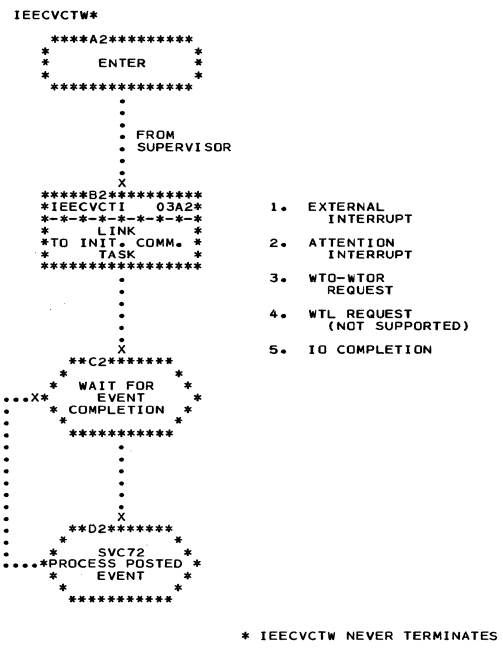


Chart 08. Communications Task Router Routine

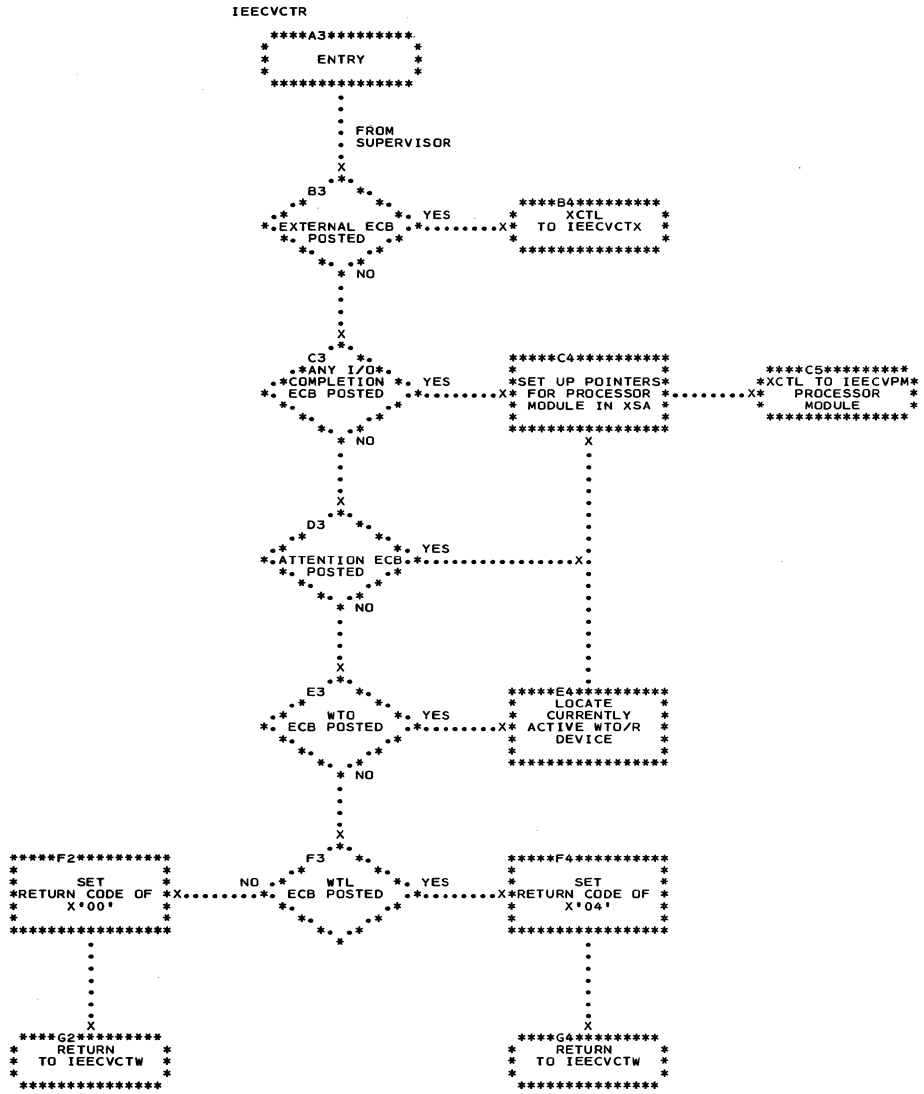


Chart 09. External Interrupt Processor Routine

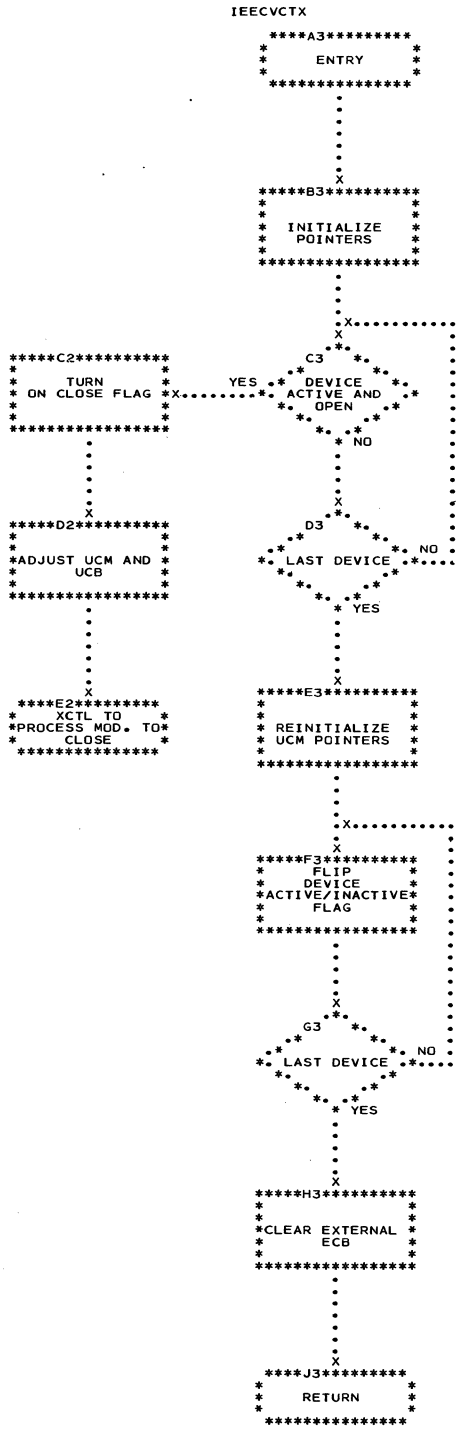


Chart 10. Communications Task Processor Routine

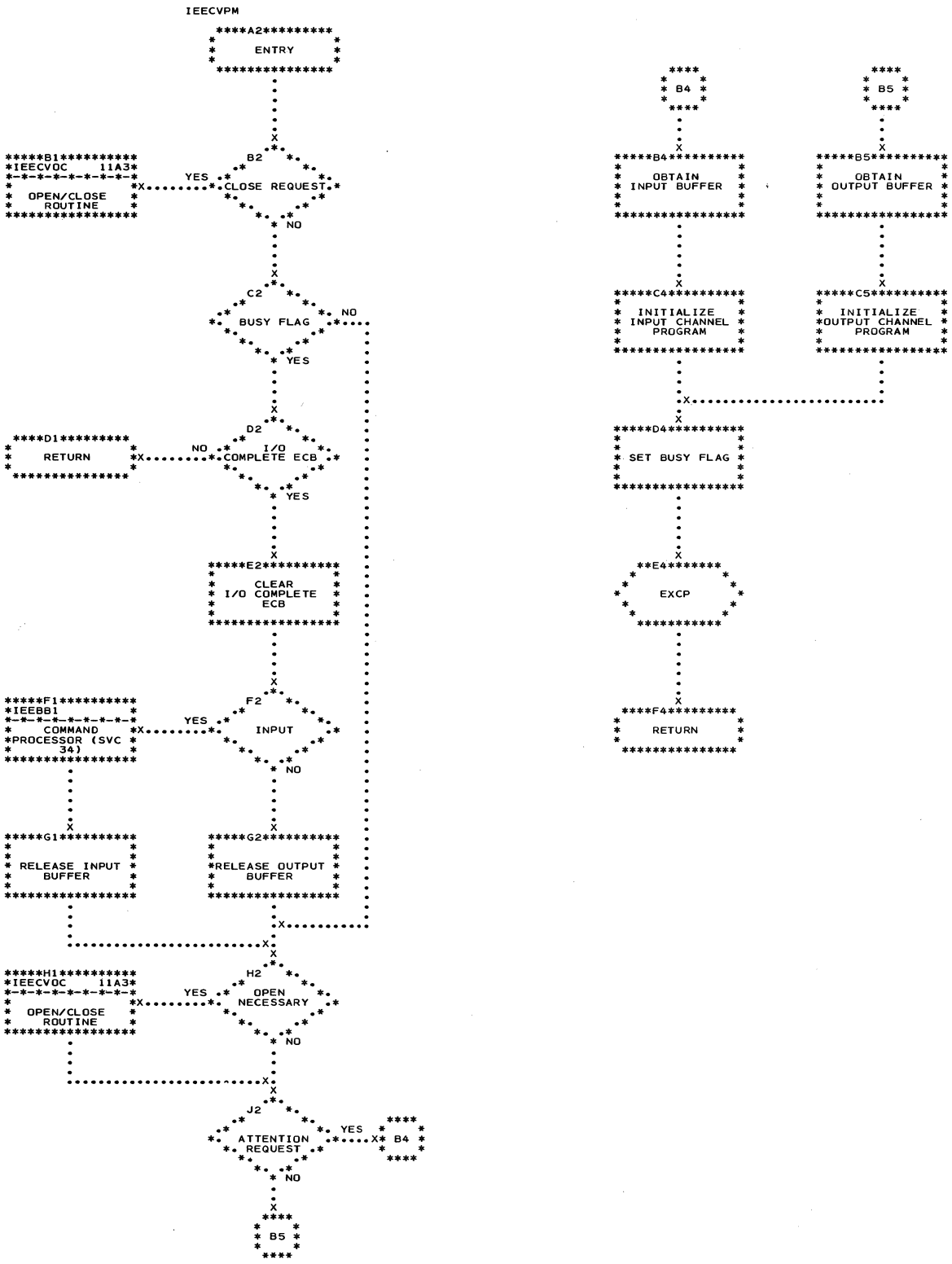


Chart 11. OPEN/CLOSE Routine

IEECVOC

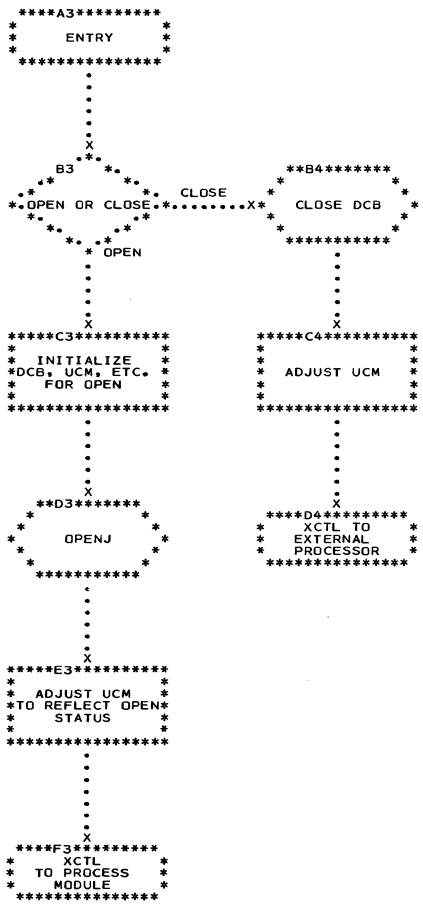
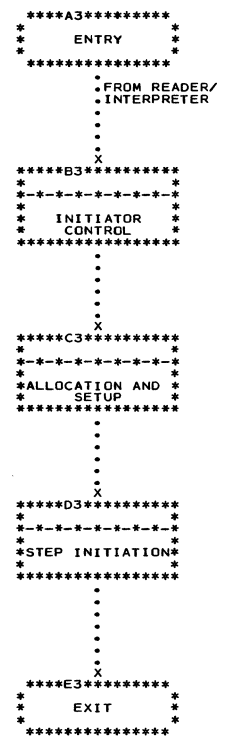


Chart 12. Initiator/Terminator Control Flow



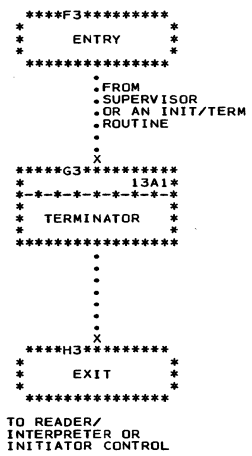


Chart 13. Pre-Termination Routine (IEFSD034)

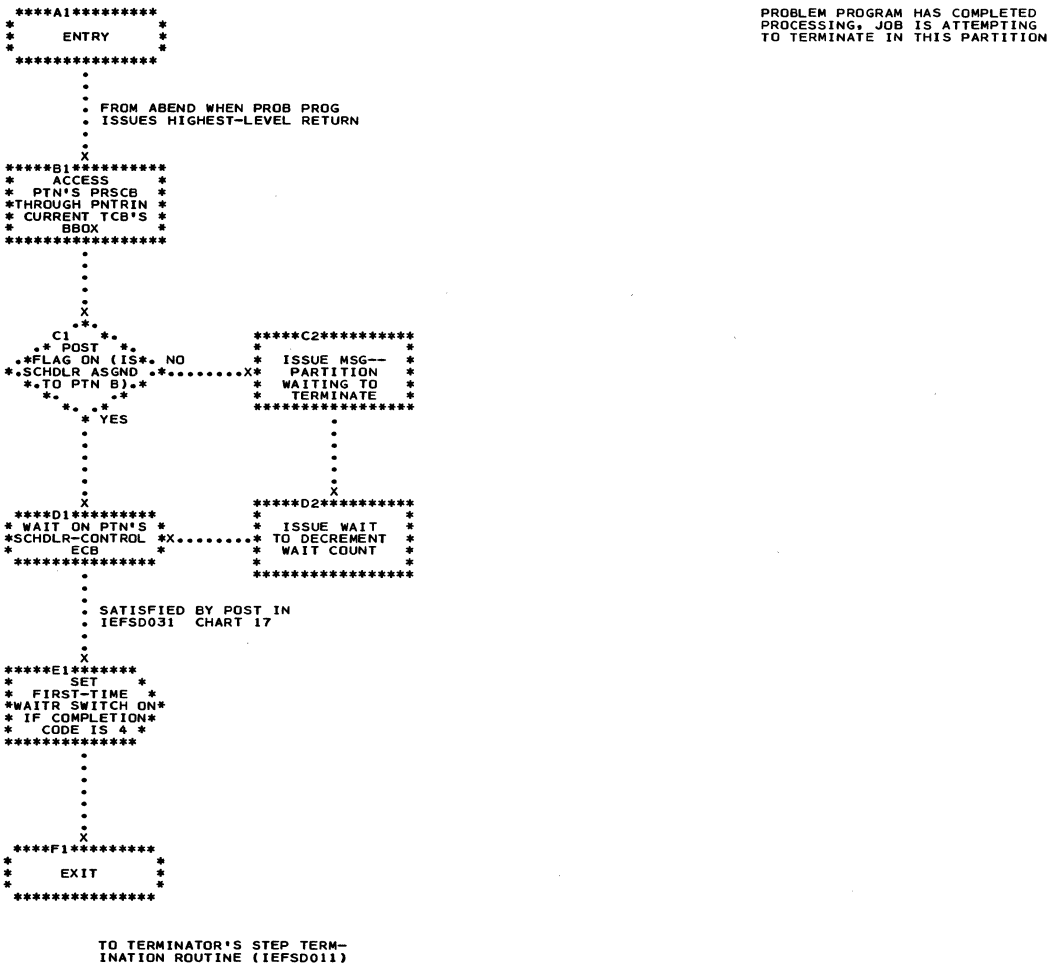


Chart 14. Termination Control Flow

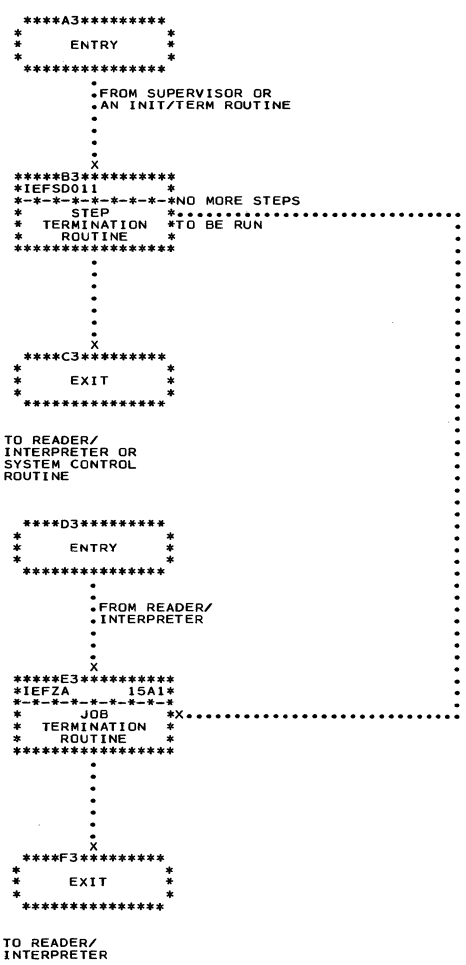


Chart 15. Job Termination Routine

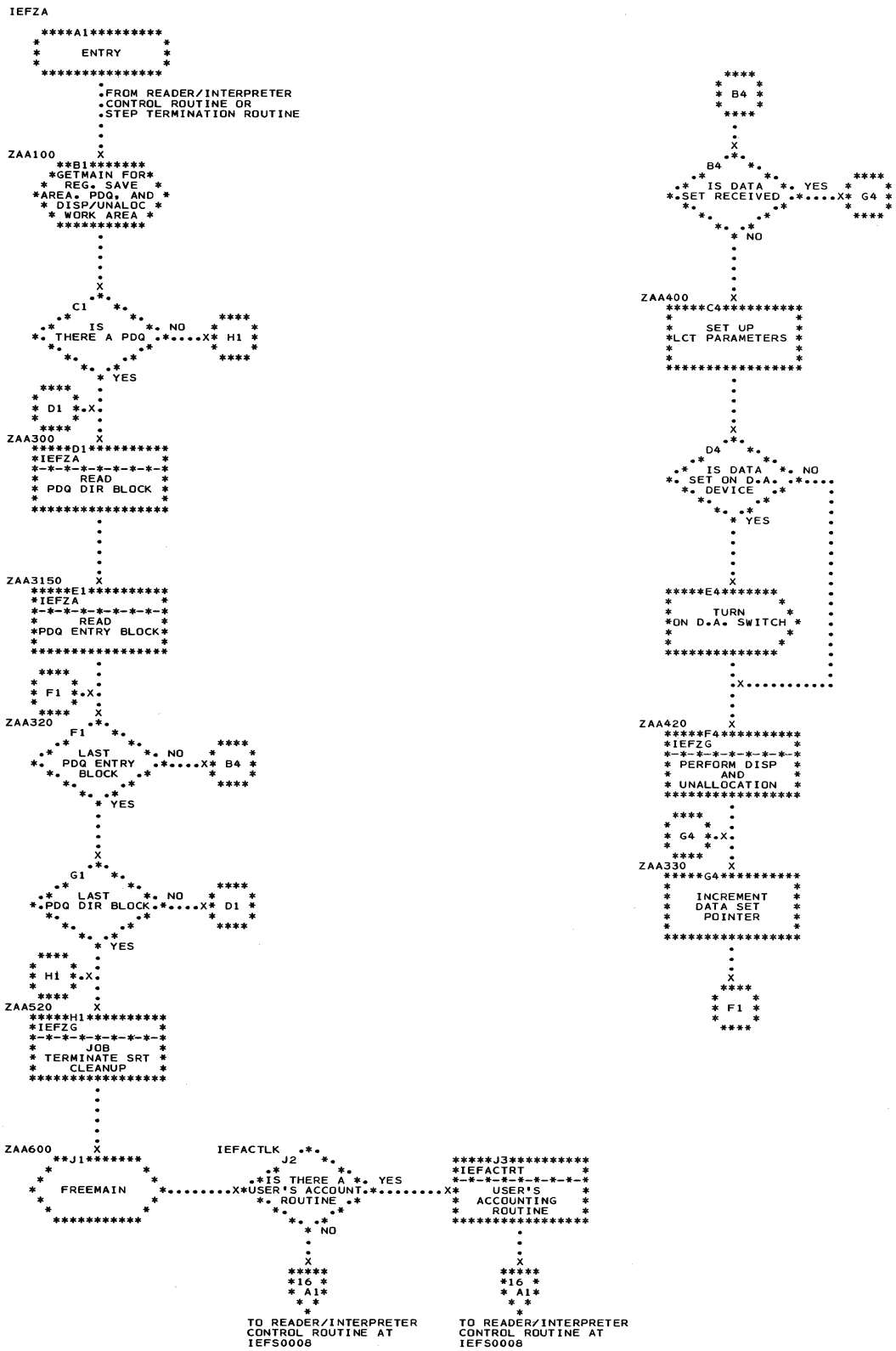


Chart 16. Shift Count Interrogator Routine (IEFSD035)

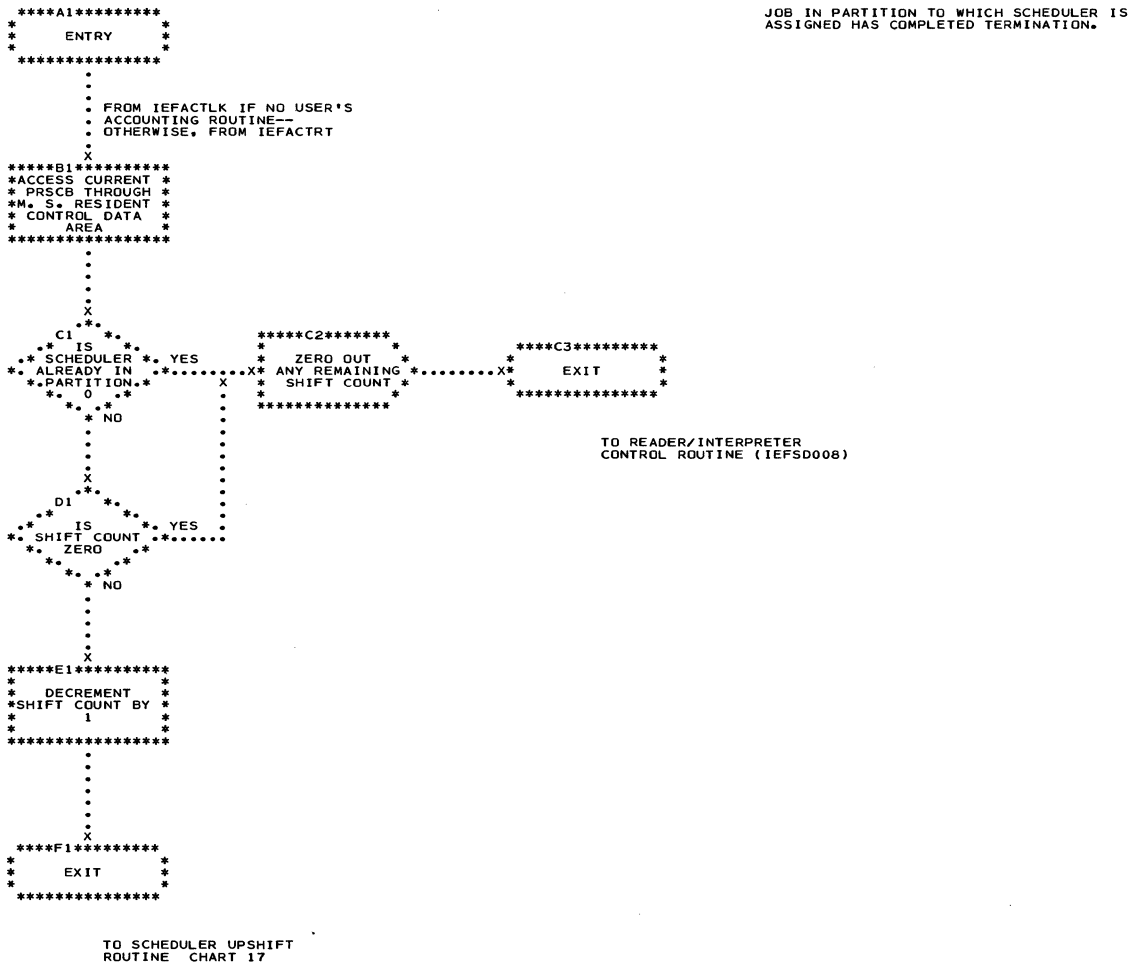
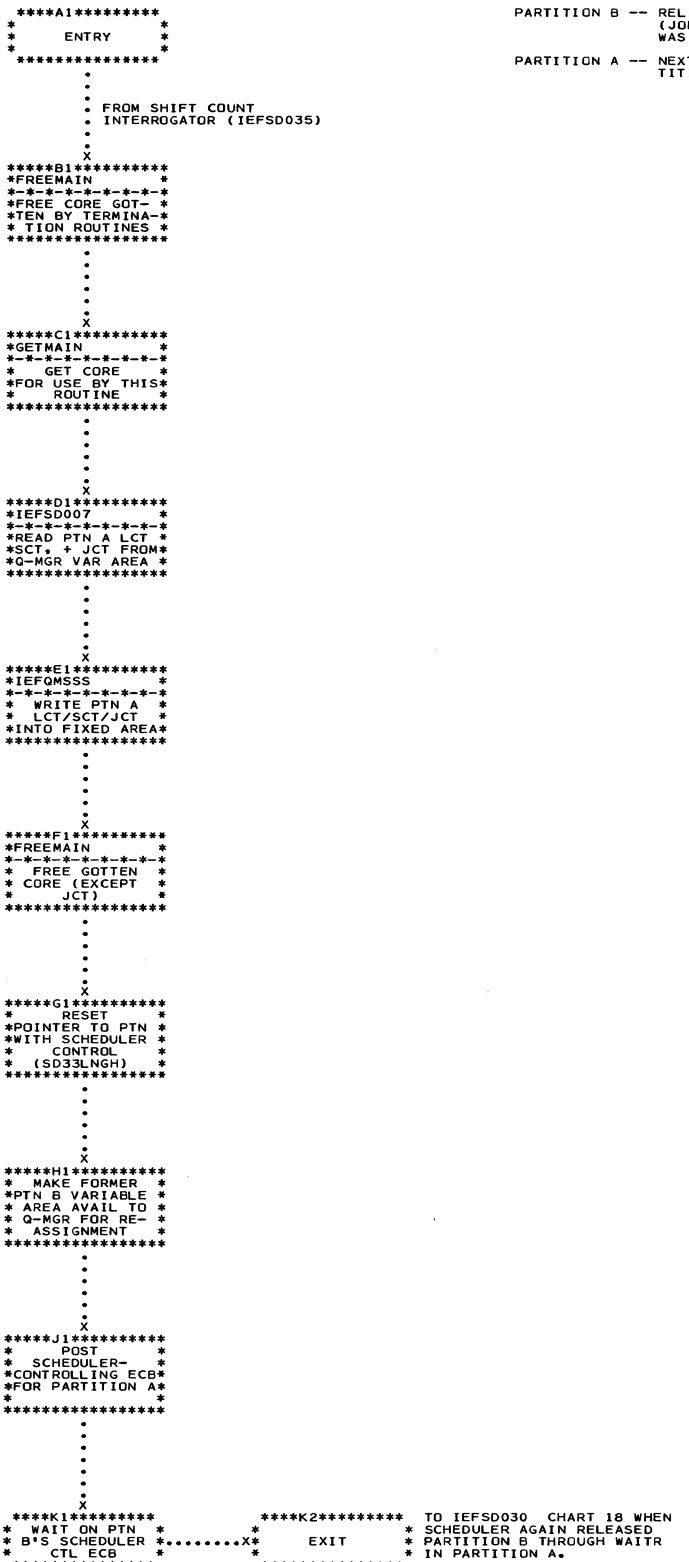


Chart 17. Scheduler Upshift Routine (IEFSD031)



PARTITION B -- RELINQUISHING SCHEDULER CONTROL (JOB HAS TERMINATED, SHIFT COUNT WAS NON-ZERO)

PARTITION A -- NEXT-HIGHER-PRIORITY PARTITION (RECEIVING SCHDLR CONTROL)

Chart 18. Scheduler Downshift Routine (IEFSD030)

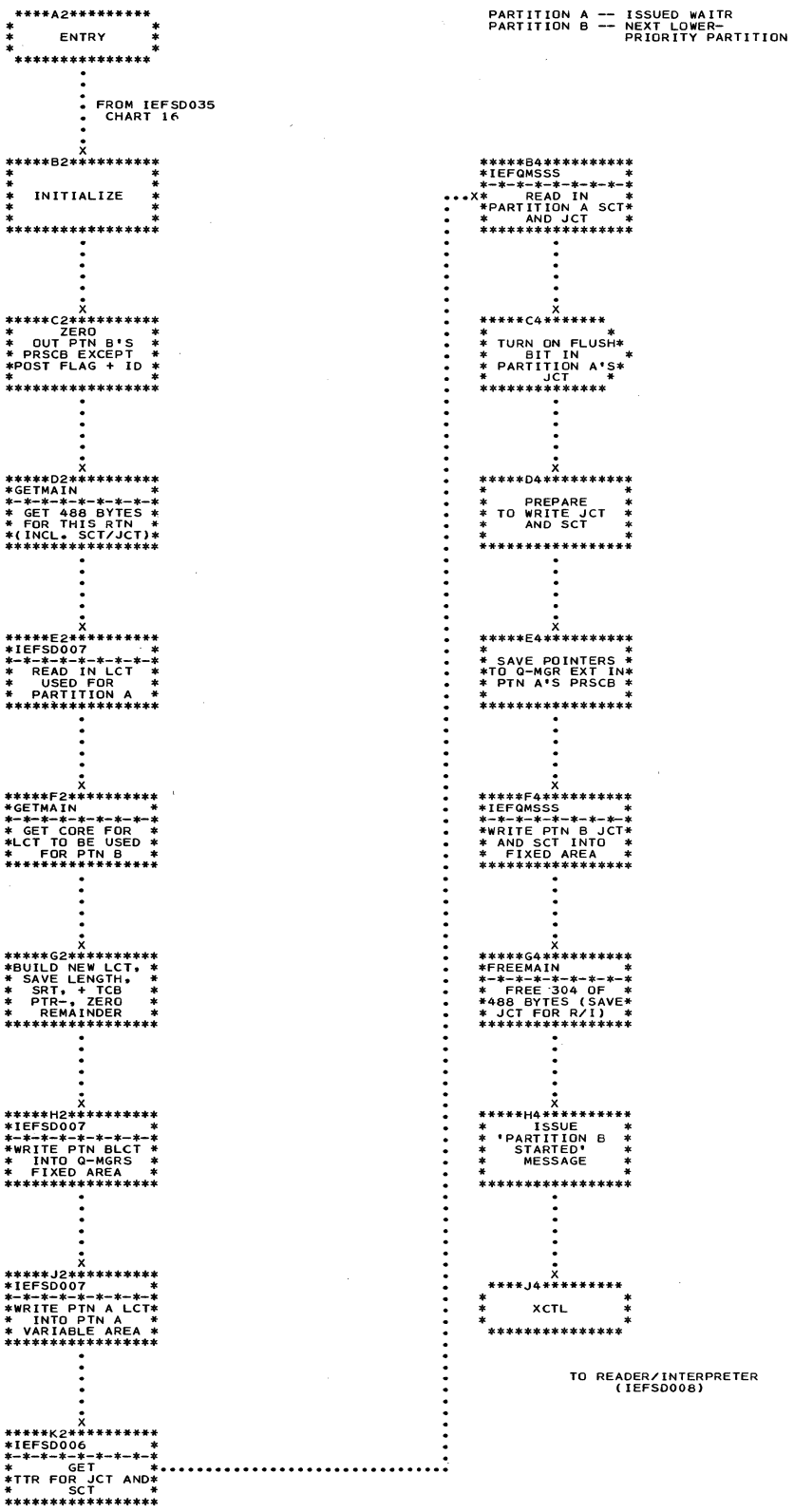


Chart 19. Enqueue Service Routine

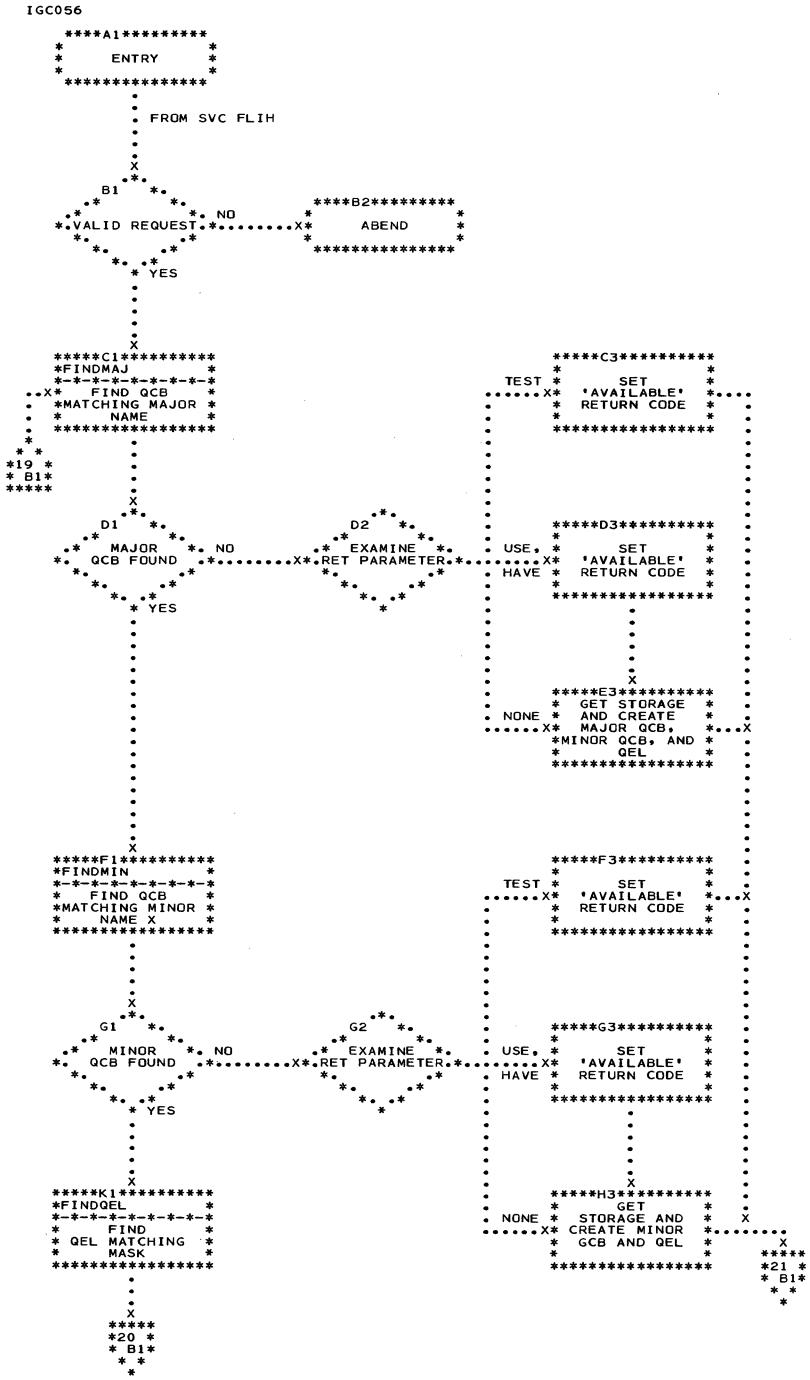


Chart 20. Enqueue Service Routine (continued)

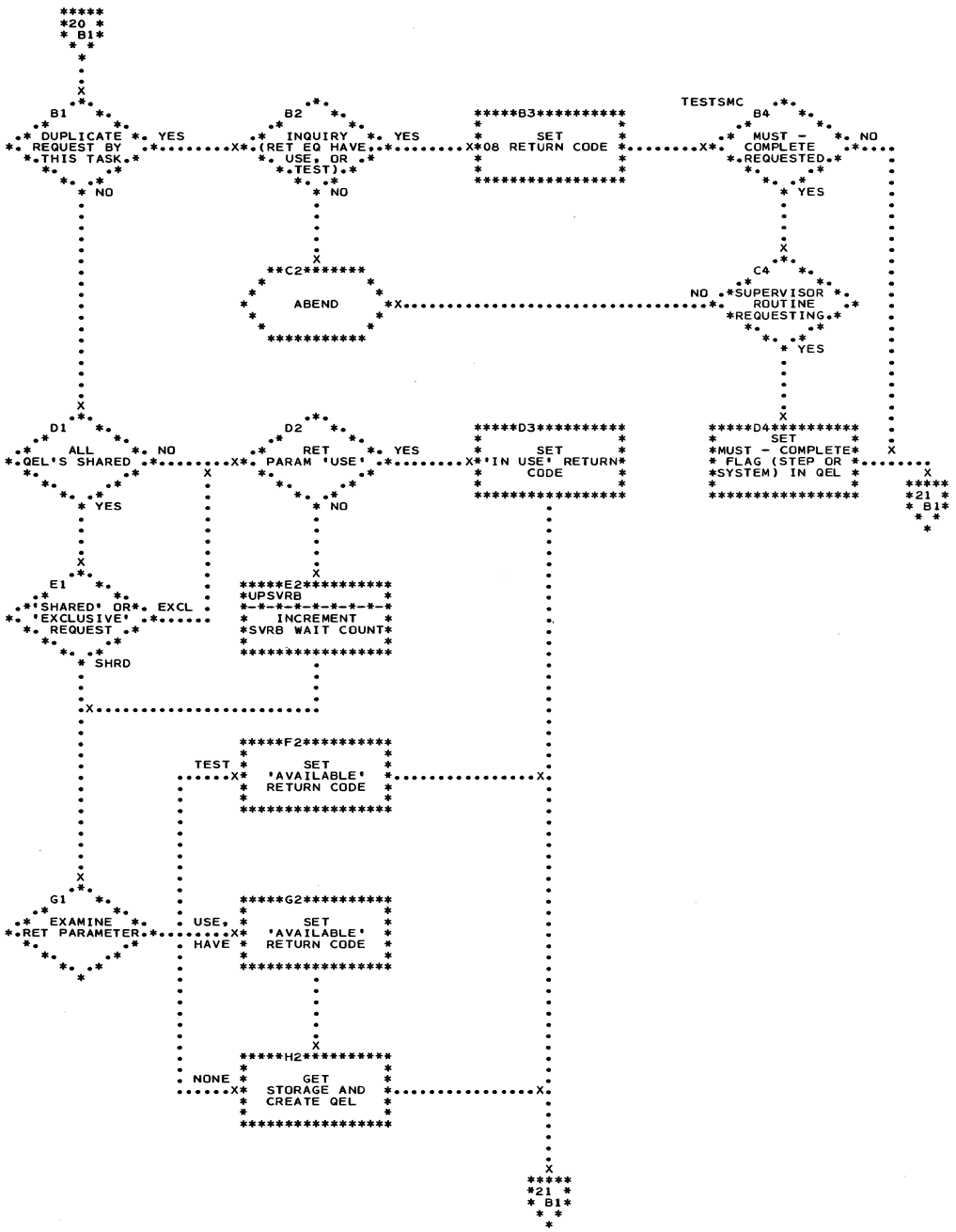


Chart 21. Enqueue Service Routine (continued)

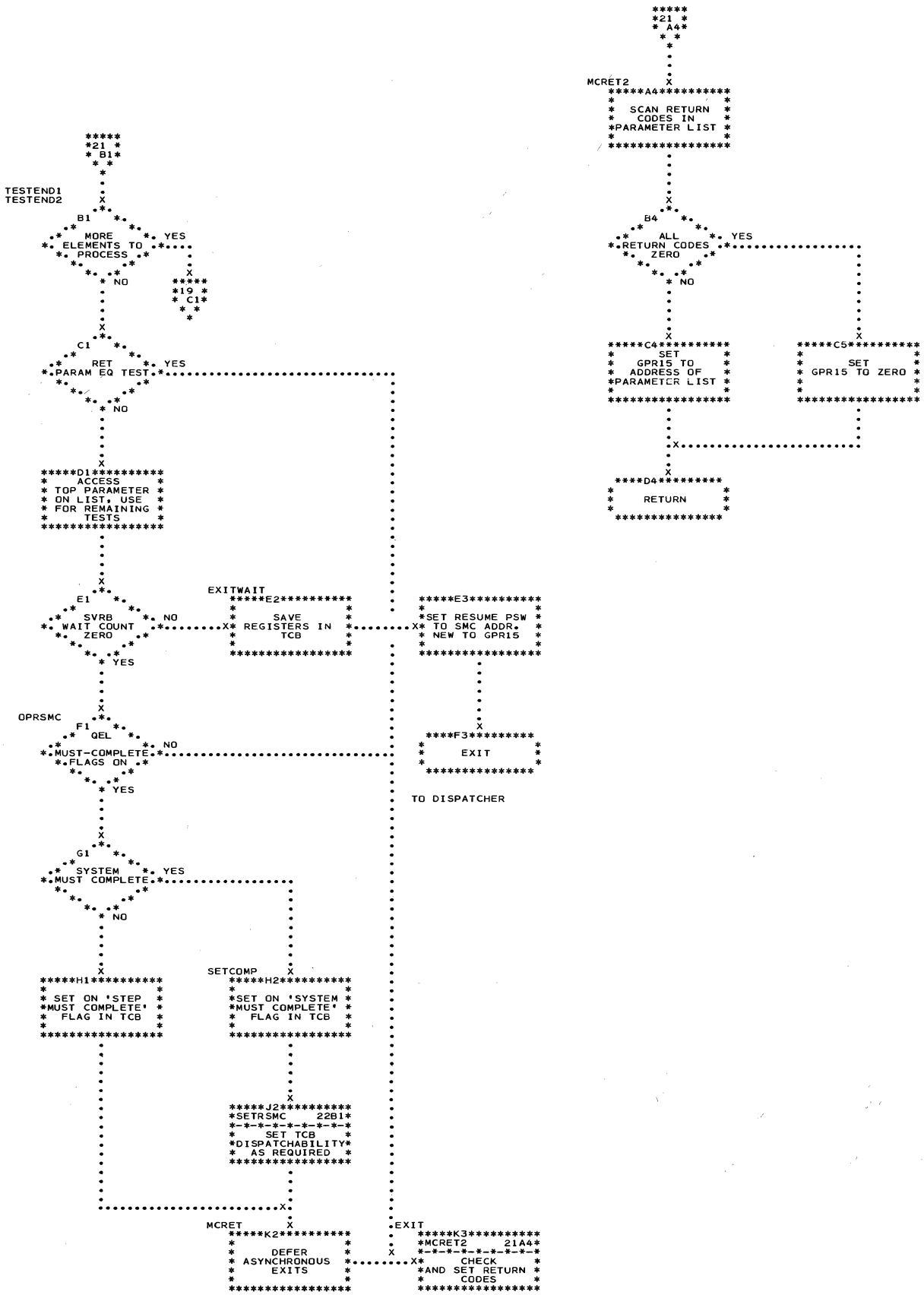


Chart 22. Must Complete Routine

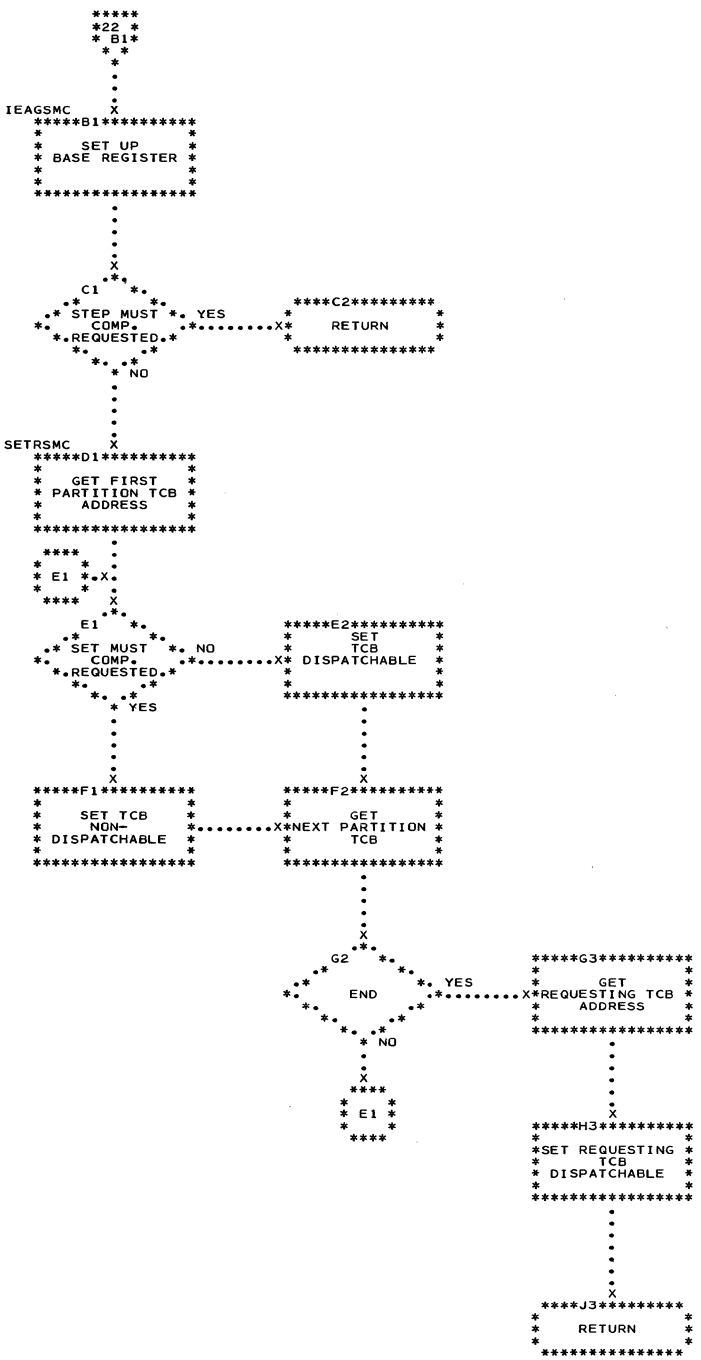


Chart 23. Dequeue Service Routine

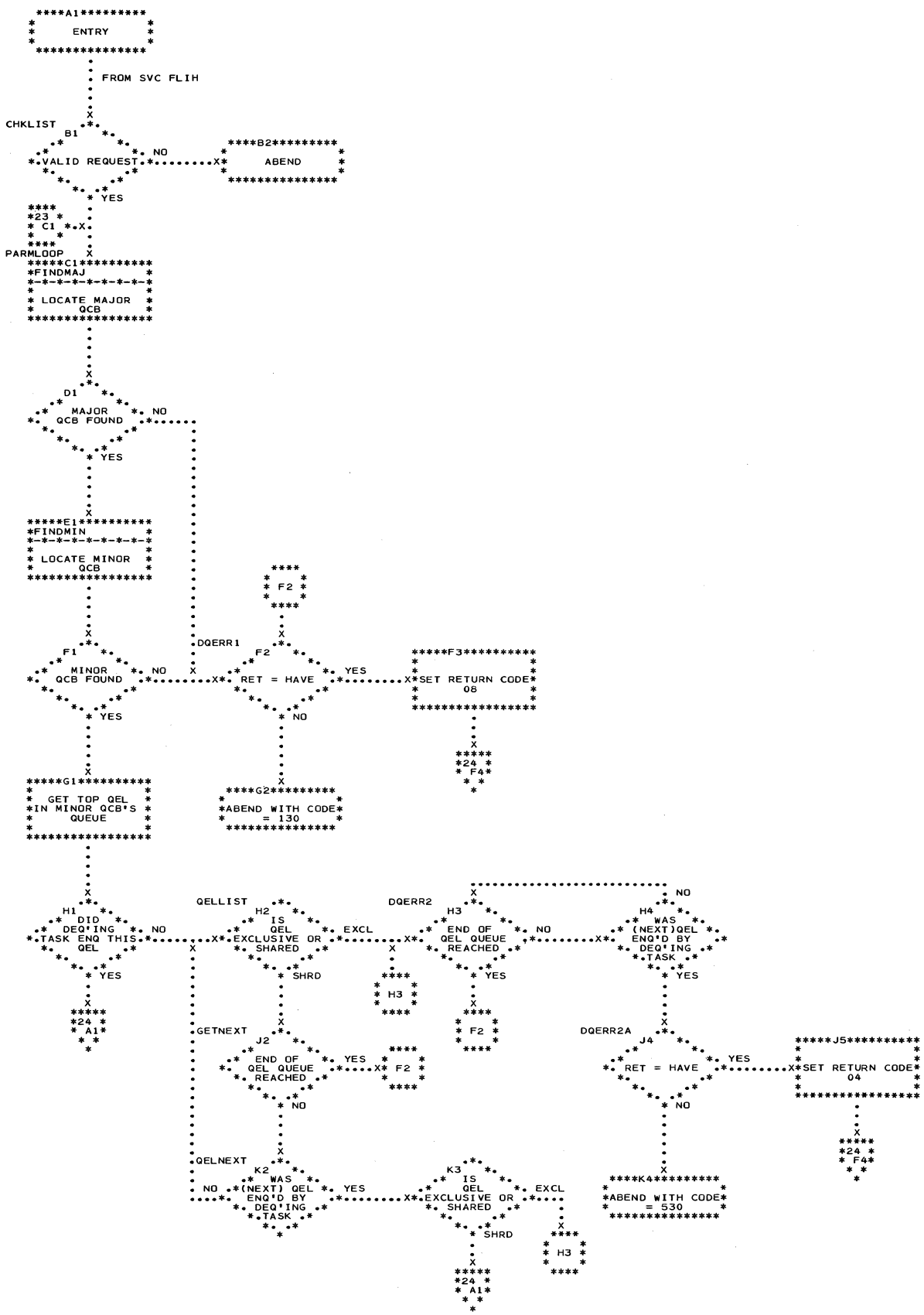


Chart 24. Dequeue Service Routine (continued)

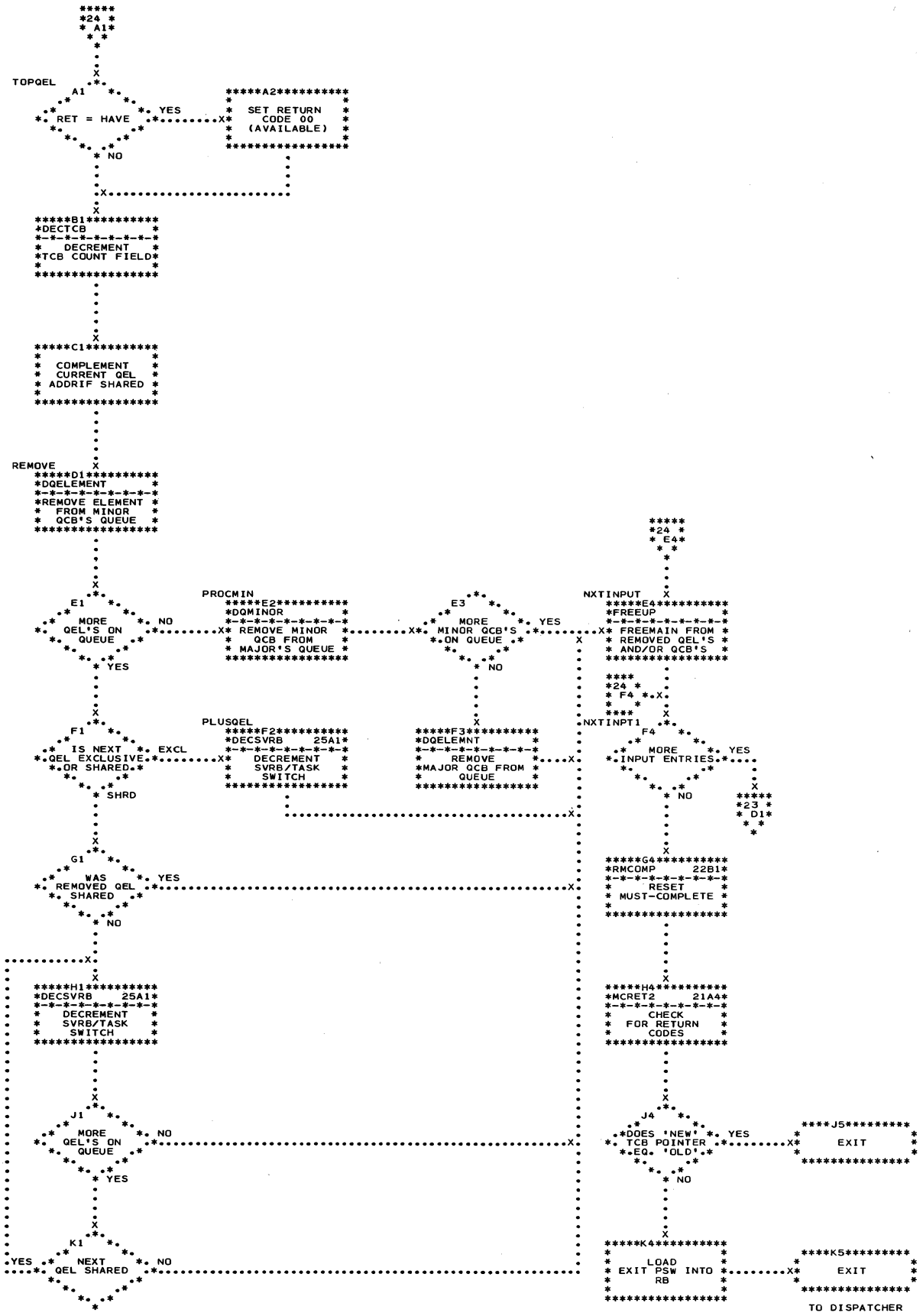


Chart 25. Decrement SVRB/TASK Switch Routine

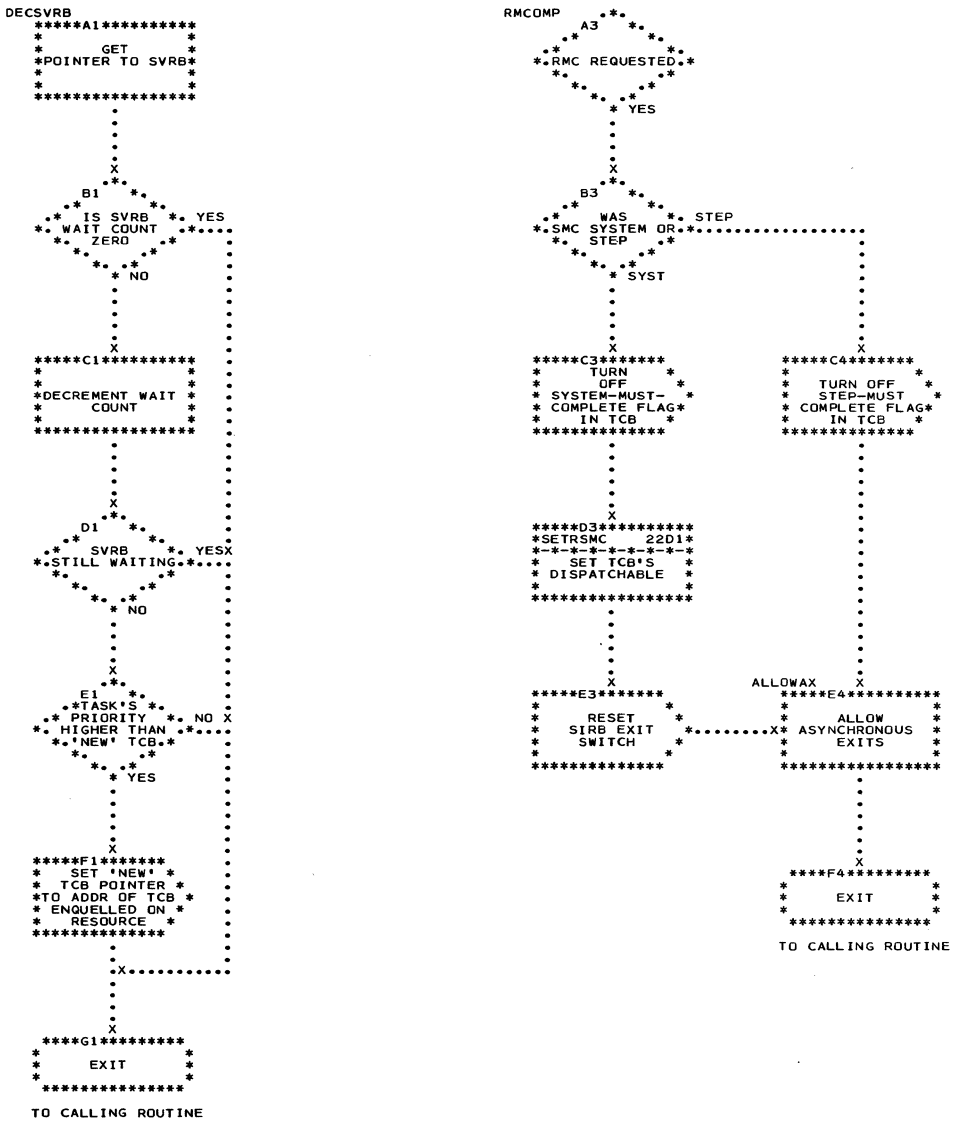


Chart 26. ENQ/DEQ Validity Check Routine

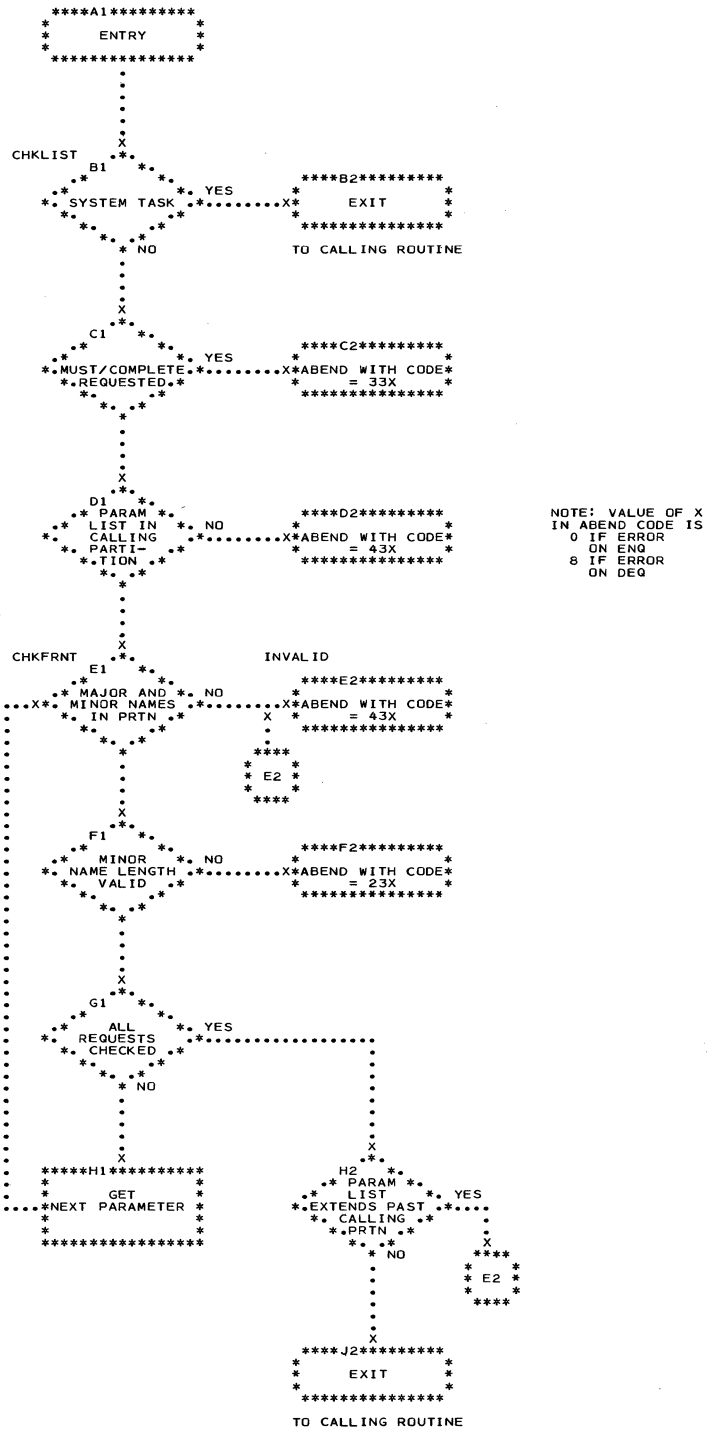


Chart 27. 18K Configuration Load Module Control Flow

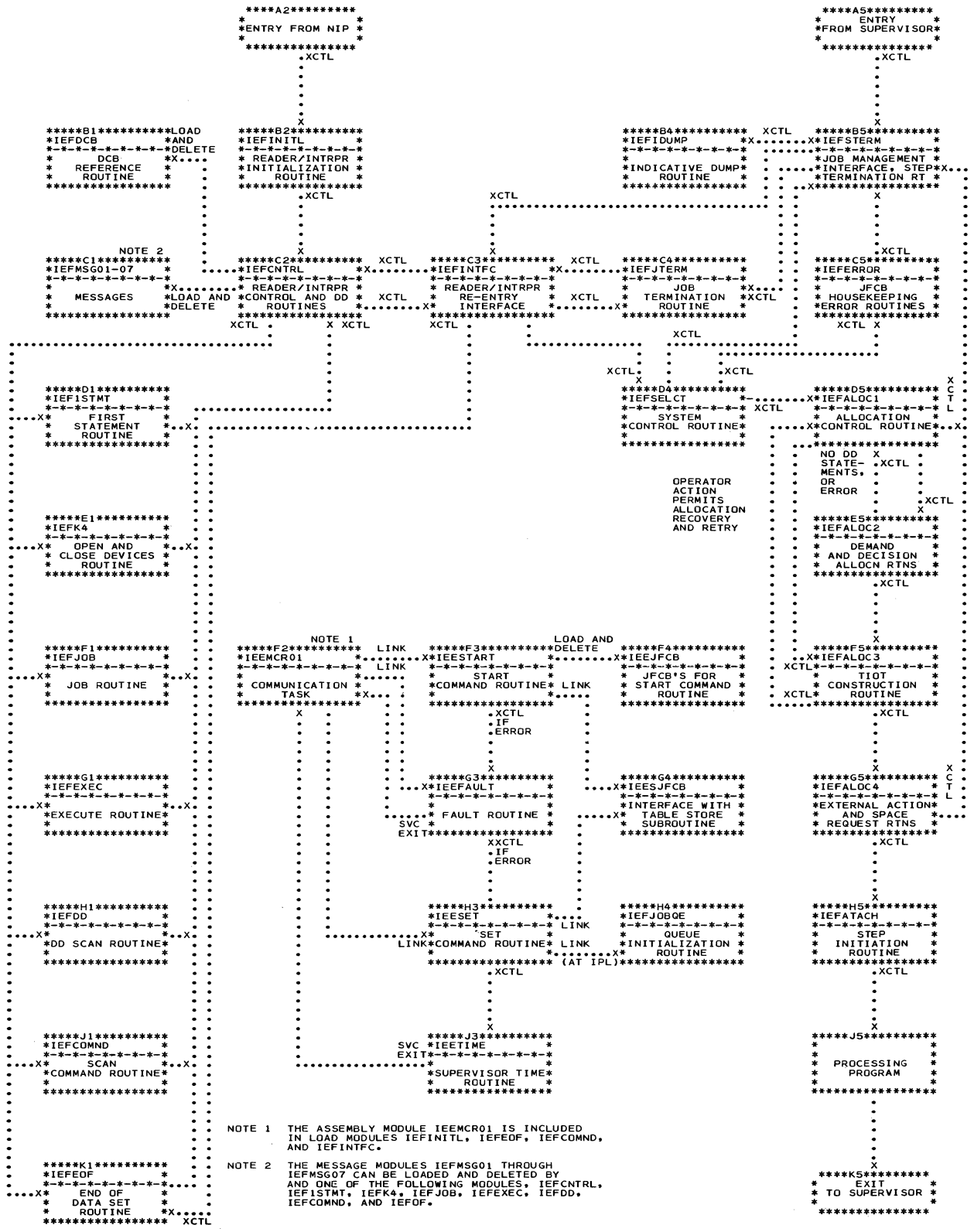
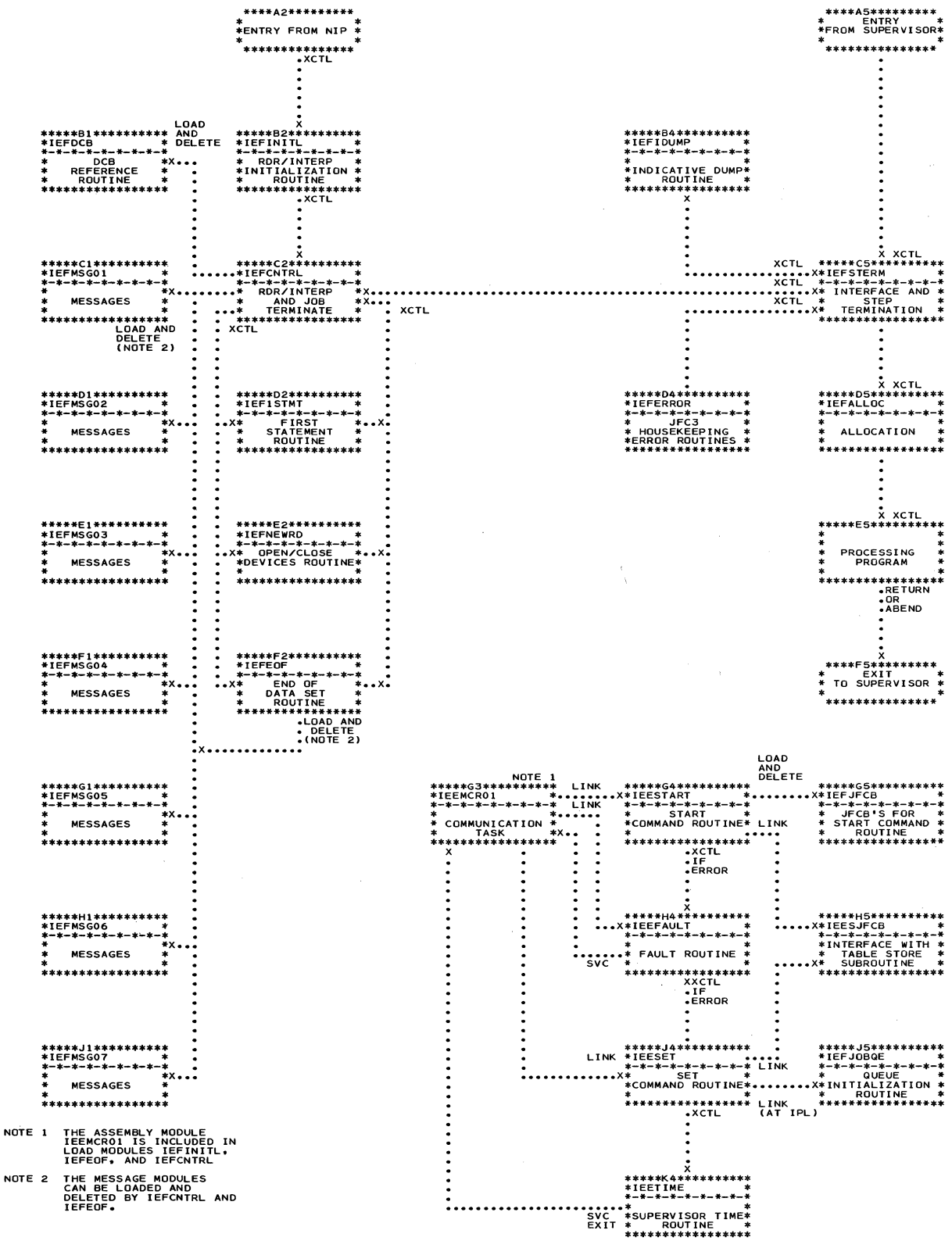


Chart 28. 44K Configuration Load Module Control Flow



NOTE 1 THE ASSEMBLY MODULE IEEMCR01 IS INCLUDED IN LOAD MODULES IEFINITL, IEFEOF, AND IEFCNTRL

NOTE 2 THE MESSAGE MODULES CAN BE LOADED AND DELETED BY IEFCNTRL AND IEFEOF.

Where more than one page reference is given, the first page number indicates the major reference.

Y27-7128-0

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y. 10601
[USA Only]

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]

READER'S COMMENTS

IBM System/360 Operating System; Control Program With Option 2
Program Numbers 360S-CI-505, 360S-DM-508
Y27-7128-0

Your comments will help us to produce better publications for your use. Please check or fill in the items below and add explanations and other comments in the space provided.

Which of the following terms best describes your job?

¤ Programmer	¤ Systems Analyst	¤ Customer Engineer
¤ Manager	¤ Engineer	¤ Systems Engineer
¤ Operator	¤ Mathematician	¤ Sales Representative
¤ Instructor	¤ Student/Trainee	¤ Other (explain) _____

Does your installation subscribe to the SRL Revision Service? ¤ Yes ¤ No

How did you use this publication?

¤ As an introduction
¤ As a reference manual
¤ As a text (student)
¤ As a text (instructor)
¤ For another purpose (explain) _____

Did you find the material easy to read and understand? ¤ Yes ¤ No (explain below)

Did you find the material organized for convenient use? ¤ Yes ¤ No (explain below)

Specific Criticisms (explain below)

Clarifications on pages
Additions on pages
Deletions on pages
Errors on pages

Explanations and Other Comments

No postage necessary if mailed in U.S.A.

FOLD FOLD

FIRST CLASS
PERMIT NO. 116
KINGSTON, N. Y.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN U. S. A.

POSTAGE WILL BE PAID BY

IBM CORPORATION
NEIGHBORHOOD ROAD
KINGSTON, N. Y. 12401

ATTN: PROGRAMMING PUBLICATIONS
DEPARTMENT 637

FOLD FOLD

IBM®

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y. 10601
[USA Only]

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]