File No. S360-36
Form Y27-7128-3

Gerhard

OBSOLETE
REL. 16

Program Logic

IBM System/360 Operating System
Control Program With MFT

Program Numbers 360S-CI-505
360S-CI-535

This publication describes the internal logic of the IBM System/360 Operating System control program to the extent that it is modified for MFT. Included are discussions of MFT system initialization, supervisor services, and job management services. The job management discussion includes explanations of the communications task, master scheduler task, queue manager, reader/interpreter, initiator/terminator, system output writers, and system task control.

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 MFT functions to the program listings (coding) for those functions. Comments in the listings provide information for thorough analysis and understanding of the coding.

Program Logic Manuals are intended for use by IBM customer engineers involved in program maintenance, and by system programmers involved in altering the program design. Program logic information is not necessary for program operation and use; therefore, distribution of this manual is limited to persons with program maintenance or modification responsibilities.

Restricted Distribution

RESTRICTED DISTRIBUTION: This publication is intended primarily for use by IBM personnel involved in program design and maintenance. It may not be made available to others without the approval of local IBM management.

Fourth Edition (May 1968)

This publication corresponds to Release 16. It is a major revision of, and renders obsolete, IBM System/360 Operating System: Control Program With MFT, Program Logic Manual, Form Y27-7128-2.

The text and illustrations have been changed to reflect the following:

- The Dispatcher and Master Scheduler descriptions have been modified to include time-slicing information relevant to MFT.
- The Small Partition Scheduling description has been revised to include more information.
- A description of the MFT modifications to the ABEND modules has been included in the Supervisor section.
- In Appendix A, three tables and/or work areas were revised to reflect changes in their format.
- In Appendix B, some module descriptions have been revised, and others added, to reflect incremental improvements.
- In Appendix C, two new flowcharts were added, including one for the ABEND control flow, and one for the Dispatcher with time-slicing included in the system. Several others were modified to reflect changes in the logic flow.

New or modified material is indicated by a vertical bar to the left of the affected text. The symbol (•) to the left of an illustration caption indicates a revision to that illustration.

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

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 train.

Copies of this and other IBM publications can be obtained through IBM branch offices.

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

This publication describes the differences in internal logic of the control program that result from the inclusion of multiprogramming with a fixed number of tasks (MFT). 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 this publication.

The manual is divided into four major sections. The Introduction describes control program functions, control program and main storage organization, and control program processing flow. The Nucleus Initialization Program section describes differences introduced by MFT into system initialization. The Supervisor section describes supervisor functions including an explanation of task dispatching in MFT.

The Job Management section contains the changes to the job management components made by MFT. Job management is divided into three major components: reader/interpreter, initiator/terminator, and output writer. Also described are the Queue Manager which is used by all three major job management components, the Communications Task which handles operator-system communication, and the Master Scheduler Task which processes operator commands.

Appendix A contains descriptions of major tables and work areas used by MFT. Appendix B contains descriptions of modules used by MFT. Appendix C contains MFT flowcharts.

PREREQUISITE PUBLICATIONS

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

- IBM System/360: Principles of Operation, Form A22-6821
- 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: MVT Job Management, Program Logic Manual, Form Y28-6660
- IBM System/360 Operating System: Initial Program Loader and Nucleus Initialization Program, Form Y28-6661
- IBM System/360 Operating System: Planning for Multiprogramming With a Fixed Number of Tasks (MFT), Form C27-6939

The following publications may be useful for reference although they are not prerequisites for this publication.

- IBM System/360 Operating System: Concepts and Facilities, Form C28-6535
- 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: MVT Control Program Logic Summary, Form C28-6658
- IBM System/360 Operating System: Input/Output Supervisor, Program Logic Manual, Form Y28-6616

CONTENTS

ILLUSTRATIONS

FIGURES

TABLES

CHARTS

In a single task environment, main storage is divided into two areas: the fixed area, and the dynamic area. In multiprogramming with a fixed number of tasks (MFT), the dynamic area is divided further into as many as fifty-two discrete areas called partitions. Figure 1 shows the division of main storage.

The system area, located in the lower portion of main storage, contains the resident portion of the control program, and control blocks and tables used by the system. The size of the system area depends on the number of partitions established by the user, and the control program options selected at system generation.

Partitions are defined within the dynamic area, located in the upper portion of main storage, at system generation. The number of partitions may be varied within the number specified at system generation, and the sizes and job classes of partitions may be redefined at system initialization or during operation. (See IBM System/360 Operating System: Planning for Multiprogramming with a Fixed Number of Tasks (MFT), Form C27-6939.) Each partition may be occupied by a processing program, or by control program routines that prepare job steps for execution (job management routines), or handle data for a processing program (access method routines).

MFT provides for the concurrent execution of as many as 15 problem programs, 3 input readers, and 36 output writers, each in its own fixed partition of main storage, as long as the total number of partitions does not exceed 52. The MFT system provides for task switching among the tasks operating in the partitions, and between those tasks and the communications task and master scheduler task in the system area.

Task dispatching in MFT differs from the primary control program (PCP) primarily in that task switching is required, and that certain system functions such as abnormal termination must be carried out so that other, unrelated, tasks are not affected. The dispatching priority of a task is determined by the relative position of the partition used to process the task. The highest-priority partition (P0) is at the highest address in storage. Successively lower partitions (P1 - P51) have correspondingly lower priorities. Control of the CPU is given to the program in the highest-priority partition that is ready.

The integrity of programs operating under MFT is preserved if the storage protection feature is included. MFT uses

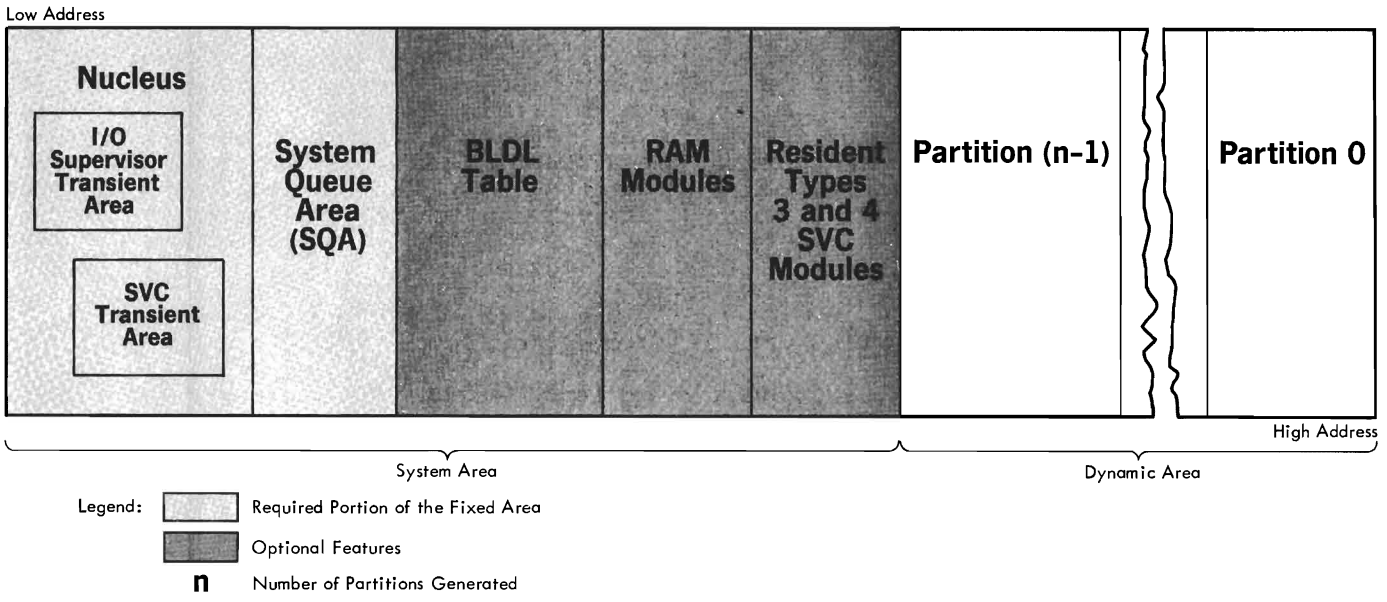


Figure 1. Main Storage Organization in MFT

the 16 protection keys to prevent a user job from modifying the control program or another job; it uses the two operating states of the CPU to restrict the use of control and I/O instructions.

Because many components of MFT are similar to those of PCP and multiprogramming with a variable number of tasks (MVT), many of the modules for a given MFT component are the same for the comparable component in either PCP or MVT. Therefore, this publication describes differences between MFT and the other configurations. The corresponding PCP and MVT routines are described in the following IBM System/360 Operating System program logic manuals and are referenced where applicable:

Fixed Task Supervisor, Form Y28-6612

MVT Supervisor, Form Y28-6659

MVT Job Management, Form Y28-6660

Information on modified or new routines for MFT is contained in the three sections that follow this introduction.

The Nucleus Initialization Program (NIP) section describes the changes that have been made to NIP for MFT. The major area of change is the deletion of some of the NIP functions, which are now performed by other system tasks (e.g., system initialization is performed by the master scheduler task in MFT).

The Supervisor section describes the task management modifications made to the supervisor for MFT. The major area of change has been in the initialization of main storage.

The Job Management section describes modifications and additions to the routines for processing communications with the programmer and the operator. The major changes are in the master scheduler task, and the MFT initiator. Other modifications have been made to the queue manager, the reader/interpreter, system output writer, and system task control routines.

FUNCTIONS OF THE CONTROL PROGRAM WITH MFT

As in PCP and MVT, the control program routines of MFT have three major functions: job management, task management, and data management.

JOB MANAGEMENT

Job management is the processing of communications from the programmer and operator to the control program. There are two types of communications: operator commands, which start, stop, and modify the processing of jobs in the system, and job control statements, which define work being entered into the system. Processing of these commands and statements is referred to as command processing and job processing, respectively.

TASK MANAGEMENT

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

- Interruption supervision
- Task supervision
- Main Storage supervision
- Contents supervision
- Overlay supervision
- Timer supervision

The task management routines are collectively referred to as the "supervisor."

DATA MANAGEMENT

Data management routines control all operations associated with input/output devices: allocating space on volumes, channel scheduling, storing, naming, and cataloging data sets, moving data between main and auxiliary storage, and handling errors that occur during input/output operations. Data management routines are used by processing programs and control program routines that require data movement. Processing programs use data management routines primarily to read and write required data, and also to locate input data sets and to reserve auxiliary storage space for output data sets of the processing program.

Data management routines are of five categories:

- Input/Output (I/O) supervisor, which supervises input/output requests and 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.

The operation of these routines is identical with MVT and is described in the following IBM System/360 Operating System program logic manuals:

Input/Output Supervisor, Form Y28-6616

Sequential Access Methods, Form Y28-6604

Indexed Sequential Access Methods, Form Y28-6618

Basic Direct Access Method, Form Y28-6617

Graphics Access Method, Form Y27-7113

Catalog Management, Form Y28-6606

Direct Access Device Space Management, Form Y28-6607

Input/Output Support (OPEN/CLOSE/EOV), Form Y28-6609

CONTROL PROGRAM ORGANIZATION

The control program resides on auxiliary storage in three partitioned data sets created when the system is generated. These data sets are:

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

RESIDENT PORTION OF THE CONTROL PROGRAM

The resident portion (nucleus) of the control program resides in SYS1.NUCLEUS. It is made up of those routines, control blocks, and tables that are brought into main storage at initial program loading (IPL) and are never overlaid by another part of the operating system. The nucleus is loaded into the fixed area of main storage.

The resident task management routines include all of the routines that perform:

- Interruption supervision
- Main storage supervision
- Timer supervision

They also include portions of the routines that perform:

- Task supervision
- Contents supervision
- Overlay supervision

These routines are described in this publication, and in the program logic manual IBM System/360 Operating System: Fixed Task Supervisor, Form Y28-6612.

The resident job management routines are those routines of the communications task that receive commands from the operator. The MFT communications task is described in this publication.

The resident data management routines are the input/output supervisor and, optionally, the BLDL routines of the partitioned access method. These routines are described in the following IBM System/360 Operating System program logic manuals:

Input/Output Supervisor, Form Y28-6616

Sequential Access Method, Form Y28-6604

The user may also select access method routines to be made resident. These routines are referred to as resident access methods (RAM), and are loaded during system initialization rather than during Open processing. RAM modules reside adjacent to the higher end of the system queue area unless the BLDL table is resident (see Figure 1).

Normally-transient SVC routines (i.e., types 3 and 4 SVC routines) can be made resident through the RSVC option, specified by the user. At IPL, NIP loads these routines adjacent to the higher end of the RAM modules. If there is no resident BLDL table or RAM modules, the routines are loaded adjacent to the higher end of the system queue area. (See Figure 1.)

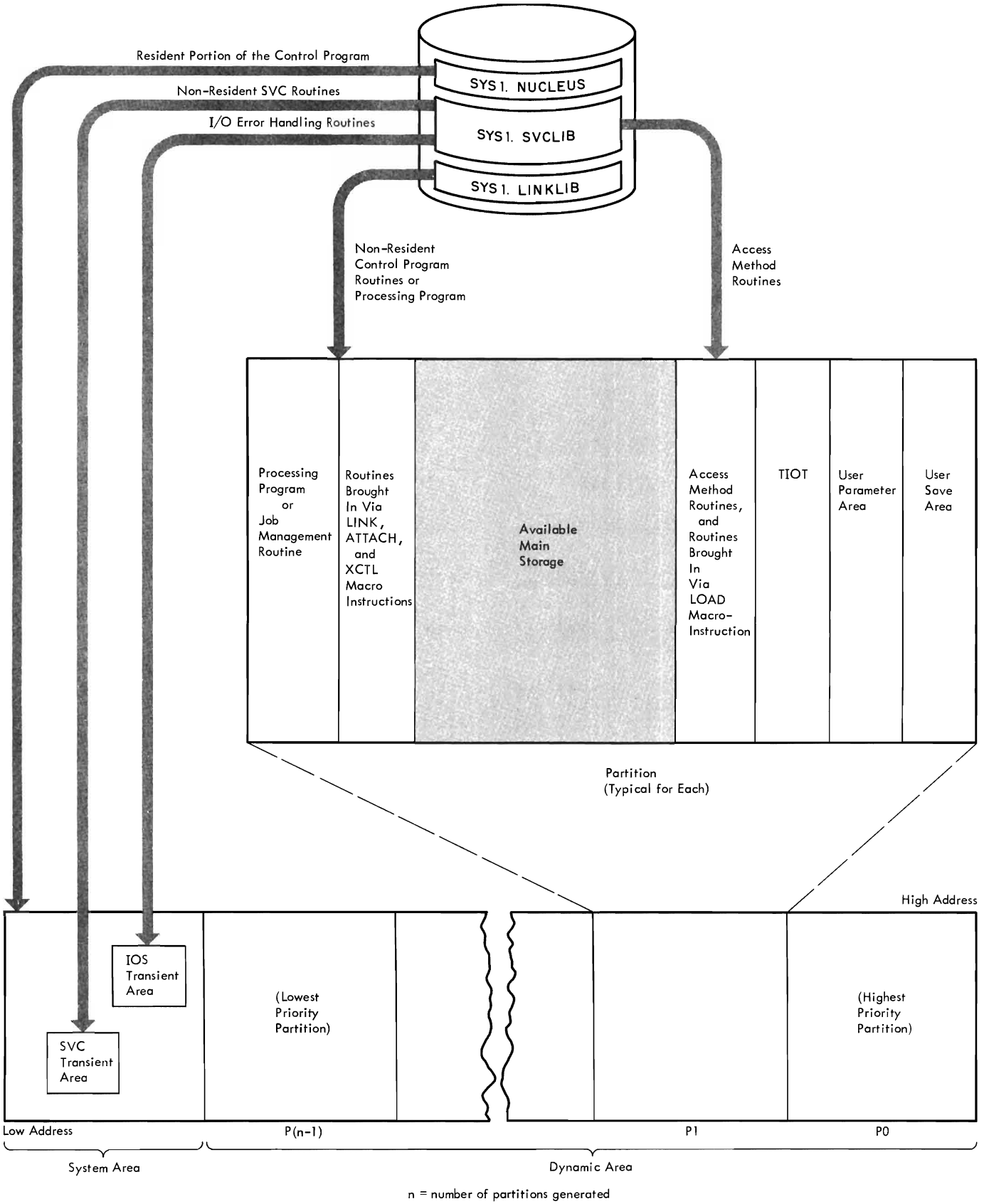


Figure 2. Division of Main Storage

NONRESIDENT PORTION OF THE CONTROL PROGRAM

The nonresident portion of the control program comprises routines that are loaded into main storage as they are needed, and which can be overlaid after their completion. The nonresident routines operate from the partitions and from two sections of the nucleus called transient areas (described below).

MAIN STORAGE ORGANIZATION

Main storage in MFT is organized similarly to main storage in MVT, except that MFT does not use a link pack area (see Figure 1).

FIXED AREA

In MFT (as in PCP and MVT) the fixed area is that part of main storage into which the nucleus is loaded at IPL. The storage protection key of the fixed area is zero so that its contents can be modified by the control program only. The fixed area also contains two transient areas into which certain nonresident routines are loaded when needed: the SVC transient area (1024 bytes) and the I/O supervisor transient area (1024 bytes). These areas are used by nonresident SVC routines and nonresident I/O error-handling routines, respectively, which are read from SYS1.SVCLIB.

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. The transient area routines operate with a protection key of zero, as do other routines in the fixed area.

System Queue Area

The system queue area (SQA) is established by NIP adjacent to the fixed area and provides the main storage space required for tables and queues built by the control program. The SQA must be at least 1600 bytes for a minimum two-partition system. Its storage protection key is zero so that it can be modified by control program routines only. Data in the system queue area indicates the status of all tasks.

DYNAMIC AREA

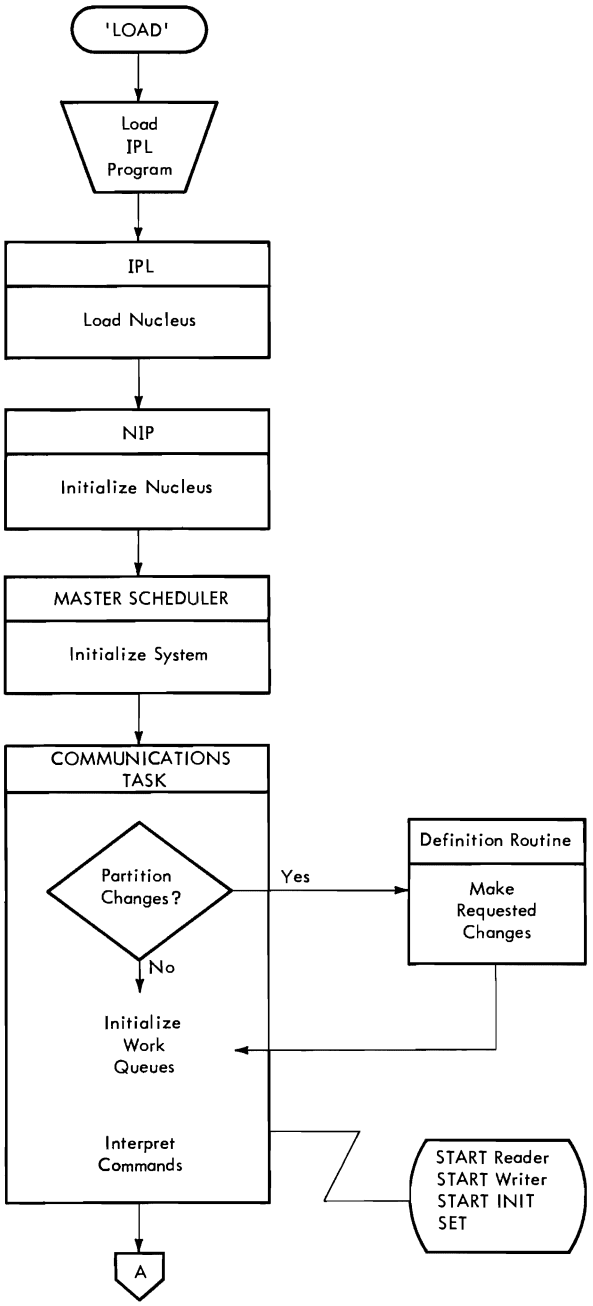
Figure 2 shows how the contents of each partition in the dynamic area are organized and how they are related to the rest of main storage. Routines are brought into the high or low portion of an MFT partition similarly to the way routines are brought into the entire dynamic area of PCP. Job management routines, processing programs, and routines brought into storage via a LINK, ATTACH, or XCTL macro instruction, are loaded at the lowest available address. The highest portion of the partition is occupied by the user parameter area and user save area. The next portion of the partition is occupied by the task input/output table (TIOT) which is built by a job management routine (I/O Device Allocation routine). This table is used by data management routines and contains information about DD statements.

Each partition may be used for a problem program as well as for system tasks (readers, initiators, and writers). When the control program requires main storage to build control blocks or work areas, it obtains this space from the partition of the processing program that requested the space. Access method routines and routines brought into storage via a LOAD macro instruction are placed in the highest available locations below the task input/output table.

Working storage and data areas are assigned from the highest available storage in a partition.

THEORY OF OPERATION

Figure 3 describes the overall processing flow through each job 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 other partitions, and terminates the job step.

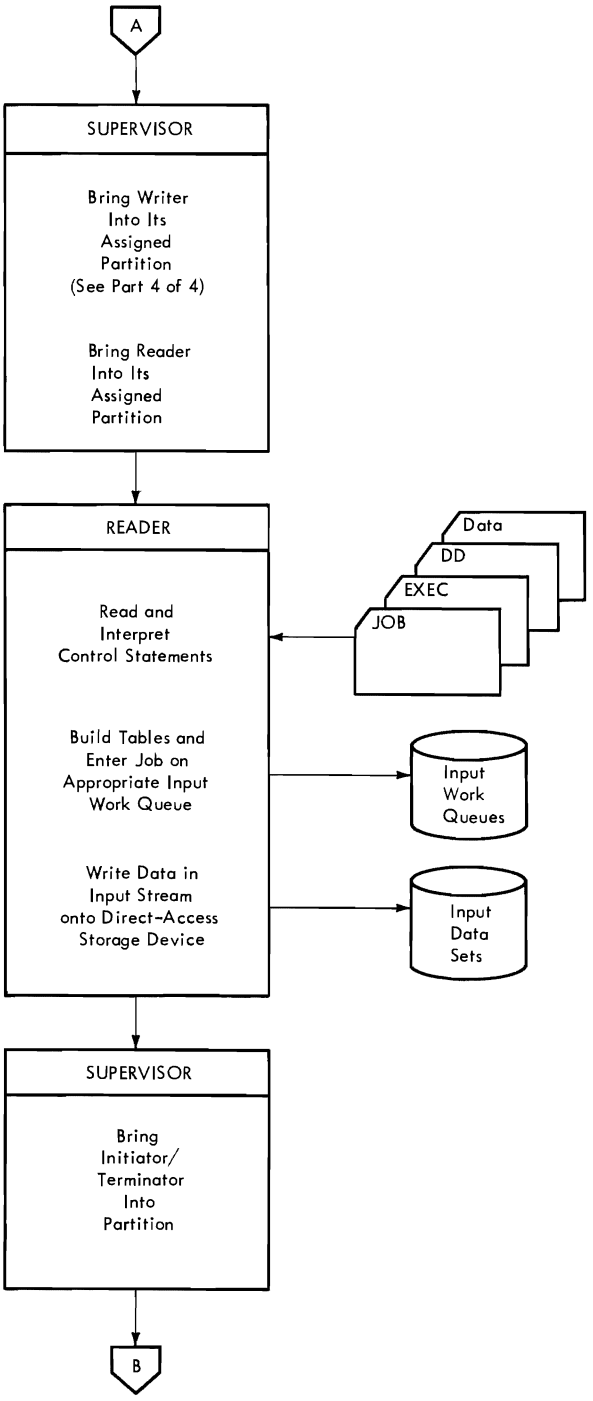


To load the nucleus, the operator sets the LOAD UNIT switches to the device on which the system residence volume is mounted, and presses the LOAD button on the operator control panel. This causes an IPL record to be read and to be given control. This record causes the second IPL record to be read, which in turn, enables the rest of the IPL program to be read into main storage.

The IPL program searches the volume label of the system residence volume to locate the volume table of contents (VTOC). The VTOC is then searched for the address of the nucleus data set (SYS1.NUCLEUS). The nucleus is brought into the system area, and NIP is brought into the dynamic area. NIP receives control from the IPL program. It performs both required and optional initialization for control program operation including initializing the Communication Vector Table (CVT), and general system initialization, such as determining user options. After completing its processing, NIP passes control to the master scheduler task (MST) which initializes main storage.

Partitions are established by the master scheduler at system initialization according to the sizes and job class(es) established at system generation by the PARTITNS macro instruction. The MST also places a copy of the Initiator/Terminator into each scheduler-size partition; a copy of the small partition module is placed in each small partition. The communications task receives control from the MST and communicates with the operator to request any partition changes. After the requested changes, if any, have been made by the definition routines, the work queues are initialized. The automatic commands are displayed, and the READY message is issued.

Figure 3. MFT Theory of Operation (Part 1 of 4)



When the required SET and START commands are entered, control is given to the master command routine. After the communications task completes processing of the commands, a copy of the Reader/Interpreter (reader) is brought into its appropriate partition. If a START writer command is entered, a copy of a writer is also brought into the specified partition(s).

When the reader gets control, it reads control statements and data from the input job stream. Information from the JOB, EXEC, and DD statements controls the execution of each job step. This information is placed in the following tables:

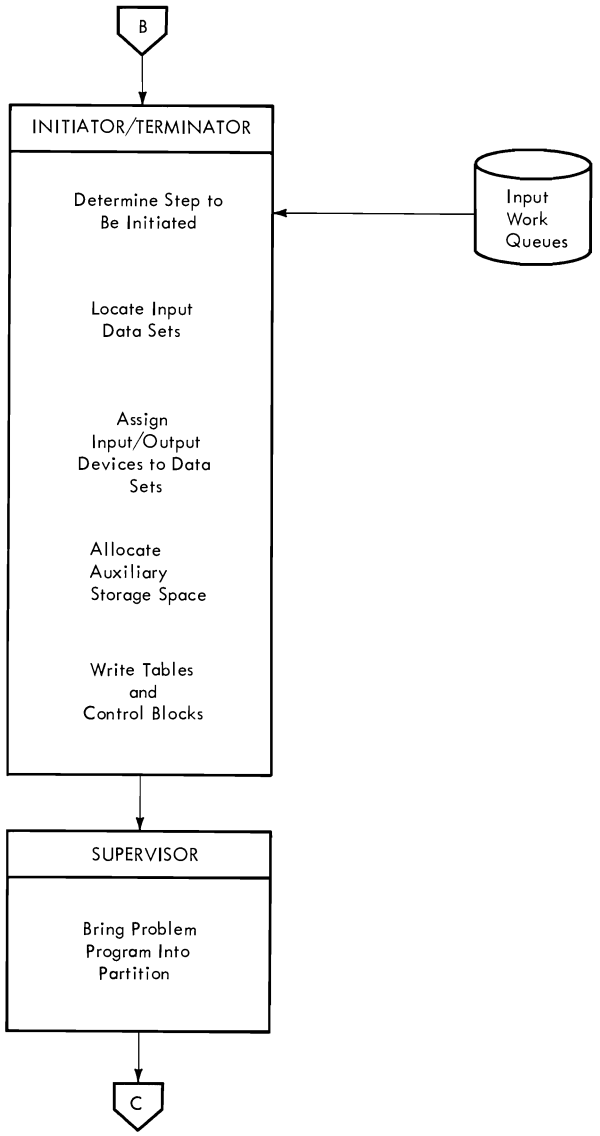
- Job control table (JCT) for the job being read
- Step control table (SCT) for the step being read
- Data set enqueue table (DSENQ) for the job being read
- Job file control block (JFCB) and step input/output table (SIOT) for each data set being used or created by the job step
- Volume table (VOLT) containing each volume serial number to be used by the job.

Information from these tables and control blocks is updated with information in the data control block (DCB) and data set control block (DSCB) or volume label when a data set is opened during step execution.

The reader then places these updated control blocks into the input work queue corresponding to the CLASS parameter on the JOB statement. Data sets in the input stream are written onto a direct-access storage device for later use by the problem program.

After the reader has completed processing all input for a job and has entered the job on an input work queue, all initiators that are waiting for that job class are posted. If the job is for a small partition, the small partition module is also posted.

Figure 3. MFT Theory of Operation (Part 2 of 4)



After receiving control, the initiator/terminator prepares to initiate the highest priority job in its primary input work queue. Using information which the reader extracted from the DD statement, the initiator/terminator processes the user accounting routine, in addition to the following:

Locates Input Data Sets: The Allocation routine, running as a subroutine of the initiator/terminator, determines the volume containing a given input data set by examining the JFCB, or by searching the catalog. This search is performed by a catalog management routine entered from allocation. (A description of the routines that maintain and search the catalog is given in IBM System/360 Operating System: Catalog Management, Program Logic Manual, Form Y28-6606.)

Allocates I/O Devices: A job step cannot be initiated unless there are enough I/O devices to fill its needs. Allocation determines whether the required devices are available, and makes specific assignments. If necessary, messages are issued to the operator to request the mounting of volumes.

Allocates Auxiliary Storage Space: Direct access volume space required for output data sets of a job step is acquired by the allocation routine, which uses the Direct Access Device Space Management (DADSM) routines. (A description of the operation of the DADSM routines is given in the publication IBM System/360 Operating System: Direct Access Device Space Management, Program Logic Manual, Form Y28-6607.)

The JFCB, which contains information concerning the data sets to be used during step execution, is written on auxiliary storage. This information is used when a data step is opened, and when it is closed, the job step is terminated.

The initiator causes itself to be replaced by the problem program it is initiating (if for a large partition), or initiates the job in a small partition.

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

Figure 3. MFT Theory of Operation (Part 3 of 4)

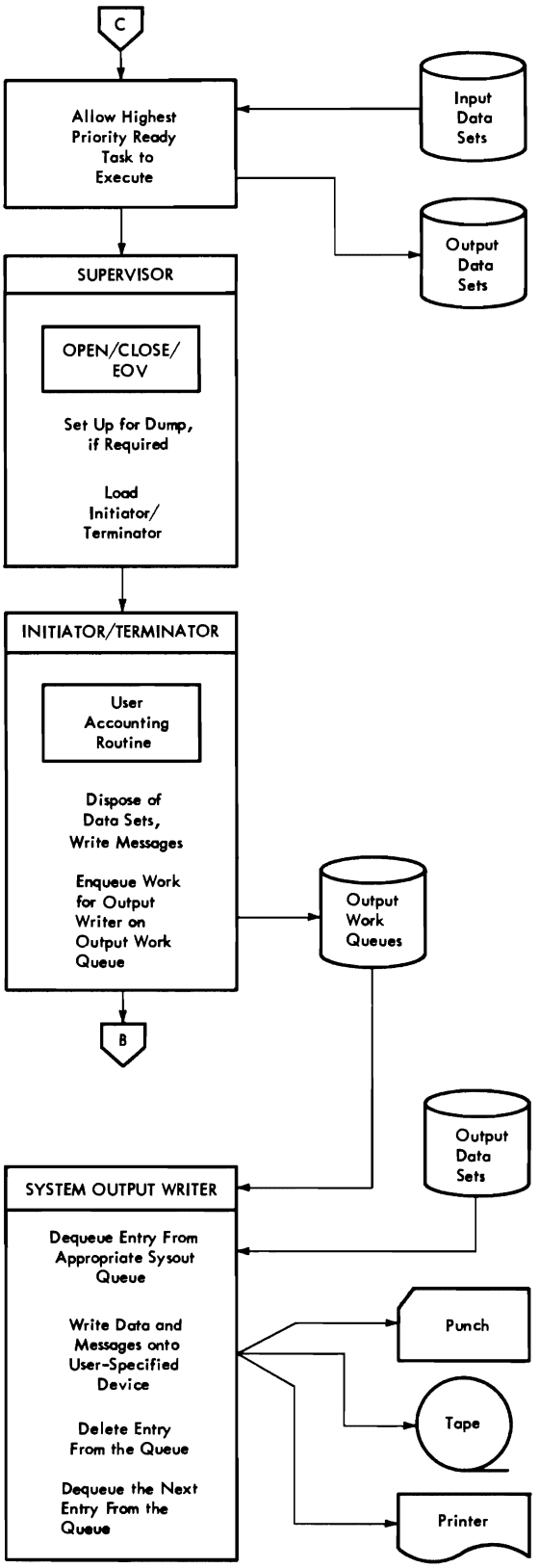


Figure 3. MFT Theory of Operation (Part 4 of 4)

The problem program processes until it terminates either normally or abnormally, though it may not retain exclusive control of the CPU. Control always is received by the highest priority task that is ready to execute.

When the problem program terminates, the supervisor receives control. The supervisor uses the OPEN/CLOSE/EOV routines to close any open data control blocks. (These routines are described in IBM System/360 Operating System: Input/Output Support (OPEN/CLOSE/EOV), Program Logic Manual, Form Y28-6609.)

Under abnormal termination conditions, the supervisor may also provide special termination procedures, such as a storage dump. The supervisor passes control to the initiator/terminator, which is either brought into the partition in which termination is to occur, or is brought into the large partition to terminate a small partition.

The initiator/terminator releases the I/O devices, and disposes of data sets used and/or created during the job step by reading tables prepared during initiation (JCT, SCT, TIOT, etc.). These tables include information such as disposition of data sets. It then executes an installation accounting routine if one is provided.

At job termination, an entry is made on the user specified output work queue; later the problem program output data can be written by a system output writer from a system direct-access storage device to a user-specified device. The initiator/terminator then initiates the next job step.

An output writer operates concurrently with readers, problem programs, and other writers. When the START command is issued for a writer, the writer dequeues the first entry in the specified output (SYSOUT) queue. If no requests have been enqueued in that output queue from the problem programs, the writer is placed in a wait condition until a job is terminated that has system messages or output data sets. After the entry is dequeued from the output queue, the writer transmits the data sets to the specified card punch, magnetic tape unit, or printer. When the last record has been processed, the writer deletes the queue entry before dequeuing the next entry.

NUCLEUS INITIALIZATION PROGRAM

The Nucleus Initialization Program (NIP) in MFT is essentially the same as that used for MVT. However, some routines have been removed, either because their functions are not required by MFT or because they are performed by other system tasks (e.g., the initialization of main storage is handled by the master scheduler task to support partition redefinition). This section describes those areas where MFT NIP differs from MVT NIP. A complete description of NIP for MVT can be found in IBM System/360 Operating System: Initial Program Loader and Nucleus Initialization Program, Program Logic Manual, Form Y28-6661.

NIP is a control section included by the linkage editor in the nucleus during system generation. The IPL program loads the nucleus and NIP and then passes control to NIP. NIP performs required and optional initialization functions for control program operation. Initialization includes nucleus table initialization and general system initialization.

NIP first operates in its own environment, using its stand-alone I/O routine. As it initializes the nucleus, NIP begins to use system routines, including the I/O Supervisor, to complete nucleus initialization.

GENERAL SYSTEM INITIALIZATION

The primary area of difference between MFT NIP and MVT NIP is in general system initialization. Differences are:

- Defining Control Program Areas
- Determining User Options
- Redefining the System Queue Area
- Locating the BLDL List and Resident Modules
- Preparing Main Storage

These differences are described in the following paragraphs.

DEFINING CONTROL PROGRAM AREAS

In MFT, NIP alters the contents of the master scheduler task boundary box to define an area including all of main storage from the end of the fixed area to the highest addressable byte of main storage. (The original boundary box contents are saved to be passed to master scheduler initialization routine IEFSD569 in a parameter list.) The area assigned to the master scheduler task is not used during NIP execution. NIP establishes the system queue area (SQA) adjacent to the nucleus. The SQA boundaries are determined by the upper boundary of the nucleus and the SQA size specified during system generation.

Main storage may be expanded by including IBM 2361 Core Storage (core storage) units in the system. Main Storage Hierarchy Support for IBM 2361 Models 1 and 2 permits access to either processor storage (hierarchy 0) or core storage (hierarchy 1). Each partition established during system generation is described by a boundary box. The first half of the boundary box describes the processor storage partition segment and the second half describes the core storage partition segment. Any partition segment not assigned main storage in the system has the applicable boundary box pointers set to zero. If a partition is established entirely within hierarchy 1, the processor storage pointers in the first half of the partition's boundary box are set to zero. If a partition segment is not generated in core storage, the core storage pointers in the second half of the partition's boundary box are set to zero. If core storage has been included in the system, but is off-line, the second half of the boundary box will contain zeros. If core storage is excluded from the system, the second half of the boundary box is not generated.

DETERMINING USER OPTIONS

After NIP issues the message SPECIFY SYSTEM PARAMETERS, the operator may enter the following MFT user options (using the keywords indicated):

- A larger or smaller system queue area (SQS=)
- Additional modules for the resident access method routines (RAM=)
- A resident module list resulting from BLDL information (BLDL=)
- Additional resident SVC routines (RSVC=)

REDEFINING THE SYSTEM QUEUE AREA

The system queue area operates under a protection key of zero. It contains system control blocks which might be destroyed by problem programs if placed in problem program partitions. These control blocks include command scheduling control blocks (CSCBs) and all control blocks associated with ENQ/DEQ. If the communications task encounters a threshold condition, write-to-operator (WTO) buffers are also constructed in the system queue area.

To respecify the system queue area, the operator enters the size as the number of total bytes required. NIP readjusts the area's free queue element (FQE) and resets the area's upper boundary accordingly. It then rounds the size to a double word boundary. If BLDL, RAM, and RSVC options were not specified during system generation or are not selected during IPL, NIP rounds the size to a 2K boundary in systems with storage protection.

Note: MFT and MVT construct the system queue area in the same place in main storage, but the SQA has a different function in each system.

LOCATING THE BLDL LIST AND RESIDENT MODULES

After the system queue area is established, the optional BLDL list is constructed, and optional RAM and RSVC modules are loaded into the area adjacent to the system queue area. If the BLDL option was chosen at system generation, a list of the SYS1.LINKLIB modules specified by the user is constructed. If the RAM and RSVC options were chosen at system generation, the optional linkage library and SVC library modules specified by the user are loaded.

The communications vector table (CVT) field CVTNUCB, containing the lowest storage address in the dynamic area, is adjusted so that the BLDL list and RAM and RSVC modules are included in the fixed area.

Note: In MVT, the BLDL list and RAM and RSVC modules are in the link pack area; there is no link pack area in MFT.

MAIN STORAGE PREPARATION

When NIP completes its functions it constructs a request block (RB) and an XCTL macro instruction (specifying master scheduler initialization routine IEFSD569) at the low address of the temporary master scheduler area defined previously (see Defining Control Program Areas). NIP places the address of this RB in master scheduler task TCB field TCBRBP. (The original contents of TCBRBP are saved and passed to IEFSD569 in a parameter list along with the original master scheduler task boundary box contents.) NIP sets master scheduler task TCB field TCBFLGS to make the master scheduler task dispatchable, and then branches to the dispatcher.

The dispatcher gives control to the master scheduler task causing execution of the XCTL instruction which NIP placed in the temporary master scheduler area. The master scheduler initialization routine is brought into the temporary master scheduler area and begins executing. Figure 4, excluding the medium shaded area, illustrates main storage at completion of NIP before branching to the dispatcher. Figure 4, excluding the light shaded area, illustrates main storage when the master scheduler initialization routine receives control from the dispatcher.

For a description of the master scheduler initialization routine see "Master Scheduler Task" in the Job Management section. Figure 5 illustrates main storage (four partition example) at completion of master scheduler initialization. When the initialization routine completes processing, it branches to the dispatcher.

Initializing the Partitions

During master scheduler initialization the operator must accept automatic START commands or enter START commands manually. When a START command is processed, the partition number specified in the command is determined, and a CSCB is built. The CSCB (see Appendix A) is used for communication between the command scheduling routines (SVC 34) and the command execution routines. The address of the CSCB is placed in the partition information block (PIB) of the specified partition, and the partition is posted. The PIB for each partition contains information used by command processing and scheduler routines. (See Appendix A for a description of the PIB, and "Initiator/Terminator" in Job Management for a discussion of its use.)

After the initialization routine completes processing, the dispatcher gives control to the master scheduler router routine. When this routine completes processing, it returns to the dispatcher which begins searching the TCB queue. The highest priority task posted through START command processing receives control. The XCTL macro instruction addressed by the partition's RB is executed and the Job Select module (IEFSD510) or Small Partition module (IEFSD599) is brought into the partition. When an interruption occurs and the partition can no longer retain control, the dispatcher gives control to the next posted partition. This process continues, enabling all posted partitions to receive control and to execute the XCTL instruction placed in them by the initialization routine.

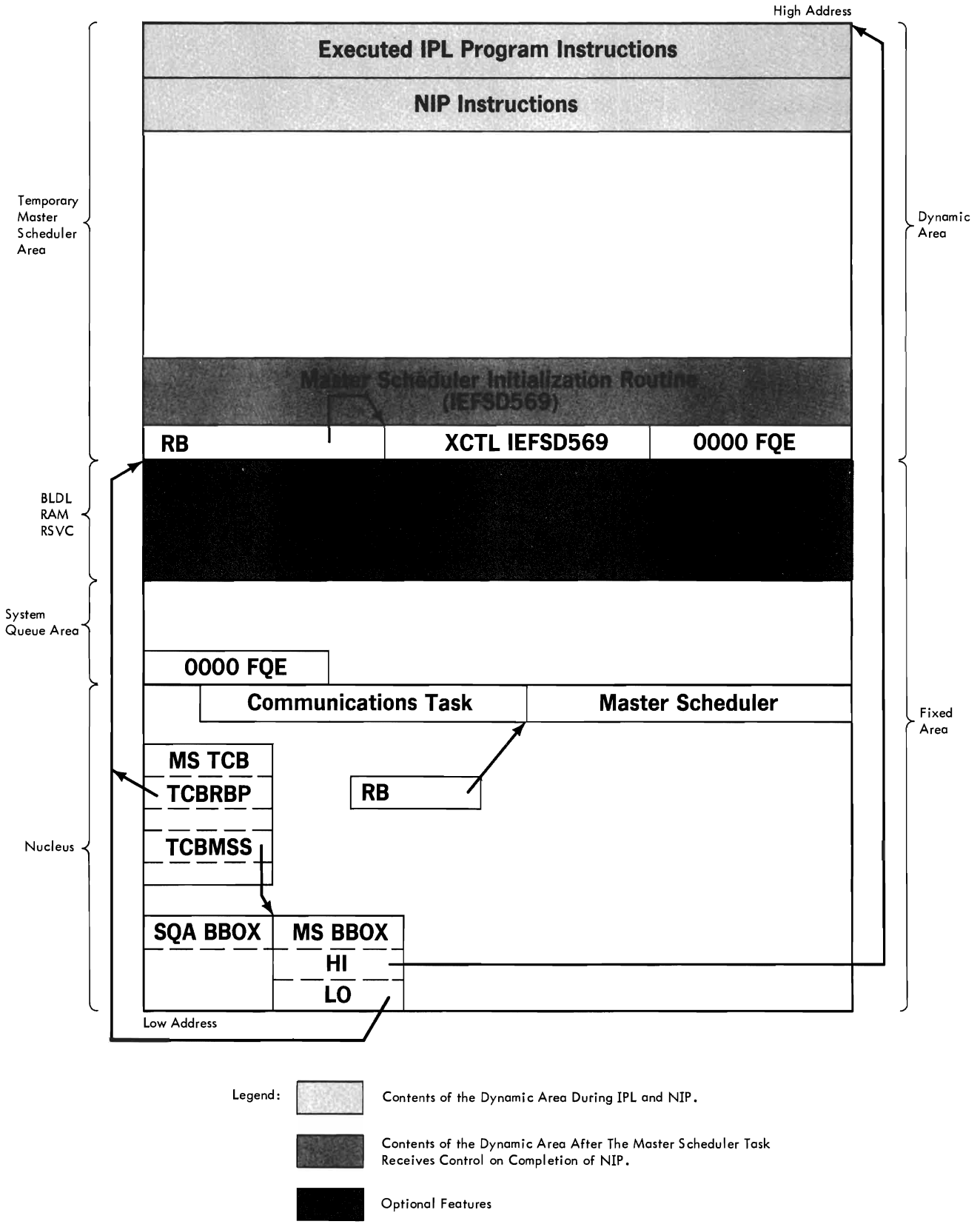


Figure 4. Main Storage During Execution of NIP

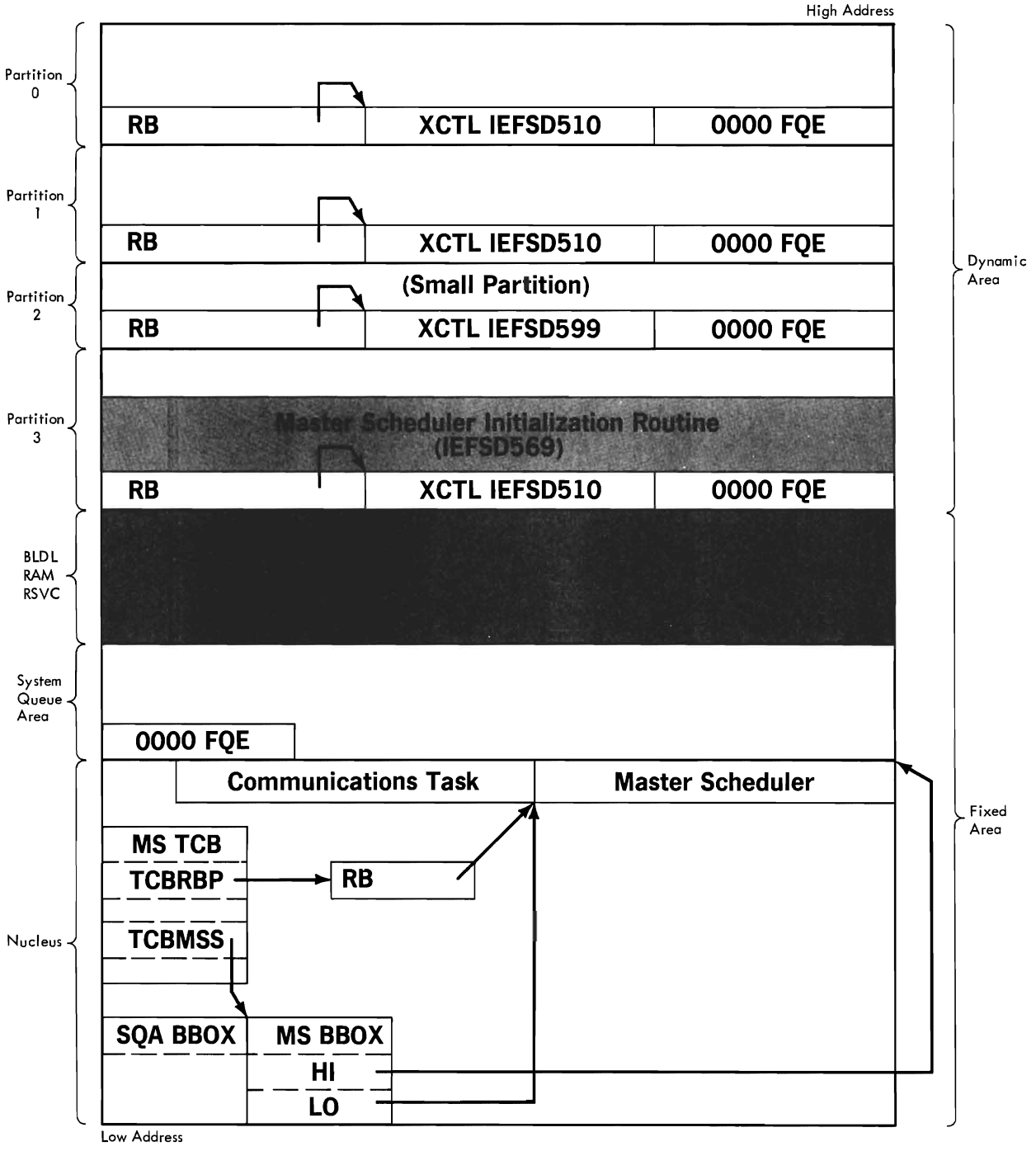


Figure 5. Main Storage at Termination of Master Scheduler Initialization

The MFT Supervisor manages the operation of the control program and processing programs. Job management selects jobs for execution, allocates devices and storage to the step to be executed, and gives control to the program that represents the step. After receiving control, a program is known as a task and becomes the responsibility of the Supervisor. As many as 15 job-step tasks may operate in the system concurrently with system tasks. Each task must be isolated so it does not interfere with any other task. To do this, each job-step task operates in its own partition in main storage. If the system has the optional storage protection feature, each partition is assigned a unique protection key (1-15). The resident portion of the control program, including some supervisor routines, occupies a fixed area of main storage and operates under a protection key of zero.

To maintain control of the computing system, the supervisor must perform many services. Routines within the supervisor are grouped into general categories depending upon the services which they perform. These categories are:

Interruption Supervision: All supervisor activity begins with an interruption. The five types of interruptions are: supervisor call, timer/external, input/output, program, and machine. When an interruption occurs, the interruption handling routine for the type of interruption that occurred gains control. The interruption handling routine then passes control to those parts of the control program that perform the services required as a result of the interruption. Many of the services which must be performed are included in other general categories of the supervisor.

Task Supervision: The supervisor maintains control information including the current status of program and interruption request blocks, task control blocks, and event control blocks.

Contents Supervision: The supervisor keeps records of the status and characteristics of all programs in each partition of main storage, initiates program fetch for the dynamic loading of programs, and maintains the active request block queue.

Main Storage Supervision: Within each partition, the supervisor allocates and releases main storage space for a task on request, and maintains a record of all free storage space within each partition.

Timer Supervision: The supervisor sets and maintains a clock, and honors requests for time intervals and exact time.

Overlay Supervision: The supervisor monitors the flow of control between segments of a program operating in an overlay structure established by the user through the linkage editor.

INTERRUPTION SUPERVISION

With the exception of the dispatcher which is described below, the interruption supervisor of MFT functions as described in IBM System/360 Operating System: Fixed-Task Supervisor, Program Logic Manual, Form Y28-6612.

When an interruption occurs and is serviced, the task which had been executing may relinquish control of the CPU. Control must always be given to the highest priority ready task. The transfer of control from one task to another is called task switching and is accomplished by the task dispatcher. When an interruption handling routine completes processing an interruption, it branches to the task dispatcher rather than returning control to the interrupted program. Type 1 EXIT is the only interruption handling routine which may return control directly to the interrupted program. Figure 6 illustrates how the task dispatcher receives control after an interruption has been serviced.

THE DISPATCHER (MACRO IEAAPS)

The dispatcher gives control to the highest priority task ready to execute. It uses information located by communication vector table (CVT) fields CVTHEAD and CVTTCBP, and if the time-slicing feature is in the system, field CVTTSCE.

Field CVTHEAD addresses a queue of task control blocks (TCBs). This TCB queue is arranged in dispatching priority order beginning with the highest priority task. The highest priority TCB, the communication task TCB, is followed by the master scheduler task TCB, and one TCB for each of the partitions generated in the system (in ascending order by partition number). Figure 7 illustrates the TCB queue.

Any number of partitions (up to 52) may be specified during system generation. Partitions must be numbered consecutively beginning with zero. Note that in Figure 7

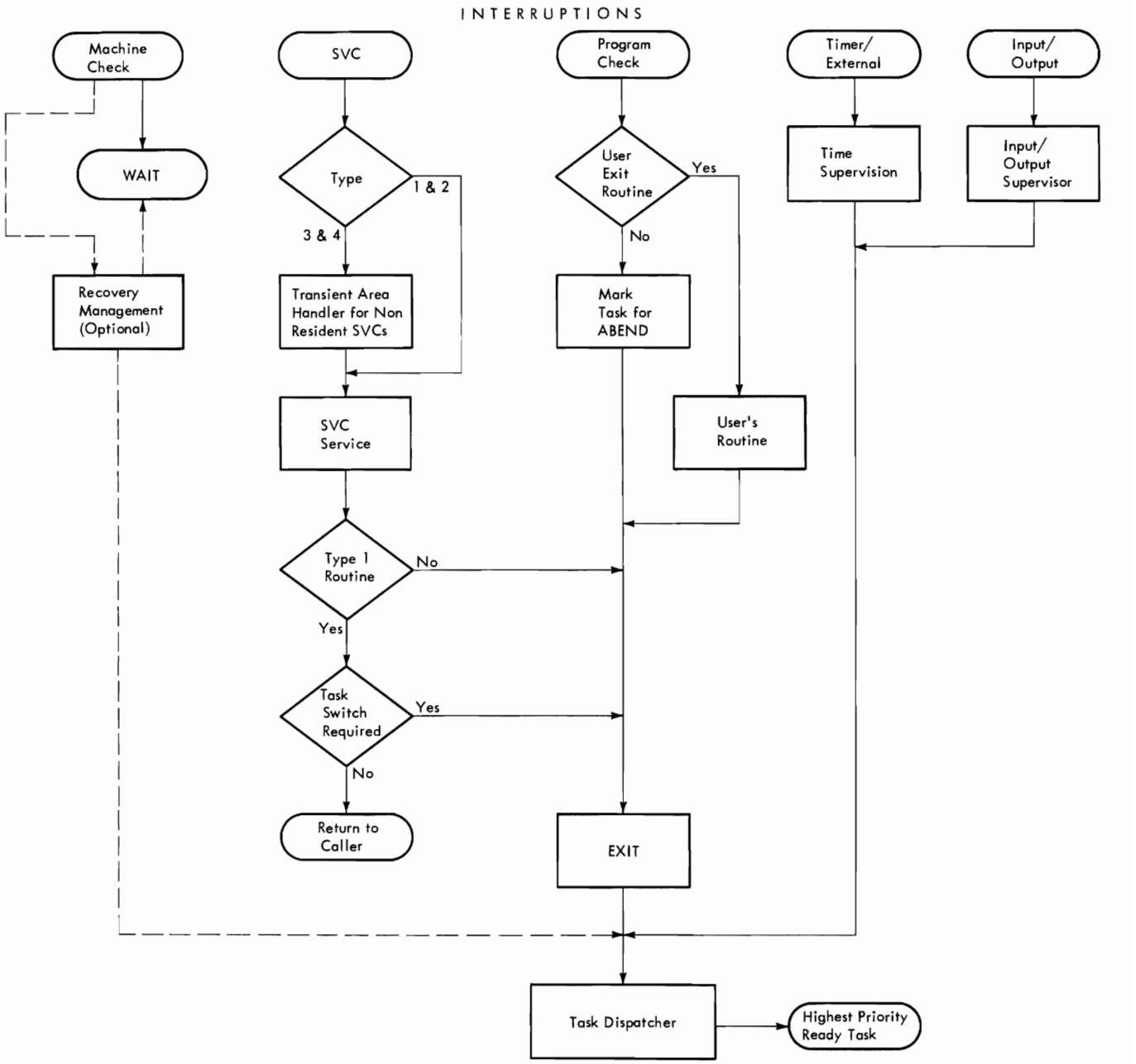


Figure 6. MFT Supervisor

there is a TCB for partition 1, but partition 1 is assigned no storage space. This illustrates a partition which was specified at system generation but which has been made inactive. If a partition is not specified during system generation, no TCB is constructed. If, for example, only 3 partitions (0 through 2) are specified at system generation, then only three TCBs are constructed and partitions 3 through 51 do not exist.

All of the TCBs in the system are chained together through TCB field TCBTCB. In each TCB, this field contains the address of the next TCB on the queue. The TCBTCB field of the last TCB on the queue contains zero.

CVT field CVTTCBP addresses two full words called NEW and OLD. The first word (NEW) contains either zero or the TCB address for the task to be given control. The second word (OLD) contains the TCB address for the task currently in control. NEW can be set by any of the supervisory routines associated with task switching (WAIT, POST, ENQ/DEQ, Manual Purge). When

a supervisory routine determines that the task currently in control can no longer retain control, it sets NEW to zero. When a supervisory routine determines the new task to be given control, it inserts the TCB address for that task in NEW.

CVT field CVTTSCE contains the address of the time-slice control element (TSCE). This field is used by the dispatcher in determining the next time-slice task to receive control, providing time-slicing was specified as a system generation option. The format of a TSCE is explained later in this section.

When the interval timer is in use and a user accounting routine is supplied, the dispatcher accumulates the total amount of time used to execute a job step. Each time a new job step is dispatched, the dispatcher stores the time from the hardware timer in the PTIMER field of IEATPC (pseudo clock area). When control is returned to the dispatcher, it calculates the elapsed time by subtracting the stored value from the current value of the hardware timer. The dispatcher adds the result to the TCBTCT field in the task's TCB. Time is not calculated for the job step if it is dispatched in a wait state.

Dispatching a Task

When the dispatcher receives control, it first schedules any requests for system asynchronous exit routines. Then it determines if NEW equals OLD (see Chart 02). If so, no task switch is indicated. If necessary, the dispatcher enqueues timer elements for the task. It then returns to the task currently in control.

If NEW does not equal OLD, a task switch is indicated. If job/step CPU timing is included in the system, the dispatcher calculates the job step time for OLD, and increments the job time accumulator in the TCB. If necessary, the dispatcher dequeues timer elements associated with the task currently in control. Then it determines if NEW equals zero.

If NEW does not equal zero, it contains the TCB address for the task to be given control. The dispatcher sets OLD equal to NEW, and enqueues timer elements if necessary. Additionally, if job/step CPU timing is included in the system, the dispatcher stores the interval timer value in the pseudo timer field of IEATPC. Control then passes to the new task.

If NEW equals zero, the dispatcher must examine the TCB queue to determine which task should be given control. This examination begins with the TCB addressed by OLD. (For a task of higher priority than OLD to receive control, the address of its TCB must be inserted in NEW by a supervisory routine.)

When examining a TCB to determine if its associated task should be given control, the dispatcher first determines if the request block (RB) of the program executing under the TCB is waiting. This is done by examining field XRBWT in the RB addressed by TCB field TCBRBP. If the RB is not waiting, the dispatcher examines TCB field TCBFLGS to determine if the task is dispatchable. If so, the dispatcher sets NEW and OLD to the address of the TCB and enqueues timer elements (if necessary). Additionally, if job/step CPU timing is included in the system, the dispatcher stores the interval timer value in the pseudo timer field of IEATPC. Control then passes to the new task.

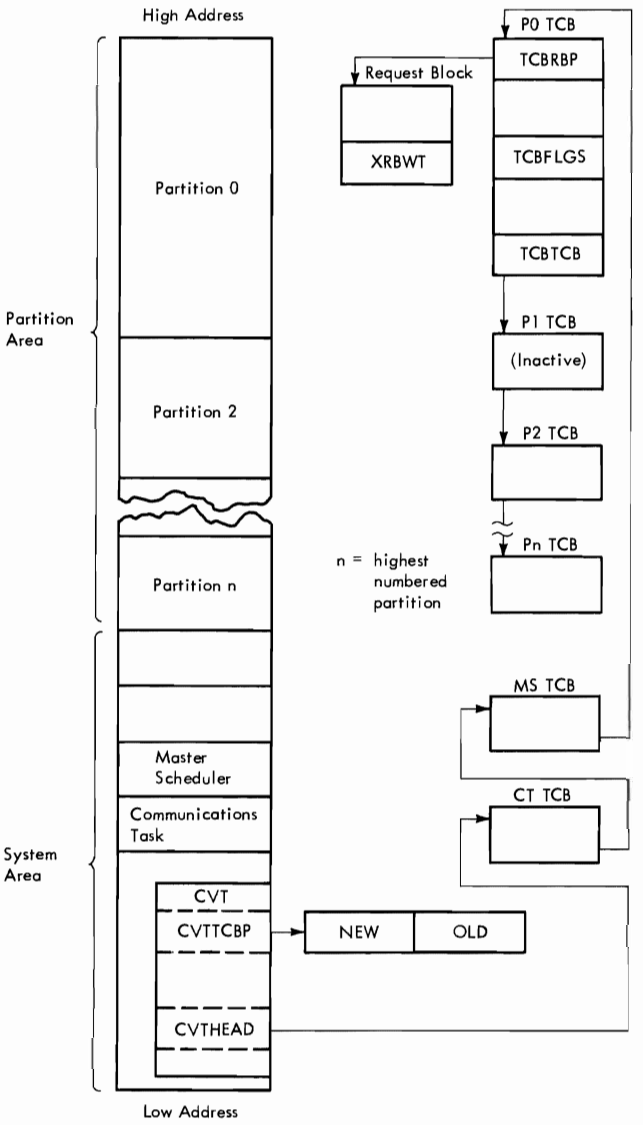


Figure 7. TCB Queue

In one case, the dispatcher does not pass control directly to the new task. If TCB field TCBRBP for the task to be given control addresses an SVRB for a transient SVC routine, a check is made to determine the contents of the double word XCNTCC (in IEAATA00) which contains the name of the routine presently in the SVC transient area. If the routine names in XCNTCC and the SVRB are identical, the dispatcher passes control to the new task. If they are not identical, the Transient SVC Refresh routine (IEAARF00) brings the required routine into the SVC transient area and then returns to the dispatcher. Since NEW and OLD have already been set equal, the dispatcher need only enqueue timer elements if necessary and pass control to the new task.

If the RB for a task is waiting or the task is nondispatchable, the task is not ready to receive control. The dispatcher examines TCB field TCBTCB to obtain the address of the next TCB on the queue. The

dispatcher then examines this TCB to identify whether it is ready to receive control. This process continues until a ready task is found or until the end of the queue is reached (indicated by a zero in TCBTCB).

If no task is able to receive control, the dispatcher sets the resume PSW wait bit of the TCB addressed by OLD. This PSW is then loaded, placing the CPU in a wait condition. The resume PSW is located in field XRBPSW of the RB addressed by TCB field TCBRBP.

Figures 8 and 9 illustrate how control is switched assuming a three partition system in which P1 is inactive (see Figure 7). All tasks are dispatchable except task P1. Initially, only the communications task and master scheduler task are waiting. Because task P0 is the highest priority task which is dispatchable and not waiting, it is given control. Task P0 has already enqueued and received exclusive control of a resource which task P2 will later enqueue (see Figure 9).

Dispatching the Communications Task and Master Scheduler Task

Figure 8 illustrates how control passes to the communications task and master scheduler task through the dispatcher. In the example illustrated, the communications task receives control in order to read a DEFINE command from the operator console.

Initially, the task in P0 has received control from the dispatcher and is executing. The operator presses the REQUEST key to indicate that he wishes to enter a command from the console. An I/O interruption is generated and control passes to the I/O supervisor which identifies the interruption as an attention signal. The I/O supervisor then passes control to the console interruption routine which issues a POST macro instruction. The POST routine posts the attention ECB and sets the communications task RB to a non-wait condition. Because the communications task is of higher priority than the task in partition 0, the POST routine places the address of the communications task TCB in location NEW. Control then passes to the dispatcher.

The dispatcher gives control to the communications task which issues SVC 72 to read the console and then issues SVC 34 to process the command. SVC 34 processes some commands completely but must pass control to the master scheduler resident command processor routine to complete processing the DEFINE command. (See "Command Processing" in the Job Management section for a complete description of SVC 34 and the master scheduler task.) SVC 34 issues a POST macro instruction to post the master scheduler task. The POST routine sets the master scheduler RB to a non-wait condition and gives control to the dispatcher. Because the master scheduler task is of lower priority than the communications task, locations NEW and OLD remain unchanged and the dispatcher returns control to the communications task.

The communications task issues a WAIT macro instruction and waits on an ECB. The WAIT routine sets the communications task RB in a wait state and sets location NEW to zero. The dispatcher then receives control and searches the TCB queue. Since the master scheduler task is the next ready task on the TCB queue, the address of the master scheduler TCB is placed in locations NEW and OLD, and the dispatcher passes control to the master scheduler.

The master scheduler completes processing the DEFINE command and then issues WAIT. The WAIT routine sets location NEW to zero and passes control to the dispatcher which searches the TCB queue until it finds a task ready to receive control. In Figure 8, control returns to the task which was executing before the operator entered the DEFINE command.

Dispatching Tasks by Partition Priority

Figure 9 illustrates task switching among tasks executing in partitions.

A. The task in partition P0 (task P0) is the highest-priority ready task and is given control by the dispatcher. When task P0 issues a WAIT on an ECB, an interruption occurs and control passes to the WAIT routine.

B. The WAIT routine places the RB for partition 0 in a wait condition and sets location NEW to zero. It then passes control to the dispatcher which searches the TCB queue beginning with the TCB for partition 0. Since task P0 is waiting and task P1 is non-dispatchable, the dispatcher passes control to task P2, the highest priority task ready to execute. When task P2 attempts to enqueue a resource through use of the ENQ macro instruction, an interruption occurs and control passes to the ENQ routine.

C. The resource is unavailable because task P0 has already enqueued it. Therefore, task P2 cannot continue executing. The enqueue routine places zero in location NEW and then passes control to the dispatcher which searches the TCB queue. Since task P2 is the last task on the queue, the dispatcher sets the wait bit in the

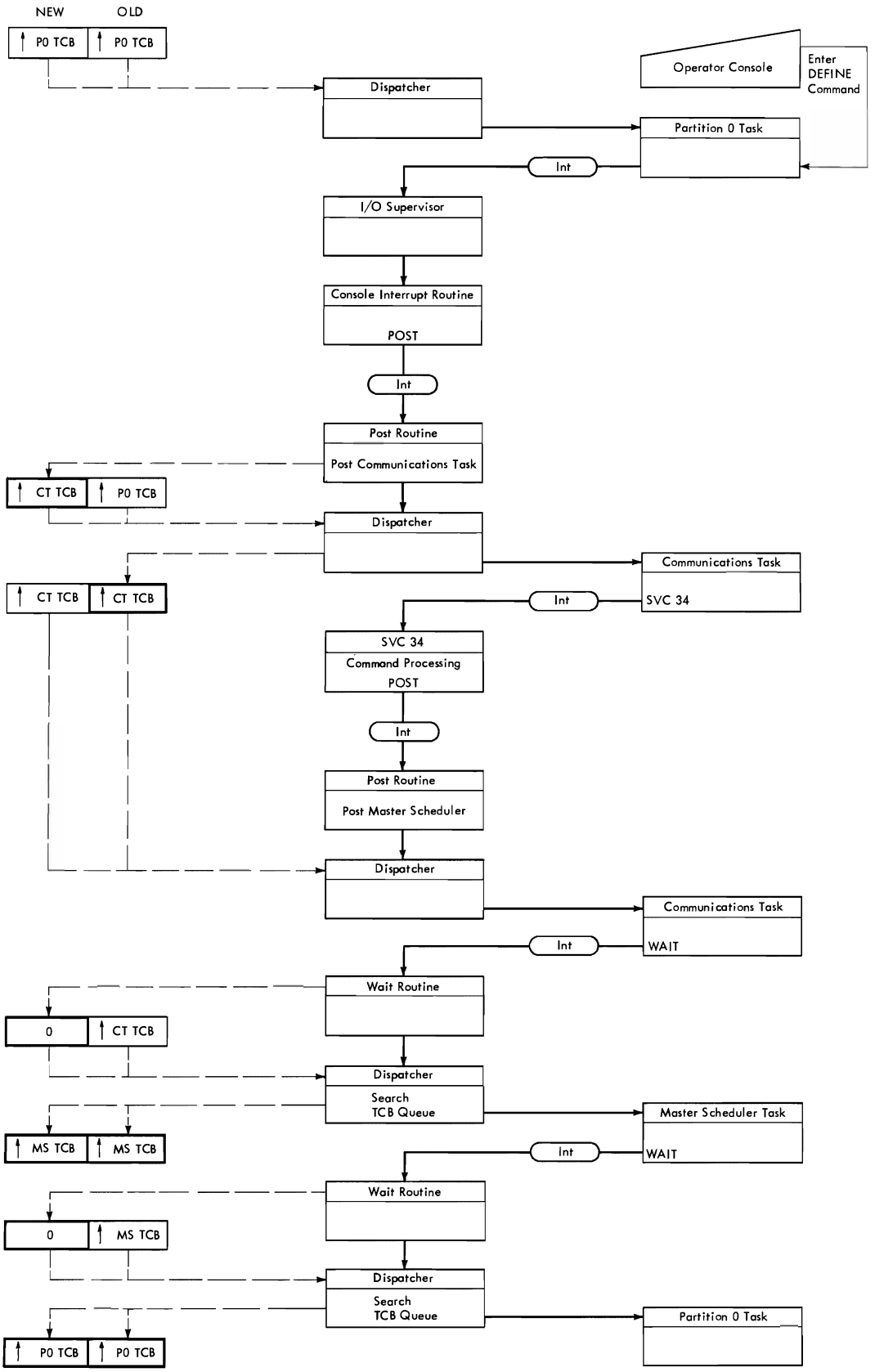


Figure 8. Dispatching Communications and Master Scheduler Tasks

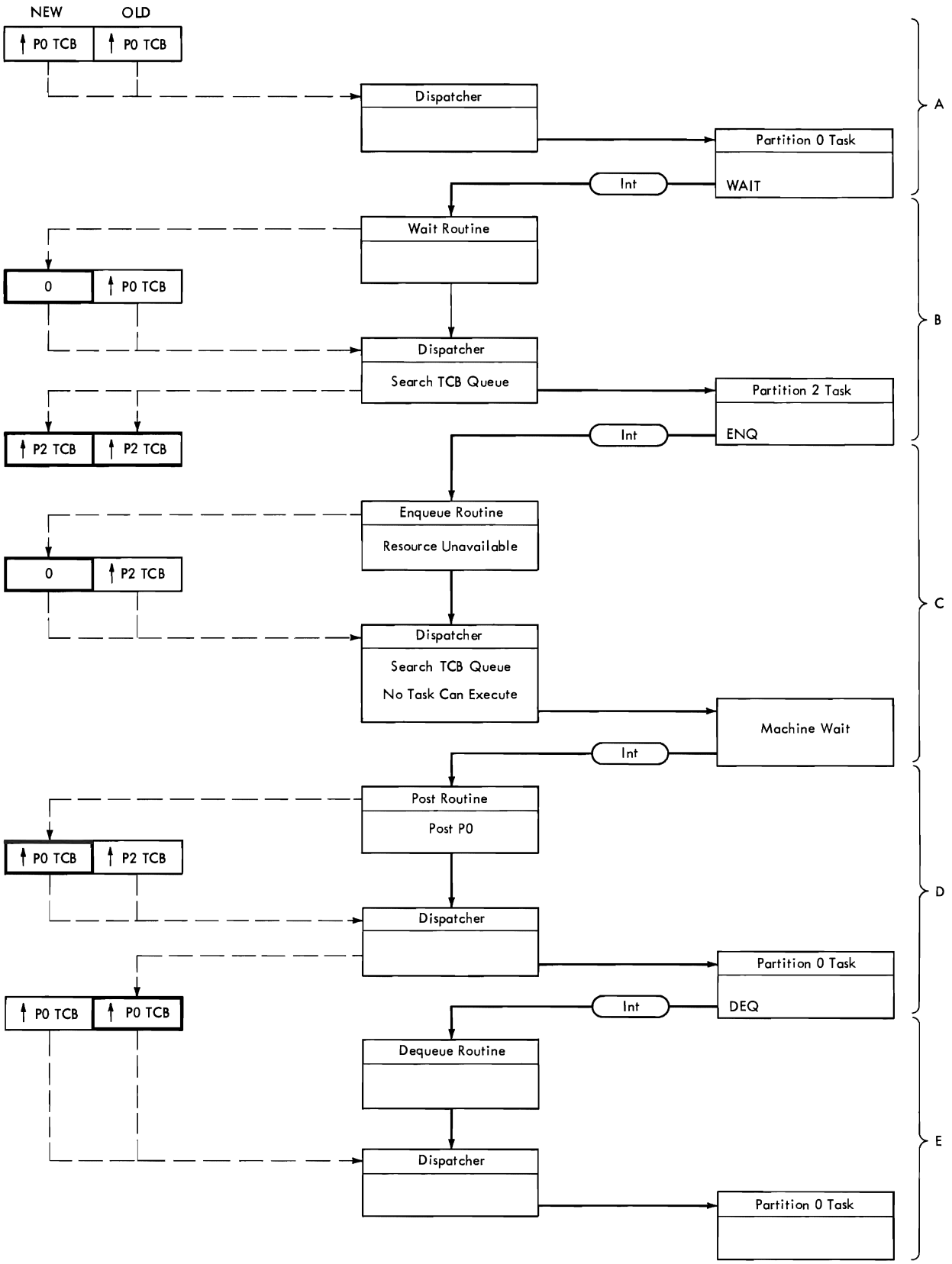


Figure 9. Task Switching

resume PSW of task P2. The dispatcher passes control to task P2, placing the CPU in a machine wait condition.

D. While the CPU is waiting, an interruption occurs signifying the completion of the event for which task P0 was waiting. The POST routine receives control and posts the ECB for task P0 which is now able to resume control. The POST routine places the TCB address for task P0 in location NEW and gives control to the dispatcher. The dispatcher sets OLD equal to NEW and gives control to task P0. Task P0 executes and when finished using the resource it has enqueued, it issues a DEQ macro instruction.

E. An interruption occurs and the DEQ routine receives control. The queue element for task P0 is removed from the resource queue. The next element on the resource queue is for task P2. The resource is assigned to task P2 and its RB is placed in a non-wait condition. The DEQ routine then compares the priority of the task which has been in control with the priority of the task which is now ready. Because task P0 has a higher priority than task P2, location NEW remains unchanged. The DEQ routine passes control to the dispatcher which returns control to task P0.

Dispatching a Task (with Time Slicing)

If time slicing was selected as a system generation option, the user can select a number of contiguous partitions to be a time-slice group. The tasks executing in time-sliced partitions have equal priority. Each ready task in the time-slice group executes for a selected amount of time, the time-slice length, and then loses control to the next ready task in the time-slice group. The time-slice group is supervised through use of a time-slice control element (TSCE) shown below.

FIRST - Address of the fist time-slice TCB on the TCB queue	4
LAST - Address of the last time-slice TCB on the TCB queue	4
NEXT - Address of the next time-slice TCB to be dispatched	4
LENGTH - Time-slice length (in milliseconds)	4

When time-slicing is selected, the dispatcher performs functions in addition to those explained in the preceding paragraphs. The following text describes the additional dispatcher functions, and parallels the flow of data shown in Chart 03.

NEW EQUALS OLD: The dispatcher first determines if NEW equals OLD. If it does, the dispatcher further determines if the task represented by OLD is a time-slice task.

OLD a Time-Slice Task: If OLD is a time-slice task, the dispatcher determines if the time-slice interval has expired; i.e., if the time-slice queue element (TQE) has been removed from the timer queue.

If the interval has expired, the next ready time-slice task must be dispatched. The dispatcher searches the time-slice group beginning with the TCB addressed by TSCE NEXT (see preceding explanation of TSCE fields). When the TCB addressed by TSCE LAST is reached, the dispatcher checks the TCB addressed by TSCE FIRST, until a ready task is found or until all time-slice TCBs have been checked.

When a ready task is found, TSCE NEXT is updated, the time-slice TQE is enqueued, and the ready task is dispatched. If no time-slice tasks are ready, the dispatcher searches the TCB queue for the highest-priority ready task.

If the interval has not expired, i.e., the time-slice TQE has not been dequeued, control is returned to the interrupted task.

OLD Not a Time-Slice Task: If OLD is not a time-slice task, control is returned to the interrupted task.

NEW NOT EQUAL TO OLD: If NEW does not equal OLD, the dispatcher determines if OLD is a time-slice task.

OLD Time-Slice Task--NEW Equal Zero: If OLD is a time-slice task and NEW equals zero, the time-slice TQE is dequeued for the current task. The dispatcher then searches (using the TSCE) for the next ready TCB in the time-slice group. If no time-slice TCBs are ready, the dispatcher searches the TCB queue for the highest-priority ready task.

OLD Time-Slice Task--NEW Not Equal to Zero: If OLD is a time-slice task and NEW does not equal zero, the dispatcher determines if NEW is a time-slice task.

If NEW is a time-slice task, the task represented by OLD, if ready, is redispatched. (The time-slice TQE remains on

the queue.) If the task represented by OLD is not ready, the time-slice TQE is dequeued, and the dispatcher searches (using the TSCE) for the next ready time-slice task. If no time-slice tasks are ready, the dispatcher searches the TCB queue for the highest-priority ready task.

If NEW is not a time-slice task, the time-slice TQE is dequeued and the NEW task is dispatched.

OLD Not a Time-Slice Task: If OLD is not a time-slice task, the dispatcher finds the next highest-priority ready task. It does this by either obtaining the TCB address from NEW or, if NEW is zero, by scanning the TCB queue. If the highest-priority ready task is not a time-slice task, it is dispatched. If the highest-priority ready task is a time-slice task, the dispatcher finds (using the TSCE) the next ready task in the time-slice group. The time-slice TQE is enqueued, and the task is dispatched.

ABEND SERVICE ROUTINE

ABEND is a type 4 SVC routine that is used for both normal and abnormal task termination. ABEND terminates the task under which it is running, resets the partition, and passes control to the job management routines for continued processing.

ABEND can be entered directly from the problem program or system task via an ABEND macro instruction, or indirectly through the ABTERM service routine. (ABTERM schedules the execution of ABEND for system routines that detect an error but cannot issue an ABEND macro instruction.) The SVC SLIH (second level interruption handler) fetches the first load module of ABEND and passes control to it. Control is passes from one ABEND load module to the next via an XCTL instruction (SVC 7). The flow of control between modules for normal and abnormal termination is shown in Chart 04.

The ABEND functions provided for MFT are similar to those provided for PCP. Modules IEAGTM0A, IEAGTM00, IEAGTM06, and IEAGTM05, exist only in MFT and are described below. The remaining modules are also used in PCP and are explained in IBM System/360 Operating System: Fixed Task Supervisor, Form Y28-6612. (For a brief description of all ABEND modules used in MFT, see "MODULE DESCRIPTIONS," IN Appendix B of this publication.)

ABEND STAE Test Routine (IEAGTM0A)

IEAGTM0A first sets a bit to prohibit asynchronous exits for this task and tests for normal end. If this is a normal end, the normal completion code is stored in the TCB, and control passes to ABEND module IEAGTM00. If the task is abnormally terminating, IEAGTM0A determines if STAE (specify task asynchronous exit) processing is indicated for this task (via a user-issued STAE macro instruction).

If a STAE was issued, i.e., TCB field TCBNSTAE does not equal zero, IEAGTM0A checks for a valid STAE and performs as follows:

- If the ABEND was issued by the Purge routine during STAE processing, i.e., the purge bit in TCB field TCBNSTAE equals one, the resume PSW of the Purge RB routine is set to the address of an EXIT instruction (SVC 3). IEAGTM0A then issues an EXIT instruction.

  Note: If ABEND was not entered from the Purge routine, i.e., the purge bit in TCB field TCBNSTAE equals zero, IEAGTM0A stores the abnormal completion code in the TCB.

- If the Task is being abnormally terminated because of a timer expiration or an operator cancel, STAE processing is bypassed and IEAGTM0A exits to ABEND module IEAGTM00. (Since task timing and cancellations from the console are directly controlled by the user, the ABEND was intentional and should not be handled by the STAE routines.)

- If STAE processing is already in progress, regular ABEND processing continues, since the STAE routine can process only once per STAE issued. IEAGTM0A thus exits to ABEND module IEAGTM00.

- If this is a valid request for STAE processing, i.e., none of the above conditions are true, IEAGTM0A tests TCB field TCBPIE for zero. If it is not zero, IEAGTM0A frees the PIE and zeros TCBPIE. Thus, subsequent program checks will not be handled by a user routine which may not be designed for such a program check. IEAGTM0A then exits to STAE module IEAATM0B.

If a STAE has not been issued, or if all STAEs have been processed, i.e., TCB field TCBNSTAE equals zero, IEAGTM0A determines if this is a graphics or an ABEND recursion and if this is a graphics job with a Graphics Abend Exit routine.

- If this is a recursion, or if the task abnormally terminating is not a graphics job, IEAGTM0A exits to ABEND module IEAGTM00.

- If this is a graphics job, IEAGTM0A passes control to the Graphics Abend routine. At the completion of this routine, control is returned either to the caller via an SVC 3 (if the task is resumable, i.e., the ABEND was issued by a user program or caused by a program check in a user routine, and if he wishes to resume processing) or to IEAGTM0A, which exits to ABEND module EAGTM00.

ABEND Initialization Routine (IEAGTM00)

This module first determines if it was entered from STAE. If so, it branches to the section of code that accomplishes the WTOR purge function. IEAGTM00 cancels the task timer element so that a timer interruption will not occur during ABEND processing. It then dequeues all interruption queue elements (IQEs) belonging to the task, since these cannot be scheduled during ABEND processing.

If a system or problem program 'must complete' bit is set in TCB field TCBFLGS, or if the task is a system task, IEAGTM00 branches to the System Quiesce routine, IEAGTWST (see "System Quiesce Routine"), which places the partition in a wait state and prints a message to the operator.

Note: Processing in the other partitions can be restarted and continue until all jobs already enqueued have been completed. The system will then be in a wait state and corrective action must be taken.) If the above conditions do not exist, IEAGTM00 purges the WQEs (WTO Queue Elements) and the RPQEs (Reply Queue Elements) for this task.

IEAGTM00 tests to see if it was entered from STAE and if this is a normal end to determine the next load module.

ABEND Input/Output Purge Routine (IEAGTM06)

IEAGTM06 purges I/O requests and I/O operations via a macro instruction version of the SVC Purge Routine, which is assembled within this module. (See "SVC PURGE ROUTINE" in Input/Output Supervisor, Form Y28-6616.) This prevents errors that can cause recursion to the ABEND routine. (Since ABEND frees main storage, an I/O operation that is not halted can cause information to be read into, or an ECB to be posted in main storage that may have been relocated, thus destroying data or programs.) RQEs (request queue elements) removed from the request queue are returned to a list of available RQEs for reuse by the I/O supervisor.

IEAGTM06 also dequeues, from the SIRB, IQEs (interruption queue elements) representing requests for the use of I/O error handling routines. IEAGTM06 passes control to ABEND module IEAATM01.

ABEND Termination Routine (IEAGTM05)

ABEND termination routine IEAGTM05 is the final ABEND module for both normal and abnormal termination. For normal termination it is entered from IEAGTM06. On abnormal termination it may have been entered from any of the previous modules of ABEND except initialization routine IEAGTM00.

If a dump message is required, i.e., if ABDUMP has been initiated but has failed to complete, IEAGTM05 causes message IEA002I, "ABEND/ABDUMP ERROR, NO ABEND OUTPUT" to be printed. IEAGTM05 issues a CLOSE macro instruction for any open data sets. The timer queue and the main storage supervisor queue are purged, and fields in the TCB are reset so that a new task may be initiated. If an indicative dump is provided by IEAATM03, it is moved to the upper boundary of the partition.

IEAGTM05 does not directly transfer control to a job management routine. In the first 72 bytes of the problem program partition, IEAGTM05 establishes a dummy PRB, an XCTL parameter list, and a set of instructions including an XCTL to a step deletion routine. The XRBLNK field of the dummy PRB contains a pointer to the TCB. The dummy PRB therefore becomes the only RB queued for this task.

The dummy PRB is then placed at the beginning of the RB queue. This ensures that the XCTL instruction will be the next operation executed for this task after IEAGTM05 has completed. For scheduler-size partitions, step deletion routine IEFSD515 gains control, at entry point GO. For small partitions, control is passed to entry point SMALLGO in small partition routine IEFSD599.

SYSTEM QUIESCE ROUTINE (IEAGTWST)

The system quiesce routine, which is part of the nucleus, is branched to during abnormal termination processing when the task scheduled for abnormal termination processing is in "must complete" status or

is a system task, or when entry to abnormal termination processing was caused by an error that occurred during ABEND processing. The system quiesce routine, whose address is located in the CVTXWTO field of the CVT, performs the same functions as in MVT: the failing task is placed in wait state, and a message indicating that a CPU wait state has been averted and is issued to the operator instructing him to allow the system to quiesce (to schedule no further jobs). Control is then returned to the supervisor enabling the system for interruptions and permitting the other partitions of the system to continue processing. See Termination Procedures in MVT Supervisor for a more detailed description of the system quiesce routine.

STAE SERVICE ROUTINE

The STAE service routine is a type 3 SVC routine which prepares the task to intercept scheduled abnormal termination (ABEND) processing. When the STAE macro instruction (resulting in an SVC 60) is issued, the STAE service routine is invoked. The STAE service routine creates a 16-byte STAE control block (SCB), which contains the addresses of a user-written STAE exit routine and parameter list. When the task becomes scheduled for abnormal termination, the ABEND/STAE interface routine (ASIR) is given control by the ABEND routine. ASIR returns control to the user at the STAE exit routine address. After the STAE exit routine has been executed, control is returned to ASIR. ABEND processing continues for the task as previously scheduled unless the STAE exit routine has requested that a STAE retry routine be scheduled. If a STAE retry routine is provided by the user, ASIR reestablishes the task scheduled for ABEND processing and exits, giving control to the dispatcher so that the STAE retry routine is executed next. See IBM System/360 Operating System: System Programmer's Guide, Form C28-6550, for further explanation of the STAE macro instruction.

The five modules which perform the functions of the STAE macro instruction are the STAE service routine (IGC00060) and the four ABEND/STAE interface modules (IGC0B01C, IGC0C01C, IGC0D01C, and IGC-0E01C). These modules perform the same functions as in MVT, (see IBM System/360 Operating System: MVT Supervisor, Form Y28-6659) with the exception both IGC0C01C and IGC0E01C pass control via the XCTL macro instruction to the ABEND module IEAG-TM0A to purge the WTOR queue before giving control the next ASIR module (IGC0D01C).

TASK SUPERVISION

The task supervisor maintains the status of tasks within the system. Task supervision service routines:

- Maintain task control blocks.
- Enter tasks into the wait state.
- Post completed events in the event control block.
- Maintain control levels indicated by request blocks.

The routines which accomplish these functions are WAIT, POST, ENQ, and DEQ.

Each task within the operating system has an associated task control block (TCB). The TCB contains task-related information and pointers to additional control blocks containing task-related information. The control blocks used by MFT are the same as those used by PCP except for the addition of the partition information block (PIB) which is described in Appendix A. The last three bytes of the word at displacement 124 (decimal) of each partition TCB contain the address of the associated PIB. Figure 10 shows the major control blocks maintained by the supervisor and their relationship to the TCB.

Task supervision is described in IBM System/360 Operating System: Fixed-Task Supervisor, Program Logic Manual, Form Y28-6612. Additional information applicable to MFT is presented in the following paragraphs.

THE ATTACH ROUTINE (MACRO IEAAAT)

In MFT, the ATTACH and LINK macro instructions are handled identically. An RB is created for the requested program, the program is brought into the requesting task's partition, and its RB is chained to the RB queue for that partition. See IBM System/360 Operating System: Supervisor and Data Management Services, Form C28-6646 for further explanation of the ATTACH macro instruction with MFT.

THE WAIT ROUTINE (MACRO IEAAWT)

The WAIT routine is not changed from that described in Fixed-Task Supervisor, Program Logic Manual. However, the user should remember the effect the optional validity checking feature has on WAIT. If validity check is included in the system and the program issuing the WAIT macro

instruction is not in supervisor mode, the WAIT routine checks that:

1. The boundary alignment of the ECBs is correct.
2. The storage protection key of the ECBs is that of the issuing program.
3. The addresses specified do not exceed main storage boundaries of the machine.

Because of point 2, it is not possible for one partition to WAIT on an ECB within another partition.

THE POST ROUTINE (MACRO IEAAPT)

The POST routine, like the WAIT routine, is unchanged from that described in Fixed-Task Supervisor, Program Logic Manual. Validity checking applies to POST in the same way it applies to WAIT.

THE ENQ/DEQ ROUTINE (IEAGENQ1)

The ENQ/DEQ routine provides a means of controlling serially reusable resources. This is done by assigning unique names consisting of a Qname and an Rname to each serially reusable resource. The ENQ/DEQ routine controls access to resources by building resource queues consisting of a queue control block (QCB) for each Qname and Rname specified in an ENQ macro instruction and a queue element (QEL) to represent each actual request. ENQ/DEQ is fully described in IBM System/360 Operating System: MVT Supervisor, Program Logic Manual, Form Y28-6659. ENQ/DEQ for MFT is identical to MVT except as described below.

In MFT, resource queues are located in the system queue area (subpool 255). Location IEAOQCB0 in the ENQ/DEQ routine contains the address of the first queue control block in the queue. There is only one TCB for each job step in MFT. Therefore, the "must complete" function of ENQ/DEQ applies only to the system, not to job steps. If "system must complete" is specified by a task, all other tasks in the

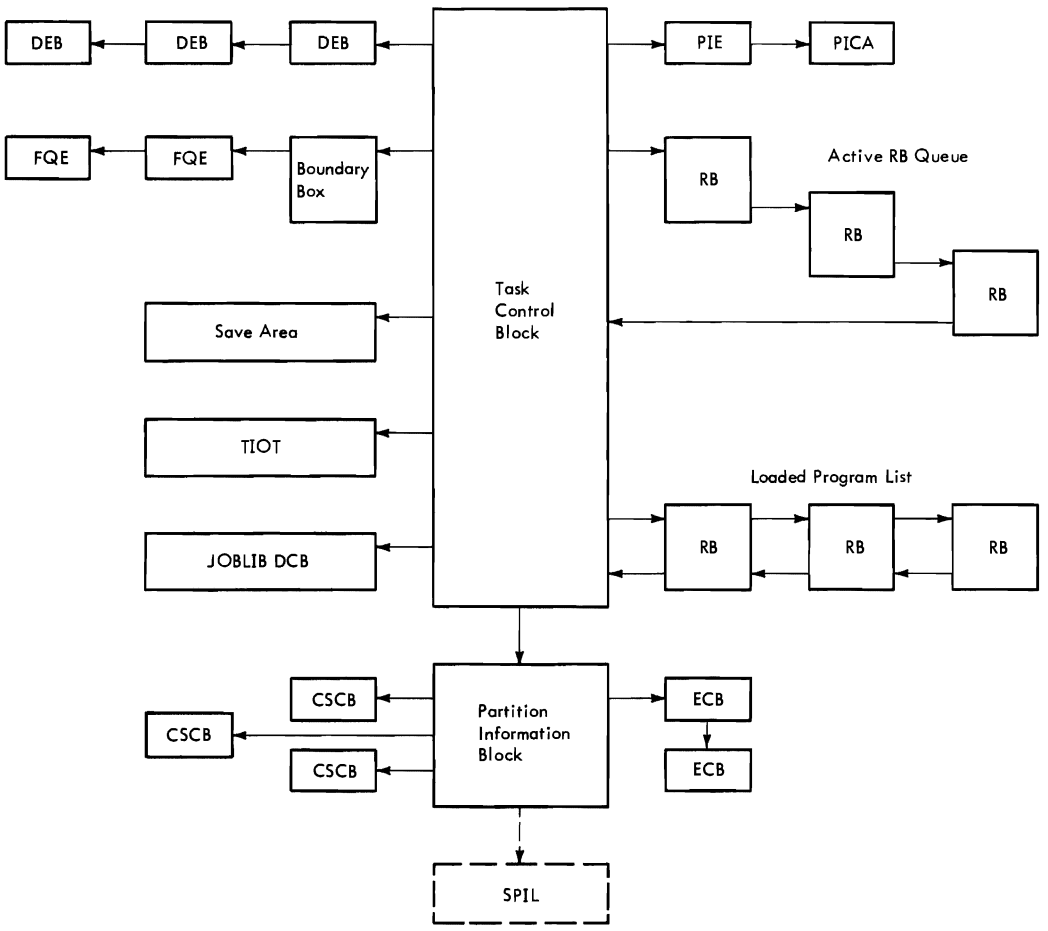


Figure 10. System Control Block Relationship

system are set non-dispatchable until the task which specified "system must complete" completes its processing.

CONTENTS SUPERVISION

Contents supervision routines determine the location of requested programs and fetch them into main storage if necessary. They also maintain records of all programs in main storage. Programs requested via LINK or XCTL macro instructions are scheduled for use by placing a request block (RB) for each program on the requesting task's active request block queue.

Programs requested via LOAD macro instructions are represented by RBs on the loaded program list.

There are six types of request blocks in MFT:

- Program Request Block (PRB) -- represents a nonsupervisory routine that must be executed in the performance of a task. PRBs are created by the contents supervision routines that perform the LINK or XCTL functions.
- Supervisor Request Block (SVRB) -- represents a supervisory routine. SVRBs are created by the SVC interruption handling routines.
- Interruption Request Block (IRB) -- controls a routine that must be executed in the event of an asynchronous interruption. IRBs are created in advance of an interruption by the CIRB routine at the user's request, but not placed on an RB queue until an interruption actually occurs.
- System Interruption Request Block (SIRB) -- used only for the system I/O error task. There is only one SIRB in the system.
- Loaded Program Request Block (LPRB) -- controls programs brought in by a LOAD macro instruction. LPRBs also control sections of programs that are specified by the IDENTIFY macro instruction. LPRBs are created by the contents supervision routines that perform the LOAD function.
- Loaded Request Block (LRB) -- a shortened form of LPRB and controls load modules that have the "load only" attribute. It is invalid to issue ATTACH, LINK, or XCTL macro instructions to these load modules. LRBs are created by the routines that perform the LOAD function.

Contents supervision alters the active RB queue and the loaded program list, and fetches programs into main storage in response to LINK, ATTACH, LOAD, and XCTL macro instructions. The routines which service these macro instructions are described below.

LINK SERVICE ROUTINE (MACRO IEAATC)

The LINK service routine determines if the RB of the requested routine is on the loaded program list. If it is and is inactive, LINK places the RB on the active RB queue. If the requested RB is not on the loaded program list (or if it is on the list, but is active), the LINK routine constructs an RB for the requested routine, places the RB on the active RB queue, and fetches the requested routine into main storage.

ATTACH SERVICE ROUTINE (MACRO IEAAAT)

The ATTACH macro instruction is handled as a LINK macro instruction. For a complete explanation, see "The ATTACH Macro Instruction" under the topic Task Supervision.

LOAD SERVICE ROUTINE (MACRO IEAATC)

The LOAD service routine first determines if the requested routine is a RAM module (if the resident access method (RAM) option was specified at system generation). If so, the entry point of the routine is passed to the requesting routine in register zero. If the routine is not a RAM module, LOAD searches the loaded program list for the RB of the requested routine. If it is found, the LOAD routine increments the RB use count by one and returns the entry point of the requested routine in register zero.

If the requested routine is not found on the loaded program list, the LOAD routine branches to the FINCH routine to load the requested routine into storage. On return from the FINCH routine, the LOAD routine initializes the requested routine's RB and places it on the loaded program list, sets the RBs use count to one and branches to the LINK routine to issue the SVC EXIT instruction.

XCTL SERVICE ROUTINE (MACRO IEAATC)

The XCTL service routine first determines if XCTL was issued by a type 3 or 4 SVC routine (if the resident SVC (RSVC) option was chosen at system generation.) The XCTL routine determines if the SVC

routine is an RSVC routine. If it is, the routine need not be brought into main storage. If the requested routine is not an RSVC, the XCTL routine branches to the FINCH routine to locate the routine on the SVC library and to bring it into the SVC transient area. The XCTL routine initializes the routine's RB and executes an SVC EXIT instruction.

If the XCTL macro instruction was not issued by a transient SVC routine, the XCTL routine dequeues the primary RB and each minor RB of the issuer from the active RB queue. The routine which issued the XCTL macro instruction and its RB are removed from storage unless the routine was brought in via a LOAD macro instruction. If the requested routine is on the loaded program list and is inactive, the XCTL routine branches to the LINK routine to place the RB on the active queue and to issue an SVC EXIT instruction.

If the XCTL routine determines that the scheduler has issued an XCTL macro instruction to branch to the problem program, the XCTL routine zeroes out the TCBTCT field of the TCB so that the optional Job/Step CPU Timing entry can be made.

If the RB of the requested routine was not found inactive on the loaded program list, the XCTL routine branches to the FINCH routine to bring in the routine. On return from the FINCH routine, the XCTL routine branches to the LINK routine to place the RB on the active queue and issue an SVC EXIT instruction.

Additional contents supervision services are provided by the IDENTIFY, DELETE, and SYNCH service routines. IDENTIFY and DELETE alter the loaded program list. SYNCH alters the active request block queue.

IDENTIFY SERVICE ROUTINE (IEAAID00)

The IDENTIFY service routine builds and initializes a minor request block to describe a routine specified in the parameters of the IDENTIFY macro instruction. The IDENTIFY routine chains this minor RB to the loaded program list and to the RB of the routine which contains the identified routine. The IDENTIFY routine returns to the issuer by issuing an SVC EXIT instruction.

DELETE SERVICE ROUTINE (IEAADL00, IEABDL00)

The DELETE service routine determines if the routine specified in the DELETE macro instruction is a RAM module. If it is, the DELETE routine exits immediately. If the routine is not a RAM module, the DELETE routine finds the routine's RB on the loaded program list and decrements the use count in the RB by one. If the use count reaches zero, the DELETE routine dequeues the routine from the loaded program list and issues a FREEMAIN macro instruction to release the storage occupied by the specified routine and its RB. On return from the FREEMAIN routine, the DELETE routine repeats the deleting process for each minor RB belonging to the specified routine. The DELETE routine returns by branching to the type 1 SVC exit.

SYNCH SERVICE ROUTINE (IEAASY00)

The SYNCH service routine uses GETMAIN to obtain 32 bytes of main storage from the lower end of the partition for the creation of a program request block (PRB). The PSW in the PRB is initialized by the SYNCH routine to address the location specified in register 15 by the issuer of the macro instruction. The SYNCH routine sets the PSW completely enabled in problem program mode, with the protection key recorded in the task control block. After the PRB is created and initialized, the SYNCH routine queues it on the active request block queue below the SVRB for SYNCH, and returns by issuing an SVC EXIT instruction.

Additional information describing PCP and MFT Contents Supervision can be found in Fixed-Task Supervisor, Program Logic Manual.

MAIN STORAGE SUPERVISION

In MFT, the main storage supervisor:

1. Allocates space via the GETMAIN SVC
2. Deallocates space via the FREEMAIN SVC
3. Allocates space in the system queue area
4. Checks validity of requests that are to be serviced
5. Maintains the pointers and control blocks necessary to supervise main storage

Each job is assigned to a partition in which it must operate. Each partition has an associated TCB which contains a pointer (TCBMSS field) to the main storage boundary box for that partition. The main storage supervisor, in response to GETMAIN or FREEMAIN macro instructions, obtains storage from either the problem program partition or the system queue area. Obtaining storage space from the system queue area is the basic difference in main storage super-

vision between MFT and PCP. In MFT, a system task can issue a GETMAIN macro instruction specifying subpool 255 and the required storage will be allocated from the system queue area. The system queue area is used to obtain space for system control blocks which might be destroyed by problem programs if they were placed in problem program partitions. The system tasks which request storage space from subpool 255 are:

- The CSCB creation module of SVC 34, for CSCBs
- The ENQ/DEQ processing routines of task supervision, for all control blocks associated with ENQ/DEQ
- The communications task, for write-to-operator (WTO) buffers if all WTO buffer storage space specified during system generation is unavailable

Note: Although subpools are not created in MFT (as in PCP and MVT), problem programs and system tasks may specify subpools in the GETMAIN macro instruction. However, all main storage requests from problem programs are allocated from the highest available main storage in the partition which issued the GETMAIN.

The boundary box for the system queue area is located in master scheduler resident data area IEESD568 (see Appendix A). The master scheduler resident data area is addressed by the CVTMSER field in the Communications Vector Table.

When problem programs issue GETMAIN macro instructions specifying a subpool from 0 through 127, storage is allocated from the high-address portion of the partition in which the GETMAIN macro instruction was issued. When problem programs attempt to issue a GETMAIN macro instruction specifying a subpool from 128 through 255, the program is abnormally terminated. When system tasks issue a GETMAIN macro instruction specifying a subpool from 0 through 127, storage is allocated from the low-address portion of the partition; when specifying a subpool from 128 through 254, storage is allocated from the high-address portion of the partition. Subpool 255 is handled as a special case as described in preceding paragraphs.

For a complete description of main storage supervisor functions, see Fixed-Task Supervisor, Program Logic Manual.

TIMER SUPERVISION

Timer supervision routines are an optional feature of MFT. If selected, the user may request timer services through the TIME, STIMER, and TTIMER macro instructions. The TIME service routine (IEA0RT00) determines the date and time of day. The STIMER service routine (IEA0ST00) sets a user specified interval, and the TTIMER service routine (IEA0ST00) determines the amount of time remaining in a previously specified interval. Whenever a timer interval is requested in an STIMER macro instruction, a timer queue element (TQE) is constructed. These elements are chained together in a timer queue. The queue is ordered so that the TQE representing the next interval to expire is always at the top of the queue. When a requested interval expires, a timer interruption occurs and the Supervisor Timer Interruption Handling Routine (IEA0TI00) takes appropriate action, depending on the type of interval which has expired. If job/step CPU timing is included in the system, IEA0TI00 adjusts the pseudo timer field in IEATPC in the same manner it adjusts the hardware timer.

TIMING PROCEDURE

The System/360 interval timer is a 32 bit word in lower main storage which continually decrements as long as the system is running and the interval timer switch is on. The timer supervision routines use this hardware timer to accomplish their functions. The timer supervision routines can set the hardware timer to any interval between zero and six hours. An interruption occurs when the hardware timer decrements to zero. Since the hardware timer never exceeds six hours, four values are needed to maintain elapsed time for a full day. These values are:

- Hardware timer
- Six Hour Pseudo Clock (SHPC)
- Twenty-four Hour Pseudo Clock (T4PC)
- Local Time Pseudo Clock (LTPC)

The SHPC is used to time intervals up to six hours; the T4PC is used to time intervals up to twenty-four hours. The LTPC contains the local time of day entered by the operator during system initialization.

When an STIMER macro instruction is issued, the STIMER supervisory routine adjusts the time interval requested relative to the intervals in the hardware timers and pseudo clocks. This enables the supervisory routines to place the newly requested timer element in the correct place on the timer queue.

TIMER PSEUDO CLOCK ROUTINE (IEATPC)

The timer pseudo clock routine (IEATPC) contains all variable information that

would normally be included in the resident timer routines. This information includes:

- Pseudo clocks
- Work space used for incrementing CVT date
- Accumulator for the job/step CPU timing feature

COMPARISON OF PCP, MFT, AND MVT TIMER SUPERVISION

Requests for timer services in PCP, MFT, and MVT are made using the same macro instructions. Timer requests are enqueued on the timer queue in the same way in all three systems. There is one difference between PCP and MFT timer supervision. Because there is only one partition in PCP, the timer completion exit routine receives control as soon as a requested task time interval expires. When a timer interval expires in MFT, the timer completion exit routine does not receive control until the task which requested the timer interval is the highest priority ready task in the system. In MVT, the maximum amount of time permitted to complete a job step or cataloged procedure may be specified on the EXEC card. This facility is not provided in MFT.

For a complete description of timer supervisor, see Fixed-Task Supervisor, Program Logic Manual, and MVT Supervisor Program Logic Manual.

OVERLAY SUPERVISION

The routines which supervise loading of overlay program segments and assist flow of control between segments of the overlay program are identical in operation for PCP and MFT. A complete description of PCP and MFT overlay supervision can be found in Fixed-Task Supervisor, Program Logic Manual.

MFT RECORDING/RECOVERY ROUTINES

Operating System Recording/Recovery routines are optional control program routines which may be selected during system generation. They handle two types of equipment malfunctions:

- Malfunctions of the central processing unit (CPU), which cause machine-check interruptions.
- Malfunctions in a channel, which cause input/output interruptions.

Operating System Recording/Recovery routines are divided into two groups: System Environment Recording and Recovery Management.

System Evironment Recording includes:

- System Environment Recording 0 (SER0, described in IBM System/360 Operating System: Fixed-Task Supervisor, Program Logic Manual, Form Y28-6612.
- System Environment Recording 1 (SER1), also described in the Fixed-Task Supervisor PLM.

Recovery Management includes:

- Machine-Check Handler (MCH), described in IBM System/360 Operating System: Machine-Check Handler for IBM System/360 Model 65, Program Logic Manual, Form Y27-7155.
- Channel-Check Handler (CCH), described in IBM System/360 Operatong System: Input/output Supervisor, Program Logic Manual, Form Y28-6616.

MACHINE-CHECK ROUTINES

There are three machine-check routines.

The recording routines:

SER0, which records information about the error and then places the system in a wait state.

SER1, which records information about the error and attempts to associate the error with a task. If it can do this, it abnormally terminates the task and allows the system to continue operation.

The recovery routine:

MCH, which records information about the error and attempts complete recovery from it, including retry of the instruction that caused the error.

For the Model 65, any one of these three routines may be selected during system generation. For the Model 40, 50, 75, and 91, either SER0 or SER1 may be selected. If no routine is selected, either SER0 or SER1 is used by default. The version used by default depends on the model (or models) specified, and on the size of the system (see IBM System/360 Operating System: System Generation, Form C28-6554).

CHANNEL-CHECK ROUTINE

There is only one channel-check routine:

CCH, which aids recovery from channel errors by:

- Providing channel error information to IBM-supplied device dependent error recovery procedures (ERP).
- Building a record entry which is later written on SYS1.LOGREC by the statistical data recorder of the I/O supervisor.

This routine may be selected during system generation for the Model 65, 75, and 91 only.

SYSTEMS WITHOUT RECORDING/RECOVERY ROUTINES

When an equipment malfunction (caused by a machine/check error or a channel error) occurs on an IBM System/360 model that does not have Recording/Recovery routines, the computer is placed in a wait state (See Figure 10A). If the system is a Model 30, the operator may then load the System Environment Recording, Editing, and Printing (SEREP) program. SEREP is a model-dependent stand-alone diagnostic program. It is described in IBM System/360: General Programming Considerations, Form Y20-0005.

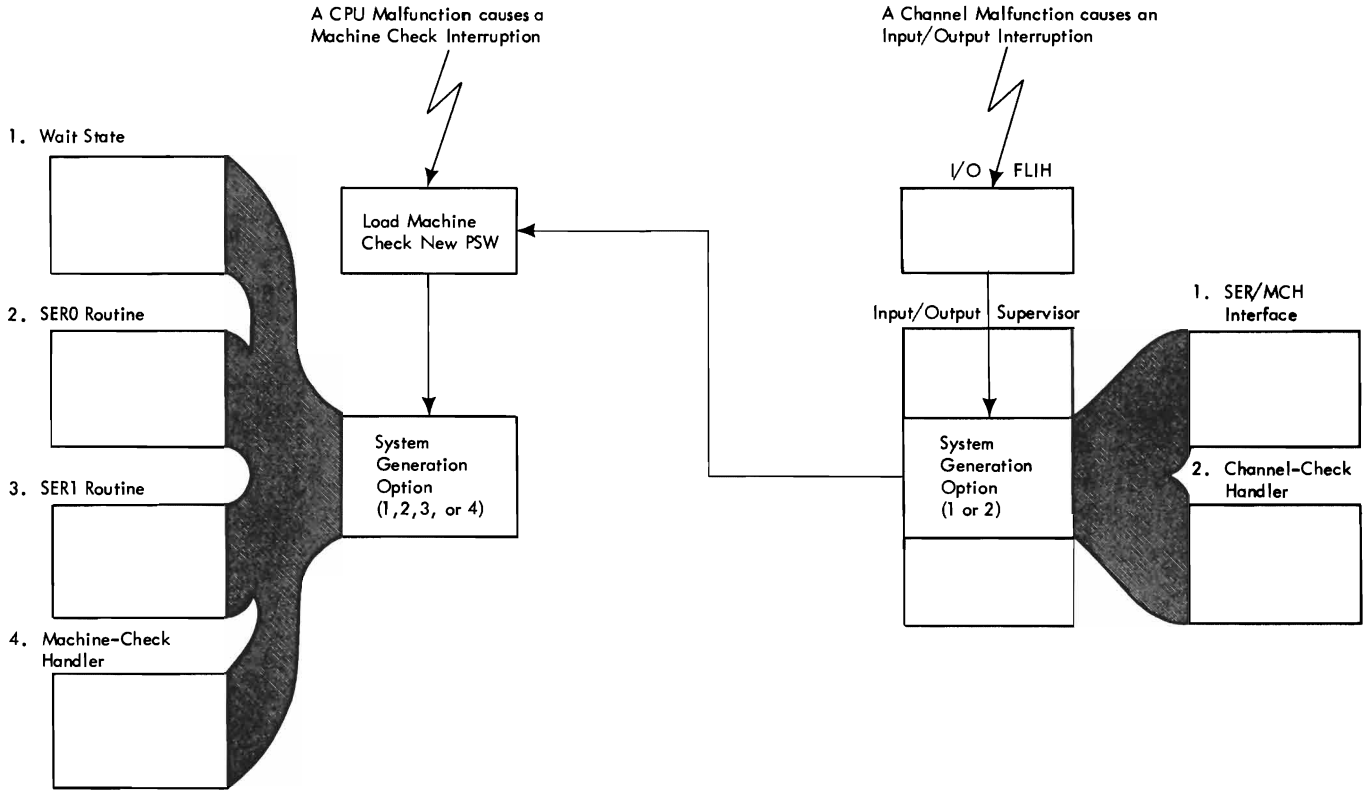
ENTRY TO RECODING/RECOVERY ROUTINES

When a machine-check interruption occurs, the machine-check new PSW is loaded. This causes control to pass directly to the Recording/Recovery routine which was selected during system generation (see Figure 10A).

When an I/O interruption occurs because of a channel error, the I/O new PSW is loaded. This causes control to pass to the I/O FLIH and then to the I/O Supervisor.

If the Channel-Check Handler option was not selected during system generation, the I/O Supervisor enters the SER Interface subroutine (SERR04) within the I/O Supervisor. This routine loads the machine-check new PSW (See Figure 10A).

If the Channel-Check Handler was selected during system generation, the I/O Supervisor enters the Channel-Check Handler Interface (SERR04) within the I/O Supervisor (see Figure 10A).



• Figure 10A. Recording/Recovery Routines

The primary job management 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 requirements
- Assigns input/output devices
- Passes control to the job step

Following step execution, job management:

- Releases main storage space occupied by the tables
- Frees input/output 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 the system (job processing), and the communications and master scheduler tasks, which handle all operator-system communication (command processing).

JOB SCHEDULER FUNCTIONS

The job scheduler includes three programs: the reader/interpreter, the initiator/terminator, and the system output writer. The functions of the reader/interpreter are similar to the MVT reader; additional information can be found in IBM System/360 Operating System: MVT Job Management, Program Logic Manual, Form Y28-6660.

After all control statements for a job have been processed, all initiators that are waiting for that job class are posted and the initiator residing in the highest priority partition is given control. The MFT initiator is described in this publication; for information on allocation and termination, refer to IBM System/360 Operating System: MVT Job Management, Program Logic Manual, Form Y28-6660.

When the job step has been executed, control is returned to the initiator/terminator which performs data set dispositions and releases input/output (I/O) resources. If the entire job is to be terminated, the terminator enqueues all data sets on the appropriate system output (SYSOUT) queues.

When the system output writer receives control, it dequeues a job from an output queue, and transcribes the data sets to the user-specified output device. (See IBM System/360 Operating System: MVT Job Management, Program Logic Manual, Form Y28-6660, for further information on the system output writer.)

COMMUNICATIONS TASK FUNCTIONS

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

- Operator commands, entered through the console or through the input stream
- Write-to-operator (WTO) and write-to-operator with reply (WTOR) macro instructions
- Interruptions caused when the INTERRUPT key is pressed, to switch functions from the primary console to an alternate console

MASTER SCHEDULER TASK FUNCTIONS

The master scheduler task handles job queue manipulation commands and partition definitions. SVC 34 and the master scheduler resident command processor routines comprise the master scheduler task.

JOB MANAGEMENT CONTROL FLOW

Figure 11 shows the major components of job management and 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 communications task which issues a message to the operator instructing him to enter commands, or to redefine the system. If he chooses to redefine the system, control passes to the master scheduler task to handle the redefinitions. If the system is not to be redefined, the initialization commands (a SET command, a START reader command, a START writer command, and a START INIT command) are issued (either automatically by the master scheduler task or by the operator), and job processing begins.

Entry to Job Management Following Step Execution

Following step execution, control is passed to the step termination routine of the initiator/terminator. If no further job steps are to be processed, control is also passed to the job termination routine of the initiator/terminator. Both routines are described in the topic "Initiator/Terminator."

MFT job management is similar in many respects to MVT job management. However, certain major differences in logic exist. These differences are described in two major topics. "Command Processing" includes the communications task and master scheduler task. "Job Processing" includes:

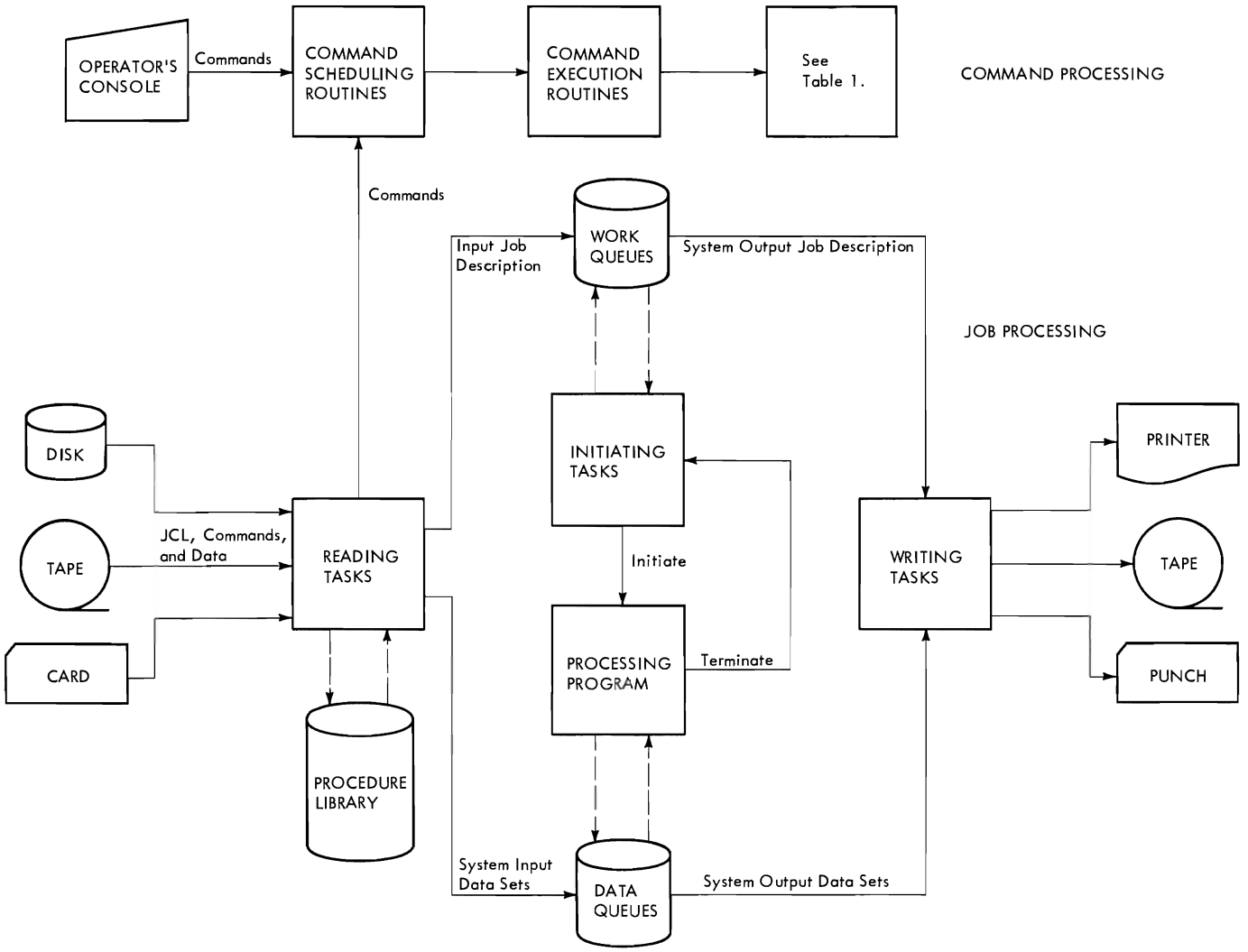


Figure 11. Job Management Data Flow

- Queue Management
- Reader/Interpreter
- Initiator/Terminator
- System output writer
- System task control
- System restart

References to IBM System/360 Operating System: MVT Job Management, Program Logic Manual, Form Y28-6660 are made in the topics where the logic is the same as in MVT.

Tables and work areas used by MFT, MFT module descriptions, and MFT flowcharts are included in the appendixes.

• Table 1. Initial Responders to Commands

Command	Initial Responder
CANCEL (active jobs)	Initiator
CANCEL (job in queue)	Master Scheduler
DEFINE	Master Scheduler
DISPLAY STATUS	Initiator
DISPLAY JOBNAMES	Initiator
DISPLAY A,Q,R,N,jobname	Master Scheduler
DISPLAY T	Timer Maintenance Routine *
HALT	Statistics Update Routine *
HOLD	Master Scheduler
MODIFY	Writer
MOUNT	Master Scheduler
RELEASE	Master Scheduler
REPLY	Master Scheduler
RESET	Master Scheduler
SET CLOCK, DATE	Timer Maintenance Routine *
SET PROC, Q, AUTO	Master Scheduler
START/STOP Reader	Reader/Interpreter
START/STOP Writer	Writer
UNLOAD	Initiator
VARY	Initiator

* See the publication IBM System/360 Operating System: MVT Supervisor, Program Logic Manual, Form Y28-6659.

COMMAND PROCESSING

Operator commands control system operation and modify system tasks. Command processing in MFT is handled by the communications task and the master scheduler task. With the exception of DEFINE, commands can be entered into the system through the console or the input job stream. The DEFINE command can be entered only through the console. Commands entered through the console are read by the communications task and routed to the master scheduler (see Figure 12). The communications task also communicates between the system and the operator; it handles WTO/WTOR macro instructions, assigns message identifiers (including partition numbers), and maintains reply queue elements.

When a command is encountered in the input stream, the reader/interpreter passes control to SVC 34 to process the command. SVC 34 processes most commands completely and returns control to the interrupted routine. For the SET and DEFINE commands, which are not processed completely by SVC 34, control is passed to the master scheduler resident command processor routine.

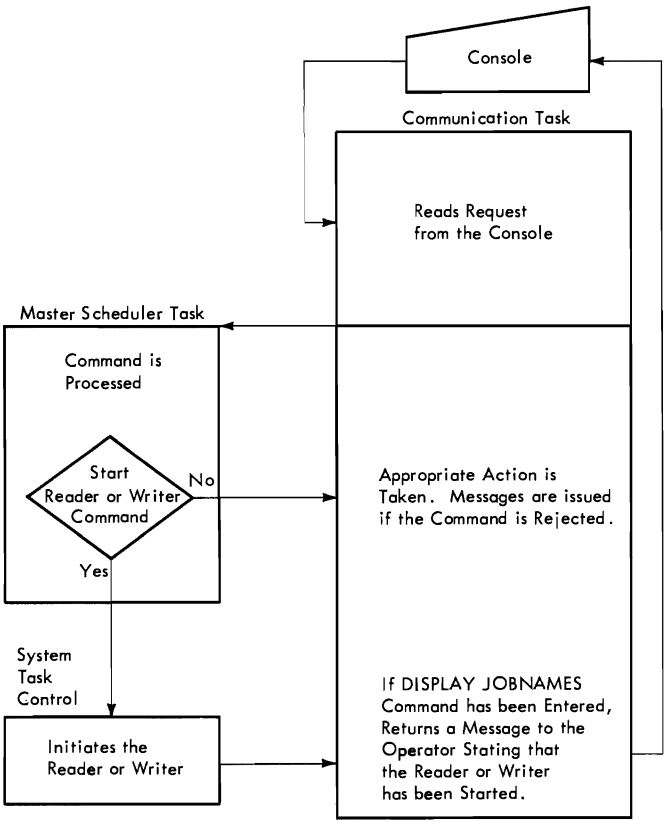


Figure 12. Command Processing Flow

The commands accepted and processed by MFT are the following:

CANCEL
DEFINE
DISPLAY
HALT
HOLD
MODIFY
MOUNT
RELEASE
REPLY
RESET
SET
START
STOP
UNLOAD
VARY

The format and syntax of these commands can be found in IBM System/360 Operating System: Operator's Guide, Form C28-6540.

COMMUNICATIONS TASK

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

WTO/WTOR MACRO INSTRUCTION PROCESSING

Whenever a WTO or WTOR macro instruction is issued, a supervisor call (SVC) interruption occurs. The supervisor identifies the type of interruption and passes control to the communications task to issue messages and/or to read replies. (See Figure 13.)

EXTERNAL INTERRUPTION PROCESSING

When the operator presses the INTERRUPT key, an external interruption occurs. The communications task then switches from the primary console device to the alternate console I/O device. (See Figure 14.)

COMMUNICATIONS TASK MODULES

The communications task (Chart 05) receives control through interruptions which occur when commands are entered or

messages are written. The following paragraphs describe the seven major routines of the communications task.

Console interruption routine (IEECVCRA): notifies the communications task wait routine that a console read has been requested.

Communications task wait routine (IEECVCTW): waits for all WTO/WTOR requests and console interrupts and calls the communications task router routine.

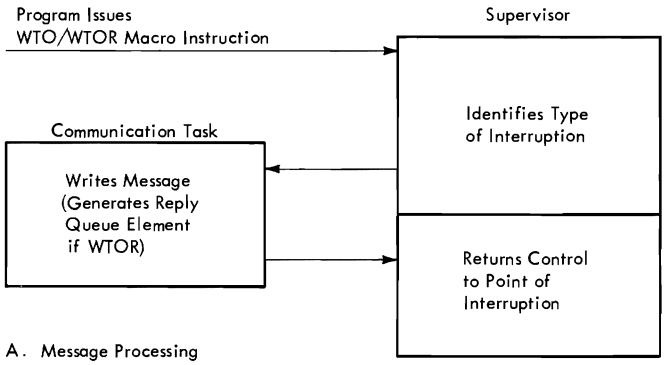
Communications task router routine (IEECVCTR): determines the type of request or interruption that occurred and passes control to the appropriate processing routine.

Console device processor routines (IEECVPM): performs console read and write operations and error checking.

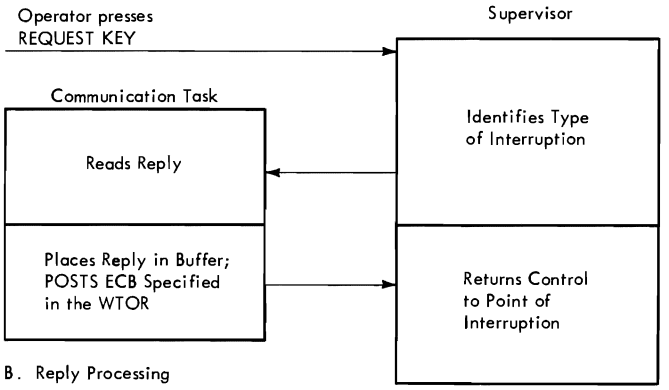
Write-to-operator routine (IEECVWTO): manages WTO buffers.

Write-to-operator with reply routine (IEECVWTOR): manages WTOR buffers.

External interruption routine (IEECVCRX): switches to the alternate console device when an external interruption occurs.



A. Message Processing



B. Reply Processing

Figure 13. WTO/WTOR Macro Instruction Processing Flow

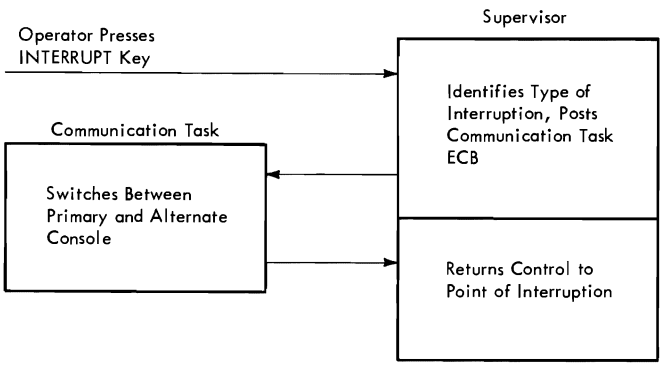


Figure 14. External Interruption Processing Flow

Commands are issued through the console device or the input reader. Before entering commands through the console device, the operator must cause an I/O interruption by pressing the REQUEST key. 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 communications task console interruption routine in the nucleus. The console interruption routine posts the attention event control block (ECB) in the unit control module (UCM) and sets the attention flag in the UCM list entry corresponding to the device from which the interruption came. Posting of the attention ECB causes the communications task wait routine to be dispatched.

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

The communications task serves a number of purposes. The first segment of SVC 72, called the router, distinguishes among these purposes and establishes the order of response. When a posted ECB is found by the router, the router passes control to the specified processor routine via an XCTL macro instruction.

The console-device processor routines read and write using the EXCP macro instruction. The processor routines consist of a routine to service external interruptions and three device-oriented routines: 1052 Printer-Keyboard routine, card reader routine, and printer routine.

Each of the three console input/output processor routines is associated with an OPEN/CLOSE support routine, which provides data management and input/output supervisor control blocks. The specified processor routine reads the input message into a buffer area and calls the master scheduler task via an SVC 34.

The write-to-operator routine moves the text from the requesting program's area to a buffer area within the nucleus and posts the communication ECB for write-to-operator.

The write-to-operator with reply routine generates a message ID, including a partition identifier, and creates a reply queue element (RPQE) to handle the operator's reply.

The external interruption routine, residing in the nucleus, switches to an alternate console device when the operator presses the INTERRUPT key on the console.

CONSOLE ATTENTION INTERRUPTION ROUTINE (IEECVCRA)

The console attention interruption routine (IEECVCRA), operating in privileged mode, posts the communications task attention ECB to request reading of the console. Input/output interruptions are disabled without destroying register contents, and without macro access to supervisor services. Using the address of the UCB (found in register 7), the UCB address is matched to a UCM entry. The attention flag for the entry is turned on. Control then passes to the POST routine, indicating the attention ECB in the UCM. The address in register 14 is used for return to the input/output supervisor (IOS).

COMMUNICATIONS TASK WAIT ROUTINE (IEECVCTW)

Upon entry from the dispatcher, the communications task wait routine (IEECVCTW) issues a WAIT (with a count of one) specifying the list of ECBs whose address is contained in the Event Indication List (EIL). Thus the communications task can respond to a variety of events since the posting of any one ECB satisfies the wait. The POST macro instruction issued in the console attention interruption routine satisfies the wait, causing the TCB to be placed on the ready queue. When next dispatched, the wait routine issues an SVC 72 which results in creation of a supervisor request block (SVRB), and fetching of the first segment of the console processor routines into the system transient area.

COMMUNICATIONS TASK ROUTER (IEECVCTR)

The communications task router (IEECVCTR) is the first segment of SVC 72 brought into the transient area. Because the communications task serves a number of purposes, and many service requests may be pending, the router establishes the order of response. The order is: external interruption, input/output list completion, attention (console interruption), and WTO/WTOR. Multiple attentions are treated in order of appearance in the UCM. Multiple input/output 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. The parameter list consists of a remote XCTL parameter list, the address of the appropriate UCM entry, and the address of (contents of CVTCUCB) the UCM. The router then passes control to a processor routine by issuing an XCTL macro instruction to the remote parameter list, using the name obtained from the unit control block (UCB) entry. The flag signifying the request to be serviced by the processor routine is turned off by the routine. Consequently, processor routines return control to the router by issuing an XCTL macro instruction to allow the router 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 device to which the message is to be sent. The router establishes the output device by checking UCB entry attribute indicators. The appropriate entry is the first active UCB entry that supports WTO. As before, the router builds a remote interface for, and passes control to, a processor routine via an XCTL macro instruction.

CONSOLE DEVICE PROCESSOR ROUTINES (IEECVPMX, IEECVPMC, IEECVPMP)

Control flow in a processor routine is determined by the setting of flags in the router-selected UCM entry. The close flag is tested first. If this flag is on, any pending input/output activity is suspended by issuing a WAIT macro instruction. Control is then passed to an associated OPEN/CLOSE support routine via an XCTL macro instruction for release of various control blocks. If the close flag is off, the busy flag is tested to determine input/output status. If there is outstanding input/output activity, error checking and buffer disposition occur if the activity has been posted complete. Otherwise, any attention request is temporarily abandoned(as are

output requests), and control returns to the router via an XCTL macro instruction. If the busy flag is off, the attention flag is tested; if it is on, the status of the device is examined. If the device has not been opened, control passes to an associated OPEN/CLOSE support routine via an XCTL macro instruction to obtain storage for a DCB and access-method dependent control blocks, and for execution of the OPEN macro instruction.

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. Input/output activity is initiated by issuing an EXCP macro instruction for a 1052, and by issuing a READ macro instruction for a unit record device. In no case does the processor routine await completion of this activity. Control immediately returns to the router via an XCTL macro instruction.

Control flow within the processor routine is as described previously up to the point at which the output request flag is tested. If the flag is 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 input/output activity. This preserves a means of retrying the message if an external interruption 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 is unaffected by previous attempts to present the message to the primary device.

Having selected a buffer, the processor routine establishes data management and input/output supervisor (IOS) control block linkages. The routine then issues an EXCP macro instruction for a 1052, or a WRITE macro instruction for a printer. Without awaiting completion of the input/output, the processor routine returns to the router via an XCTL macro instruction.

WRITE-TO-OPERATOR ROUTINES (IEECVWTO AND IEEVWTOR)

The write-to-operator routine (SVC 35) writes operator messages on the console when a WTO or WTOR macro instruction is issued by system component programs or problem programs. Messages and replies are buffered; the period of time between issuing the message and receiving the reply is available for processing. Issuance of either macro instruction causes an SVC interruption. When the SVC interruption is handled, the supervisor causes the write-to-operator routine to be loaded into the transient area of the nucleus and passes control to it.

There are two console queues: the buffer queue and the reply queue. The extent of both queues is defined by specifying the number of buffers at system generation. An attempt to exceed this value results in the requesting task being placed on a queue to wait for service; i.e., the task is placed in a wait condition. Each WTO and WTOR macro instruction results in the addition of a WTO Queue Element (WQE) to the buffer queue; each WTOR results in the addition of a Reply Queue Element (RPQE) to the reply queue. SVC 35 (IEECVWTO) sets up the problem program message. If it is a WTOR, the write-to-operator-with-reply routine (IEEVWTOR) inserts the message identification (ID) in addition to a partition identifier. The same message ID (which the operator must use for his reply) is placed in the RPQE with other information to insure passing the reply, when received, to the proper area. WTO messages are always written; 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. Like WTOR elements in the buffer queue, RPQES contain a TCB address to permit their being purged from the queue if the issuing task is abnormally terminated.

For a REPLY (to WTOR), the processor issues SVC 34 (see "Master Scheduler Task"). The SVC routine determines that the incoming command is a REPLY, processes the reply, posts the user's ECB and branches back to the processor.

EXTERNAL INTERRUPTION ROUTINE (IEECVCRX)

The external interruption routine assigns functions performed by the primary console device to an alternate console device. When the operator presses the INTERRUPT key on the console, an external interruption occurs and control passes to the supervisor. The supervisor identifies the interruption and passes control to the external interruption routine which 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 interruption.

MASTER SCHEDULER TASK

The MFT master scheduler task (MST) processes all commands, and initializes main storage at system initialization. It is composed of the SVC 34 routines and the master scheduler resident command processor routines. SVC 34 processes all commands directly except HOLD, RELEASE, RESET, CANCEL (inactive jobs), DISPLAY (A,Q,N, jobname) and DEFINE. SVC 34 calls the resident command processor to complete the processing of these commands.

The master scheduler resides in the nucleus and operates under control of its own TCB. The master scheduler TCB is always dispatchable and is of higher priority on the TCB queue than the TCBs for the partitioned area (the problem program area) of storage. Therefore, when a command is issued, the master scheduler always gains control of the CPU after the communications task for processing the command.

When processing commands, interruptions are disabled so that command processing may be completed before any other interruptions are serviced. Although commands are processed when issued, the command may not take effect immediately. An example of this is the STOP writer command. The master scheduler marks a command scheduling control block (CSCB) which is checked by the writer between jobs. The command does not take effect until the writer completes the job it was processing when the command was issued.

SVC 34 FUNCTIONS

SVC 34 (Chart 06) is called to process all commands. As previously noted, it processes some of these commands completely and calls the resident command processor to process the remaining commands. The commands processed completely by SVC 34 are:

START
STOP
MODIFY
CANCEL (active jobs only)
HALT
MOUNT
VARY
UNLOAD
REPLY
DISPLAY (JOBNAMES, T, or STATUS)

For CANCEL (inactive jobs), HOLD, RELEASE, RESET, DISPLAY (A, Q, N, R, jobname), and DEFINE commands, SVC 34 does preliminary processing before passing control to the resident command processor.

The following paragraphs describe two modified MVT routines and two new MFT routines within SVC 34.

Validity Check Command Routine (IEE0403D)

The validity check command routine (IEE0403D) scans and checks the validity of commands for proper syntax and content. The buffer is scanned for the first character of the verb. If the verb is eight characters or less and properly delimited, it is placed in the verb substitute of the extended save area. Then a check is made for a parameter list. If a parameter list is found, its address is put in the parameter position of the extended save area. If there is no parameter list, zero is placed in the parameter list position of the extended save area. Next, the verb is compared against a table of valid verbs. If a match is not found, the COMMAND INVALID error message is written. If a match is found, the validity check routine passes control to the appropriate processing module via an XCTL macro instruction.

Reply Processor Routine (IEE1203D)

The reply processor routine attempts to check the validity of the operator's reply command by matching the command with an outstanding reply request (RPQE). If valid, the reply command is then moved to the buffer of the user that issued the respective WTOR. The RPQE is freed, and chain relinkage is performed. The user's ECB and RPQE count ECB in the UCM are posted. The routine then returns to the calling routine.

DEFINE, MOUNT, and CANCEL Routine (IEESD571)

This routine processes the DEFINE command by setting the necessary indicators in the master scheduler resident data area. It then posts the ECB for the resident command processor. This routine also processes the CANCEL command (for active jobs), and the MOUNT command.

MOUNT processing parallels that of PCP by building a parameter list for, and issuing an XCTL macro instruction to the PCP master command EXCP routine (IGC0103D).

Cancelling of an active job is handled by scanning the CSCBs for a jobname compare. If the compare is equal and the CSCB is marked cancellable, IEESD571 issues a BALR to ABTERM with the job's TCB address and proper completion code dump indication. If the CSCB is not marked cancellable, the CSCB is marked canceled and is posted. If the job is not found, IEESD571 passes control to the CSCB creation routine

(IEE0803D) via an XCTL macro instruction, to CANCEL the jobname on the job queue. (See IBM System/360 Operating System: MVT Job Management, Program Logic Manual, Form Y28-6660, for a description of IEE0803D.)

START and STOP INIT Command Routine (IEESD561)

This routine processes the START command and the STOP INIT command. For a START command, the routine first extracts the partition number from the command and determines if the partition can accept the command; i.e., the routine determines if the partition is established as a reader, writer, or problem program, and if it is large enough to contain the requested task. The routine builds and chains a CSCB, passes the address of the CSCB to the partition's PIB, and posts the partition. (See Figure 15.)

This routine also processes the STOP INIT command. After verifying that the partition number is correct, the routine marks the partition's PIB to indicate STOP INIT.

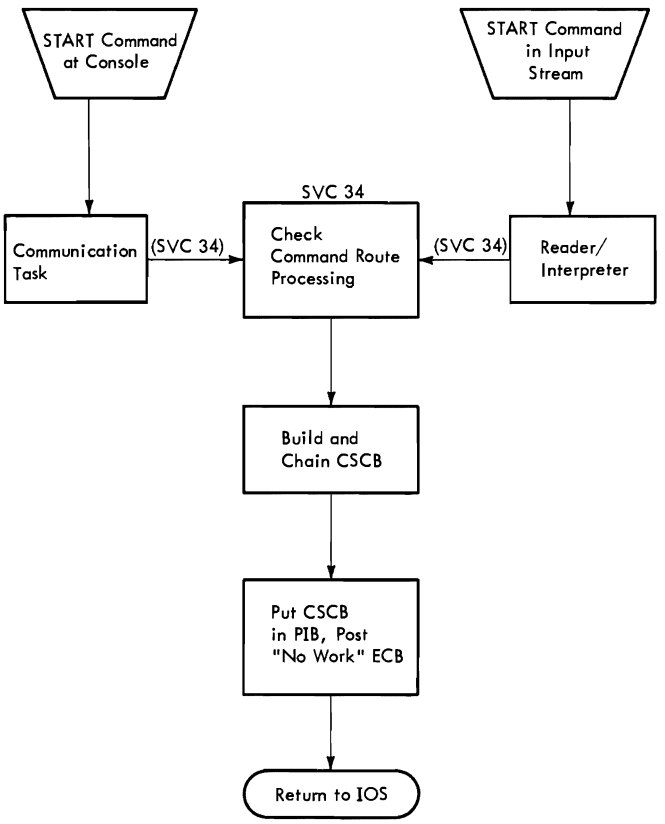


Figure 15. START Command Processing Flow

SYSTEM INITIALIZATION

The master scheduler task (Chart 07) performs the function of initializing main storage. In MVT this is done by NIP. In MFT it is done by the master scheduler to facilitate redefinition of main storage. The following paragraphs describe the action of the master scheduler in defining main storage at system initialization.

The master scheduler task is loaded with the nucleus. Its task control block (TCB) points to the master scheduler request block (RB) in the nucleus. NIP saves the RB address and the contents of the boundary box describing the normal master scheduler task partition, for later use by the master scheduler initialization routine IEFSD569. (Note: IEFSD569 is brought into main storage by the macro instruction SGIEEOVV generated during system generation.)

The boundary box (BBX) is then changed by NIP to describe a partition including all of storage except the nucleus. The address of an RB at the low address of this partition is placed in the master scheduler TCB. NIP then creates the RB. The RB points to an XCTL to IEFSD569. NIP then sets the master scheduler task dispatchable and branches to the dispatcher.

The master scheduler initialization routine is given control to perform scheduler initialization. First it passes control to the communications task initialization routine (IEECVCTI) via a LINK macro instruction. It then uses the communications task to request any partition changes. On return from the communications task, the master scheduler initialization routine passes control to the definition routine, IEEDFIN1, via a LINK macro instruction. IEEDFIN1 communicates with the operator, or prepares the partition as it was described at system generation. IEFSD569 then issues the READY message, types the automatic commands, and issues a WAIT macro instruction.

When the operator presses the REQUEST key, control is given to the supervisor which recognizes the interruption and passes control to the input/output supervisor. The input/output supervisor determines that the interruption is an attention signal and passes control to communications task console attention interrupt routine (described above). The interrupt routine posts the communications task attention ECB to request reading of the console. The operator enters a SET command. SVC 34 posts the WAIT and places the parameters of the SET command in the master scheduler resident data area. The master scheduler initialization routine then regains control to continue processing. Control blocks for

the job queue and procedure library are created. To format the job queue, the routine passes control to queue initialization routine IEFSD055 via a LINK macro instruction which places a queue control record (QCR) in the nucleus after the DCB and DEB. Control then passes to queue manager formatting routine IEFORMAT which formats the job queue and returns control to the queue initialization routine. (For a discussion of these two modules, see the topic "Queue Manager".) After return from the queue manager initialization routine, the master scheduler initialization module displays and processes any automatic commands.

The master scheduler initialization routine then establishes partitions based on information in the TCBs. It constructs an RB in each partition, with an XCTL macro instruction addressing job selection module IEFSD510 (for large partitions), or small partition module IEFSD599 (for small partitions). The master scheduler initialization routine then readjusts the pointers to the master scheduler area, and returns to the dispatcher. The dispatcher returns control to the master scheduler task, but the TCB now points to master scheduler router routine IEECIR50, in the nucleus.

Master Scheduler Router Routine (IEECIR50)

Resident master scheduler router routine IEECIR50 waits on an ECB which is posted by SVC 34 when a command has been scheduled for processing. This router (Chart 08) scans the CSCB chain for any outstanding commands to be processed. If a command is found, the CSCB is removed from the chain. The router routine then passes control to syntax check routine IEESD562 via a LINK macro instruction, passing the address of the CSCB.

After all commands are processed, or if none are found, the router routine determines if a DEFINE command has been entered. If so, the router routine passes control to IEEDFIN1, the first module of the definition routines, via a LINK macro instruction. If no DEFINE command has been issued, the router routine returns to wait on its ECB. No test is made for DEFINE command scheduling until all other commands have been processed.

Syntax Check Routine (IEESD562)

Syntax check routine IEESD562 checks the syntax of the command parameter in the CSCB (Chart 09). If a search of the input work queues (SYS1.SYSJOBQE) is required for processing the command, the syntax check routine sets internal codes for the queue search, issues a GETMAIN to obtain storage, and constructs an event control block (ECB) and an input/output block (IOB). Control is then passed to queue search setup routine IEESD563. If the command was a DISPLAY A command, control is passed to DISPLAY A routine IEESD566. If it was a DISPLAY R command, control is passed to DISPLAY R routine IEESD567.

Queue Search Setup Routine (IEESD563)

Queue search setup routine IEESD563 determines which of the queues is to be searched and reads the queue control record (QCR) for that queue. If the queue must be searched, the queue search setup routine establishes parameters for the search. The queue search setup routine then passes control to queue search routine IEESD564 via an XCTL macro instruction. When the queue search setup routine regains control, the QCR is scanned and if any information in the record has been changed, the updated information is rewritten on SYS1.SYSJOBQE. The queue search setup routine then establishes a parameter list and passes control to service routine IEFSD565 via an XCTL macro instruction.

Queue Search Routine (IEESD564)

Queue search routine IEESD564 reads the entries of a queue based on the parameter information passed by setup routine IEESD563. If the command processing requires changes in the chaining information in a queue entry or control record, the updated information is written on the queue. Action indicators are passed as parameters when control returns to the setup routine.

Service Routine (IEESD565)

Based on the information passed by the calling routine, service routine IEESD565 performs the following:

1. Passes control to queue manager enqueue routine IEFQMNQQ via a LINK macro instruction to enqueue an entry or QCR.

2. Issues a FREEMAIN macro instruction to free the ECB/IOB which was used to read SYS1.SYSJOBQE.

3. Passes control to the master scheduler message module (IEE0503D) via a LINK macro instruction to write a message.

After the requested processing has been performed, the service routine transfers control to router routine IEECIR50.

DISPLAY A Routine (IEESD566)

DISPLAY A routine IEESD566 receives control from syntax check routine IEESD562

when the DISPLAY A (active) command is entered. This routine constructs WTO messages containing the active job and stepnames. The DISPLAY A routine returns control to the router routine.

DISPLAY R Routine (IEEDS567)

DISPLAY R routine IEESD567 receives control from syntax check routine IEESD562 when the DISPLAY R command is entered. This routine constructs WTO messages containing the ID of each unreplied-to WTOR message, the unit number of each device for which a mount has been requested but not complied with, and an indication if an AVR mount message is pending. If none of these conditions exists, the operator is advised that there are no outstanding requests. DISPLAY R routine returns control to router routine IEECIR50.

PARTITION DEFINITION BY THE MASTER SCHEDULER

The master scheduler uses the DEFINE command processing routines (shown in Figure 16) to initialize and change partition definitions in MFT.

DEFINE Command Initialization Routine (IEEDFIN1)

The master scheduler passes control to DEFINE command initialization routine IEEDFIN1 whenever a DEFINE command is entered by the operator. The routine also receives control from the master sheduler during system initialization, after the nucleus initialization program (NIP) completes its preparation of the system. In either case the routine builds the DEFINE data area containing the size and description (job classes A-O, or R or W) of each partition If Main Storage Hierarchy Support is included in the system, the data area contains the size of the partitions in terms of hierarchies. Hierarchy 0 represents processor storage and hierarchy 1 represents 2361 Core Storage.

If the time-slicing feature is included in the system, the data area also contains a doubleword of time-slicing information, including the first and last partition numbers in the time-slicing group and the time interval (in milliseconds) assigned to the group of partitions. This data is used at completion of DEFINE processing to define the partitioning of main storage.

If the DEFINE command initialization routine was entered as the result of a DEFINE command, it then determines whether LIST was specified, and, if so, passes control to listing routine IEEDFIN4 via an XCTL macro instruction. If not, the routine passes control to message routine IEEDFIN5 via an XCTL macro instruction.

If the routine was entered during system initialization, it determines whether partition redefinitions of LIST was specified by the operator, and, if not, passes control to validity check routine IEEDFIN3 via an XCTL macro instruction. If either LIST or partition redefinitions were specified, the routine continues processing as if a DEFINE command had been entered by the operator.

Listing Routine (IEEDFIN4)

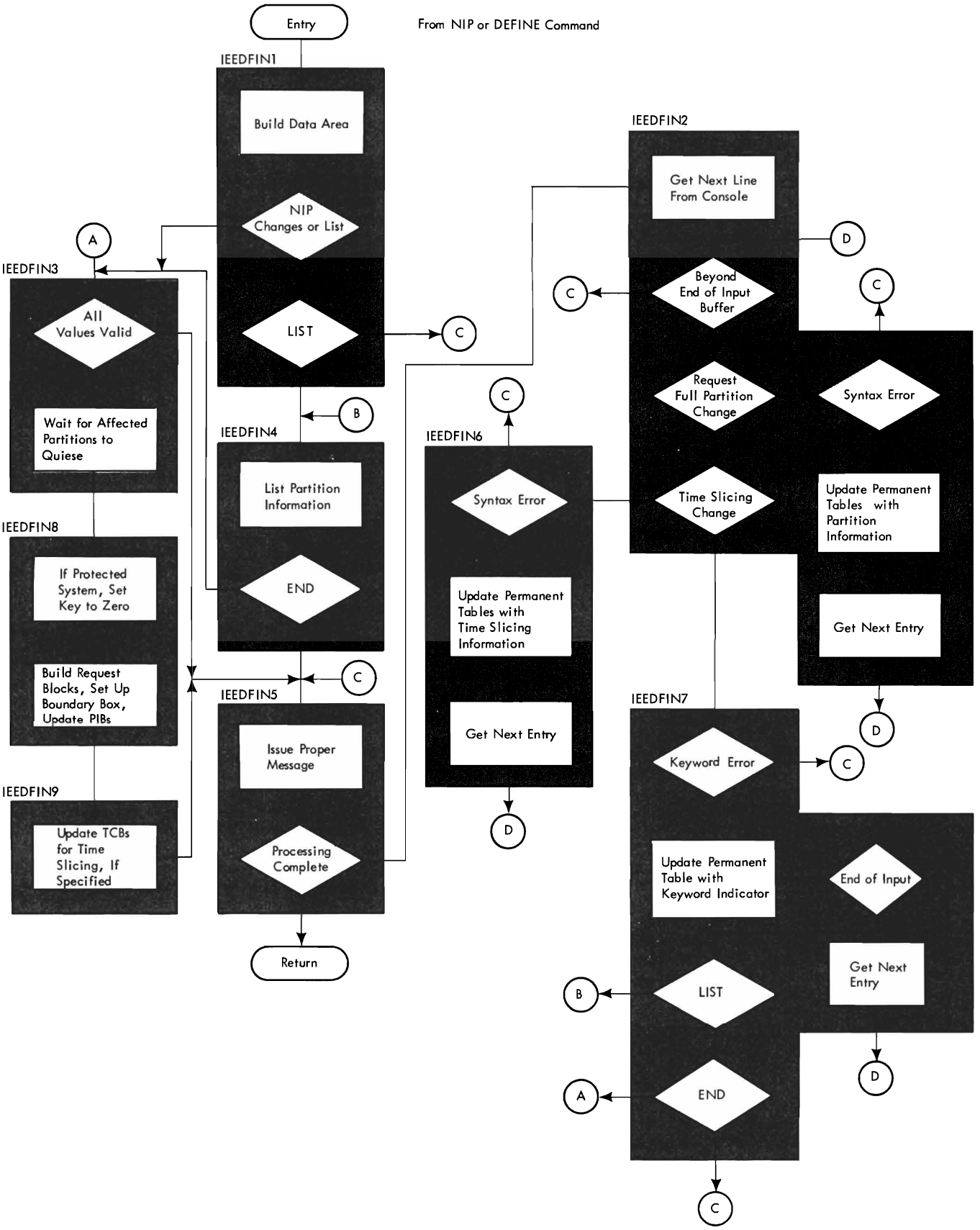
Listing routine IEEDFIN4 lists partition definitions. If the time-slicing feature is in the system, it also lists the time-slicing attributes. After performing the listing function, the routine determines whether an END keyword has been read from the console, and, if so, passes control to validity check routine IEEDFINI3 via an XCTL macro instruction. If not, it passes control to message routine IEEDFIN5 via an XCTL macro instruction.

Message Routine (IEEDFIN5)

The messages issued by message routine IEEDFIN5 when entered by other DEFINE processing routines are:

- IEEDFIN1: The routine issues an ENTER DEFINITION message.
- IEEDFIN2: The routine issues the appropriate error message.
- IEEDFIN3: The routine issues the appropriate error message.
- IEEDFIN4: The routine issues a CONTINUE DEFINITION message.
- IEEDFIN6: The routine issues the appropriate error message.
- IEEDFIN7: The routine issues either the appropriate error message or a CONTINUE DEFINITION message.
- IEEDFIN9: The routine issues a DEFINITION COMPLETED message.

After issuing the message, the routine determines whether processing is complete, and if so, returns to the dispatcher. If not, the routine passes control to syntax check routine IEEDFIN2 via an XCTL macro instruction.



• Figure 16. DEFINE Command Processing Flow

Syntax Check Routine (IEEDFIN2)

Syntax check routine IEEDFIN2 scans the statement entered by the operator. Each entry in the statement -- a partition definition, a time-slicing change, or a keyword -- is processed separately.

If the entry is a partition definition, the routine checks the entry for syntax errors. If a syntax error is found, the routine passes control to message routine IEEDFIN5 via an XCTL macro instruction, thus ignoring the erroneous entry and all following entries. If the partition definition is valid, the routine updates the DEFINE data area with the partition information, then gets the next entry for processing.

If the entry is a time-slicing change, the routine passes control to time-slice check routine IEEDFIN6 via an XCTL macro instruction.

If the entry is neither a partition definition nor a time-slicing change, the routine assumes that it is a keyword and passes control to keyword scan routine IEEDFIN7 via an XCTL macro instruction.

Time-Slice Syntax Check Routine (IEEDFIN6)

Time-slice syntax check routine IEEDFIN6 checks the time-slicing entry for syntax errors. If a syntax error is found, the routine passes control to message routine IEEDFIN5 via an XCTL macro instruction, thus ignoring the erroneous entry and all following entries. If the time-slicing change is valid, the routine updates the DEFINE data area with the time-slicing information, gets the next entry in the statement being processed, and passes control to syntax check routine IEEDFIN2 via an XCTL macro instruction.

Keyword Scan Routine (IEEDFIN7)

Keyword scan routine IEEDFIN7 determines whether the entry being processed is a valid keyword. If it is not a valid keyword, the routine passes control to message routine IEEDFIN5 via an XCTL macro instruction, thus ignoring the erroneous entry and all following entries. If a valid keyword is found, the routine sets the appropriate keyword indicator in the DEFINE data area.

If there are more entries to be processed, the routine gets the next entry and passes control to syntax check routine IEEDFIN2 via an XCTL macro instruction.

If there are no more entries to be processed (end of input), the routine determines whether a LIST keyword had been entered, and, if so, passes control to listing routine IEEDFIN4 via an XCTL macro instruction. If LIST was not specified, a check for the END keyword is made. If an END entry is found, the routine passes control to validity check routine IEEDFIN3 via an XCTL macro instruction; otherwise, control is passed via an XCTL macro instruction to message routine IEEDFIN5.

Validity Check Routine (IEEDFIN3)

Validity check routine IEEDFIN3 makes final checks to determine that the information entered by the operator is correct (e.g., that no more than 15 problem program partitions have been defined). The routine then determines the partitions affected and constructs a list of ECBs (one for each affected partition). The address of the ECB, which must be posted by the affected partition, is placed in the PIB of the partition. If time-slicing information was changed, the new information is placed in the DFTMSL word of the data area, and the time-slice control element (TSCE) is updated accordingly. (See "Dispatching a Task (with Time Slicing)" in the Supervisor section for a description of the TSCE). The routine then issues a WAIT macro instruction on the ECB list. When the ECB list has been posted, the routine passes control to system reinitialization routine IEEDFIN8 via an XCTL macro instruction.

If Main Storage Hierarchy Support is included in the system, IEEDFIN3 also determines whether a partition was defined in two segments. If it was, before setting the task inactive, IEEDFIN3 determines if the operator has reduced both segments of the partition to zero. If both segments have been reduced to zero, the partition is made inactive after the current task terminates. The data area is then altered and message routine IEEDFIN5 receives control to issue the DEFINITION COMPLETED message.

System Reinitialization Routine (IEEDFIN8)

System reinitialization routine IEEDFIN8 sets the protection key to zero if the system is protected. Using the information in the DEFINE data area, the routine builds request blocks and boundry boxes for the defined partitions and updates the partition information blocks. The routine then passes control to command final processor routine IEEDFIN9 via an XCTL macro instruction.

Command Final Processor (IEEDFIN9)

Command final processor routine IEEDFIN9 updates the task control blocks affected by time-slicing if time-slicing is specified. The routine then passes control to message routine IEEDFIN5 via an XCTL macro instruction.

JOB PROCESSING

Job processing is accomplished by three types of tasks:

- Reading tasks, which control the reading of input job streams and the interpreting of control statements in these input streams.
- Initiating tasks, which control the initiating of job steps whose control statements have been read and interpreted. (Terminating procedures are also part of initiating tasks.)
- Writing tasks, which control the transferring of system messages and user data sets from direct-access volumes on which they were written initially to some other external storage medium.

These tasks are created in response to START commands entered for readers, initiators, and writers. Whenever a START reader or writer command is entered, the resulting command processing brings a reader or writer into the associated partition. Initiators are brought into all scheduler-size partitions at system initialization, and after a START INIT command has been issued following partition redefinition. An initiator is also brought into a partition that is specified in a STOP INIT command to terminate the initiator.

There may be more than one of each of the job processing tasks. Input job streams may be read simultaneously from three input devices by issuing a START reader command for each input stream. System messages or data sets may be written to as many as 36 output devices by issuing a START command for each device. Up to 15 initiating tasks can exist concurrently. Each initiating task is created in response to a START INIT command issued for a specific partition, or a START INIT.ALL command. (See IBM System/360 Operating System: Operator's Guide, Form C28-6540.)

Note: The total number of tasks may not exceed 52.

This section is divided into six topics, including the three major tasks discussed above, and three other areas associated with the major tasks: Queue Manager, System Task Control, and System Restart.

QUEUE MANAGER

MFT uses the MVT Queue Manager. However, to reduce possible interlocks due to unavailability of requested tracks, the assign routine (IEFQASGQ) has been modified, and a new module (IEFSD572) has been added. A table breakup routine (IEFSD514) has also been added to subdivide variable size tables located in main storage into 176-byte data records on disk. Descriptions of some MVT modules have also been included to provide a more complete explanation of the relationship of these modules to the entire system.

WORK QUEUES

An MFT system contains 54 work queues which form the job queue data set (SYS1.SYSJOBQE). These 54 work queues are:

- Free-track queue
- HOLD queue
- Remote job entry (RJE) queue
- 36 output class queues
- 15 input job class queues

The job entries are enqueued in priority order within each job class on the appropriate job class queue. Jobs are selected for processing according to the job class designation of the partition requesting work.

QUEUE MANAGEMENT

Queue Manager is a general term describing a group of routines used by various system components, such as the reader/interpreter, initiator/terminator, and output writer. The queue manager performs some common functions for all system components. It performs all input/output for accessing the job queue data set and keeps track of all space on this queue. The queue manager assigns space on the job queue in logical track increments for control blocks, tables, and system messages built by the scheduler. When the control blocks and tables have been created, the reader/interpreter enqueues (ENQs) the job using the queue manager. After the job is enqueued, the initiator dequeues (DEQs) the job for execution when a partition that is assigned to service that job class becomes available for work. The terminator places control information needed by the system output writer on the job queue. At job termination, the terminator enqueues the output work description. The writer then dequeues the output work according to output class and priority within the class, and transcribes it to the appropriate device, specified by the user.

At system generation, the space for the job queue data set is allocated. The

device upon which the job queue resides is considered a non-demountable system residence volume.

JOB QUEUE INITIALIZATION

At system initialization, queue initialization routine IEFSD055 receives control from the SET command processor to construct a data control block (DCB) in the nucleus, and to issue an OPEN macro instruction which causes a data event block (DEB) to be built for accessing SYS1.SYSJOBQE. It also places a queue manager master queue control record (master QCR) in the nucleus after the DCB and DEB. (See Figure 17 for the format of the master QCR.) Control then passes to queue formatting routine IEFORMAT.

The queue formatting routine divides the job queue data set into a control record area and a logical track area. The control record area contains a copy of the master QCR, a control record for the free track queue, a control record for the HOLD queue, a control record for the Remote Job Entry (RJE) queue, a control record for each of the 36 SYSOUT writer classes, and a control record for each of the 15 input work queues. (See Figure 18 for the format of an input queue control record.)

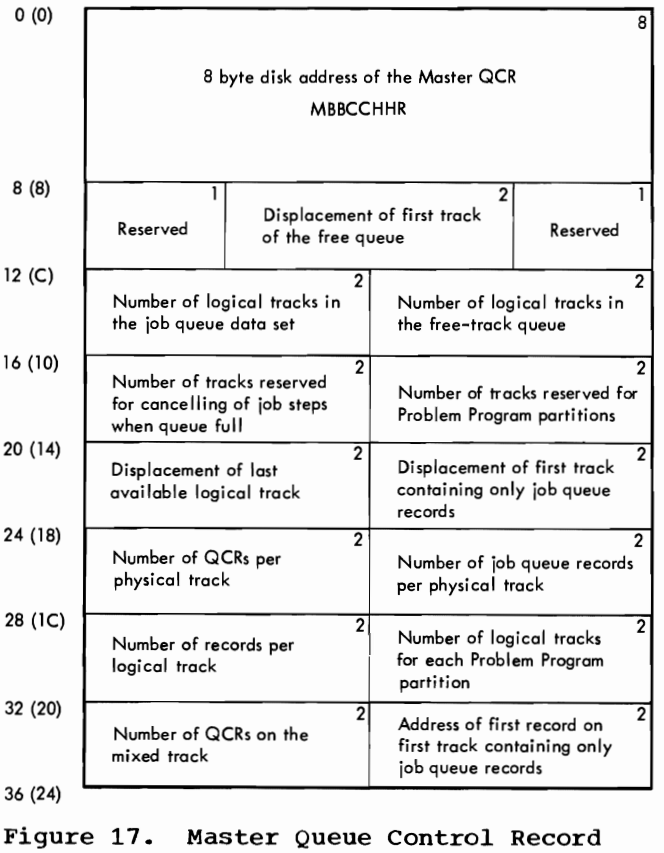


Figure 17. Master Queue Control Record (Master QCR) Format

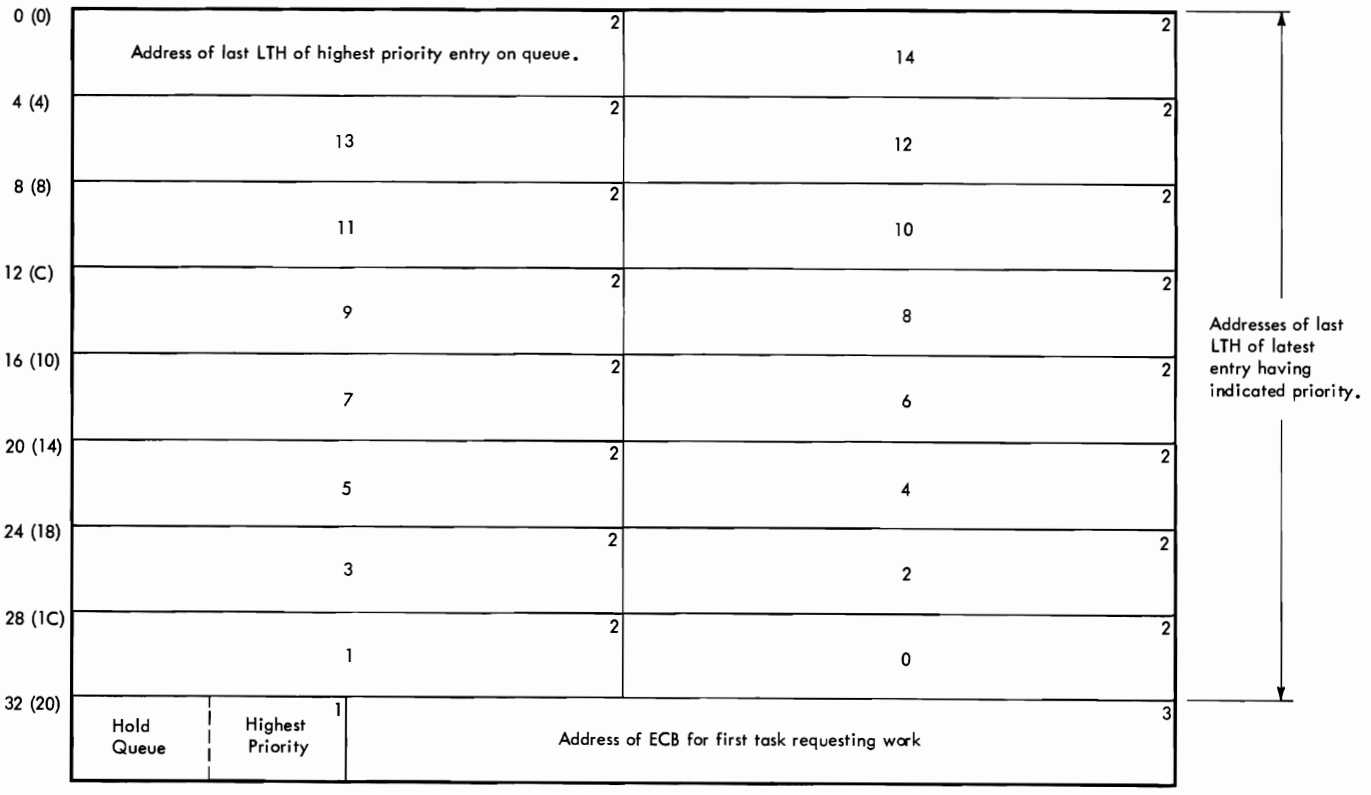


Figure 18. Job Queue Control Record (QCR)

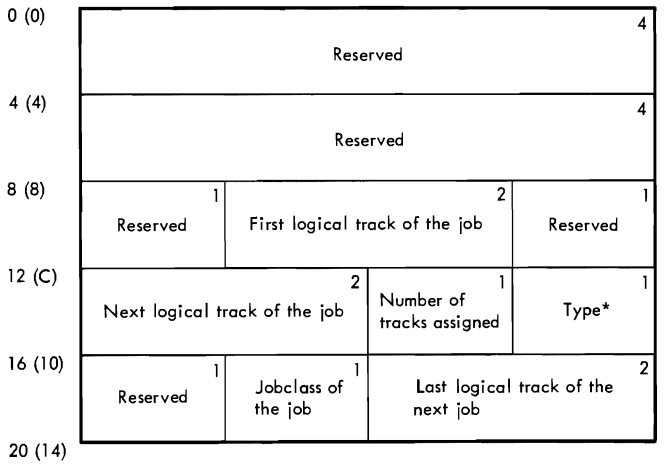
Note: The first position of the job queue control record (job QCR) contains zeros if no work exists. The job QCR contains a minimum of two entries if work exists for at least one priority.

The job class specified by the user (on the JOB statement or in a START command) is converted by the system to match the system-assigned job class identifiers. The user-assigned job class and corresponding system job class identifiers are:

User-assigned job class	System-assigned identifier (hexidecimal)
A	28
B	29
C	2A
D	2B
E	2C
F	2D
G	2E
H	2F
I	30
J	31
K	32
L	33
M	34
N	35
O	36

The logical track area length is variable. Logical tracks are used instead of physical tracks so that the job queue can reside on different device types. Each logical track contains a 20-byte header record (as shown in Figure 19) which includes a pointer to the next track. The header record is used to chain all tracks of a job together. When the job is enqueued, the header record is used to chain jobs first-in/first-out (FIFO) according to priority. All jobs of the same job class are chained together. Following the header record are a variable number of 176-byte data records. The number of records per logical track is determined at system generation. All tables, control blocks, and system messages are in 176-byte increments.

At system initialization, all tracks are members of the free track queue. The free track queue is a list of logical tracks available for assignment to work queues. As tracks are needed, they are taken from the free track queue. When the system is finished with tracks, they are returned to the free track queue. After system initialization, SYS1.SYSJOBQE appears as shown in Figure 20. Figure 21 illustrates typical input and output work queues. Each input and output QCR contains the address of the last entry in each priority queue.



• Figure 19. Logical Track Header (LTH) Record Format

QUEUE MANAGER MODULES

As jobs are read into the system, they are placed into each job class queue according to priority (established by the PRTY parameter on the JOB statement). When the reader/interpreter reads a job or establishes a new queue for an output class, it establishes a queue entry. This is done by Assign/Start Routine IEFQASGT.

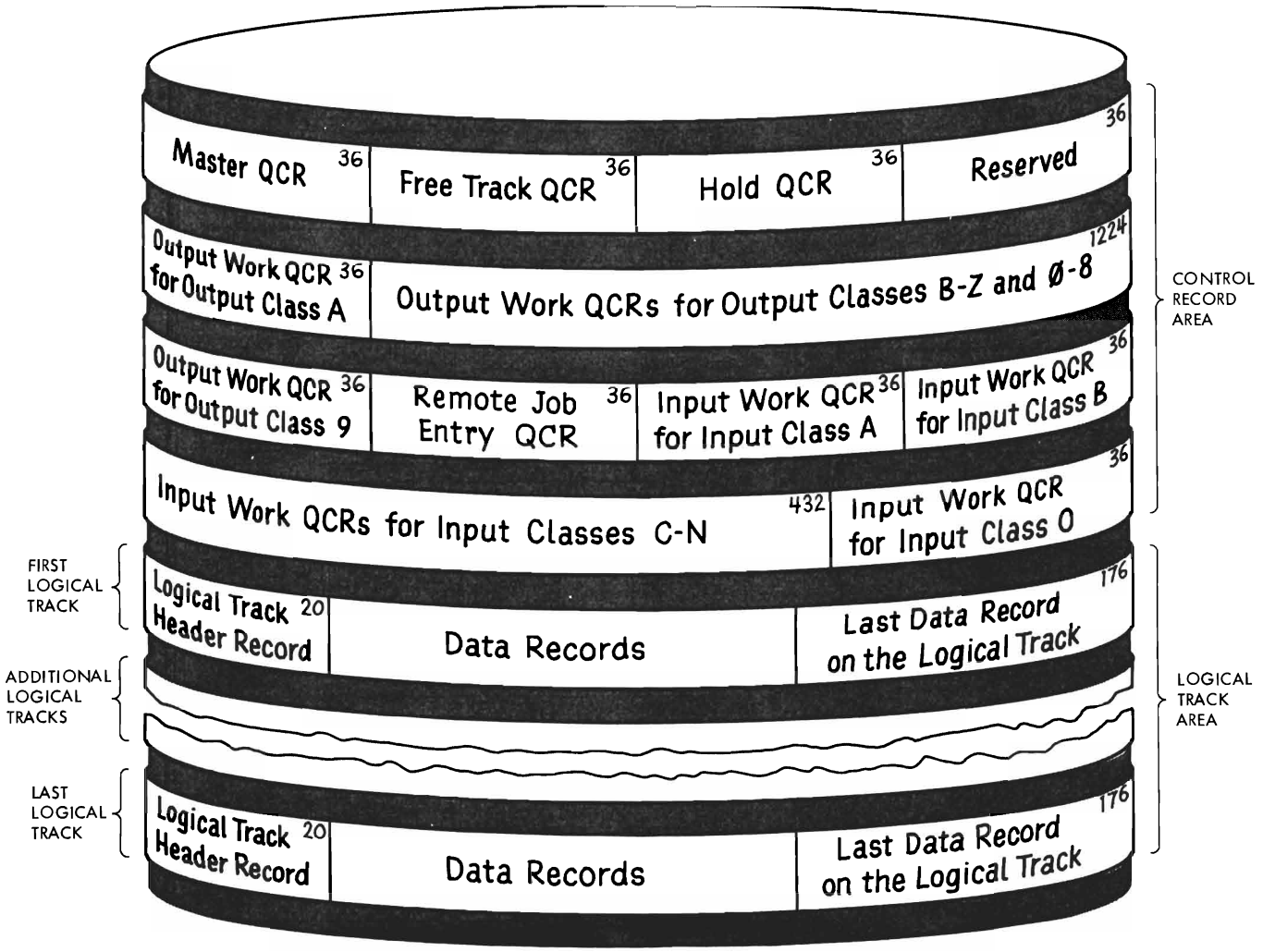
Assign/Start Routine (IEFQAGST)

The Assign/Start routine takes the first track from the available track pool and establishes it as the first track for a job. The queue manager parameter area (QMPA) is updated accordingly. (See IBM System/360 Operating System: MVT Job Management, Program Logic Manual, Form Y28-6660, for a description of QMPA.) An IOB and an ECB are created for subsequent input/output operations. The actual reserving of tracks is done by the assign routine, IEFQASGQ.

Note: MFT does not support the track-stacking facility of MVT.

Assign Routine (IEFQASGQ)

The assign routine assigns record space on the job queue, and determines whether the requested blocks can be assigned to the current track. If so, the record addresses are placed in the external parameter list



• Figure 20. Sample Job Queue (SYS1.SYSJOBQE) Format After Initialization

of the QMPA, and the records-available field of the QMPA is decremented to reflect this assignment. If additional logical tracks must be assigned, this routine issues an ENQ macro instruction on the master QCR to prevent concurrent access by other tasks. The master QCR is read into main storage.

The primary user of this assign routine is the reader/interpreter, although the initiator/terminator also uses it. To prevent the possibility of the reader/interpreter taking all the space and making it impossible for jobs to be initiated or terminated, two limit values have been added: the number of tracks reserved for initiating a job, and the number of tracks reserved for terminating a job.

If logical tracks are available, the requested tracks are acquired. The address of the first available logical track is updated and the newly assigned tracks are chained to the tracks assigned to the job. The master QCR is written to the control record area of the job queue data set. A DEQ macro instruction is issued to make the master QCR available to the next user.

If there are no available logical tracks, this routine passes control to the queue manager/interpreter interlock routine (IEFSD572) which issues a message to the operator requesting him to reply 'WAIT' or 'CANCEL'. If the reply is WAIT, control returns to this assign routine to wait for tracks to become available. When the system component is assigned the requested record TTRs, it can read and write records on the job queue. The master QCR is written, and a DEQ macro instruction is issued to make the master QCR available to

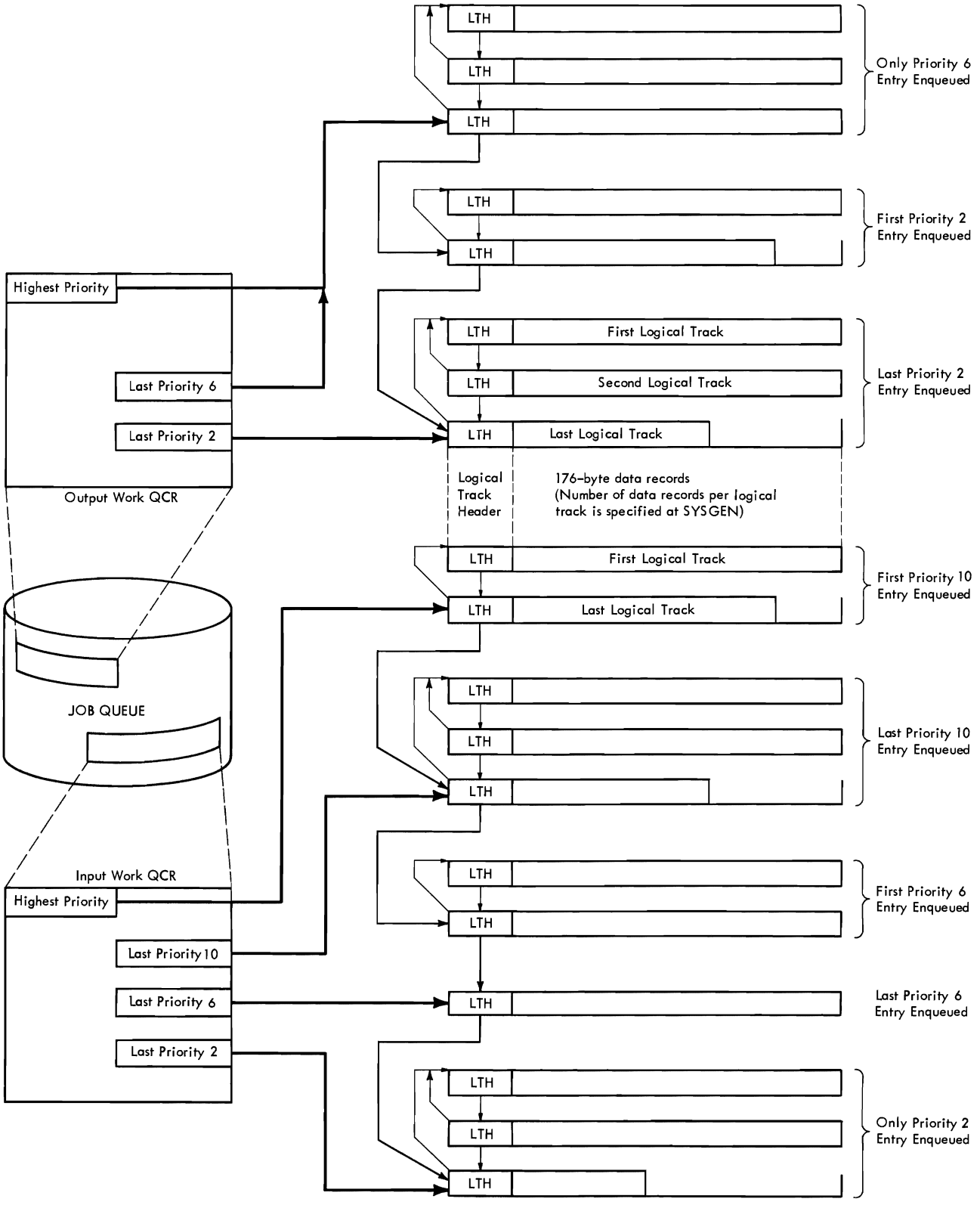


Figure 21. Input and Output Queue Entries

the next user. The record addresses in storage and TTR pointers are contained in the external parameter list of the QMPA. When available space on the job queue becomes critical, a warning is sent to the requesting task. Logical tracks are removed from the pool of available tracks and assigned to the job.

If the reply is CANCEL, the interlock routine deletes all queue space assigned to the job, cancels the job, and returns control to the assign routine. Normal initiator operation recovers the partition for further use.

Interpreter/Queue Manager Interlock Routine (IEFSD572)

When the reader/interpreter requests tracks for the job it is processing, and nc space is available, IEFQASGQ passes control to interlock routine IEFSD572 to identify whether an interlock can occur. If the reader is transient, the possibility exists that space needed by the reader/interpreter can be provided only by the termination routines, which must operate in the partition that the reader occupies. Because the requested space is not available, the routine issues a message to the operator requesting a reply of 'WAIT' or 'CANCEL'. If the reply is WAIT, this routine returns to the assign routine to wait for available space. (If the reader requesting space is a resident reader, no message is issued, and a reply of WAIT is assumed.)

If the reply is CANCEL, control passes to the delete routine (IEFQDELQ) to delete all queue space assigned to the job being processed (if any space had already been assigned). When control returns, the interlock routine abnormally terminates the job with a job-canceled code of 222. Normal initiator operation recovers the partition for further use.

Queue Manager Enqueue Routine (IEFQMNQQ)

After all control blocks for a job have been written, the job is eligible for selection by an Initiator. Declaring a job ready for selection (enqueuing) is done by Queue Manager Enqueue routine IEFQMNQQ.

When an interpreter has completed the processing of a job, (all records generated by the interpreter have been written on the queue), it uses this routine to enqueue the job, in priority order, on the appropriate job class input work queue. When a job completes processing, the terminator uses this routine to enqueue output data sets, in priority order, on the appropriate output work queues.

To prevent concurrent updates, this routine issues an ENQ macro instruction for the queue control record (QCR) of the proper queue. When the QCR becomes available, it is read into main storage. The enqueue routine then places the new queue entry after the last entry with the same priority as shown in Figure 21. The address of the new entry is then placed in the track header of the prior entry (maintaining a chain), and in the QCR position for that priority. The job control table (JCT) is written. The updated QCR is written on the job queue. A DEQ macro instruction is issued making the QCR available. Control is then returned to the calling routine.

Dequeue Routine (IEFQMDQQ)

In addition to dequeuing a job from the input queue for an initiator, the dequeue routine (IEFQMDQQ) removes the output data from an output queue for processing by a system output writer.

The routine issues an ENQ macro instruction on the QCR of the selected queue. When the QCR becomes available, the dequeue routine reads it into main storage. The QCR is examined for a job belonging to the same job class as the partition. Upon finding a job, this routine adjusts the chain. If none is found, the requesting task tries the next job class. If no work is found on any of the selected queues (up to three), the requester places itself in a wait state. In the case of an output writer, a pointer to the "no work" ECB is placed in the QCR. If a pointer already exists, the ECB is chained to the last ECB waiting for that output class. Then the updated QCR is written and a DEQ macro instruction is issued making the QCR available.

Once a job has completed processing, or the output writer has written all records for a job, the tracks are returned to the system. This is known as deleting a job and is handled by the queue manager delete routine IEFQDELQ.

Delete Routine (IEFQDELQ)

The Delete routine first issues an ENQ macro instruction on the master QCR of the free chain of tracks. After control is returned, the record is updated to reflect the new available tracks. The prior last track of free storage is updated to point to the new set of free tracks. After the master QCR is updated, it is written and a DEQ macro instruction is issued against it. The ECB indicating wait-for-space is posted.

Table Breakup Routine (IEFSD514)

When a reader must be suspended, the job scheduler must prevent the destruction of variable size tables in main storage. To do this, it calls the queue manager table breakup routine, IEFDS514, (Chart 10) which subdivides tables in main storage and writes them on disk as 176-byte data records. The data records are written in a queue entry related to the caller. The job scheduler calls IEFSD514 to retrieve the 176-byte data records and to reconstruct the tables in main storage. Whether reading or writing tables, the caller must build a parameter list (see Figure 22) and place the address of the list in general register 1 before calling the TBR.

When the tables are written initially, the TBR parameter list must contain the address of a QMPA specifying the queue entry into which the tables are to be written. The function code field (QMPOP) of QMPA must specify a write operation. The TBR parameter list must also contain the address, subpool, and size of each table to be written. The last word of the TBR parameter list must be zero. The TBR returns a Head TTR address which locates the beginning of the tables on disk. This TTR must be saved for subsequent retrieval of the tables.

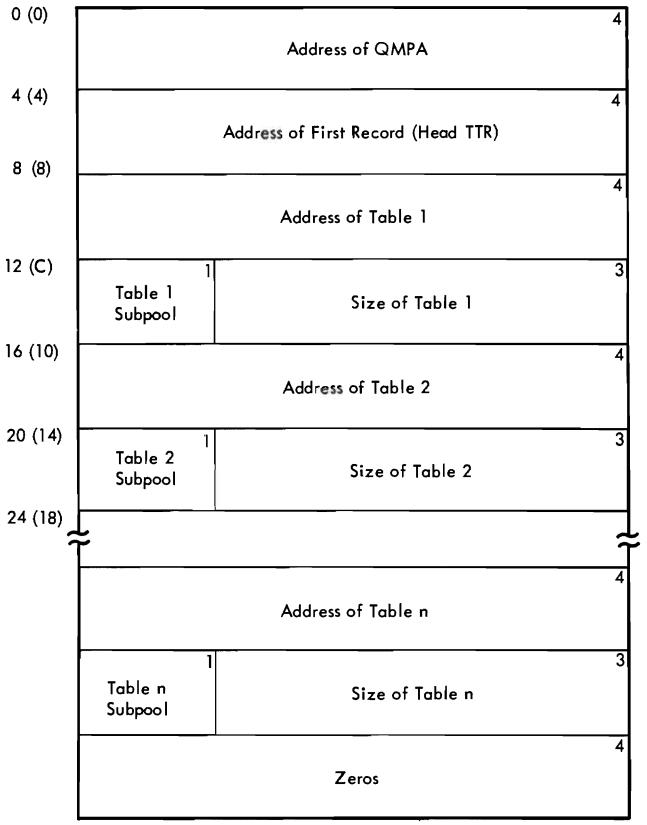


Figure 22. Table Breakup Parameter List

The initial write establishes disk data records for the tables for the duration of the associated queue entry (i.e., until the entry is deleted). Therefore, further write requests must specify the Head TTR in the TBR parameter list. Before issuing a write request, the caller must retrieve any previously written tables to prevent their being overlaid by the new write request.

If the request is for output of tables, (transferring from main storage to direct-access device), the Head TTR (passed in the parameter list) is used to read the first table queue control record (TQCR). If the Head TTR is zero, the assign routine, IEFQASGQ, is called to assign space for a new TQCR. The TQCR is a 176-byte record containing a 4-byte forward-chain pointer and space for 43 TTRs. These spaces are filled in as the tables are written, using the assign routine to assign the TTRs, and the Read/Write routine, IEFQMRAW, to write the tables in 176-byte segments. If more than 43 records are required to hold the tables, a new TQCR is chained to the first, and processing continues. The low-order byte of the last TTR used in writing the tables is set to 'FF' (hexadecimal) to indicate end-of-tables. After these TTRs are assigned, they are used each time the table breakup routine is called to write tables, as long as the Head TTR is preserved by the caller.

Once a queue entry has been deleted, a caller must issue another initial write request (Head TTR is zero in the table breakup routine parameter list) to establish a new string of table data records. IEFSD514 does not free table storage areas.

In retrieving tables, the TBR parameter list must contain the address of an associated QMPA. The function code (QMPOP) field must specify a read operation. The TBR parameter list must also contain the Head TTR address. Sufficient space must be allowed for the TBR to return the new main storage address of each table, and the subpool and size of each table as specified when they were written by the TBR.

If the request is for input (reading into storage) of tables, the first TQCR is read into storage using the Head TTR passed in the parameter list. The first record of the first table is read, using the first record in the TQCR. This record contains the size of the table and the number of the desired subpool. IEFSD514 issues a GETMAIN specifying the subpool and the amount of storage required for the table. The remainder of the table is then read into the storage obtained, using read/write routine IEFQMRAW. Each table specified in the parameter list is processed in this manner until 'FF' (hexadecimal), indicating end-

of-tables, is found. As each table is read into main storage, the parameter list is updated with the main storage address of that table. When all tables have been read, control is returned to the caller. The address of the updated parameter list is returned in register 1. Tables are always written in the same sequence that they appear in the TBR parameter list, beginning with the Head TTR. They are retrieved, totally, in the same sequence; they cannot be read selectively.

READER/INTERPRETER

MFT uses the MVT reader/interpreter (reader). However, because of job class, possible MFT interlocks, and the capability of using transient readers, some modifications have been made to the MVT modules, and six new modules have been added. These modifications and additions are described below.

MFT allows as many as three input readers to execute concurrently with problem programs and writers. Resident readers operate in previously defined reader partitions, and transient readers operate in problem program partitions large enough to accommodate them. Input stream data for the step being read is transcribed onto direct-access storage where it is held until execution of the associated job begins. Problem programs retrieve this data directly from the storage device.

In MFT there are three types of system input readers:

- Resident reader
- User-assigned transient reader
- System-assigned transient reader

Resident and transient readers may operate in the same system, provided no more than one system-assigned reader is specified, and the total number of readers does not exceed three. The primary difference between the user-assigned and system-assigned transient readers is the manner in which the transient reader resumes operation after it is suspended.

RESIDENT READERS

A resident reader operates in a partition designated as such at system generation (by replacing the job class identifier with R), or during system initialization or partition definition (by specifying RDR for the job class identifier). A resident reader reads its input stream, enqueuing jobs until the input stream reaches end-of-file or until it is terminated by a STOP command entered for that partition.

Note: The STOP command does not take effect until the current job is completely read.

TRANSIENT READERS

A transient reader operates in a problem program partition large enough to accommodate it. A transient reader can be terminated by issuing a STOP command or by reaching end-of-file, as can the resident reader. In addition, a transient reader is suspended when a job is enqueued either for the partition occupied by the reader, or for a small partition. (Note that this is possible only when a reader completes reading an entire job.)

If a transient reader is started in a specific partition by including the partition assignment in the START command, it always resumes operation in that same partition, and only when that partition becomes free. This type of transient reader is referred to as user-assigned. If 'S' is substituted for the partition number in the START command, the system assigns the reader to any available large problem program partition. This type of transient reader is called system-assigned.

READER CONTROL FLOW

After a START command is entered to activate a reader, master scheduler routine IEECIR50 determines if the size of the requested partition is large enough, and posts the partition. Job selection routine IEFSD510 determines that a START command has been entered, and passes control to system task control (STC) syntax check routine IEEVSTAR. The syntax check routine validates the syntax of the START command, builds job control language tables, and retrieves the reader cataloged procedure specified in the START command. Control is then passed to interface routine IEFSD533 which sets up an interpreter entrance list (NEL) for a reader. It also allocates job queue space for a transient reader by issuing a dummy WRITE macro instruction. Control is then passed to linkage module IEFSD537 which issues a LINK macro instruction to reader initialization routine IEFVH1 to begin reading the input job stream (Charts 11-15).

Each reader is assigned to an input device specified in the START command.

When the reader initialization routine receives control, it reads its input stream using QSAM, and translates job processing information into convenient form for subsequent processing by an initiator and system output writer. Each job read in by the readers is converted into tables that are placed in the appropriate job class input work queue specified by the CLASS parameter on the JOB statement. One input work queue exists for each of the fifteen problem program job classes (A through O).

After the reader has completed reading a job, control passes to the queue manager enqueue routine, IEFQMNQQ, which enqueues the job on the appropriate input work queue according to the PRTY parameter on the JOB statement (see "Queue Management" in this section).

Note: If the reader is being used as a subroutine by a problem program, it does not enqueue the job on the input work queue, but returns control to the problem program passing the addresses of the JCT constructed for that job, and the QMPA associated with that input queue entry.

If data is encountered in the input stream, control is passed to the interpreter CPO routine (IEFVHG) to transcribe the data onto direct-access storage for later retrieval by the problem program. If there is no space for the data, control passes to the interpreter operator message routine (IEFSD536) to issue a DISPLAY active command and a WTOR message. The operator replies with either 'WAIT' or 'CANCEL'. If 'WAIT' is specified, the reader waits for space to become available. If 'CANCEL' is specified, the reader is canceled and a READER CLOSED message is issued. IEFSD536 then sets indicators which cause cleanup of the current job, and control to be passed to interpreter termination routine IEFVHN to terminate the reader.

After a reader enqueues each job, control passes to the transient-reader suspend tests routine (IEFSD532). This routine decides whether to 1) terminate the reader, 2) suspend the reader, or 3) have the reader continue reading the job stream. (The decision to suspend the reader would never be made if the reader is resident.) If the reader is to be terminated, control passes to the termination routine (IEFVHN). If the reader is to be suspended, control passes to the transient reader suspend routine (IEFSD530). Otherwise, control returns to the job and step enqueue routine (IEFVHH) to continue reading the job stream.

Transient Reader Suspend Routine (IEFSD530)

When a transient reader is suspended, transient reader suspend routine IEESD530 (Chart 16) writes the tables and work areas used by the reader onto the work queue data set (SYS1.SYSJOBQE).

The routine closes the reader and procedure library. Data needed to restore the reader is temporarily saved in the interpreter work area (IWA). The IWA is then written to the work queue data set. When a user-assigned transient reader is suspended, the address of the reader space on the work queue is placed in the partition information block (PIB). When a system-assigned transient reader is suspended, the address of the IWA is placed in the master scheduler resident data area (IEFSD568). (See Appendix A for the format of IEFSD568.) The work queue data set is later used by transient reader restore routine IEESD531 to restore the reader when the assigned partition becomes available after job termination. "No work" ECBs for problem program partitions are posted (see "Job Selection"), and suspend routine IEESD530 returns control to system task control.

Transient Reader Restore Routine (IEFSD531)

Once a partition is again free for the reader, transient reader restore routine IEFSD531 (Chart 17) receives control and issues a GETMAIN for the IWA, Local Work Area (LWA), reader DCB, and procedure library DCB. The direct-access device address of the IWA is retrieved from the PIB if a user-assigned reader is to be restored, or from the master scheduler resident data area, if a system-assigned reader is to be restored. The IWA is then read in from the job queue. The TIOT is read into storage and the TCB pointer is updated; other tables and work areas necessary to restore the reader are reset from the information saved in the IWA. The reader and procedure library DCBs are opened and the reader resumes operation to start reading at the point in the job stream where it was suspended. Control is then passed to interpreter module IEFVHCB to continue reading the job stream.

INITIATOR/TERMINATOR (SCHEDULER)

To provide independent scheduling, schedulers operate in any problem program partition of sufficient size. A partition large enough to accommodate the scheduler is referred to as a "large partition." A partition not large enough to accommodate the scheduler is referred to as a "small partition". Within a given large parti-

tion, a scheduler operates independently of schedulers in other large partitions. Because small partitions cannot accommodate the scheduler, they rely on large partitions to perform their initiation, allocation, and termination operations. Sched uling for small partitions is described in "Small Partition Scheduling" in this section.

An MFT initiator (Chart 18) dequeues a job (entry) for its partition based on a job class designated for the partition. Once dequeued, the job is scheduled according to the information contained in the entry.

During allocation and termination of each job step, the allocation and termination routines place messages and output data set pointer blocks in a specified output queue. The queue entry is created by the reader/interpreter. (The output queue entry becomes input to an output writer when the job is completed.)

An initiator functions as a control program for the scheduling process, using the allocation and termination functions as closed subroutines. The MFT initiator is composed of the following routines:

- Job Selection
- Small Partition
- Job Initiation
- Data Set Integrity
- Step Initiation
- Problem Program Interface
- Step Deletion
- ENQ/DEQ Purge Routine
- Alternate Step Deletion
- Job Deletion

JOB SELECTION (IEFSD510)

The job selection routine (Charts 19-23) acts as the control routine for the MFT initiator. The routine is brought into all large problem program partitions by the master scheduler at system initialization, by the job deletion routine when a job has terminated, or by system task control when a writer has been scheduled for a small partition or a reader has been suspended.

Job selection first waits on a "no work" ECB in the PIB. This ECB is posted complete by the command processing routines, the job deletion routine, system task control, or the small partition module when a small partition needs scheduler services.

When the "no work" ECB has been posted complete, the job selection routine checks the PIB to determine if a life-of-task (LOT) block exists (see Appendix A for a description of the LOT block). If not, it creates one for the task.

Job selection then checks the PIB for a small partition information list (SPIL) pointer (see Appendix A for a description of SPIL). If one exists, scheduling is performed for the small partition by passing control to IEFSD599. If no SPIL pointer exists, the PIB is checked to determine if the partition is involved in partition redefinition; if so, appropriate action is taken to inhibit scheduling in the partition. (See "Master Scheduler Task".)

If the partition in which the initiator is operating is not part of a partition redefinition, a test is made to determine if a reader or writer (system tasks) is to be started in the partition. If a reader or writer is to be started, control passes to system task control which initiates readers and writers.

If no small partition is requesting service, no reader or writer is to be started, and the partition is not part of a redefinition operation, a final check is made to determine if a START INIT command has been issued; if so, job selection attempts to dequeue work from the input work queue (see Figure 23). If a STOP INIT command has been issued, the attempt to dequeue a job is bypassed.

A threshold check is then made to determine if enough logical tracks are available on SYS1.SYSJOBQE to start the initiator. If not, a START INIT REJECTED message is sent to the operator and job selection again waits on the "no work" ECB.

The job selection routine obtains storage for the job control table (JCT) and then uses the queue manager dequeue routine (IEFQMDQQ) to obtain work from one of the input job queues according to the job class assignment of the partition. If work is found, IEFQMDQQ constructs a CSCB for the job and an IOB to be used when reading or writing the input queue. The CSCB is constructed in the system queue area and the address of the CSCB is placed in the LCT. The address of the IOB is placed in QMGR1 When a user accounting routine is supplied, the job selection routine sets the LCT fields LCTTMWRK and LCTTMWRK+4 to zero. These fields are used in calculating the execution time of a job step. Job selection then branches to job initiation routine IEFSD511.

If the search for work for the partition is unsuccessful (i.e., no work has been enqueued for any of the job classes assigned to the partition) tests are made to determine if a transient reader is to be restored in the partition or if a START

command has been entered for a system-assigned transient reader. If so, system task control is called. If a reader is to be restored in the partition, job selection passes control to special entry point IEE534SD in system task control.

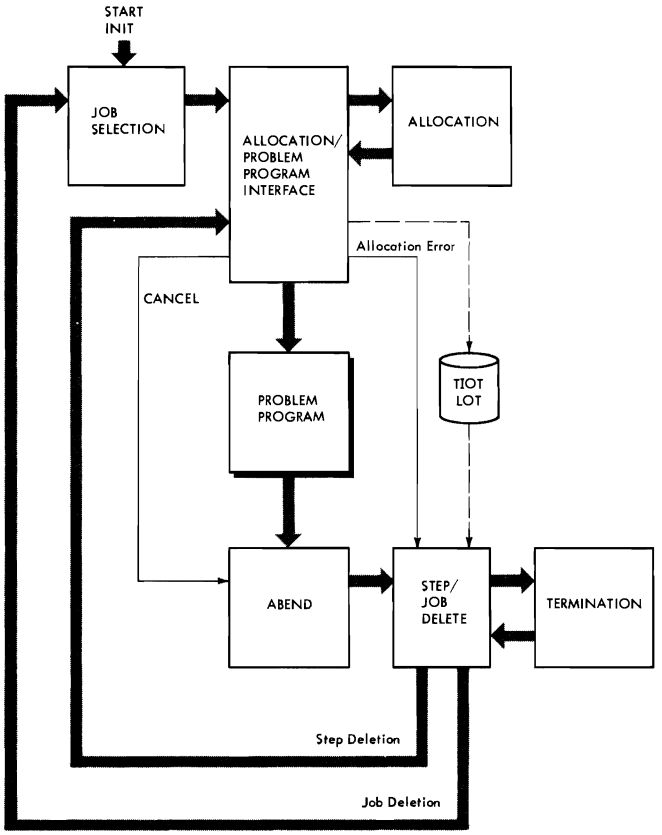


Figure 23. Scheduling a Problem Program in a Large Partition

Command Processing Services

In response to system commands entered in the input stream or from the console, the command processing routines request a service by storing information in the PIB of the affected partition or in the master scheduler resident data area for START and STOP commands issued for system-assigned transient readers and writers. The job selection routine recognizes these requests and takes one of the following actions:

- Inhibits further job scheduling for the partition in preparation for the processing of a DEFINE command. (The DEFINE command can be entered only from the console.)
- Prevents execution of problem programs in large partitions in response to a STOP INIT command.
- Passes control to system task control in response to a START reader or START writer command.
- Schedules problem program execution in response to a START INIT command.

SMALL PARTITION SCHEDULING

A partition is defined as "small" when its size is at least 8K bytes but less than the job scheduler (30K or 44K) generated for the system. Small partition scheduling is performed by an initiator in a scheduler-size partition at the request of small partition module IEFSD599 (IEFSD599 is described later in the topic "Small Partition Module"). The small partition is therefore temporarily dependent on a large partition while scheduler services are being performed. Scheduling for a small partition is independent of scheduling for other small partitions in the system.

The small partition module interfaces with job selection module IEFSD510 to schedule a problem program, or with system task control to schedule a writer in a small partition. Communication between the small partition module and job selection or system task control is maintained through a small partition information list (SPIL). (The format of a SPIL is shown in Appendix A.)

Small partition module IEFSD599 requests the scheduling function by placing the address of a SPIL in the partition information block (PIB) of each scheduler-size partition in the system. Each time job selection is entered between jobs, the PIB is checked for a non-zero SPIL address. If the PIB contains a valid address, the SPIL is analyzed, the job class queues for small partitions are searched for work, and control is passed to one of the following:

- Job Initiation (IEFSD511), if work has been found for a small partition.
- Step Deletion (IEFSD515), if a small partition is waiting for termination.
- System Task Control (IEEVSTAR), if a writer is to be started in the small partition.

These routines perform the requested service in the large partition and use the SPIL to indicate their action to IEFSD599. When the requested service has been performed, these routines return to IEFSD510.-

Initiating a Problem Program

As shown in Figure 24, initiation of a problem program in a small partition is performed by a large partition. If a small partition is waiting for work, job selection module IEFSD510 dequeues a job from an input work queue that the small partition is assigned to service. The large partition posts a completion code in field ECBA of the SPIL when initiation services have been performed.

A completion code of one indicates that no work was found for the small partition. The small partition then waits on the ECB list in the SPIL. The posting of any of the listed ECBs causes the small partition to request initiation services.

A completion code of zero indicates that initiation services have been performed and the problem program job step is ready to be executed. The small partition, using the allocate parameter list (APL), moves the task input/output table (TIOT) and life-of-task (LOT) block from the large partition, opens required DCBs, and establishes problem program mode. (If the system has the storage protection feature, the protection key is set.) If the job has not been canceled, control passes to the problem program, thus freeing the large partition to continue processing.

Initiating a Writer

As shown in Figure 25, if a writer is to be started in the small partition, small partition module IEFSD599 requests initiation of the writer by system task control. A large partition responds to the request by bringing system task control routine IEEVSTAR into the large partition. IEEVSTAR initiates the small partition to the point of calling in the writer. IEEVSTAR then posts ECBA in the SPIL with a completion code of zero to indicate to IEFSD599 that initiation services have been performed, and the writer is ready to be executed. Small partition module IEFSD599, using the link parameter list (LPL), moves the TIOT from the large partition to the small partition. ECBC in the SPIL is posted, thus freeing the large partition to continue normal processing. Problem pro-

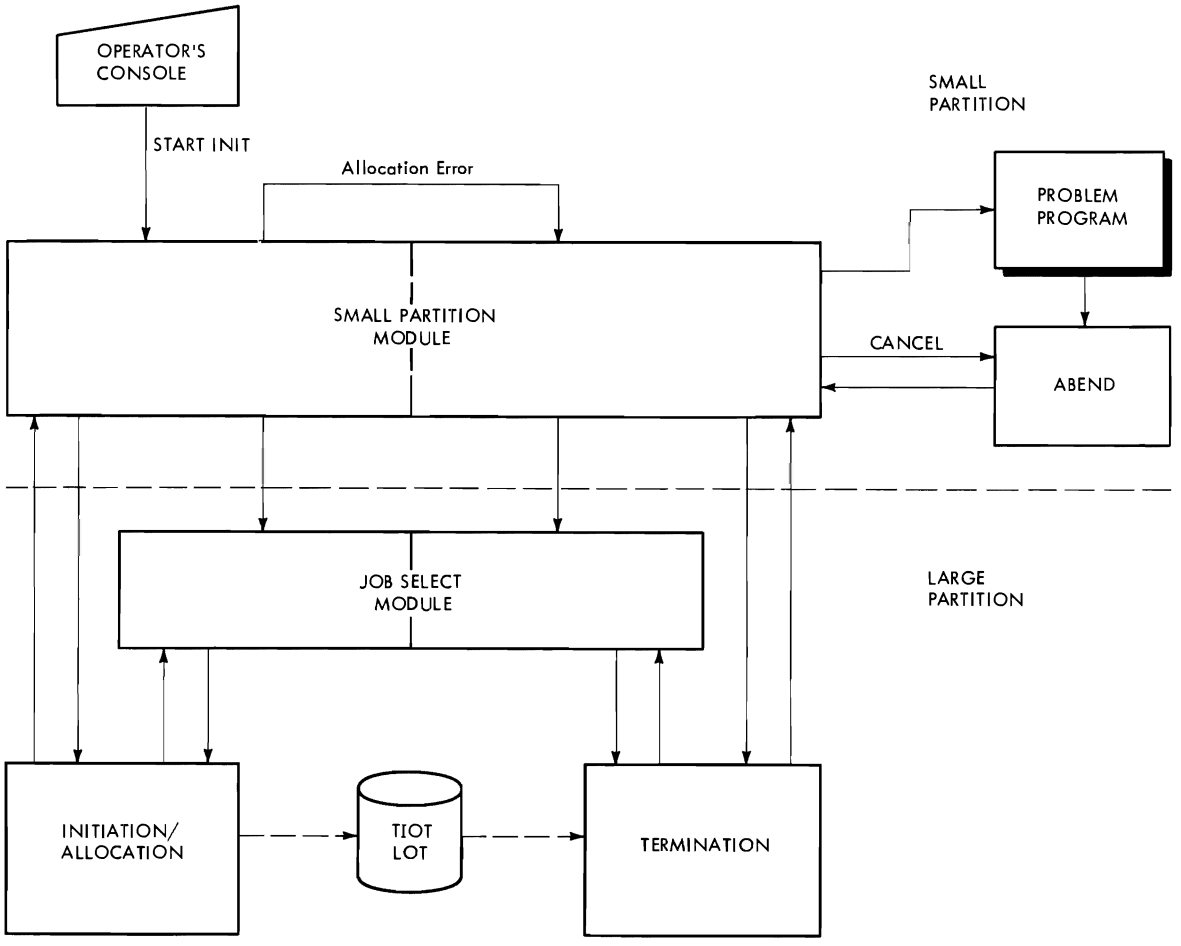


Figure 24. Scheduling a Problem Program in a Small Partition

gram mode is established, the SPIL is freed, and control passes to the writer via an XCTL macro instruction.

Terminating the Small Partition

When the job step is completed, or a writer is stopped, small partition module IEFSD599 is brought back into the partition and entered at special entry point SMALLGO. A check is made to determine whether a scheduler ABEND occurred. If it did, a message is issued to the operator with a completion code, and all CSCBs associated with that job are removed from the CSCB chain. Control then passes to the normal entry point of IEFSD599. If no scheduler ABEND occurred, the SPIL is created, and a status bit is set indicating that termination services are requested. The small partition module then begins a search for a large partition to perform the job termination services or writer end-of-job processing.

After an initiator in a large partition has performed the termination services, ECBA in the SPIL is posted with a completion code of two to indicate that job termination has taken place. A check is made to determine if the small partition is involved in a redefinition operation. If it is, the small partition is made quiescent. If the small partition is not associated with a redefinition operation, it requests additional services from an initiator in a large partition.

Note: If the initiator in a large partition performs step termination instead of job termination, the next step of the job in the small partition is scheduled before the initiator schedules a job into its partition, or before it performs scheduling services for another small partition.

Small Partition Module (IEFSD599)

Small partition module IEFSD599 (Charts 24-27) is entered at special entry point SMALLGO from either the master scheduler or redefinition routines (at system initialization), or the ABEND routines (when a step has completed execution). IEFSD599 first waits on a "no work" ECB located in the partition's PIB. When this ECB is posted

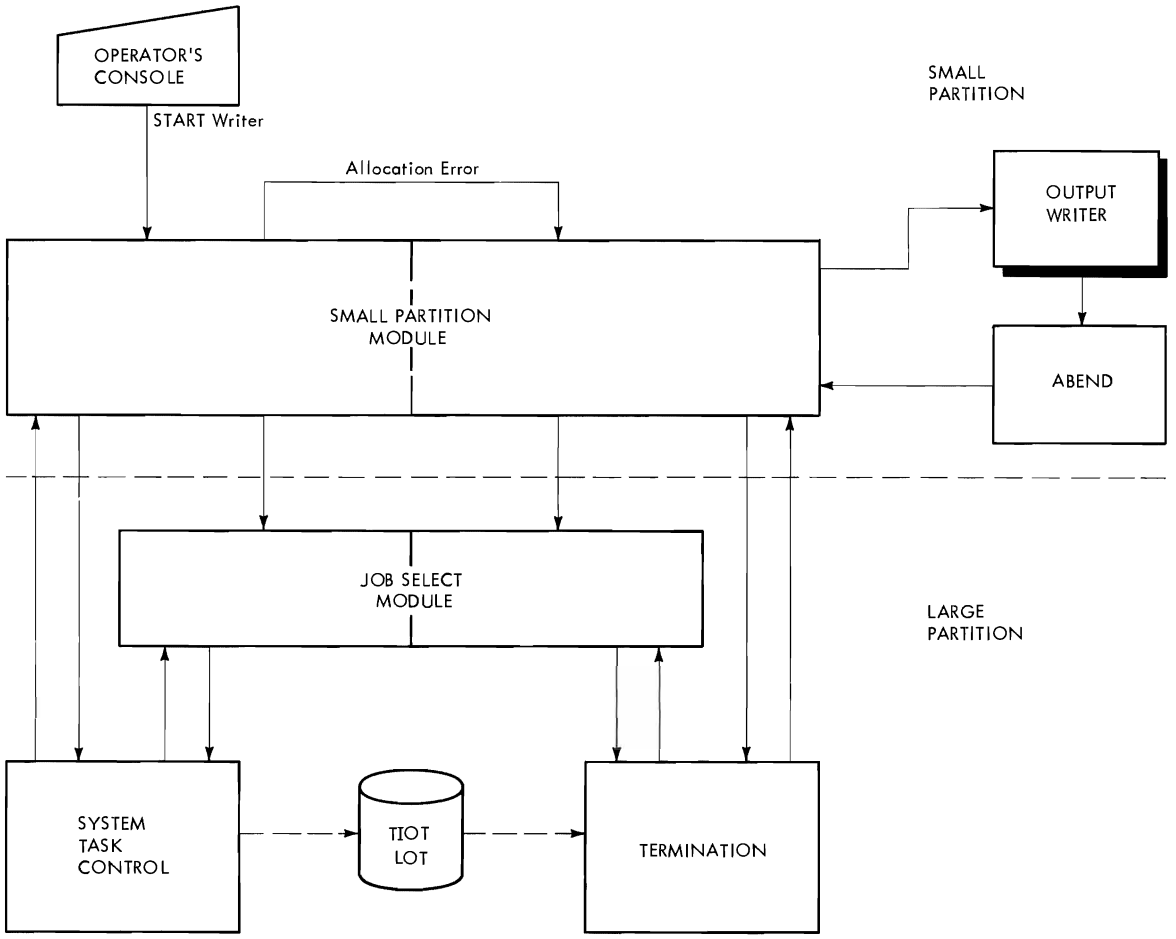


Figure 25. Scheduling a Writer in a Small Partition

complete, the PIB is checked to determine if a SPIL has been created. If not, one is created and an indicator is set in the PIB. The PIB is checked to determine if the partition is involved in a redefinition operation. If so, assigned tracks are deleted, the SPIL is freed, and pending CSCBs are freed. The 'DEFINE' ECB in the PIB is posted to indicate that the partition has been made quiescent, and a return is made to wait on the "no work" ECB.

If no redefinition operation is pending, the PIB is checked to determine if a writer is to be started in the partition. If so, an indicator is set in the SPIL, assigned tracks are deleted, and a request for scheduling is made to a large partition (described below). If a writer is not to be started, the STOP INIT bit in the PIB is checked. If this bit is on, assigned tracks are deleted, the SPIL is freed, and a return is made to wait on the 'no work' ECB. If the STOP INIT bit is not on, the PIB is checked for track assignment. If needed, tracks are assigned and indicated in the PIB. The SPIL is updated to indicate a request for initiation of a problem program.

A request is made for a large partition to service the small partition based on the contents of the SPIL. First, an exclusive ENQ macro instruction is issued to prevent concurrent service requests by small partitions. Interruptions are disabled to prevent interference with the address of the SPIL in the large partition's PIB. IEFSD599 then searches for a scheduler-size partition. The TCBs are tested for problem program status; when a scheduler-size partition is found, a determination is made of whether the small partition is involved in a DEFINE operation.

If the small partition is involved in a DEFINE operation, the test for the large partition involved in a DEFINE operation is bypassed. If the small partition is not involved in a DEFINE operation, the large partition is tested to determine if it is involved in a DEFINE operation. If so, the large partition is bypassed and the TCB search is continued.

The address of the SPIL is stored in the PIB of the large partition, thus constituting a request. An indication is made when storing occurs. If a large partition is waiting on its 'no work' ECB (in its PIB), SPIL addresses are cleared in all other large partition PIBs, and the large partition is posted. When a large partition is posted, or all applicable TCBs are checked, interruptions are enabled.

If no SPIL pointers were stored during the search, a DEQ macro instruction is issued (to allow other small partitions to make requests), and a WAIT macro instruction is issued on a 'dormant' ECB in the small partition's PIB. (When later posted by the command processing routines, the small partition module will repeat its search). If at least one SPIL pointer was stored, a WAIT macro instruction is issued on ECBB in the SPIL. This allows a large partition, immediately upon recognition of the request, to post the ECB complete. The small partition module may then issue a DEQ macro instruction to release the SPIL pointer field so other small partitions may make requests.

Next, a WAIT macro instruction is issued on ECBA (in the SPIL) to delay the small partition until the requested service has been performed. When ECBA is posted complete by the large partition, the completion code is tested to determine the action which occurred. If the completion code is two, job termination occurred and return is made to the point of determining the DEFINE status of the small partition. If the completion code is one, 'no work' was found for the small partition and a return is made to WAIT on the ECB list in the SPIL. If the completion code is zero, the large partition is at the point of calling either the problem program or a writer. The large partition is waiting on ECBC (in the SPIL) to allow transfer of information into the small partition by the small partition module.

If a problem program is to be initiated, IEFSD599 uses the allocate parameter list (APL) to move the TIOT and user parameter area into the small partition. It then posts ECBC (freeing the large partition), and opens Fetch and/or JOBLIB DCBs if required. The partition is established in problem program protection mode. The SPIL is freed.

A check is made to determine if the job has been canceled. If so, an ABEND macro instruction is issued. If the job has not been canceled, an XCTL macro instruction is issued to call the problem program into the small partition (the problem program passes control to ABEND at completion of its execution).

ABEND recalls small partition module and enters at special entry point SMALLGO. The small partition protection key is changed to zero and a SPIL is created. A termination request is indicated in the SPIL, and IEFSD599 begins the search for a large partition to service the request.

If a writer is to be initiated, the control flow is the same as described above in "Initiating a Writer".

INITIATOR/TERMINATOR CONTROL FLOW

In addition to IEFSD510 and IEFSD599, several other initiator routines are unique to MFT. These are described in the following paragraphs. Included also are the MVT modules that have been modified by MFT. Descriptions of the MVT allocation and step initiation routine then passes control to termination modules that have not been modified by MFT can be found in IBM System/360 Operating System: MVT Job Management, Program Logic Manual, Form Y28-6660.

Job Initiation Routine (IEFSD511)

Job initiation routine IEFSD511 issues a GETMAIN specifying subpool 0 to obtain space for the system output class directory (SCD). The SCD is then read into the area and the contents of the SCD are used to initialize QMGR2 in the LOT block. (QMGR2 is the queue manager parameter area which is used for referencing the output data set.) After QMGR2 has been initialized, the storage obtained for the SCD is freed. A GETMAIN is then issued to obtain storage for IOB2, the IOB used in conjunction with QMGR2. A GETMAIN is issued (specifying subpool 253) to obtain space for the step control table (SCT). The SCT is read into the area thus obtained. Job initiation then branches to data set integrity routine IEFSD541.

Data Set Integrity Routine (IEFSD541)

The data set integrity routine is entered only once per job, from job initiation routine IEFSD511. It first determines whether data set integrity processing is required.

If the JCT indicates a 'failed' job or if there are no explicit data sets (DSNAME parameter in a DD statement) for the job, processing is bypassed and exit is made to step initiation routine IEFSD512. If data set integrity processing is required, the DSENQ table records are read from the job's entry in the input job queue (SYS1.SYSJOBQE). Duplicate DSNAMEs are eliminated from the table and each unique DSNAME is placed in a minor name list. The most restrictive attribute (exclusive or share) is chosen for each DSNAME placed in the minor name list. After this processing is complete, an ENQ supervisor list is constructed which contains an entry for each DSNAME in the minor name list. Each entry is initialized with the following:

- RET=TEST option of ENQ
- SYSTEM option of ENQ
- Attribute (E/S) of the corresponding DSNAME
- Address of the common major name 'SYSDSN'
- Address of the corresponding DSNAME (considered the minor name) in the minor name list

The DSNAME (minor name) length is contained in the first byte of each DSNAME field in the minor name list.

When the ENQ supervisor list is constructed, the system is disabled and an ENQ supervisor call is issued against the list to test the availability of the DSNAMEs. If the DSNAMEs are available, the ENQ supervisor list is updated so that each entry reflects the RET=NONE option of ENQ. A second ENQ supervisor call is issued against the list to reserve DSNAMEs for the job. The system is enabled and exit is made to step initiation routine IEFSD512.

If the DSNAMEs are unavailable for the job (already reserved with conflicting attributes by other task(s) in the system), the operator is notified of the condition. In notifying the operator, the return code field of each entry in the ENQ supervisor list is tested for a non-zero setting. If the setting is non-zero, the associated DSNAME (minor name) is identified to the given the following reply options:

- RETRY, in case the resources have been freed by the other task(s) (processing is delayed until the operator replies)
- CANCEL the job.

If RETRY is entered by the operator, processing continues at the initial ENQ supervisor call to again test the availability of the DSNAMEs. The operator is again notified, and he can reply either RETRY or CANCEL. If the job is canceled by the operator, the 'job fail' bit in the JCT is set and exit is made to step initiation routine IEFSD512.

Step Initiation Routine (IEFSD512)

Step initiation routine IEFSD512 first issues a GETMAIN specifying subpool 253 to obtain storage for an allocate register save area (ARSA) and an allocate parameter list (APL). The APL (Figure 26) is initialized containing addresses of the LOT, JCT, and SCT, and two words of zeros. The step initiation routine then passes control to allocation via a LINK macro instruction. Allocation returns the addresses of a task input/output table (TIOT) list (which points to the TIOT) in the first word of zeros in the APL. On return from allocation, the return code is tested to determine if allocation was successful. If not, step initiation branches to alternate step

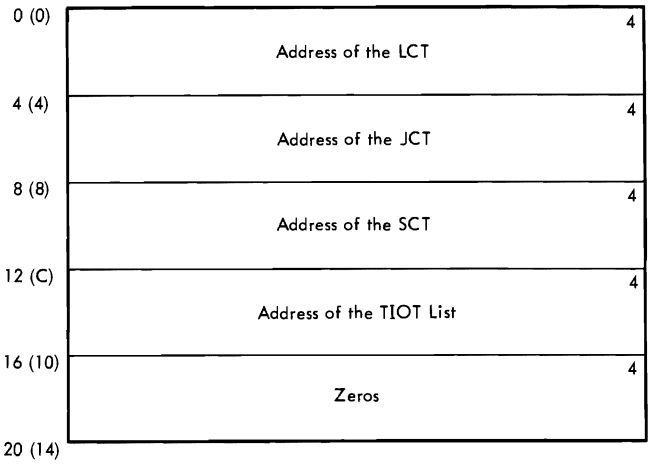


Figure 26. Allocate/Terminate Parameter List

deletion routine IEFSD516 via an XCTL macro instruction. If allocation was successful, the ARSA is freed, and the "step started" bit in the SCT is turned on. The address of the job's CSCB is stored in the APL (in the last word of the list).

Step initiation then uses queue manager read/write routine IEFQMRAW to write the JCT and SCT back on the input queue. The disk addresses of the JCT and SCT are saved in the LCT. A GETMAIN specifying subpool 253 is issued for the table breakup routine (TBR) parameter list and register save area. The TBR parameter list is initialized with the address, size, and subpool specifications for the TIOT and LOT block. The TIOT and LOT are then written into the job's entry in the job queue, and the Head TTR is saved in the JCT. The storage obtained for the TBR parameter list and register save area, IOB1, and IOB2 is freed. The JCT is then written out. Step initiation then passes control to problem program interface routine IEFSD513 via an XCTL macro instruction.

Problem Program Interface Routine (IEFSD513)

The problem program interface routine prepares the partition for execution of the job step. A test is made to determine if scheduling was performed for a small partition. If so, the address of the APL is placed in the SPIL, ECBA in the SPIL is posted to indicate that scheduling is complete, and a WAIT is issued on ECBC. This WAIT allows the small partition module to copy tables and work areas into the small partition. When the tables have been copied, ECBC is posted complete, and the interface routine frees all storage obtained for tables and work areas except for the LOT block, which is retained. The address of the LOT block is placed in register 1 and this routine passes control to job selection, IEFSD510, via an XCTL macro instruction.

If scheduling was not performed for a small partition, a test is made to determine if the job has been canceled. If so, exit is made by issuing an ABEND macro instruction.

If the job has not been canceled, the LOT block is freed, the TIOT is moved to the lowest possible location (subpool 0) in the partition, and a GETMAIN macro instruction specifying subpool 253 is issued for the user's parameter list (UPL). The UPL (Figure 27) is initialized from the SCT. Another GETMAIN macro instruction (subpool 253) is issued to create a register save area for the user's problem program. If STEPLIB, JOBLIB, and/or FETCH have been specified, their DCBs are created (but not opened) in subpool 253. The JCT, SCT, and APL are now freed, the STEPLIB or JOBLIB and FETCH DCBs are opened, and the TIOT is then moved to subpool 253. A single DCB is used for STEPLIB or JOBLIB, with STEPLIB overriding JOBLIB if both are present.

Note: The use of subpools, and the order in which control blocks and tables are created, moved, or deleted, follows a particular sequence even though this handling occurs within different modules. This is done to prevent fragmenting main storage within the partition.

After the TIOT has been moved to the highest available position within the partition, the task control block (TCB) is updated and the problem program's protection key is set (if the system has storage protection). The problem program interface routine then passes control to the problem program via an XCTL macro instruction.

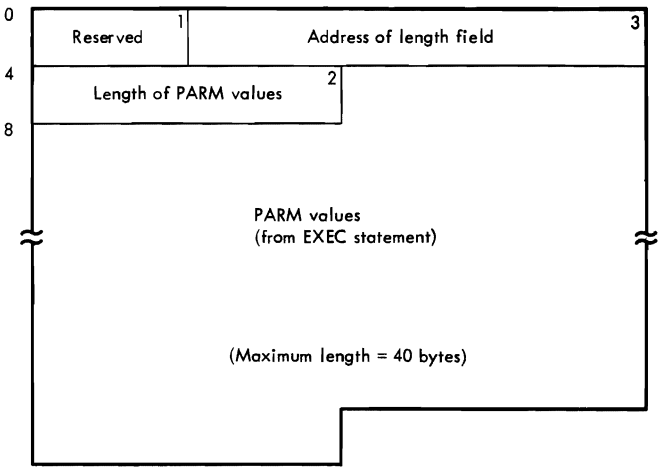


Figure 27. User's Parameter List

Step Deletion Routine (IEFSD515)

Step deletion routine IEFSD515 is entered at the end of step execution to prepare the partition for continued execution of the job, to interface with the termination subroutine, to prepare for the initiation of the next step, or to branch to job deletion if there are no more steps in the current job. When step deletion is entered, a check is made to determine whether the routine was entered due to an ABEND with the scheduler in control. If so, a message is issued to the operator and all CSCBs are removed from the CSCB chain. Control passes to IEFSD510.

If an ABEND did not occur, the step deletion routine branches to ENQ/DEQ purge routine IEFSD598 via a BALR instruction to remove any control blocks which were enqueued, but not dequeued, by the problem program step.

Step deletion then issues a series of GETMAIN requests to obtain storage for queue manager IOBs (IOB1 and IOB2), a temporary QMPA, and a register save area and parameter list for the table breakup routine. These blocks and tables are initialized and step deletion branches to queue manager table breakup routine IEFSD514, to read in the TIOT and LOT blocks for the job step. The addresses in these blocks are restored and the storage obtained for the temporary work areas is freed.

A GETMAIN (subpool 253) is issued to obtain storage for the SCT and JCT. The SCT is read into storage from the job queue, the JCT from its temporary area. Storage is obtained for a terminate register save area and a terminate parameter list. The terminate parameter list is initialized with addresses of control blocks (LOT, JCT, SCT, and TIOT list) and the step deletion routine branches to the termination subroutine via a BALR instruction. When termination returns control, step deletion frees the terminate register save area and terminate parameter list and then checks the return code.

If the return code indicates that job termination was entered, step deletion branches to job deletion routine IEFSD517. If job termination was not entered, the SCT for the next step of the job is read from the job queue, and step deletion passes control to IEFSD512 via an XCTL macro instruction.

Note: If a small partition is requesting termination, entry to the step deletion routine is made at special entry point SMALTERM. Entry at this point causes pointers to the SPIL and the small partition's TCB to be established before the step deletion routine invokes ENQ/DEQ Purge routine IEFSD598.

ENQ/DEQ Purge Routine (IEFSD598)

At job termination, this routine purges all ENQ/DEQ control blocks associated with the TCB address passed in Register 4 by the caller. If step termination was completed instead, this routine purges all ENQ/DEQ control blocks except the data set integrity blocks associated with the major name SYSDSN.

When a given resource is dequeued for the subject TCB, a task switch may occur for a higher priority requestor whose wait count becomes zero, due to availability of the resource. (This purge routine operates in a disabled state to prevent concurrent updating of the ENQ/DEQ control blocks.)

Alternate Step Deletion Routine (IEFSD516)

Alternate step deletion routine IEFSD516 is entered from step initiation routine IEFSD512 when allocation for a step has not been successful. Using the APL and ARSA (created by the step initiation routine) as the terminate parameter list and terminate register save area, this routine branches to termination subroutine IEFSD22Q via a BALR macro instruction. When control is returned from termination, the storage used for the parameter list and register save area is freed and a test is made to determine if job termination was entered. If so, this routine branches to job deletion routine IEFSD517. If job termination was not entered, the SCT for the next job step is read from the job queue and this routine branches to step initiation routine IEFSD512.

Job Deletion Routine (IEFSD517)

The job deletion routine is called at job termination to delete the job from the input queue and to prepare the partition for initiation of the next job. The routine sets the high-order byte of the LCTTCBAD field of the LCT to '80' (hexadecimal) to indicate to the ENQ/DEQ purge routine that it is job termination instead of step termination. The routine then branches to ENQ/DEQ purge routine IEFSD598 to purge the control blocks. On return from the purge routine, the high-order byte is reset to '00'.

The job deletion routine then deletes the job from the input queue, using queue manager delete routine IEFQDELQ. All areas of storage in the partition which were used for the job (except the LOT block) are freed, and the job's CSCB is freed by issuing an SVC 34. The PIB fields used for

the disk address of the TIOT and the LOT block are set to zero. If termination was for a small partition, ECBA in the SPIL is posted with a code of two (indicating job termination for the small partition). If termination was for a large partition (or after ECBA has been posted) the "no work" ECB in the PIB is posted and the job deletion routine branches to job selection routine IEFSD510.

SYSTEM OUTPUT WRITERS

MFT uses the MVT system output writer (Charts 28-29) with minor changes to five of the modules. As in MVT, the user may have up to 36 system output writers operating concurrently in the system. Each output writer can handle eight output classes; output classes may be shared by writers. However, in MFT, system output writers are classified as either resident or non-resident. A resident writer operates in its own partition. A non-resident writer operates in any problem program partition large enough to accommodate it.

RESIDENT WRITERS

Resident output writer partitions are designated in the partition information block (PIB) pointer to the TCB by setting bits one and two of the first byte to '10'. This designation is made at system generation by assigning W to the partition in place of the job class or by redefining a partition and assigning WTR to it.

A resident writer is activated by issuing a START command specifying a partition designated previously as a writer partition. A resident writer can be terminated only by issuing a STOP command specifying the device assigned to that writer.

NON-RESIDENT WRITERS

A non-resident system output writer may be started in a problem program partition large enough to hold the writer by issuing a START command specifying either that partition or by replacing the partition number with an 'S' to specify a system-assigned non-resident writer. This causes a "command pending" flag to be set in the partition's PIB.

When the writer has started, it executes in the same way as a resident writer and must be terminated by a STOP command to allow processing of problem programs to be resumed in the partition.

SYSTEM OUTPUT WRITER MODULES

The following five MVT system output writer modules are modified for MFT.

- IEFSD070 - Data Set Writer Linkage Routine
- IEFSD079 - Linkage to Queue Manager Delete Routine
- IEFSD084 - Wait Routine
- IEFSD085 - DSB Handler Routine
- IEFSD087 - Standard Writer Routine

Descriptions of all other system output writer modules can be found in IBM System/360 Operating System: MVT Job Management, Program Logic Manual, Form Y28-6660.

Data Set Writer Linkage Routine (IEFSD070)

This routine passes control to the appropriate writer routine via a LINK macro instruction. The normal linkage is to the standard writer, IEFSD087. If a special user-written output writer routine is requested, this routine passes control to that writer. Upon return from either writer, the routine passes control to data set delete routine IEFSD171 via an XCTL macro instruction which deletes the output data sets from the output queue.

Linkage to Queue Manager Delete Routine (IEFSD079)

Upon completion of a job, linkage module IEFSD079 passes control to queue manager delete routine IEFQDELQ via an XCTL macro instruction to delete all control blocks and SMBs associated with the output job from the job queue. Following deletion, the routine then posts all reader ECBs that are waiting for space to indicate that space is now available. (The reader ECB chain address is obtained from the master scheduler resident data area.) When all ECBs have been posted, control is returned to main logic routine IEFSD082.

Wait Routine (IEFSD084)

This routine serves as a multiple WAIT when there is no work in any of the output classes associated with the writer. It issues a WAIT macro instruction on the ECB list created by class name setup routine IEFSD081. When the system output writer enters a wait state, the wait routine issues a message informing the operator that the writer is waiting for work. Any posting (such as a command, or work for the writer) causes control to be given to IEFSD082.

DSB Handler Routine (IEFSD085)

DSB handler routine IEFSD085 is the setup module for printing data sets. It issues a GETMAIN macro instruction for the input DCB if it was not obtained before, and constructs a new TIOT containing an entry for the input data set. It also sets up any user-written output writer program. A check is then made to determine if a pause is required between data sets or only at forms change. If a special form is to be used, the routine writes a message to the operator telling him what form to put in the output device. The form change only occurs if the output device is unit record. This routine then passes control to linkage routine IEFSD070 via an XCTL macro instruction.

Standard Writer Routine (IEFSD087)

This routine first issues an OPEN macro instruction to open the output data set. If the data set was not opened by the problem program, no attempt is made to process the data set. After OPEN, a test is made to check for machine control characters. A switch is set that is interrogated by PUT routine IEFSD089. The writer then passes control to transition routine IEFSD088 which creates header and trailer records. Upon return from IEFSD088, the writer routine checks the CANCEL ECB in the CSCB to determine if a CANCEL command has been issued for this writer. If the CANCEL ECB has been posted complete, control passes to transition routine IEFSD088 to create a trailer record. When control is returned from IEFSD088, the writer is closed. Control is then returned to linkage routine IEFSD078 via a RETURN macro instruction.

If the writer is not to be canceled, the writer routine issues a GET macro instruction to read a record and checks for a control character. If no control character exists, the writer puts one in which causes the printer to skip one line or the punch to feed into the normal pocket. If the printer has overflowed, a skip is made to the next page.

The writer then adjusts the pointer to the record so that it points to the first data character (instead of control character) and passes control to transition routine IEFSD088 for trailer records. It then issues a CLOSE macro instruction to close the input data set, a FREEPOOL macro instruction to free the buffers, and returns control to linkage module IEFSD078 via a RETURN macro instruction.

SYSTEM TASK CONTROL

System task control (STC) (Chart 30) initiates all tasks except the initiator (START INIT). When the master scheduler determines that a START command with an identifier operand has been issued, it checks the validity of the partition specified in the command, builds and chains a CSCB, places a pointer to the CSCB in the partition's PIB, and posts the partition.

Note: If the procedure being started is for a system-assigned reader or writer, the CSCB pointer is placed in the master scheduler resident data area. (See Appendix A for the format of the master scheduler resident data area).

As shown in Figure 28, job selection module IEESD510 responds when the partition is posted, and calls STC when a START command for a reader or writer is recognized. If a reader or system output writer is to be started, STC must process a job description similar to a user's job description.

The job description information for a reader or writer comes from three sources: the procedure library, JCL statements, and the operator. The procedure library contains standard descriptions of a reader and

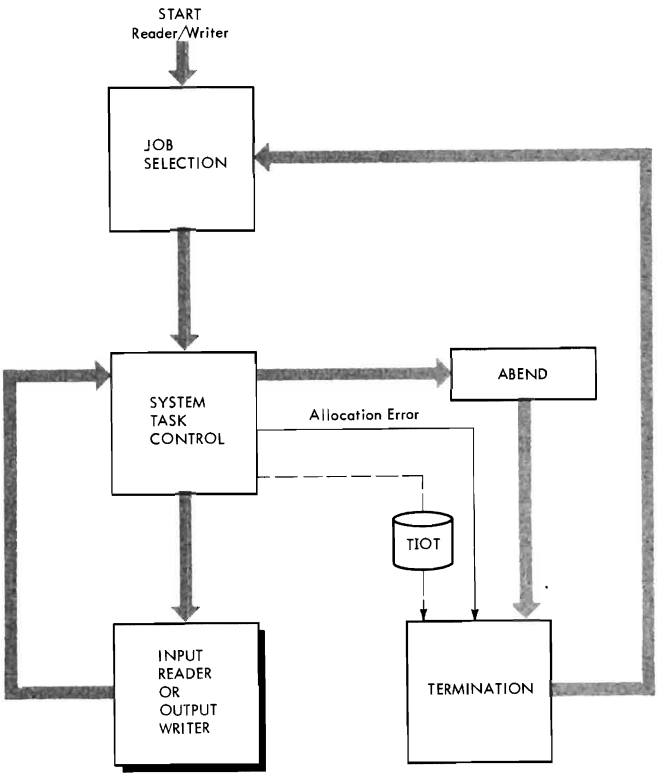


Figure 28. Scheduling a Writer in a Large Partition

writer. JCL statements (corresponding to input stream JCL) are stored internally; these statements invoke and modify the reader or writer procedure. The operator furnishes additional information in the operand of the START command; this information is edited into the internally stored JCL statements before they are used to invoke and modify the procedure.

INITIATING SYSTEM TASKS

When initiator job selection module IEESD510 determines that a START command for a reader or writer has been entered, it passes control to START syntax check module IEEVSTAR via an XCTL macro instruction.

START Syntax Check Routine (IEEVSTAR)

The START syntax check module gets main storage for, and builds, the start descriptor table (SDT). This table (see Figure 29) contains JCL statements constructed from information in the START command. These statements will be placed in the internally-stored job control language set (JCLS).

The START syntax check module passes control to JCL Build module IEEVJCL which builds the Job Control Language Set. Each statement is built in an 88-character buffer (obtained with a GETMAIN macro instruction) from information in the SDT. A pointer to the first buffer is placed in the CSCB associated with the command; each buffer contains a pointer to the next, and, in the last 80 bytes, a card image of the JCL statement.

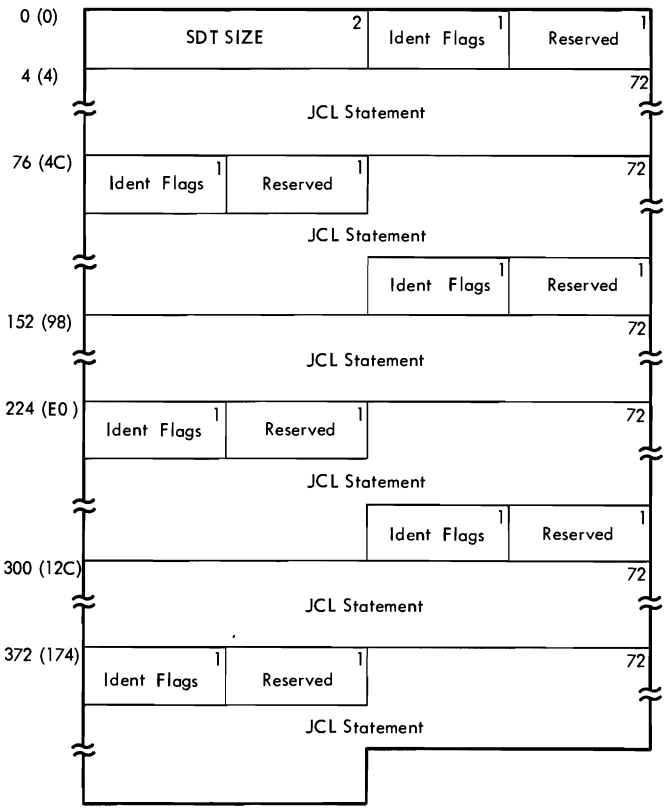


Figure 29. START Descriptor Table (SDT)

Reader Control Routine (IEEVRCTL)

Reader control routine IEEVRCTL then receives control and builds the interpreter entrance list (NEL), option list, and exit list. The interpreter entrance list contains the address of the JCLS in its third word. The reader control routine passes control to the reader via a LINK macro instruction.

The reader, used as a closed subroutine, is the same routine that performs the reading task. The non-zero value of the third word of the entrance list indicates that the input stream is an internal data set. Since the input stream is internal, the reader issues a pseudo OPEN macro instruction to bring a special access method (a modified QSAM) into storage and places a pointer to the access method in the input DCB. This special access method reads the JCLS; it is entered from the expansion of the standard GET macro instruction.

The internally-stored job control language statements, and the statements from the procedure library are analyzed and combined. The standard job description tables are built, and an input queue entry is constructed; however, because bit 7 of the option switches field of the option list is off, the entry is not enqueued, and the reader or writer "job" cannot be selected by an initiator. If errors are detected during reader processing, appropriate messages are placed in system message blocks, which are enqueued in the message class queue. When processing is complete, the reader places the main storage address of the job control table (JCT) in the NEL and returns control to the reader control routine with a code that indicates whether processing was successful. The reader control routine then passes control to allocation interface control routine IEEVACTL.

Allocation Interface Control Routine (IEEVACTL)

The reader control routine passes control to allocation interface control routine IEEVACTL, with an indication of whether the reader had encountered errors. The allocation interface control routine uses the WTO macro instruction to inform the operator of any errors that have been found. The routine then constructs the

required allocate parameter list, and passes control to the I/O device allocation routine via a LINK macro instruction.

I/O device allocation routine IEFSD21Q uses the JCT to find the appropriate tables in the input queue, allocates the necessary devices to the reader or writer, and issues any necessary mounting messages. The allocation recovery routines issue WTO macro instructions to inform the operator of any errors found during allocation. When allocation is complete, or if allocation cannot be performed, control is returned to the allocation control interface routine.

Allocation control interface routine IEEVACTL determines if the module to be given control is an authorized module and then transfers control to Write TIOT routine IEESD590.

Note: A list of "authorized" modules is contained in a table in link-table module IEEVLKNT.

Write TIOT on Disk Routine (IEESD590)

Write TIOT on disk routine IEESD590 checks that a reader has not been started in a small partition, writes the TIOT which is used for job selection, and checks for a small partition writer. If a writer is to be started in a small partition, this module issues a POST macro instruction and a WAIT macro instruction for the SPIL and then passes control to job selection routine IEFSD510 via an EXIT macro instruction. If it is not for a small partition writer, control is transferred to linkor routine IEESD591.

Linkor Routine (IEESD591)

The linkor routine passes control to the requested routine via a LINK macro instruction. When the reader or writer stops, it returns control to the linkor routine, which checks for a small partition writer. If a small partition writer returned control to the linkor routine, control then passes to IEFSD510. If a resident reader or large partition writer returned control, termination interface routine IEEVTCTL is given control via an XCTL macro instruction. If a transient reader was suspended, IEFSD591 returns to job selection routine IEFSD510.

POST Routine (IEESD592)

POST routine IEESD592 checks the CSCB to determine if it has been freed; if not, it is freed. It also checks for a small partition. The valid condition is posted in the SPIL or the PIB. The post routine then passes control to IEFSD510 via an EXIT macro instruction.

SYSTEM RESTART

The system restart functions may be requested at any time that a system restart becomes necessary; e.g., end-of-day, end-of-shift, system malfunction, power failure. This feature provides a means whereby a maximum amount of information concerning input work queues, output work queues, and jobs in interpretation, initiation, execution, or termination can be preserved. System restart permits reinitialization, rather than a complete reformatting, of the job queue data set (SYS1.SYSJOBQE).

MFT uses the MVT system restart modules. For a complete description of these modules, and how they function, see IBM System/360 Operating System: MVT Job Management, Program Logic Manual, Form Y28-6660.

This appendix contains descriptions and format diagrams of the major tables and work areas that are used by MFT job management. The tables and work areas are in alphabetical order, as shown below:

- Command Scheduling Control Block (CSCB)
- Data Set Enqueue (DSENQ) Table
- Interpreter Work Area (IWA)
- Job Control Table (JCT)
- Job File Control Block (JFCB)
- Job File Control Block Extension (JFCBX)
- Life-of-Task Block (LOT)
- Linkage Control Table (LCT)
- Master Scheduler Resident Data Area
- Partition Information Block (PIB)
- Small Partition Information List (SPIL)
- Step Control Table (SCT)
- Step Input/Output Table (SIOT)
- Task Input/Output Table (TIOT)

Tables and work areas are shown four or eight bytes wide for convenience, but are not necessarily drawn to scale. Tables that are stored in work queue entries are limited, by convention, to a length of 176 bytes.

The names of most fields are sufficient to describe the fields; those that require further explanation are described in the text accompanying the table. Where a macro instruction may be used to include a DSECT of a table in routines using the table, the name of the mapping macro instruction is also given. The displacement of each field is shown to the left of each table; the values in parentheses show the hexadecimal displacement.

COMMAND SCHEDULING CONTROL BLOCK (CSCB)

Description: A command scheduling control block (CSCB) (Figure 30) is an area for communications between the command scheduling routine (SVC 34) and the command execution routines. CSCBs are created (in the input format shown above) by several system routines. When a CSCB is created, it is placed in a chain of CSCBs by the command scheduling routine. It remains in the chain until it is deleted from the chain by the command scheduling routine, which may also free the main storage occupied by the CSCB. A CSCB is created under the following circumstances:

- A CSCB is created by the command scheduling routine each time a task-creating command is encountered. If the task is a reading or writing task, the CSCB is deleted from the chain, and its main storage released, when the task terminates.

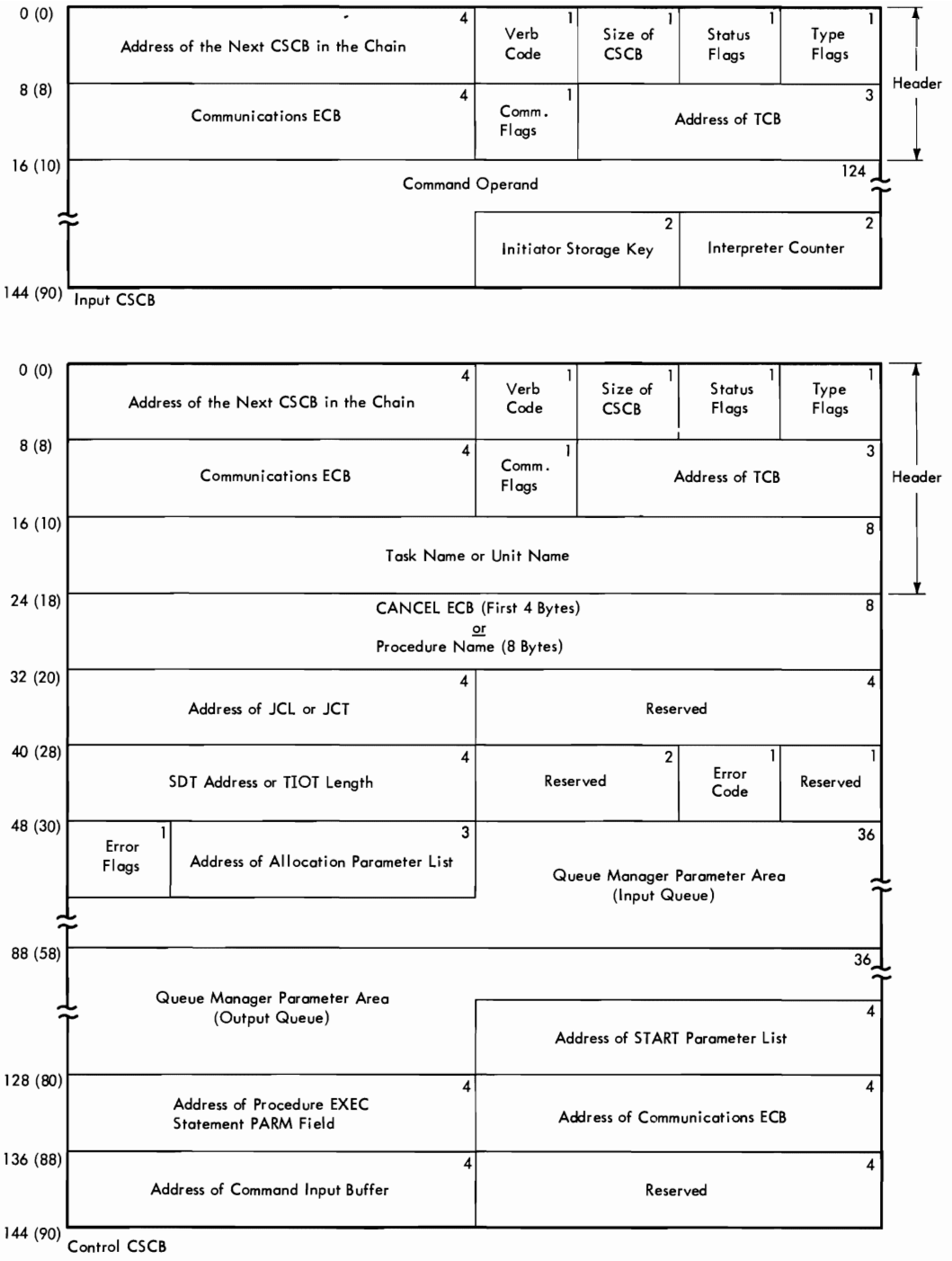


Figure 30. Command Scheduling Control Block (CSCB)

- A CSCB is created by the queue management dequeue routine each time the initiator dequeues a job. This CSCB is deleted from the chain, and its main storage released, when the last step of the job has terminated.

- A CSCB is created by a system output writer each time it encounters a DSB that was not preceded by another DSB in the current queue entry. The CSCB serves as a communication area, allowing the cancellation (by operator command) of the subtasks established by the writer. The CSCB is deleted from the chain, and its main storage released, when

the writer encounters an SMB (or the last block in the current queue entry).

A CSCB is updated (and changed to the control format shown above if necessary) by the command scheduling routine when a CANCEL jobname (job selected), CANCEL writer device, MODIFY, or STOP command is encountered.

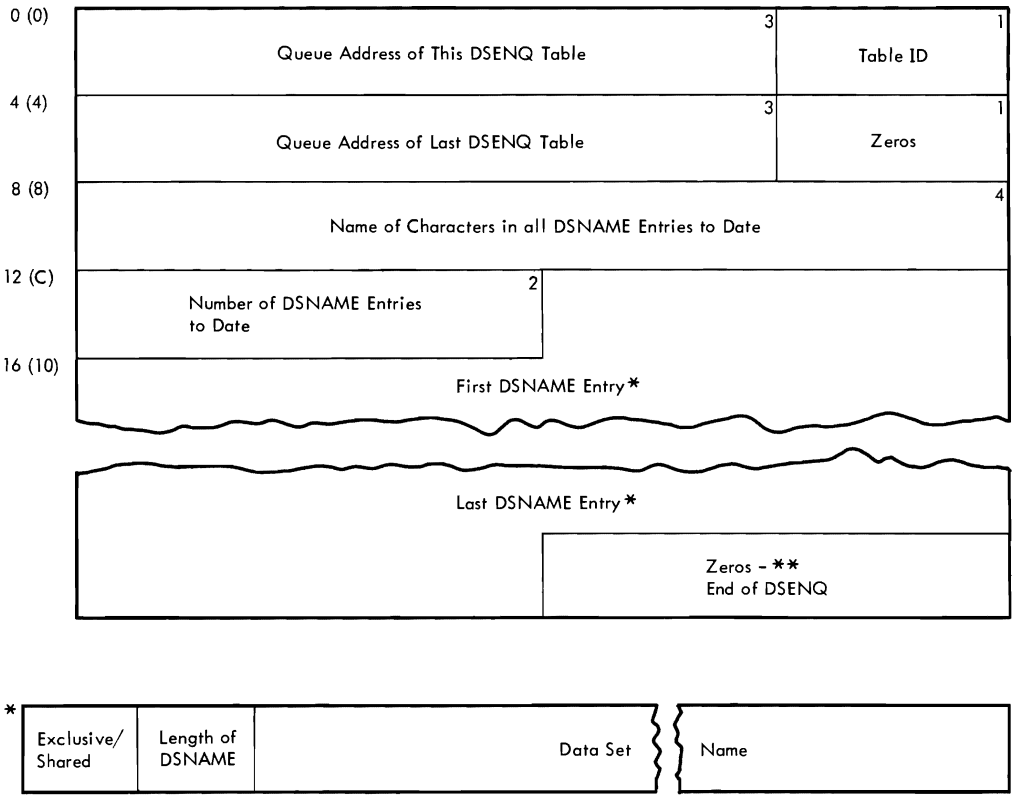
Although most of the fields are self-explanatory, the following require further description:

- Status Flags: This byte indicates the status (pending/not pending) of the CSCB, and the action to be taken by the command scheduling routine. In addition to command processing, the command scheduling routine may be entered to add the CSCB to the chain, delete it, free its main storage, or to branch to the abnormal termination routine.
- Type Flags: This byte indicates the type of activity with which the CSCB is associated.
- Communication Flags: This byte indicates the function to be performed by the command processing routine.

Mapping Macro Instruction: IEECHAIN

DATA SET ENQUEUE TABLE (DSENQ)

Description: The data set enqueue table (DSENQ) (Figure 31) is built by the DD statement processor routine of the interpreter, and is used by the initiator to construct an ENQ macro instruction parameter list to prevent



** If the last entry uses the last available space in the tables but no overflow occurs, the zero bytes are omitted.

Figure 31. Data Set Enqueue Table (DSENQ)

routines performing different tasks from using the same exclusive data sets concurrently. The table contains an entry for each data set (except temporary data sets) required for a job.

INTERPRETER WORK AREA (IWA)

Description: The 2044-byte interpreter work area (IWA) (Figure 32) is obtained from subpool zero by a GETMAIN macro instruction in the interpreter initialization module (IEFVH1). The IWA contains information used by the interpreter routines; it is the area in which job description tables are built before they are placed in the work queues.

Although most of the fields in the interpreter work area are self-explanatory, the following require further description:

- Default Parameters: The PARM field of the EXEC statement in the reader procedure contains parameters to be used when no explicit specification is made. These parameters specify whether the installation requires a programmer's name or account number on each JOB statement, the priority to be assigned to a job if no priority has been specified, whether commands in the input stream should be processed (or ignored), and the device, primary quantity, and secondary quantity to be allocated to system output data sets.

- Switches A-F: These fields contain internal switches used for communicating status information among the interpreter routines.

- System Input Allocation Table: This area contains a list of pointers to the UCBs corresponding to units available for allocation to system input data sets.

- Queue Address Table: This area contains the addresses (in TTR form) of the next two records assigned to the job's input queue entry, and the addresses (in TTR form) of the first joblib SIOT, the first scan dictionary record, and the DD override table.

- Input Stream Parameter List: This area describes the statement last encountered in the input stream, and contains a pointer to the field currently being processed.

- Procedure Library Parameter List: This area describes the statement last read from the procedure library, and contains a pointer to the field currently being processed.

- Procedure Library Merge Control Data: This area contains information used in merging statements from the input stream with statements from the procedure library. The information includes the statement names, the step names, and the names of the previous and next procedure steps.

Mapping Macro Instruction: IEFVMIWA

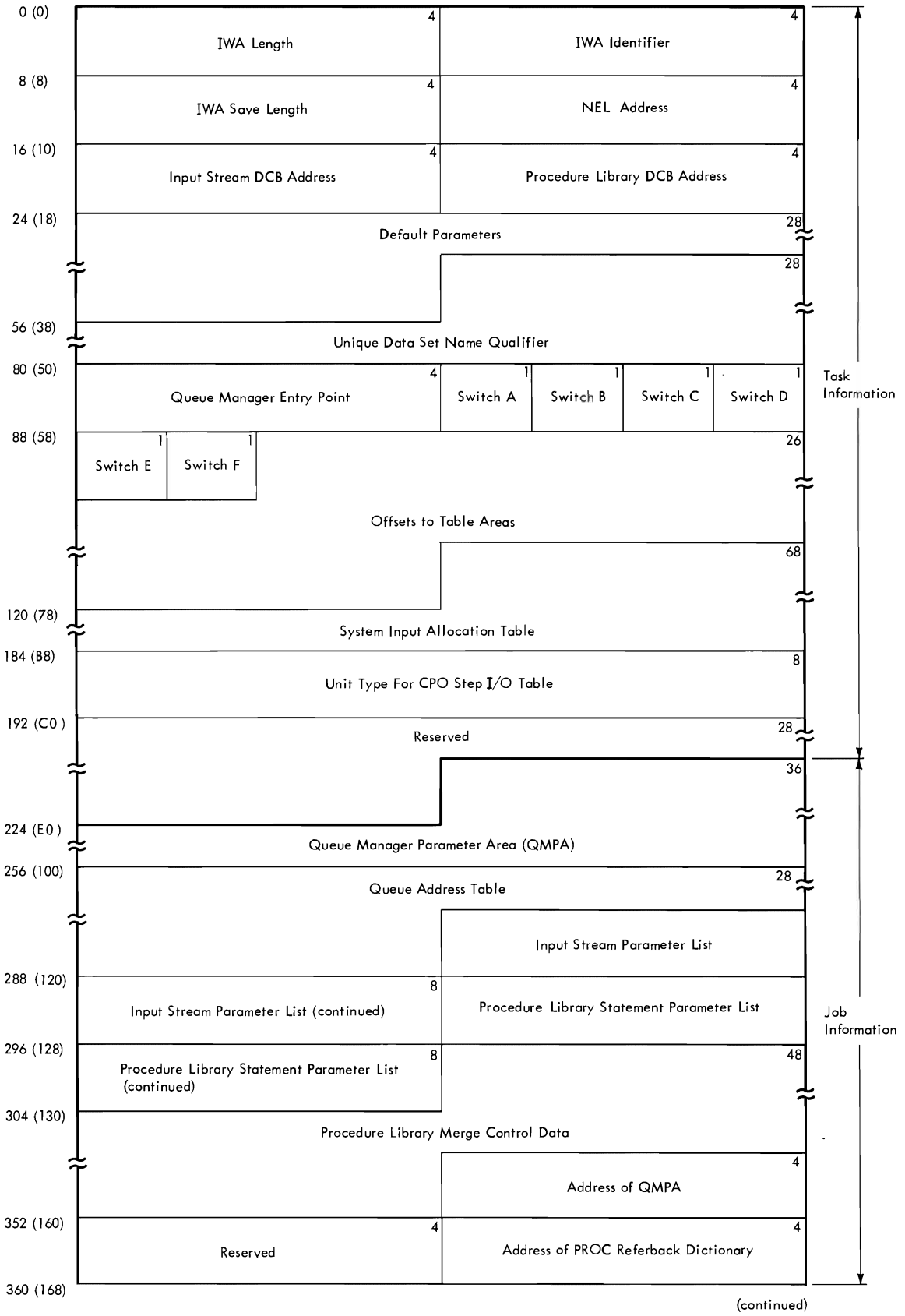


Figure 32. Interpreter Work Area (IWA) (Part 1 of 2)

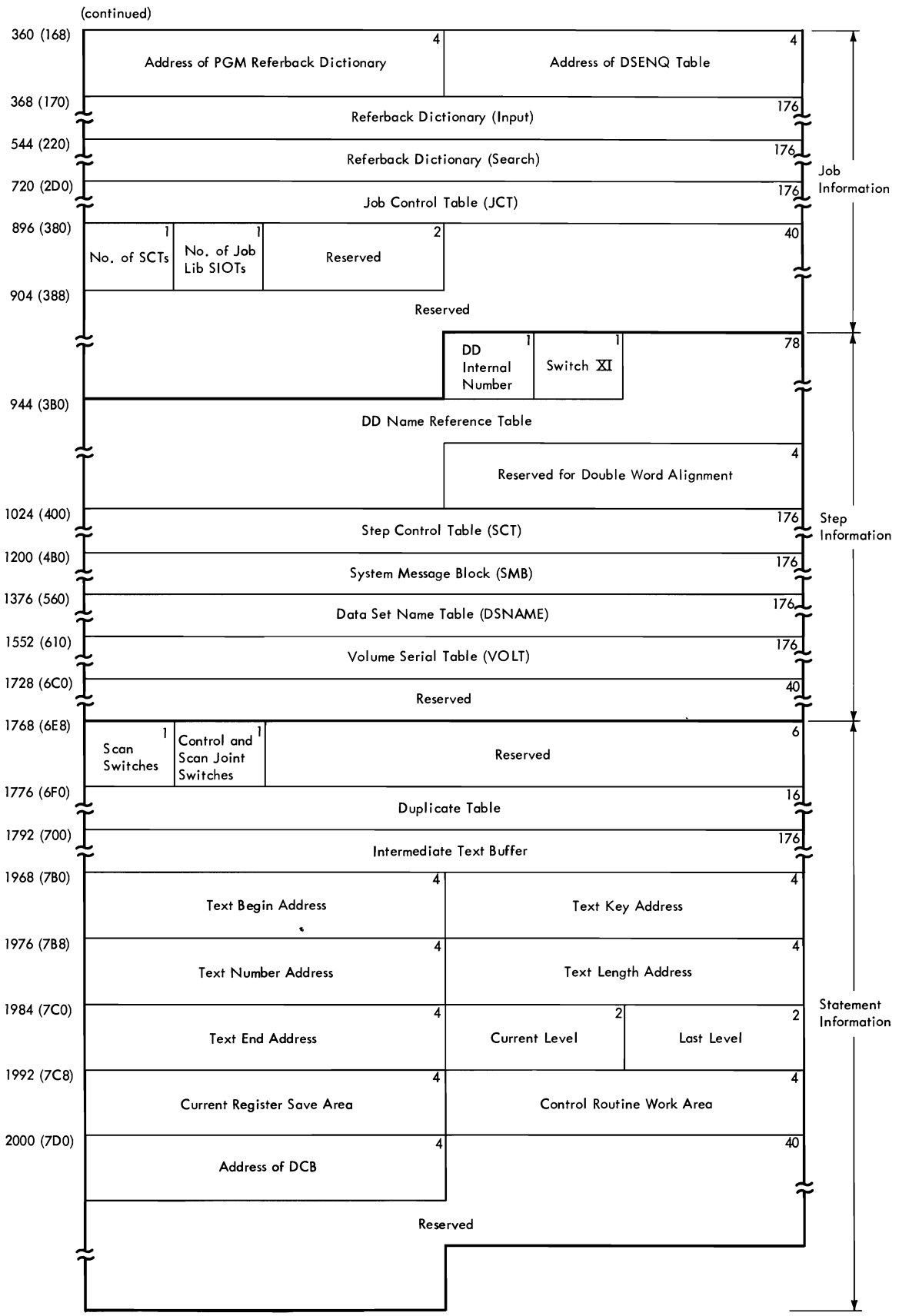


Figure 32. Interpreter Work Area (IWA) (Part 2 of 2)

JOB CONTROL TABLE (JCT)

Description: The job control table (JCT) (Figure 33) is created in the interpreter work area by the job statement processor routine of the interpreter. It contains information from the JOB statement, job status information, and pointers to other tables in the job's input queue entry. When the interpreter has processed all steps of a job, the JCT is written into the appropriate input queue according to priority; it is read back into main storage by the initiator job selection and job delete routines.

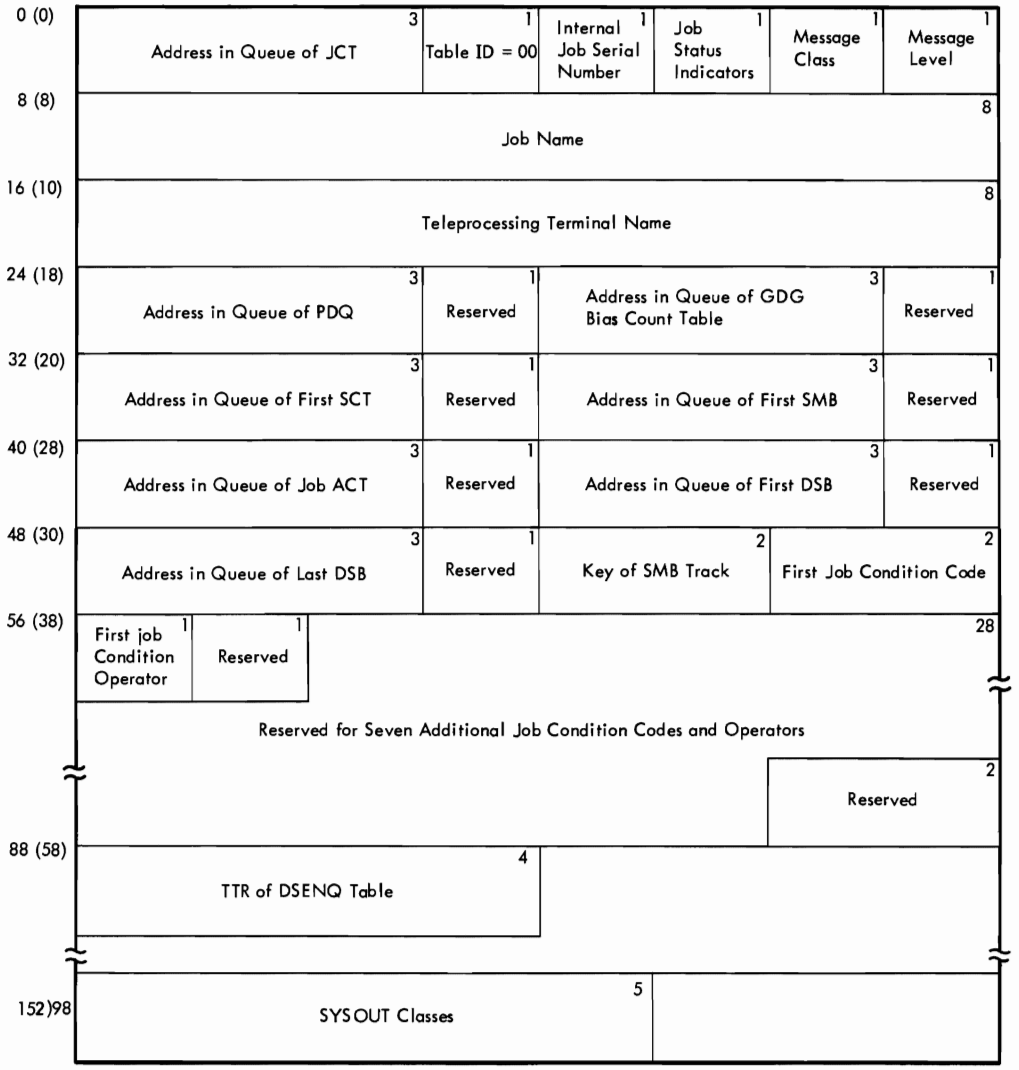
Although most of the fields in the job control table are self-explanatory, the following require further description:

- Job Status Indicators: The sixth byte of the JCT indicates the status of the job as shown below:

  Bit 0 is set to 1 of a JOBLIB DD statement is included with the job.
  Bit 5 is the job-failed bit. It is set to 1 if an error condition is encountered that causes the job to be terminated.
  Bit 6 is set to 1 if the job includes a cataloged procedure.

- SYSOUT Classes: The first 36 bits of the five-byte field are used to indicate the system output classes that contain data. The four remaining bits are reserved.

Mapping Macro Instruction: IEFAJCTB



• **Figure 33. Job Control Table (JCT)**

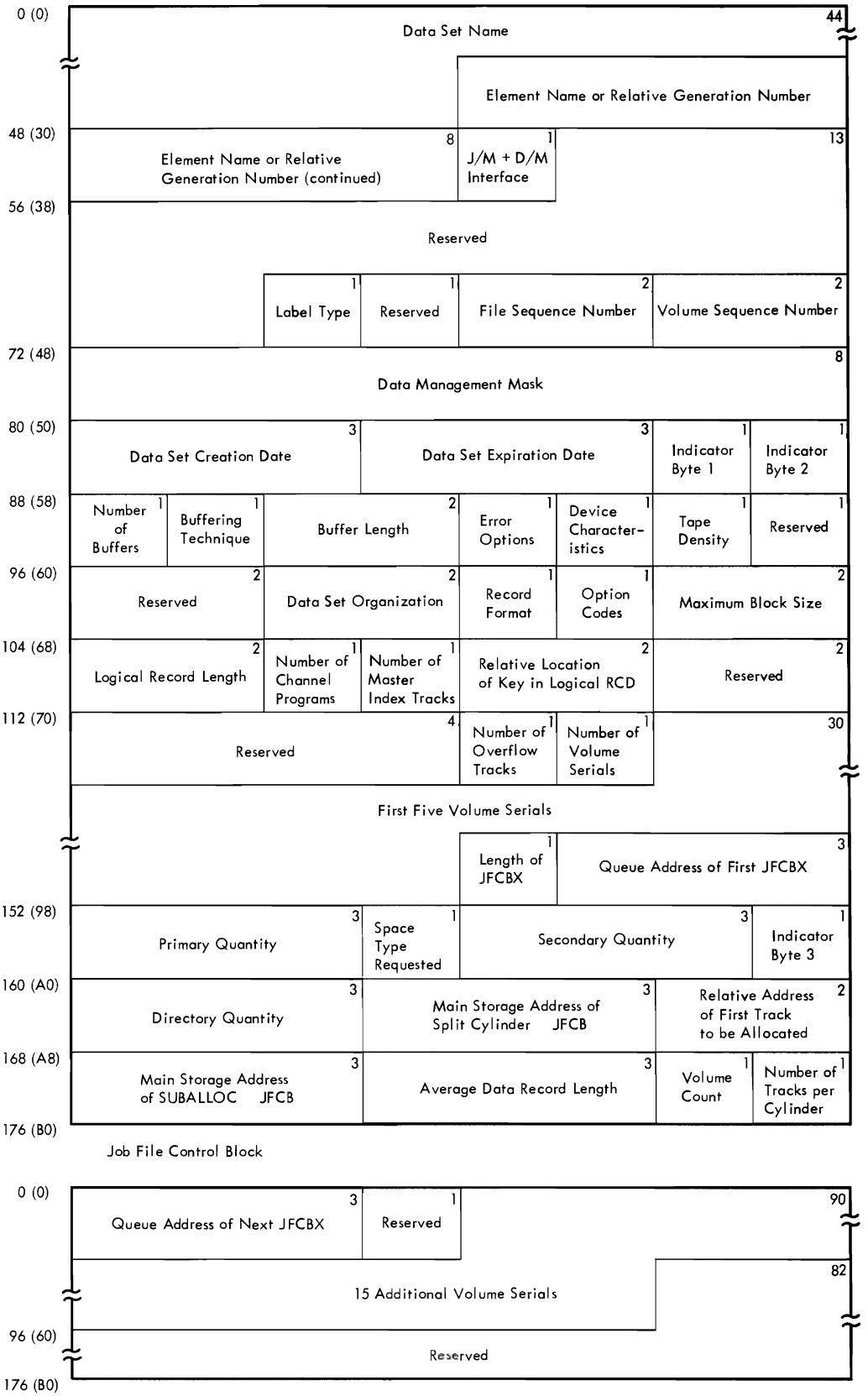


Figure 34. Job File Control Block (JFCB) and Extension (JFCBX)

JOB FILE CONTROL BLOCK (JFCB) AND EXTENSION (JFCBX)

Description: A job file control block (JFCB) (Figure 34) is constructed in subpool zero (from information in a DD statement) by the interpreter DD statement processor routine. The JFCB is written into the job's input queue entry, and retrieved when a DCB with the corresponding name is opened. The information in the JFCB, which describes the characteristics of a data set, may be modified by the open routine.

A JFCB contains enough space to record five volume serials. If more than five volume serials are specified, enough job file control block extensions (JFCBXs) to contain the additional volume serials are constructed; each JFCBX can contain up to fifteen additional volume serials.

Additional information on the contents of the JFCB and JFCBX may be found in the publication, IBM System/360 Operating System: System Control Blocks, Form C28-6628.

Mapping Macro Instruction: IEFJFCBN

LIFE-OF-TASK (LOT) BLOCK

Description: The 348-byte life-of-task (LOT) block (Figure 35) is built in a main storage area obtained from subpool 253. It stores information for scheduling functions, and is used by system task control and initiators. It is created by the Job Select module for initiating problem programs, and by system task control for initiating readers and writers.

The LOT block contains the linkage control table (LCT), a two-level register save area (REGSAVE), an input queue manager parameter area (QMGR1), an output queue manager parameter area (QMGR2), the address of the ECB list, the address of the PIB, and the address of the SPIL.

LINKAGE CONTROL TABLE (LCT)

Description: The linkage control table (LCT) (Figure 36) is built in a main storage area obtained from subpool 253 by the initiator initialization routine. It is a communications area used by the routines of the initiator.

Most of the fields in the LCT are self-explanatory; it should be noted, however, that the job termination status bit is the low-order bit of the one-byte device features field.

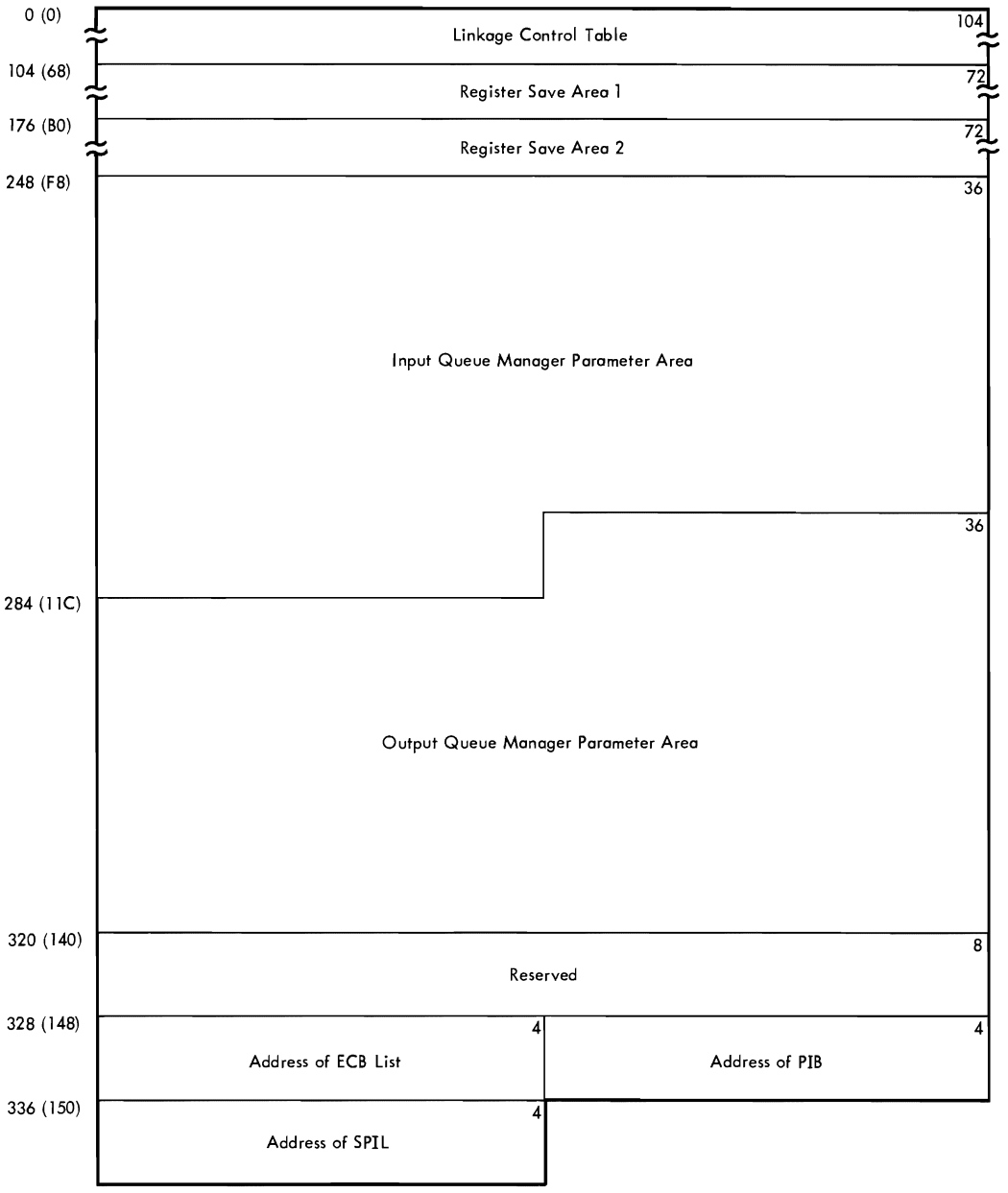
Mapping Macro Instruction: IEFALLCT

MASTER SCHEDULER RESIDENT DATA AREA

Description: The master scheduler resident data area (Figure 37), which is in the nucleus area of main storage, contains information used by the queue initialization, command scheduling, initiator, and I/O device allocation routines. Its location is stored in the CVTMSER field of the communication vector table.

Most of the fields in the master scheduler resident data area are self-explanatory; those fields that require further explanation are described below:

- Queue Formatting Switch: If the high-order bit of this field is on, it indicates that the queue data set must be formatted.
- Status Flags: The high-order bit is set on during system initialization; the second bit is set on by the DISPLAY JOBNAMES command, and set off by the STOP JOBNAMES command; the third bit is set on by the



• Figure 35. Life-of-Task (LOT) Block

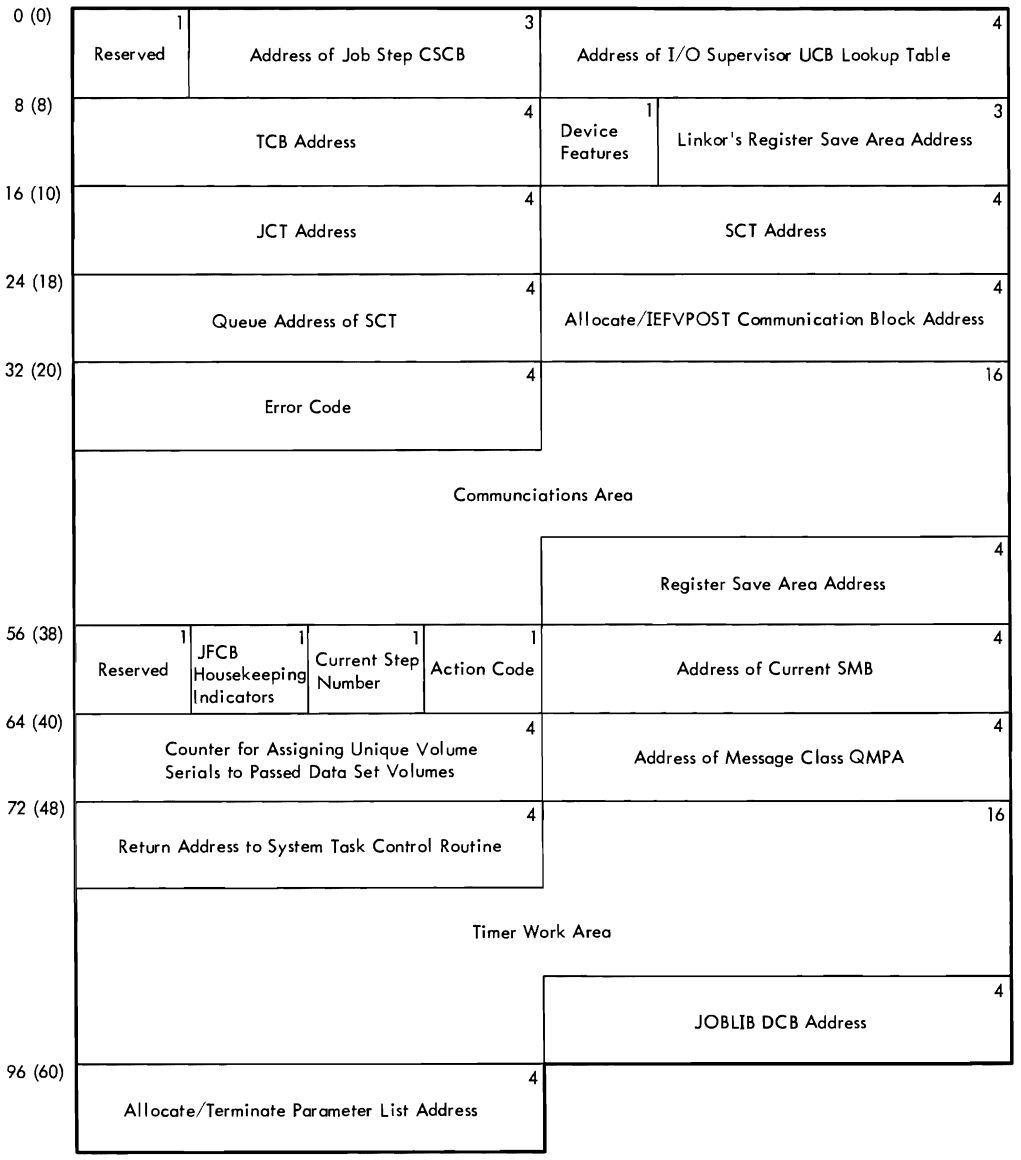


Figure 36. Linkage Control Table (LCT)

VARY command to indicate that type I/O device allocation routine is to search the UCBs for a change in unit status.

- MFT Switches: If the high-order bit of this field is on, it indicates that there is an active transient reader; the second bit is set on when there is a transient reader in main storage; the third bit is set on when there is a pending START reader command for a transient reader; the fourth bit is set on to indicate that it is an MFT environment; the fifth bit indicates that a system-assigned transient reader is running.

- Transient Reader TTR: This field is used by the transient reader suspend routine to store the address of the work queue data set where the reader information was placed when the reader was suspended.

- DEFINE Control Information: If the high-order bit of this field is on, it is a DEFINE operation; if off, it is IPL time. The second bit indicates that a list of the partitions' sizes and job class(es) has been requested; the third bit indicates that there is an adjacent partition check; the fifth bit is set on when the operator has requested partition changes at IPL; the sixth bit indicates that a

small partition cannot terminate because of the DEFINE operation; the seventh bit indicates that a DEFINE command has been issued during operation; the eighth bit indicates that the system has storage protection.

- Mapping Macro Instruction: IEEBASEB.

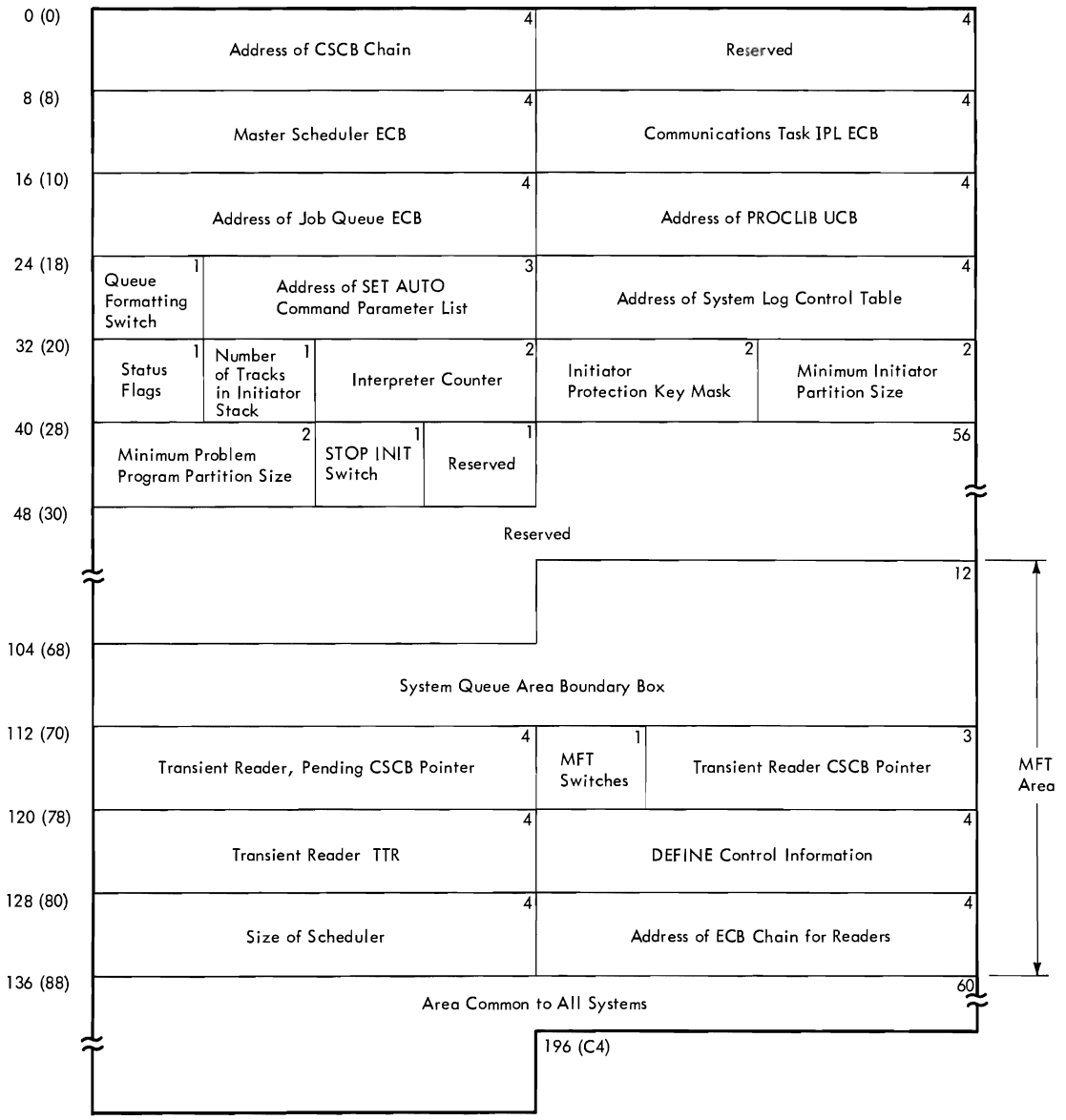
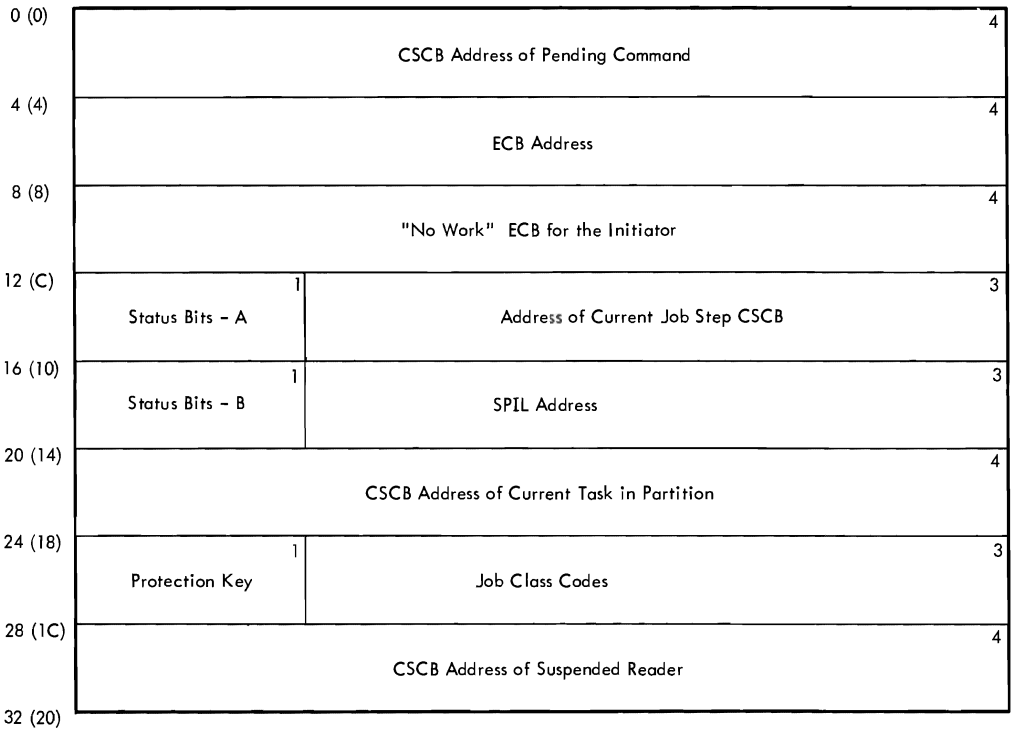


Figure 37. Master Scheduler Resident Data Area

PARTITION INFORMATION BLOCK

The 32-byte partition information block (PIB) (Figure 38) contains information used by the command processing and scheduler routines. Its location is stored in the TCBPIB field at displacement 124 (decimal) of the task control block (TCB).

Although most of the fields in the partition information block are self-explanatory, the following require further description:



• Figure 38. Partition Information Block (PIB)

- ECB Address: Contains the address of ECB to be posted by job selection when the partition is made quiescent for partition redefinition.

- "No Work" ECB for the Initiator: This ECB is posted by small partitions requesting service, the queue manager when a job has been enqueued, and by the DEFINE and START command routines.

- Status A Information:

Bit	Setting	Meaning
0	0	Stop initiator
	1	START INIT issued
1	1	Partition active
2	1	Pending command
3	1	Transient reader operating
4	1	Reserved
5	1	Partition is involved in redefinition
6	1	System-assigned transient reader operating in this partition
7	1	Problem program is running

- Status B Information:

Bit	Setting	Meaning
0	1	Logical tracks added for initiator
1	1	LOT block exits
2	1	SPIL has been created

- SPIL Address: The small partition information list (SPIL) is applicable to large partitions only.

- Job Class Codes: Contains one to three codes for the partition, arranged in descending numerical order, i.e., GRP3 is in the second byte of the field, followed by GRP2 and GRP1. The first byte contains the protection key for the partition, if the system has the storage protection feature.

SMALL PARTITION INFORMATION LIST (SPIL)

Description: The 32-byte small partition information list (SPIL) (Figure 39) is a storage area for information pertaining to small partition scheduling. It is built in main storage obtained from subpool 0. The address of the ECBs provides for information to be passed between the small partition and the large partition that is performing initiation, allocation, or termination functions for the small partition.

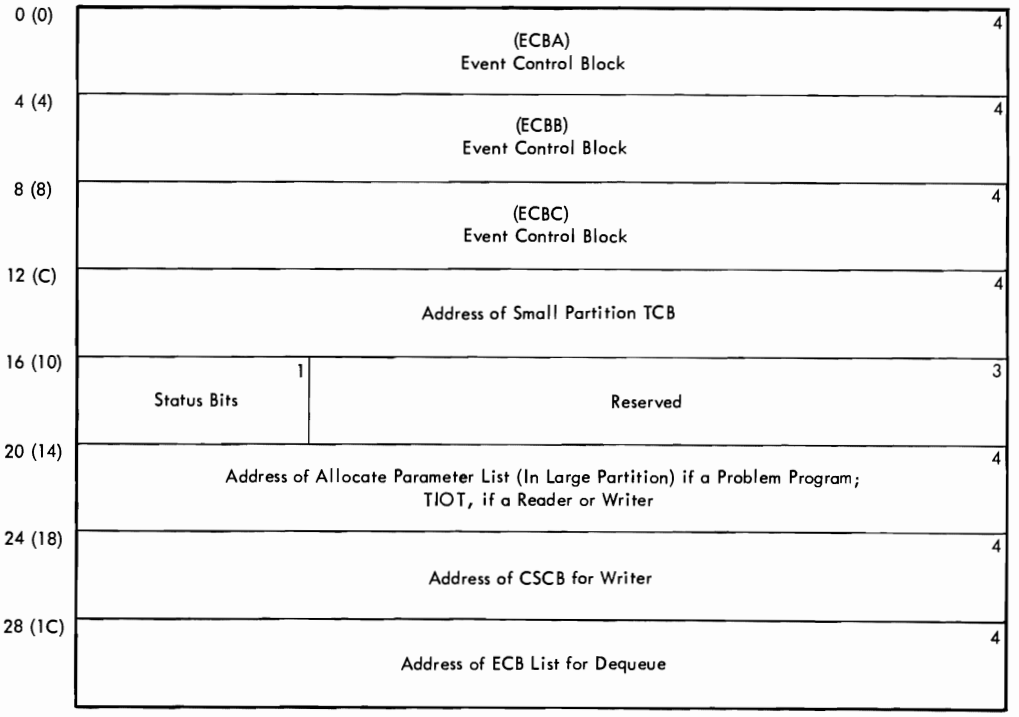
Most of the fields in the small partition information block are self explanatory; however, the status bits field is described below.

Bits 0 and 1 contain ones if a START writer command has been entered.

Bit 2 contains a one if a SPIL pointer has been stored in the PIB.

Bit 5 contains a one if a problem program has requested termination.

Bits 0-7 contain zeros if a START INIT command was entered.



• Figure 39. Small Partition Information List (SPIL)

STEP CONTROL TABLE (SCT)

Description: The step control table (SCT) (Figure 40), is used to pass control information to the DD routine of the interpreter and to the initiator routines, which also contribute information to the table. This table is created and initialized by the execute statement processor routine of the interpreter when an EXEC statement is read. One SCT is created for each step of a job.

If the step is part of a previously cataloged procedure, the name of the step that called the procedure, if any, is entered. The following variable-content and indicator fields are included in the table:

1. Internal Step Status Indicators:

   Bit 7 contains a one if an error condition caused the step to be terminated.

2. PARM Count or Step Status Code:

   a. Interpreter: The number of characters specified in the PARM parameter of the EXEC statement is placed in this entry.

   b. Initiator: This table entry contains the condition code returned by the processing program.

3. Step Type Indicators:

   Bit 0 contains a one if the following parameter definition appears in the EXEC statement:

   PGM=*.stepname.ddname

   Bit 1 indicates SYSIN is specified (DD *).

   Bit 2 indicates SYSOUT is specified.

   Bit 3 contains a 1 if JFCB housekeeping is complete.

   Bits 4, 5, and 6 are unused.

Mapping Macro Instruction: IEFASCTB

STEP INPUT/OUTPUT TABLE (SIOT)

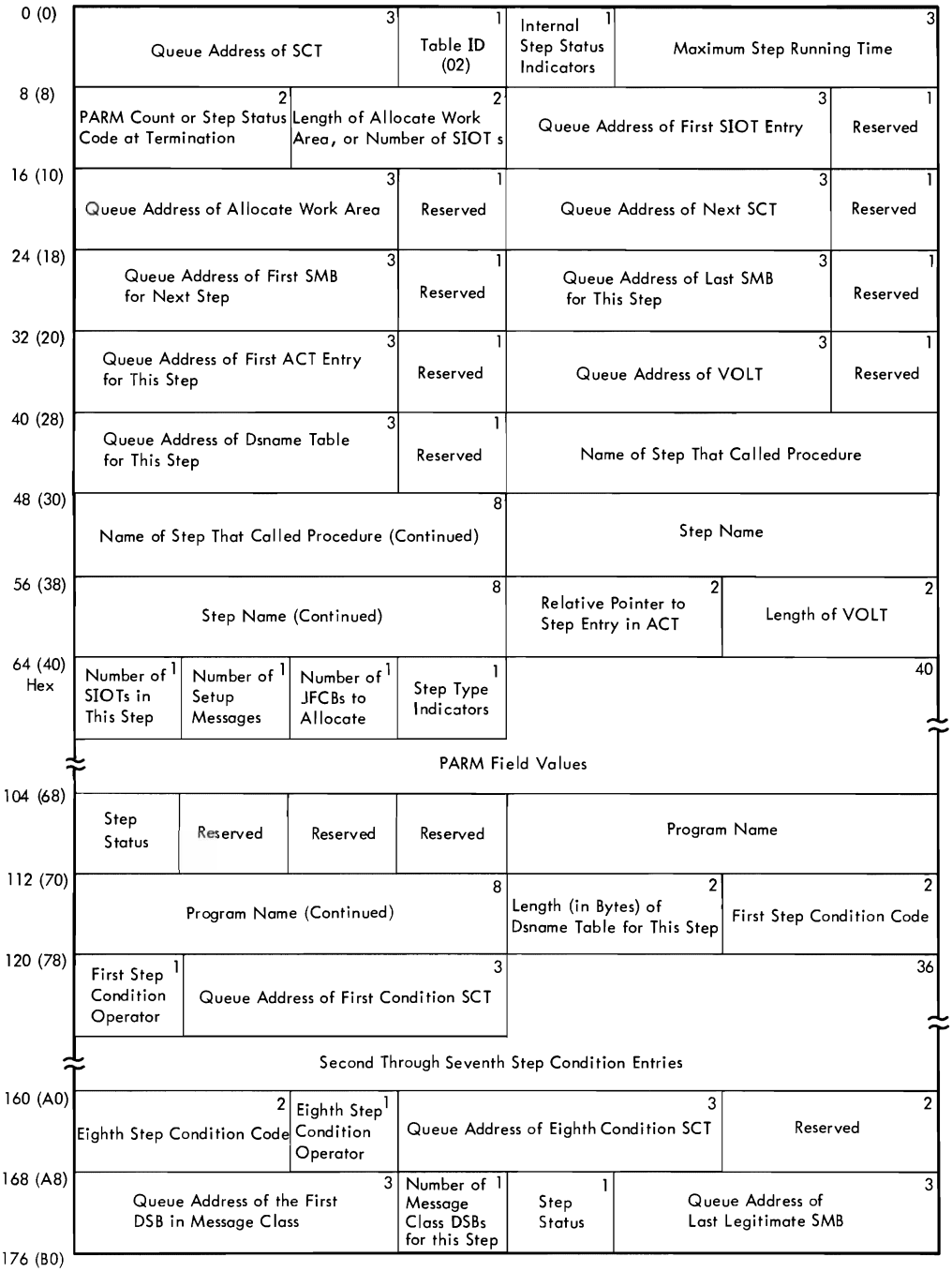
Description: The Step Input/Output Table (SIOT) (Figure 41), makes DD statement available to the initiator for use as a source of information for the TIOT and for providing DD information to allocation and disposition routines. When a DD statement is read, the interpreter creates a new SIOT and places the DD information into it. The individual bits of the disposition byte and of indicator bytes 57 through 60 in the SIOT are set to one to indicate the following conditions:

BYTE 56: Scheduler Disposition

Bit 0 Reserved
Bit 1 Retain volume
Bit 2 Private volume
Bit 3 Pass data set
Bit 4 Keep data set
Bit 5 Delete data set
Bit 6 Catalog data set
Bit 7 Uncatalog data set

BYTE 57: Indicator Byte Number 1

Bit 0 Dummy data set
Bit 1 SYSIN data set
Bit 2 Split (primary)
Bit 3 Split (secondary)
Bit 4 Suballocate
Bit 5 Parallel mount
Bit 6 Unit affinity
Bit 7 Unit separation



•Figure 40. Step Control Table (SCT)

BYTE 58: Indicator Byte Number 2

Bit 0 Channel affinity
Bit 1 Channel separation
Bit 2 Volume affinity
Bit 3 JOBLIB DD statement
Bit 4 Unlabeled (no labels)
Bit 5 Pool DD statement
Bit 6 Defer mounting
Bit 7 Received data set

BYTE 59: Indicator Byte Number 3

Bit 0 Volume reference
Bit 1 SYSIN expected (procedures only)
Bit 2 Reserved
Bit 3 Volume reference in step
Bit 4 SYSOUT was specified
Bit 5 NEW data set
Bit 6 MOD data set
Bit 7 OLD or SHR data set

BYTE 60: Indicator Byte Number 4

Bit 0 Set by reader to indicate GDG single
Bit 4 Step processed
Bit 5 Intra-step volume affinity
Bit 6 Data set is in PDQ
Bit 7 1 = old or modified data set
0 = new data set

BYTE 93: Conditional Disposition

Bits 0-3 Reserved
Bit 4 Keep data set
Bit 5 Delete data set
Bit 6 Catalog data set
Bit 7 Uncatalog data set

BYTE 104: Step Status

Bit 0 Reserved
Bit 1 Reserved
Bit 2 SCTMCVOL
Bit 3 Reserved
Bit 4 SCTSTPLB
Bit 5 Reserved
Bit 6 Reserved
Bit 7 Reserved

Mapping Macro Instruction: IEFASIOT

TASK INPUT/OUTPUT TABLE (TIOT)

Description: The Task Input/Output Table (TIOT) (Figure 42) provides data management routines with the addresses of the JFCBs and devices allocated to the data sets in a job step or system task. It is constructed by the I/O device allocation routine in main storage obtained from subpool zero. The allocation routine also places a copy of the TIOT on the appropriate job class queue with the other tables for the job step. After the step completes processing, the TIOT is brought in from the job queue and placed in the upper portion of the partition. The step is then terminated, and the TIOT is deleted.

For further information on the TIOT, see IBM System/360 Operating System: System Control Blocks, Form C28-6628.

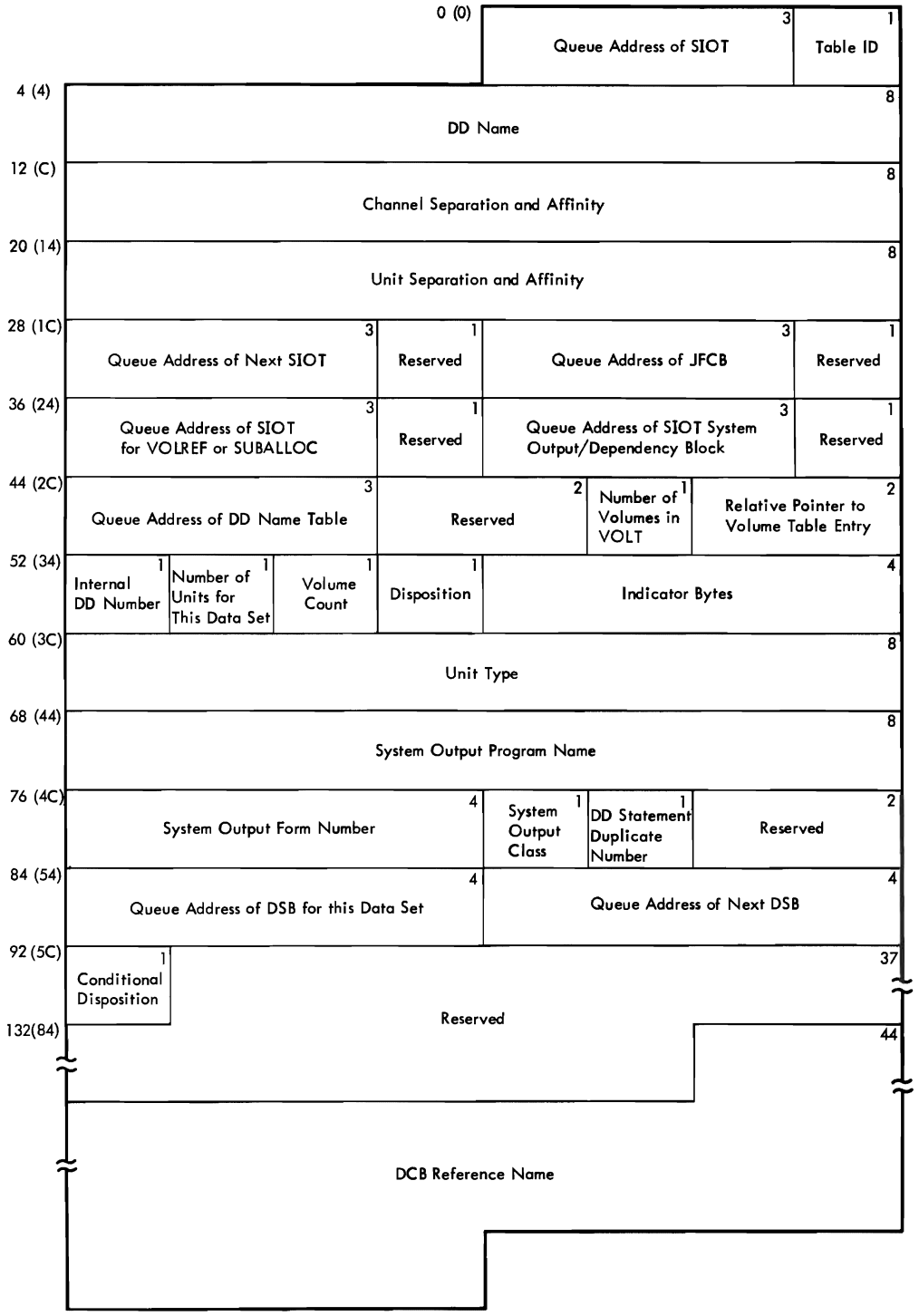


Figure 41. Step Input/Output Table (SIOT)

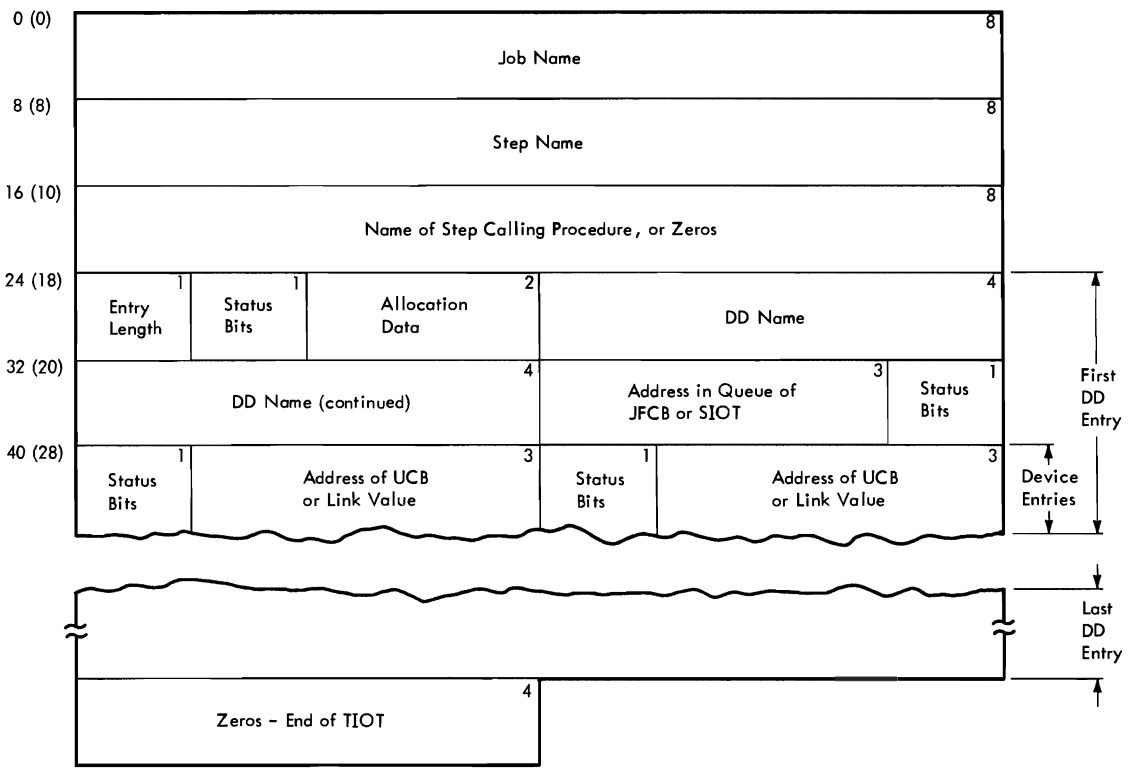


Figure 42. Task Input/Output Table (TIOT)

APPENDIX B: MFT MODULES

This appendix contains a table of new MFT modules, a group of tables showing the modules of each major component, and a brief description of each of the modules used by MFT. If you are looking for a specific module and know only the major component and routine name, use Tables 3-14 which give a cross-reference to the source module. The source modules are in turn listed alphanumerically for easy access. If you know the source module name, go directly to the module descriptions.

NEW MFT MODULES

Table 2 lists all new modules implemented by MFT. This table is organized alphabetically by major component.

Table 2. New MFT Modules

ABEND:	I/O Device Allocation:	Queue Management:
IEAGTM05	IEFSD551	IEFSD514
IEAGTM06	IEFSD552	IEFSD572
	IEFSD557	
Communications Task:		
IEEVWTOR	**Master Scheduler Task:**	
Initiator:		**Reader/Interpreter:**
	IEECIR50	
IEFSD510	IEEDFIN1	IEFSD530
IEFSD511	IEEDFIN2	IEFSD531
IEFSD512	IEEDFIN3	IEFSD532
IEFSD513	IEEDFIN4	IEFSD533
IEFSD515	IEEDFIN5	IEFSD536
IEFSD516	IEEDFIN6	IEFSD537
IEFSD517	IEEDFIN7	
IEFSD540	IEEDFIN8	
IEFSD541	IEEDFIN9	
IEFSD553	IEESD561	**System Task Control:**
IEFSD554	IEESD562	
IEFSD555	IEESD563	IEESD590
IEFSD556	IEFSD564	IEESD591
IEFSD558	IEESD565	IEESD590
IEFSD559	IEESD566	IEESD591
IEFSD589	IEESD571	IEESD592
IEFSD598	IEFSD569	IEFSD534
IEFSD599		IEFSD535
	Nucleus:	
	IEESD568	IEFSD587
	IEFSD567	IEFSD588

MAJOR COMPONENT MODULES

Tables 3 through 14 list all MFT modules according to major component. The tables appear in alphabetical order by component name. Within each component, routine names are listed alphabetically with a cross-reference to the module name.

Table 3. ABEND Modules

Routine	Source Module
ABDUMP	IEAATM04
Indicative Dump	IEAATM03
Initialization	IEAGTM00
Input/Output Purge	IEAGTM06
Linkage	IEAATM01
Main Storage Allocation	IEAATM02
Termination	IEAGTM05

Table 4. Communication Task Modules

Routine	Source Module
Console Device Processor	IEECVPM
Console Interrupt	IEECVCRA
External Interrupt	IEECVCRX
Initialization Routine	IEECVCTI
Purge RQE	IEECVED2
Router	IEECVCTR
Wait	IEECVCTW
Write-to-Operator	IEECVWTO
Write-to-Operator-With-Reply	IEEVWTOR

Table 5. Initiator Modules

Routine	Source Module
Alternate Step Deletion	IEFSD516
Data Set Integrity	IEFSD541
ENQ/DEQ Purge	IEFSD598
Job Deletion	IEFSD517
Job Initiation	IEFSD511
Job Selection	IEFSD510
Linkage to IEFSD510	IEFSD555
Linkage to IEFSD511	IEFSD558
Linkage to IEFSD512	IEFSD553
Linkage to IEFSD515	IEFSD559
Linkage to IEFSD516	IEFSD554
Linkage to IEFSD534	IEFSD589
Linkage to IEFSD541	IEFSD540
Problem Program Interface	IEFSD513
Set Problem Program State	IEFSD556
Small Partition Module	IEFSD599
Step Deletion	IEFSD515
Step Initiation	IEFSD512

Table 6. I/O Device Allocation Modules

Routine	Source Module
Allocation Control	IEFXCSSS
Allocation Entry	IEFSD210
Allocation Exit	IEFSD41Q
Allocation Recovery Messages	IEFSJMSG
Allocation Recovery	IEFXJIMP
Automatic Volume Recognition	IEFXV001
Automatic Volume Recognition Messages	IEFVMSG
Automatic Volume Recognition Non-standard Label Routine	IEFXVNSL
DADSM Error Recovery	IEFXT003
Decision Allocation	IEFS5000
Demand Allocation	IEFWA000
Device Strikeout	IEFX300A
EXEC Statement Condition Code Processor	IEFVKIMP
EXEC Statement Condition Code Processor Messages	IEFVKMSG
External Action Messages	IEFWD001
External Action	IEFWD000
Interface	IEFSD557
JFCB Housekeeping Control and Allocate Processing	IWDCMLA1
JFCB Housekeeping Error Message Processing	IEFVMLS6
JFCB Housekeeping Error Messages	IEFVMLS7
JFCB Housekeeping Fetch DCB Processing	IEFVM2LS
JFCB Housekeeping GDG All Processing	IEFVM4LS
JFCB Housekeeping GDG Single Processing	IEFVM3LS
JFCB Housekeeping Patterning DSCB	IEFVM5LS
JFCB Housekeeping Unique Volume ID	IEFVM76
Mount Control-Volume Routine	IEFMCVOL
Linkage Module	IEFWCFAK
Linkage Module	IEFWDFA
Linkage Module	IEFWSWIN
Linkage Module	IEFXJFAK
Linkage to JFCB Housekeeping	IEFVMMS1
Linkage to JFCB Housekeeping	IEFVMFAK
Linkage to IEFXJIMP	IEFSD551
Linkage to IEFXJIMP	IEFSD552
Linkage to IEFXV001	IEFAVFAK
Linkage to Mount Control Volume	IEFCVFAK
Message Module	IEFWSTRT
Message Module	IEFXAMSG
Non-Recovery Error	IEFXKIMP
Non-Recovery Error Messages	IEFXKMSG
Separation Strikeout	IEFXH000
Space Request	IEFXT00D
TIOT Compression	IEFXT002
TIOT Construction	IEFWCIMP
Unsolicited Device Interrupt Handler	IEFVPOST
Wait for Space Decision	IEFSD097
Wait for Unallocation	IEFSD195

Table 7. Interpreter Modules

Routine	Source Module
Command Statement	IEFVHM
CPO Allocation Subroutine	IEFVSD12
CPO	IEFVHG
Continuation Statement	IEFVBC
DD* Statement Generator	IEFVHB
DD Statement Processor	IEEFVDA
Data Set Name Table Construction	IEFVDBSD
Dictionary Entry	IEFVGI
Dictionary Search	IEFVGS
End-of-File	IEFVHAA
EXEC Statement Processor	IEFVEA
Get Parameter	IEFVGK
Get	IEFVHA
Housekeeping	IEFVHHB
Initialization	IEFVH1
Initialization	IEFVH2
Interface	IEFSD533
Job and Step Enqueue	IEFVHH
Job Statement Processor	IEFVHA
Job Validity Check	IEFVHEC
Linkage Module	IEFSD537
Message Module	IEFVGM1
Message Module	IEFVGM2
Message Module	IEFVGM3
Message Module	IEFVGM4
Message Module	IEFVGM5
Message Module	IEFVGM6
Message Module	IEFVGM7
Message Module	IEFVGM8
Message Module	IEFVGM9
Message Module	IEFVGM10
Message Module	IEFVGM11
Message Module	IEFVGM12
Message Module	IEFVGM13
Message Module	IEFVGM14
Message Module	IEFVGM15
Message Module	IEFVGM16
Message Module	IEFVGM17
Message Module	IEFVGM18
Message Module	IEFVGM70
Message Module	IEFVGM78
Message Processing	IEFVGM
Null Statement	IEFVHL
Operator Message	IEFSD536
Post-Scan	IEFVHF
Pre-Scan Preparation	IEFVHEB
Queue Management Interface	IEFVHQ
Router	IEFVHE
Scan	IEFVFA
SCD Construction	IEFVSD13
Symbolic Parameter Processing	IEFVFB
Termination	IEFVHN
Test and Store	IEFVGT
Transient Reader Restore	IEFSD531
Transient Reader Suspend	IEFSD530
Transient Reader Suspend Tests	IEFSD532
Vary Identification	IEFVHCB

Table 8. Master Scheduler Modules

Routine	Source Module
DEFINE Final Processor	IEEDFIN3
DEFINE Initialization	IEEDFIN1
DEFINE Listing	IEEDFIN4
DEFINE Message	IEEDFIN5
DEFINE Syntax Check and Router	IEEDFIN2
DISPLAY A	IEESD566
DISPLAY R	IEESD567
Queue Search	IEESD564
Queue Search Setup	IEESD563
Service	IEESD565
Syntax Check	IEESD562
Time-Slice Syntax Check	IEEDFIN6
Wait/Router	IEECIR50

Table 9. Queue Management Modules

Routine	Source Module
Assign	IEFQASGQ
Assign/Start	IEFQAGST
Branch	IEFQMLK1
Control	IEFQBVMS
Delete	IEFQDELQ
Dequeue	IEFQMDQQ
Dummy	IEFQMDUM
Enqueue	IEFQMNQQ
Interpreter/Queue Manager Interlock	IEFSD572
Message Module	IEFSD311
Queue Formatting	IEFORMAT
Queue Initialization	IEFSD055
Read/Write	IEFQMRAW
Resident Main Storage Reservation	IEFPRESD
Unchain	IEFQMUNQ

Table 10. SVC 34 Modules

Routine	Source Module
CSCB Creation	IEE0803D
CSCB Marking	IEE0703D
DEFINE, MOUNT, CANCEL	IEFSD571
HALT	IEE1403D
Message Assembly	IEE0503D
Message Assembly	IEE2103D
Reply Processor	IEE1203D
Router	IEE0403D
SET Command Handler	IEE0903D
SET Command	IEE0603D
START and STOP INIT	IEFSD561
Translator/Chain Manipulator	IEE0303D
VARY and UNLOAD	IEE1103D

Table 11. System Output Writer Modules

Routine	Source Module
Class Name Setup	IEFSD081
Command Processing	IEFSD083
Data Set Delete	IEFSD171
Data Set Writer Interface	IEFSD070
DSB Handler	IEFSD085
Initialization	IEFSD080
Job Separator	IEFSD094
Linkage Module	IEF078SD
Linkage Module	IEF079SD
Linkage Module	IEF082SD
Linkage Module	IEF083SD
Linker	IEFSD078
Linkage to Queue Manager Delete	IEFSD079
Main Logic	IEFSD082
Message Module	IEFSD096
Print Line	IEFSD095
Put	IEFSD089
SMB Handler	IEFSD086
Standard Writer	IEFSD087
Transition	IEFSD088
Wait	IEFS084

Table 12. System Restart Modules

Routine	Source Module
Delete	IEFSD303
Initialization	IEFSD300
Jobnames Table	IEFSD302
Linkage Module	IEF300SD
Linkage Module	IEF304SD
Message Module	IEFSD312
Purge Queue Construction	IEFSD301
Scratch Data Sets	IEFSD304
Scratch Data Sets	IEFSD308
TTR and NN to MBBCCHHR Conversion	IEFSD310

Table 13. System Task Control Modules

Routine	Source Module
Allocation Interface	IEEVACTL
Internal JCL Reader	IEEVICLR
Interpreter Control	IEEVRCTL
JCL Edit	IEEVJCL
Linkage to IEFSD535	IEFSD587
Linkage to IEE534SD	IEFSD588
Linker	IEESD591
Link-Table	IEEVLNKT
LPSW	IEFSD534
Message Writer	IEEVMSG1
Message Writer	IEEVSMSG
Message Writing	IEEVOMSG
POST	IEESD592
Problem Program Mode	IEFSD535
QMPA Builder	IEEVSMRA
START Syntax Check	IEEVSTAR
Termination Interface	IEEVTCTL
Write TIOT on Disk	IEESD590

Table 14. Termination Modules

Routine	Source Module
Disposition and Unallocation Messages	IEFZGMSG
Disposition and Unallocation Messages	IEFZHMSG
Disposition and Unallocation	IEFZGJB1
Disposition and Unallocation	IEFZGST1
DSB Processing	IEFYTVMS
Job Statement Condition Code Processor	IEFVJIMP
Job Statement Condition Code Processor Messages	IEFVJMSG
Job Termination Control	IEFZAJB3
Job Termination Exit	IEFSD31Q
Message Blocking	IEFYSVMS
Message Module	IEFWTERM
Message	IEFIDMPM
Step Termination Control	IEFYNIMP
Step Termination Control Routine Messages	IEFYNMSG
Step Termination Data Set Driver	IEFYPJB3
Step Terminate Exit	IEFSD22Q
Step Termination Messages	IEFYPMSG
System Output Interface	IEFSD017
Termination Entry	IEFSD42Q
User Accounting Routine Linkage	IEFACTLK
User Dummy Accounting	IEFACTFK

MODULE DESCRIPTIONS

This section contains a brief description of each of the modules used by MFT. Modules are listed alphanumerically by module name; associated with each module is a descriptive name, which indicates the major component of the system to which the module belongs. Each module contains a brief statement of the purpose of the module. Where applicable, the description includes the names of the modules entry points, the names of the modules to which it passes control, the major tables and work areas to which it refers, its attributes, the names of the control sections it contains, and a page reference to the detailed writeup in the Job Management section. The names of the first and last modules on each page appear at the top of that page.

IEAGTM0A: Supervisor -- ABEND STAE Test Routine

This routine prevents asynchronous exits and stores the completion code (if not previously stored). It determines if control should be returned to STAE after a purge error, if the Graphics Abend Exit routine should be entered, and if a valid STAE is in effect.

- Entry: IGC0001C
- Exit: XCTL to IEAGTM00 to continue ABEND processing
  to IEAATM0B if a valid STAE is in effect
  EXIT to IEAATM0B if an ABEND was issued by the Purge routine during STAE processing
  EXIT to caller if graphics program
- Attributes: Reentrant, disabled, privileged
- Control Section: IGC0001C

IEAGTM00: Supervisor -- ABEND Initialization Routine

This routine provides purging for IQEs and WTOR requests, and cancels the task timer element. On detecting certain abnormal conditions, IEAGTM00 passes control to the System Quiesce routine.

- Entry: IGC0701C
- Exit: XCTL to IEAATM0D if IEAGTM00 was entered from STAE
  to IEAGTM05 if this is a normal end
  to IEAGTM06 if this is an abnormal end
  branch to IEAGTWST if a system task is attempting ABEND, or if the task or system has been specified 'must complete'
- Attributes: Reentrant, disabled for SVC interruptions, privileged
- Control Section: IGC0701C

IEAGTM06: Supervisor -- ABEND Input/Output Purge Routine

This routine purges I/O operations in process and outstanding I/O requests.

- Entry: IGC0601C
- Exit: XCTL to IEAATM01
- Attributes: Reentrant, disabled for I/O and external interruptions, privileged
- Control Section: IGC0601C

IEAATM01: Supervisor -- ABEND Validity Check Routine

This routine performs validity checking of the MSS (main storage supervisor) queue, the load list, the active RB list, and the DEB queue. It dequeues invalid control blocks, or terminates the queue at the point of error, and sets bits in the ABEND appendage to the boundary box to indicate invalid control blocks found on one or more lists.

- Entry: IGC0101C
- Exit: XCTL to IEAATM0A
- Attributes: Reentrant, disabled for external and I/O interruptions, privileged
- Control Section: IGC0101C

IEAATM0A: Supervisor -- ABEND Linkage Routine

This routine checks for valid and invalid recursions. For an invalid recursion, control is passed to the System Quiesce Routine. For a valid recursion, a bit is set in the TCBFLGS field of the TCB to prevent an ABDUMP from being attempted. IEAATM0A determines the amount of main storage required by ABEND, and transfers control to the appropriate ABEND load module.

- Entry: IGC0111C
- Exits: XCTL to IEAATM02 if main storage must be 'stolen'
  to IEAATM03 if main storage is available and an indicative dump is requested
  to IEAATM04 if main storage is available and ABDUMP is requested
  to IEAATM05 if main storage is available and no dump is requested
  branch to IEAGTWST (System Quiesce routine) if an invalid ABEND recursion has been detected
- Attributes: Reentrant, disabled for external and I/O interruptions, privileged
- Control Section: IGC0111C

IEAATM02: Supervisor -- ABEND Steal LRB Main Storage Routine

This routine 'steals' main storage needed for ABEND functions that cannot be obtained via a GETMAIN macro instruction. The main storage is stolen from programs represented by LRBs on the loaded program list.

- Entry: IGC0201C
- Exits: XCTL to IEAATM2A if there is no loaded program list or if enough main storage is not available from the LRBs
  to IEAATM2B if IEAATM02 has acquired the necessary main storage
- Attributes: Reentrant, disabled for external and I/O interruptions, privileged
- Control Section: IGC0201C

IEAATM2A: Supervisor -- ABEND Steal Problem Program Main Storage Routine

This routine 'steals' main storage needed for ABEND functions from the lower end of the partition when it cannot be acquired either by a GETMAIN macro instruction or by the steal routine provided by IEAATM02.

- Entry: IGC0211C
- Exits: XCTL to IEAATM03 if indicative dump is requested
  to IEAATM04 if ABDUMP is requested
  to IEAGTM05 if no dump is requested or if a dump was previously attempted and failed
- Attributes: Reentrant, disabled for external and I/O interruptions, privileged
- Control Section: IGC0211C

IEAATM2B: Supervisor -- ABEND LRB Stack Routine

This routine moves the LRBs whose main storage was stolen by IEAATM02 to contiguous locations in the low end of the freed area and resets the chain pointers in the LRBs.

- Entry: IGC0221C
- Exits: XCTL to IEAATM03 if indicative dump is requested
  to IEAATM04 if ABDUMP is requested
  to IEAGTM05 if no dump is requested or if a dump was previously attempted and failed
- Attributes: Reentrant, disabled for external and I/O interruptions, privileged
- Control Section: IGC0221C

IEAATM03: Supervisor -- ABEND Indicative Dump Routine

This routine accumulates the information for an indicative dump and stores it in main storage.

- Entry: IGC0301C
- Exit: XCTL to IEAGTM05
- Attributes: Reentrant, disabled for external and I/O interruptions, privileged
- Control Section: IGC0301C

IEAATM04: Supervisor -- ABEND/ABDUMP Routine

This routine determines if the user wants a full or partial ABDUMP, initiates the ABDUMP output, and calls an SVC 51 (ABDUMP).

- Entry: IGC0401C
- Exits: XCTL to IEAATM03 for an indicative dump if the DCB has failed to open
  to IEAGTM05 for initialization of the next task
- Attributes: Reentrant, disabled for external and I/O interruptions, privileged
- Control Section: IGC0401C

IEAGTM05: Supervisor -- ABEND Termination Routine

This routine closes all data sets, purges the timer queue, resets the TCB fields, frees main storage, and transfers control to the job scheduler.

- Entry: IGC0501C
- Exits: XCTL to IEFSD51K for scheduler-size partitions
  to IEFSD599 for small partitions
- Attributes: Reentrant, disabled for external and I/O interruptions, privileged
- Control Section: IGC0501C

IEAGENQ1: Supervisor -- Enqueue Service Routine

This routine constructs and processes control blocks to serialize the use of resources in a multiprogramming environment.

- Entry: IEAGENQ1
- Exit: EXIT routine or to the dispatcher
- Tables/Work Areas: Minor QCB, Major QCB, Queue element
- Attributes: Reenterable
- Control Sections: IGC048 and IEG056

IEAGENQ2: Supervisor -- Shared DASD Enqueue Service Routine

This routine is the enqueue service routine for systems that include the Shared DASD option. It is identical to IEAGENQ1 except that additional processing is performed when a shared direct-access device is requested through the RESERVE macro instruction.

- Entry: IEAGENQ2
- Exits: EXIT routine or to the dispatcher
- Tables/Work Areas: Minor QCB, Major QCB, Queue element
- Attributes: Reenterable
- Control Sections: IGC048 and IEG056

IEAGTWST: Supervisor -- System Quiesce Routine

This routine places the failing task in wait state, and permits the system to quiesce.

- Entry: IECIWTST
- Exit: To the supervisor
- Tables/Work Areas: TCB
- Attributes: Resident in nucleus, disabled, reusable
- Control Section: IEAF03BP

IEA0TI01: Supervisor -- Timer Second Level Interruption Handler

This routine maintains the timer queue when the timer option is not specified during system generation. It handles only the normal six hour interruptions.

- Entry: IEA0TI01
- Exit: To Timer/External FLIH
- Tables/Work Areas: SHPC, T4PC, LTPC
- Attributes: Reenterable, disabled for system interruptions, resident, supervisor mode
- Control Section: IEA0TI01

IEECIR50: Master Scheduler -- Wait/Router Routine

This routine waits until a command is issued, analyzes the command and passes control to the appropriate processing module.

- Entry: IEECIR50
- Exits: IEESD562, IEEDFIN1
- Attributes: Read-only, reenterable, resident in nucleus.
- Control Section: IEECIR50
- Page Reference: 44

IEECVCRA: Communications Task -- Console Interruption Routine

This routine notifies the wait routine that a console read has been requested.

- Entry: IEEBA1
- Exit: Return to IOS
- Tables/Work Areas: ECB, UCM, UCB
- Attributes: Reenterable
- Control Section: IEEBA1
- Page Reference: 40

IEECVCRX: Communications Task -- External Interruption Routine

This routine switches control from the primary console device to an alternate console device when an external interruption occurs.

- Entry: IEEBC1PE
- Exit: Return to IOS
- Tables/Work Areas: UCM
- Attributes: Reenterable
- Control Section: IEEBC1PE
- Page Reference: 41

IEECVCTI: Communications Task -- Initialization Routine

This routine sets up control blocks and attributes in the unit control module (UCM).

- Entry: IEECVCTI
- Exit: IEECVCTW
- Tables/Work Areas: UCM
- Attributes: Read-only, reenterable
- Control Section: IEECVCTI
- External References: IEEVRFRX

IEECVCTR: Communications Task -- Router Routine

This routine determines the type of request or interruption that occurred, and passes control to the appropriate processing routine.

- Entry: IEECVCTR
- Exits: XCTL to IEECVPMX (IGC0107B), IEECVPMC (IGC1107B), or IEECVPMP (IGC2107B)
- Tables/Work Areas: UCM, SVRB, UCB
- Attributes: Reenterable
- Control Section: IEECVCTR
- Page Reference: 40

IEECVCTW: Communications Task -- Wait Routine

This routine waits on all communications task ECBs associated with WTO/WTOR macro instructions.

- Entry: IEECIR45
- Exit: None
- Tables/Work Areas: TCB, ECB, UCM
- Attributes: Reenterable
- Control Section: IEECIR45
- Page Reference: 40

IEECVED2: Communications Task -- Purge RQE Routine

This routine scans and purges all outstanding request queue elements (RQEs) pertaining to the terminating task.

- Entry: IEECVPRG
- Exits: End-of-task, and ABEND
- Tables/Work Areas: RQE, WQE, JCM, CVT
- Attributes: Reenterable
- Control Section: IEECVPRG

IEECVPM: Communications Task -- Console Device Processor Routine

This routine performs console read and write operations and checks for errors.

- Entry: IEECVPM
- Exit: XCTL to IEECVCTR (IGC0007B)
- Tables/Work Areas: DCB, UCB, UCM
- Attributes: Reenterable
- Control Section: IEECVPM
- Page Reference: 40

IEECVWTO: Communications Task -- Write-to-Operator Routine

This routine processes all WTO macro instructions.

- Entry: IGC0003E
- Exit: Return to calling program
- Tables/Work Areas: WQE, UCM, CVT, RQE
- Attributes: Reenterable
- Control Section: IGC0003E
- Page Reference: 41

IEEDFIN1: Master Scheduler -- DEFINE Command Initialization Routine

This routine sets up data areas for partition definition and passes control to the appropriate processing module.

- Entry: IEEDFIN1
- Exits: IEEDFIN3, IEEDFIN4, IEEDFIN5
- Attributes: Read-only, reenterable
- Control Section: IEEDFIN2
- Page Reference: 45

IEEDFIN2: Master Scheduler -- DEFINE Command Syntax Check and Router Routine

This routine checks the syntax of DEFINE command statements. If a syntax error is discovered, the statement is ignored and an error message is issued. If the syntax is correct, the information is stored and control is passed to the appropriate routine.

- Entry: IEEDFIN2
- Exits: IEEDFIN5, IEEDFIN6, IEEDFIN7
- Attributes: Read-only, reenterable
- Control Section: IEEDFIN2
- Page Reference: 45

IEEDFIN3: Master Scheduler -- DEFINE Command Final Processor

This routine determines that all information for the partition redefinition is correct, constructs a list of ECBs (one for each of the affected partitions) to be posted when the jobs in the partitions have terminated, and issues a message that DEFINE processing is complete.

- Entry: IEEDFIN3
- Exits: IEEDFIN5, IEEDFIN8
- Attributes: Read-only, reenterable
- Control Section: IEEDFIN3
- Page Reference: 47

IEEDFIN4: Master Scheduler -- DEFINE Command Listing Routine

This routine lists partition definitions.

- Entry: IEEDFIN4
- Exits: IEEDFIN3, IEEDFIN5
- Attributes: Read-only, reenterable
- Control Section: IEEDFIN4
- Page Reference: 45

IEEDFIN5: Master Scheduler -- DEFINE Command Message Module

This routine contains texts for operator messages required for DEFINE command processing. The message is constructed according to a code passed by the calling routine. IEEDFIN5 issues the requested message and passes control to IEEDFIN2 or the dispatcher.

- Entry: IEEDFIN5
- Exits: IEEDFIN2 or return to calling program
- Attributes: Read-only, reenterable
- Control Section: IEEDFIN5
- Page Reference: 45

IEEDFIN6: Master Scheduler Task -- Time-Slice Syntax Check Routine

This routine checks the TMSL subparameters for proper syntax.

- Entry: IEEDFIN6
- Exits: IEEDFIN2, IEEDFIN5
- Attributes: Read-only, reenterable
- Control Section: IEEDFIN6
- Page Reference: 47

IEEDFIN7: Master Scheduler -- Keyword Scan Routine

This routine checks keyword parameters for syntax errors. If a syntax error is discovered, the statement is ignored and an error message is generated. If the syntax is correct, the information is stored.

- Entry: IEEDFIN7
- Exits: IEEDFIN2, IEEDFIN3, IEEDFIN4, IEEDFIN5
- Attributes: Read-only, reenterable
- Control Section: IEEDFIN7
- Page Reference: 47

IEEDFIN8: Master Scheduler -- System Reinitialization Routine

This routine updates system control blocks and boundary boxes with the entered partition definition information.

- Entry: IEEDFIN8
- Exits: IEEDFIN9
- Attributes: Read-only, reenterable
- Control Section: IEEDFIN8
- Page reference: 47

IEEDFIN9: Master Scheduler -- Command Final Processor Routine

This routine updates the task control blocks affected by time-slicing if time-slicing is specified.

- Entry: IEEDFIN9
- Exits: IEEDFIN5
- Attributes: Read-only, reenterable
- Control Section: IEEDFIN9
- Page Reference: 47

IEESD561: SVC 34 -- START and STOP INIT Routine

This routine processes the START and STOP INIT commands.

- Entry: IEESD561
- Exit: Return to caller
- Tables/Work Areas: CSCB, PIB, M/S resident data area, CVT
- Attributes: Reenterable, Transient read- only
- Control Section: IEESD561
- Page Reference: 43

IEESD562: Master Scheduler -- Syntax Check Routine

This routine checks syntax of the command and sets internal codes for queue search, if required.

- Entry: IEESD562
- Exits: XCTL to IEESD563 for queue search, or XCTL to IEESD566 for DISPLAY active
- Attributes: Read-only, reenterable
- External References: None
- Control Section: IEESD562
- Page Reference: 44

IEESD563: Master Scheduler -- Queue Search Setup Routine

This routine determines which queue is to be searched, reads and scans the queue control record, establishes parameters for the search, and transfers control to the queue search module. IEESD563 will write out updated queue control records.

- Entry: IEESD563
- Exits: XCTL to IEESD564 to search queue; XCTL to IEESD565 at completion
- Tables/Work Areas: QCR, QMPA, CVT, CSCB
- Attributes: Read-only, reenterable
- Control Section: IEESD563
- Page Reference: 44

IEESD564: Master Scheduler -- Queue Search Module

This routine searches the work queues for the execution of the queue manipulation commands.

- Entry: IEESD564
- Exit: XCTL to IEESD563
- Tables/Work Areas: QCR, CSCB, CVT, QMPA, XSA
- Attributes: Read-only, reenterable

- Control Section: IEESD564
- Page Reference: 44

IEESD565: Master Scheduler -- Service Routine

This routine frees storage obtained by IEESD563, links to the queue manager to enqueue an entry or queue control record on SYS1.SYSJOBQE, or links to write a message.

- Entry: IEESD565
- Exit: Return to caller
- Tables/Work Areas: QMPA, CSCB, QCR, CVT
- Attributes: Read-only, reenterable
- External References: IEFQMNQ2, IEE0503D
- Control Section: IEESD565
- Page Reference: 44

IEESD566: Master Scheduler -- DISPLAY A Routine

This routine builds a table and constructs operator messages according to the processing required by a DISPLAY A command.

- Entry: IEESD566
- Exit: Return to caller (IEECIR50)
- Tables/Work Areas: QMPA, CSCB, XSA, QCR, CVT
- Attributes: Read-only, reenterable
- Control Section: IEESD566
- Page Reference: 44

IEESD567: Master Scheduler -- DISPLAY R Routine

This routine constructs operator messages according to the processing required by a DISPLAY R command.

- Entry: IEESD567
- Exit: Return to caller (IEECIR50)
- Attributes: Read-only, reenterable
- Control Section: IEESD567
- Page Reference: 45

IEESD568: Nucleus -- Master Scheduler Resident Data Area

This routine contains the master scheduler resident data area.

- Entry: IEEMSER
- Exit: None
- Attributes: Not reusable
- Control Section: IEESMER
- Page Reference: 77

IEESD571: SVC 34 -- DEFINE, MOUNT, CANCEL Routine

This routine schedules the execution of the DEFINE, MOUNT, and CANCEL (for active jobs only) commands.

- Entry: IEESD571
- Exits:
  MOUNT - XCTL to IG0103D
  DEFINE - Return to caller
  CANCEL - Active and cancellable -- Enter ABTERM to force cancel
  Active and not cancellable -- POST and mark CSCB inactive; XCTL to IEE0803D
- Tables/Work Areas: CSCB, PIB, M/S resident data area, CVT
- Attributes: Reenterable
- Control Section: IEESD571
- Page Reference: 42

IEESD590: System Task Control -- Write TIOT on Disk

This routine writes the TIOT which is used by Job Selection (IEESD510) and checks for a small partition writer.

- Entry: IEESD590
- Exits: XCTL to IEFSD510 (small partition writer) or XCTL to IEFSD591
- Tables/Work Areas: TIOT, SPIL
- Attributes: Reenterable
- Control Section: IEESD590
- Page Reference: 68

IEESD591: System Task Control -- Linker Routine

This routine transfers control between system task control and an interpreter or system output writer.

- Entry: IEESD591
- Exit: XCTL to IEEVTCTL
- Tables/Work Areas:
- Attributes: Reenterable
- Control Section: IEESD591
- Page Reference: 68

IEESD592: System Task Control -- POST Routine

This routine checks for an error indication in the CSCB. It posts the error condition or a valid condition.

- Entry: IEESD592
- Exit: XCTL to IEFSD510
- Tables/Work Areas: None
- Attributes: Reenterable
- Control Section: IEESD592
- Page Reference: 68

IEEVACTL: System Task Control -- Allocation Interface Routine

This routine sets up the interface between system task control and the I/O device allocation routine.

- Entry: IEEVACTL
- Exits: To IEFW21SD or IEEVRWTC
- Attributes: Reenterable
- Control Section: IEEVACTL
- Page Reference: 67

IEEVICLR: System Task Control -- Internal JCL Reader

This routine reads the internal job control language used in starting a reader or writer.

- Entry: IEEVICLR
- Exit: Return to caller
- Tables/Work Areas: DCBD
- Attributes: Read-only, reenterable
- Control Section: IEEVICLR

IEEVJCL: System Task Control -- JCL Edit Routine

This routine constructs the internal job control language used in the START reader and START writer command execution routines.

- Entry: IEEVJCL, from IEEVSTRT
- Exit: XCTL to IEERCTL
- Tables/Work Areas: SDT, CSCD
- Attributes: Reenterable
- Control Section: IEEVJCL

IEEVLNKT: System Task Control -- Link-Table Module

This routine contains the table of routines that is scanned by IEEVACTL as a validity check for program linking.

- Entry: IEEVLNKT
- Attributes: Non-executable
- Control Section: IEEVLNKT

IEEVMSG1: System Task Control -- Message Writer Routine

This routine writes messages to the operator as required by system task control.

- Entry: from IEEVRCTL, IEEVACTL, or IEEVTCTL
- Exit: Return to caller
- Control Section: IEEVMSG1

IEEVOMSG: System Task Control -- Message Writing Routine

This routine assembles and writes messages to the operator.

- Entry: IEEVOMSG
- Exit: Return to caller
- Control Section: IEEVOMSG

IEEVRCTL: System Task Control -- Interpreter Control Routine

This routine provides an interface between system task control and an interpreter.

- Entry: IEEVRCTL
- Exits: To IEFVH1 and IEEVACTL
- Tables/Work Areas: CVT, CSCB
- Control Section: IEEVRCTL
- Page Reference: 67

IEEVSMBA: System Task Control -- QMPA Builder

This routine constructs a queue manager parameter area (QMPA) referring to the message class queue for the use of the I/O Device Allocation routine.

- Entry: IEEVSMBA
- Exit: To IEEVACTL
- Tables/Work Areas: QMPA, LCT, SMB, IOB
- Control Section: IEEVSMBA

IEEVSMSG: System Task Control -- Message Writer Routine

This routine writes messages to the operator as required by the master scheduling task and system task control.

- Entry: IEEVSMSG, from IEEVSOPT, IEEVATT1, IEEVMSG1, or IEEVRTTC
- Exit: Return to caller
- Control Section: IEEVSMSG

IEEVSTAR: System Task Control -- Start Command Syntax Check Routine

This routine checks the syntax of a START command, and builds a start descriptor table (SDT) containing the parameters of the command.

- Entry: IEEVSTRT
- Exits: To IEEVJCL, or IEE0503D
- Tables/Work Areas: SDT, M/S Resident Data Area, CVT, M/S TIOT, UCB XSA, and CSCB.

- Attributes: Reenterable
- Control Section: IEEVSTRT
- Page Reference: 67

IEEVTCTL: System Task Control -- Termination Interface Routine

This routine initializes the necessary tables for terminating a task that was established via a START or MOUNT command.

- Entry: IEEVTCTL, from IEEVWILK or IEFW31SD
- Exit: To IEFW42SD, then return to supervisor
- Tables/Work Areas: TCB, JCT, SCT, LCT, and CSCB
- Attributes: Reenterable. Character Dependence Type C
- Control Section: IEEVTCTL

IEEVWTOR: Communications Task -- Write-to-Operator With Reply Routine

This routine processes all WTOR macro instructions.

- Entry: IGC0103E
- Exit: Return to calling program
- Tables/Work Areas: WQE, RQE, UCM, CVT
- Attributes: Reenterable
- Control Section: IGC0103E
- Page Reference: 41

IEE0303D: SVC 34 -- Translator/Chain Manipulator

This routine translates lowercase letters into uppercase, and manipulates the CSCB chain as requested by the caller of SVC 34.

- Entry: IEE0303D
- Exit: To IEE0403D, or return to caller
- Tables/Work Areas: CVT, M/S resident data area, CSCB, XSA
- Control Section: IEE0303D

IEE0403D: SVC 34 -- Router Routines

This routine identifies the command verb and passes control the appropriate routine, or manipulates the chain of CSCBs.

- Entry: IEE0403D
- Exit: Depending on command verb, via XCTL to another SVC 34 module
- Tables/Work Areas: M/S resident data area, XSA, CSCB
- Control Section: IEE0403D
- Page Reference: 42

IEE0503D: SVC34 -- Message Assembly Routine

This routine assembles and edits messages for the command scheduling routine, and writes the messages to the operator.

- Entry: IEE0503D
- Exit: Branch on register 14
- Control Section: IEE0503D

IEE0603D: SVC 34 -- SET Command Routine

This routine processes the SET command.

- Entry: IEE0603D
- Exits: To IEE0903D or IEE0503D
- Tables/Work Areas: XSA, CVT, M/S resident data area
- Attributes: Reenterable
- Control Section: IEE0603D

IEE0703D: SVC 34 -- CSCB Marking Routine

This routine schedules the execution of the STOP and MODIFY commands by finding and updating the appropriate CSCB and by issuing a POST macro instruction to the master scheduling task.

- Entry: IEE0703D
- Exits: Branch on register 14, or XCTL to IEE0803D or IEE0503D.
- Tables/Work Areas: M/S Resident Data Area, XSA, CVT, CSCB
- Attributes: Reenterable
- Control Section: IEE0703D

IEE0803D: SVC34 -- CSCB Creation Routine

This routine schedules the execution of RESET, HOLD, RELEASE, and DISPLAY commands by adding a CSCB to the CSCB chain and issuing a POST macro instruction to the master scheduling task. The CANCEL command is also processed if the job is active.

- Entry: IEE0803D
- Exit: Branch on register 14
- Tables/Work Areas: XSA, M/S resident data area, CVT and CSCB
- Attributes: Reenterable
- Control Section: IEE0803D

IEE0903D: SET Command Handler

This routine processes the date and time operands of the SET command.

- Entry: IEAQOT00
- Exit: SVC 3
- Tables/Work Areas: CVT
- Attributes: Reenterable, supervisor state, disabled for system interrupts, transient
- Control Section: IEAQOT00

IEE1103D: SVC 34 -- VARY and UNLOAD Routines

This routine schedules the execution of the VARY and UNLOAD commands.

- Entry: IEE1103D
- Exits: Branch on register 14, or XCTL to IEE0503D
- Tables/Work Areas: M/S resident data area, XSA, CVT, UCB
- Attributes: Reenterable, read-only, self-relocating
- Control Section: IEE1103D

IEE1203D: SVC 34 -- Reply Processor

This routine checks the validity of the operator's reply command, and moves the operator's reply (if valid) to the buffer of the user that issued the respective WTOR.

- Entry: IEE1203D
- Exit: Return to caller
- Tables/Work Areas: CVT, UCM, WQE, RQE, CXSA
- Attributes: Reenterable
- Control Section: IEE1203D
- Page Reference: 42

IEE1403D: SVC 34 -- HALT Routine

This routine schedules the execution of the HALT command by adding a CSCB to the CSCB chain and by issuing a POST macro instruction to the master scheduling task.

- Entry: IEE1403D
- Exit: IFBSTAT
- Tables/Work Areas: XSA, M/S resident data area, CVT, and CSCB
- Attributes: Reenterable
- Control Section: IEE1403D

IEE2103D: SVC 34 -- Message Assembly Routine

This routine assembles and edits messages for the command scheduling routine, and writes the messages to the operator.

- Entry: IEE2103D
- Exit: Branch on register 14
- Control Section: IEE2103D

IEFACTFK: Termination -- User Dummy Accounting Routine

This routine takes the place of the user's accounting routine when a user accounting routine was specified at system generation, but none was supplied.

- Entry: IEFACTLK
- Exit: Return to caller
- Control Section: IEFACTLK

IEFACTLK: Termination -- User Accounting Routine Linkage Routine

This routine provides linkage between the termination routine and the user's accounting routine. It also sets up the required parameter list -- including the execution time of the job step -- and reads the first record of the account control table.

- Entry: IEFACTLK
- Exits: To user's accounting routine, return to caller.
- Tables/Work Areas: LCT, JCT, SCT, JACT, SACT, QMPA
- Control Section: IEFACTLK

IEFACTRT: Termination -- Dummy Accounting Routine

This routine takes the place of the user-supplied accounting routine.

- Entry: IEFACTRT
- Exit: Return to caller
- Control Section: IEFACTRT

IEFAVFAK: I/O Device Allocation -- Linkage to IEFXV001

This routine passes control to the AVR routine (IEFXV001) via and XCTL macro instruction.

- Entry: IEFXV001
- Exit: XCTL to IEFXV001
- Control Section: IEFXV001

IEFCVFAK: I/O Device Allocation -- Linkage to IEFMCVOL

This routine passes control to Mount Control-Volume Routine IEFMCVOL via an XCTL macro instruction to one of three entry points, IEFCVOL1, IEFCVOL2, or IEFCVOL3.

- Entries: IEFCVOL1, IEFCVOL2, IEFCVOL3
- Exits: XCTL to IEFCVOL1, IEFCVOL2, IEFCVOL3
- Control Section: IEFCVOL1

IEFIDMPM: Termination -- Message Module

This routine contains the messages used by the Indicative Dump routine.

- Entry: IEFIDMPM
- Attributes: Non-executable
- Control Section: IEFIDMPM

IEFMCVOL: I/O Device Allocation -- Mount Control-Volume Routine

This routine will have a control volume mounted when a data set called for in a job step cannot be located on any currently mounted control volume.

- Entries: IEFCVOL1, IEFCVOL2, IEFCVOL3
- Exits: IEFVM1, IEFVMCVL, IEFVM6, IEFYN (IEFW41SD)
- Tables/Work Areas: LCT, JCT, SCT, SIOT, JFCB, VOLT, QMPA, UCB
- Attributes: Reusable
- Control Sections: IEFCVOL1, IEFCVOL2, IEFCVOL3

IEFORMAT: Queue Management -- Queue Formatting Routine

This routine places the work queue data set in the format required by the MFT queue management routines.

- Entry: IEFORMAT, from IEFSD055
- Exit: Return to IEFSD055
- Tables/Work Areas: DCB, DEB
- Attributes: Reusable
- Control Section: IEFORMAT
- Page Reference: 49

IEFQAGST: Queue Management -- Assign/Start Routine

This routine sets up an ECB/IOB and prepares the queue manager parameter area for the assign routine.

- Entry: IEFQAGST
- Exit: Return to caller
- Tables/Work Areas: Q/M resident data area, QMPA, CVT
- Attributes: Reenterable
- Control Section: IEFQAGST
- Page Reference: 50

IEFQASGQ: Queue Management -- Assign Routine

This routine assigns records to a queue entry and assigns logical tracks as required.

- Entry: IEFQASGN
- Exit: Return to caller
- Tables/Work Areas: Q/M resident data area, QMPA, CVT
- Attributes: Reenterable
- Control Sections: IEFQASGN, IEFQASNM
- Page Reference: 51

IEFQBVMS: Queue Management -- Control Routine

This routine inspects the function code in the queue manager parameter area and, on the basis of this code, branches to the appropriate queue management routines.

- Entry: IEFQMSSS
- Exits: To IEFQAGST, IEFQMRAW, IEFQMNQQ, or IEFQASGQ, return to caller
- Tables/Work Areas: QMPA
- Attributes: Reenterable
- Control Section: IEFQMSSS

IEFQDELQ: Queue Management -- Delete Routine

This routine makes those logical tracks assigned to a queue entry available for assignment to other queue entries.

- Entry: IEFQDELE
- Exit: Return to caller
- Tables/Work Areas: LTH, QMPA, QCR, Q/M resident data area, CVT
- Attributes: Reenterable
- Control Section: IEFQDELE
- Page Reference: 53

IEFQMDQQ: Queue Management -- Dequeue Routine

This routine removes the highest priority entry from an input queue or a system output queue.

- Entry: IEFQMDQ2
- Exit: Return to caller
- Tables/Work Areas: CVT, Q/M resident data area, QCR, LTH
- Attributes: Reenterable
- Control Section: IEFQMDQ2
- Page Reference: 53

IEFQMDUM: Queue Management -- Dummy Module

This routine prevents the occurrence of an unresolved external reference to module IEFQMSSS during system generation.

- Entry: IEFQMDUM
- Attributes: Non-Executable
- Control Section: IEFQMSSS

IEFQMLK1: Queue Management -- Branch Routine

This routine branches to the appropriate queue management routine on the basis of an assign or read/write function code issued by an initiator.

- Entry: IEFQMSSS
- Exits: To IEFQASGQ or IEFQMRAW
- Tables/Work Areas: QMPA
- Attributes: Reenterable
- Control Section: IEFQMSSS

IEFQMNQQ: Queue Management -- Enqueue Routine

This routine places an entry in an input queue or an output queue at the requested priority.

- Entry: IEFQMNQ2
- Exit: Return to caller
- Tables/Work Areas: CVT, Q/M resident data area, QMPA, QCR, LTH
- Attributes: Reenterable
- Control Section: IEFQMNQ2

IEFQMRAW: Queue Management -- Read/Write Routine

This routine performs the conversion of a TTR into a MBBCCHHR and reads or writes up to 15 records of the work queue data set.

- Entry: IEFQMRAW
- Exit: Return to caller
- Tables/Work Areas: Q/M resident data area, QMPA, CVT, IOB/ECB
- Attributes: Reenterable
- Control Section: IEFQMRAW

IEFQMUNQ: Queue Management -- Unchain Routine

This routine removes a task from the queue management no-work chain.

- Entry: IEFQMUNC
- Exit: Return to caller

- Tables/Work Areas: CVT, Q/M resident data area, QCR
- Attributes: Reenterable
- Control Section: IEFQMUNC

IEFQRESD: Queue Management -- Resident Main Storage Reservation Module

This routine reserves 140 bytes of resident main storage for the queue-management-opened DCB/DEB and the master queue control record at nucleus initialization time.

- Attributes: Non-executable
- Control Section: IEFJOB

IEFSD017: Termination -- System Output Interface Routine

This routine provides an interface between the termination entry routine and system output processing.

- Entry: IEFSD017
- Exit: To IEFSD42Q
- Control Section: IEFSD017

IEFSD055: Queue Management -- Queue Initialization Routine

This routine constructs a resident DEB/DCB, passes control to the queue formatting routine or the first phase of system restart, initializes the queue manager

resident data area, and (if required) passes control to the second phase of the system restart routine.

- Entry: IEFSD055, from IEFQINTZ
- Exits: To IEFORMAT, IEF300SD, or IEF304SD
- Attributes: Reusable
- Control Section: IEFSD055
- Page Reference: 49

IEFSD070: System Output Writer -- Data Set Writer Interface Routine

This routine passes control to the standard data set writer or to the user-supplied data set writer routine.

- Entry: IEFSD070
- Exits: To IEFSD087 or user-supplied routine via LINK, or to IEFSD171 via XCTL
- Attributes: Reenterable
- Control Section: IEFSD070
- Page Reference: 65

IEFSD078: System Output Writer -- Linker Routine

This routine determines whether the record obtained from the output queue entry is a DSB or SMB, and passes control, accordingly, to the DSB or SMB processor.

- Entry: IEFSD078
- Exits: To IEFSD085, IEFSD086, or IEFSD079
- Attributes: Reenterable
- Control Section: IEFSD078

IEFSD079: System Output Writer -- Link to Queue Manager Delete Routine

This routine passes control to the delete routine to delete the current output queue entry.

- Entry: IEFSD079
- Exits: To IEFQDELQ and IEFSD082
- Tables/Work Areas: QMPA
- Attributes: Reenterable
- Control Section: IEFSD079
- Page Reference: 65

IEFSD080: System Output Writer -- Initialization Routine

This routine initializes the system output writer by obtaining main storage for a parameter list and the output DCB, and opening the output DCB.

- Entry: IEFSD080
- Exit: To IEFSD081
- Tables/Work Areas: DCB, CSCB, TIOT, JFCB
- Attributes: Reenterable
- Control Section: IEFSD080

IEFSD081: System Output Writer -- Class Name Setup Routine

This routine obtains main storage for, and initializes, a list of ECB pointers, ECBs, and queue management communication elements, depending on the system output classes specified for the writer.

- Entry: IEFSD081
- Exit: To IEFSD082
- Tables/Work Areas: CSCB, ECB
- Attributes: Reenterable
- Control Section: IEFSD081

IEFSD082: System Output Writer -- Main Logic Routine

This routine obtains main storage for QMPAs and internal work areas, dequeues output queue entries, checks for operator commands, and passes control to the appropriate routine.

- Entry: IEFSD082
- Exits: IEFSD083, IEFSD084, IEFSD078
- Tables/Work Areas: CSCB, ECB
- Attributes: Reenterable
- Control Section: IEFSD082

IEFSD083: System Output Writer -- Command Processing Routine

This routine processes MODIFY and STOP commands that apply to the writer.

- Entry: IEFSD083

- Exits: To IEFSD081 or IEEVTCTL
- Tables/Work Areas: CSCB, DCB, QMPA, ECB
- Attributes: Reenterable
- Control Sections: IEFSD083, IEFSD83M

IEFSD084: System Output Writer -- Wait Routine

This routine waits for an entry to be enqueued in an output queue corresponding to a class available to the writer.

- Entry: IEFSD084
- Exit: To IEFSD082
- Attributes: Reenterable
- Control Section: IEFSD084
- Page Reference: 65

IEFSD085: System Output Writer -- DSB Handler Routine

This routine initializes for data set processing, and informs the operator of the pause option in effect.

- Entry: IEFSD085, IEF085SD, or IEF850SD
- Exit: To IEFSD171
- Attributes: Reenterable
- Control Sections: IEFSD085, IEFSD85M
- Page Reference: 66

IEFSD086: System Output Writer -- SMB Handler

This routine initializes for message processing, and extracts each message from the current SMB.

- Entry: IEFSD086, IEF086SD
- Exits: To IEFSD088, IEFSD089, IEFQMNQQ, IEFQMRAW, IEFSD085, IEFSD078
- Tables/Work Areas: SMB, UCB, QMPA, TIOT, CSCB, TCB
- Attributes: Reenterable
- Control Sections: IEFSD086, IEFSD86M

IEFSD087: System Output Writer -- Standard Writer Routine

This routine gets records from a data set.

- Entry: IEFSD087
- Exits: To IEFSD088, IEFSD089, IEFSD078
- Tables/Work Areas: DCB
- Attributes: Reenterable
- Control Sections: IEFSD087, IEFSD87M
- Page Reference: 66

IEFSD088: System Output Writer -- Transition Routine

This routine handles the transition between messages and data sets, and between data sets.

- Entry: IEFSD088
- Exit: To IEFSD089
- Tables/Work Areas: DCB
- Attributes: Reenterable
- Control Section: IEFSD088

IEFSD089: System Output Writer -- Put Routine

This routine formats records as required and issues PUT macro instructions to write them on the output unit.

- Entry: IEFSD089
- Exit: To IEFSD088
- Tables/Work Areas: DCB
- Attributes: Reenterable
- Control Sections: IEFSD089, IEFSD89M

IEFSD094: System Output Writer -- Job Separator Routine

This routine prints or punches a job name and system output class designation on the writer's output device.

- Entry: IEFSD094
- Exits: To IEFSD088, IEFSD089, IEFSD095, IEFSD078
- Attributes: Reenterable
- Control Section: IEFSD094

IEFSD095: System Output Writer -- Print Line Routine

This routine constructs the block letters used to separate jobs processed by a

system output writer when the output data set is to be printed.

- Entry: IEFSD095
- Exit: Return to caller
- Attributes: Reenterable
- Control Section: IEFSD095

IEFSD096: System Output Writer -- Message Module

This routine contains message headers and texts for messages to the operator.

- Entry: IEFSD096
- Attributes: Non-executable
- Control Section: IEFSD096

IEFSD097: I/O Device Allocation -- Wait for Space Decision Routine

This routine makes the decision whether to wait for direct access space, and provides an interface with the I/O device allocation space request routine so that retry and additional recovery passes may be made.

- Entry: IEFSD097
- Exit: Branch on register 14
- Tables/Work Areas: LCT, TIOT, UCB
- Attributes: Read-only, reenterable
- Control Section: IEFSD097

IEFSD171: System Output Writer -- Data Set Delete Routine

This routine obtains records from an output queue entry, and deletes system output data sets.

- Entry: IEFSD071
- Exits: To IEFQMNQ2, IEF850SD, IEF086SD, IEFSD078, or IEFQMRAW
- Tables/Work Areas: DCB, SMB, UCB, CVT, QMPA, TIOT, CSCB, TCB
- Attributes: Reenterable
- Control Sections: IEFSD071, IEFSD71M

IEFSD195: I/O Device Allocation -- Wait for Deallocation Routine

This routine provides the I/O device allocation routine with the ability to wait for deallocation to occur during the execution of another task, when allocation cannot be completed because of current allocations.

- Entry: IEFVAWAT
- Exit: Return to caller
- Tables/Work Areas: JCT, SCT, SIOT, LCT, ECG, CSCB
- Attributes: Read-only, reenterable
- Control Section: IEFSD095

IEFSD21Q: I/O Device Allocation -- Allocation Entry Routine

This routine provides an interface for entry to the I/O device allocation routine operating in a multiprogramming environment.

- Entry: IEFW21SD
- Exits: To IEFVK, IEFVM or IEFWD000
- Tables/Work Areas: JCT, LCT, SCT, SMB, QMPA, CVT
- Attributes: Read-only, reenterable
- Control Section: IEFWLISD

IEFSD22Q: Termination Routine -- Step Terminate Exit Routine

This routine provides an interface between the termination routine and the step delete or alternate step delete routine when a step has been terminated.

- Entry: IEFW22SD
- Exit: Return to caller of termination routine
- Tables/Work Areas: JCT, SCT, SMB, LCT, QMPA, ECB
- Attributes: Read-only, reenterable
- Control Section: IEFW22SD
- Page Reference: 64

IEFSD300: System Restart -- Initialization Routine

This routine reads all QCRs and logical track header records into main storage, builds tables A, B, and C, and removes from Table A all the LTH entries corresponding to logical tracks in the free-track queue or in one of the other queues.

- Entry: IEFSD300

- Exit: To IEFSD301
- Tables/Work Areas: System restart work area, Table A, Table B, Table C
- Attributes: Reenterable
- Control Section: IEFSD300

IEFSD301: System Restart -- Purge Queue Construction Routine

This routine searches Table A for the last ITH corresponding to each queue entry, determines the type of entry, and constructs the purge queue.

- Entry: IEFSD301
- Exit: To IEFSD302
- Tables/Work Areas: System restart work area, Table A, Table C purge queue, interpreter jobnames table
- Attributes: Reenterable
- Control Section: IEFSD301

IEFSD302: System Restart -- Jobnames Table Routine

This routine removes from Table A all logical tracks assigned to dequeued input or RJE queue entries, and builds a table of jobnames for incomplete input and RJE queue entries and dequeued input queue entries.

- Entry: IEFSD302
- Exit: To IEFSD303
- Tables/Work Areas: System restart work area, Table A, Table C, and the interpreter/initiator jobnames table
- Attributes: Reenterable
- Control Section: IEFSD302

IEFSD303: System Restart -- Delete Routine

This routine creates a queue entry of the remaining logical tracks and deletes that entry, thus assigning those tracks to the free-track queue.

- Entry: IEFSD303
- Exit: Return to caller
- Tables/Work Areas: System restart work area, QMPA, Table A
- Attributes: Reenterable
- Control Section: IEFSD303

IEFSD304: System Restart -- Scratch Data Sets Routine

This routine informs the operator of the names of jobs being processed by an interpreter, and scratches temporary data sets generated for incomplete input queue entries.

- Entry: IEFSD304
- Exits: To IEFSD055, IEFSD308
- Tables/Work Areas: CVT, UCB address look-up table
- Attributes: Reenterable
- Control Section: IEFSD304

IEFSD305: System Restart -- Reenqueue Routine

This routine dequeues the entries in the purge queue and reenqueues them in the appropriate input or output queue and informs the operator of the names of jobs in the process of initiation.

- Entry: IEFSD305
- Exit: IEFSD304
- Tables/Work Areas: System restart work area, purge queue, JCT, SCT, JFCB, DSB, SCD, SIOT.
- Attributes: Reenterable
- Control Section: IEFSD305

IEFSD308: System Restart -- Scratch Data Sets Routine

This routine scratches the temporary data sets generated for incomplete input queue entries.

- Entry: IEFSD308
- Exit: Return to caller
- Tables/Work Areas: DSCB, DCB, UCB, CVT, VTOC, DEB
- Attributes: Reenterable
- Control Section: IEFSD308

IEFSD31Q: Termination Routine -- Job Termination Exit Routine

This routine provides an interface between the termination routine and the step delete or alternate step delete routine when the last step of a job has been terminated.

- Entry: IEFW31SD
- Exit: Return to caller of termination routine
- Tables/Work Areas: JCT, SCT, SMB, QMPA, ECB, CVT, M/S resident data area
- Attributes: Read-only, reenterable
- Control Section: IEFW31SD

IEFSD310: System Restart -- TTR and NN to MBBCCHHR Conversion Routine

This routine converts a relative record address (NN) or a relative track and record address (TTR) to an actual disk address (MBBCCHHR).

- Entry: IEFSD310
- Exit: Return to caller
- Tables/Work Areas: CVT
- Attributes: Reenterable
- Control Section: IEFSD310

IEFSD311: Queue Management -- Message Module

This routine contains the messages required by the queue initialization routine (IEFSD055).

- Entry: IEFSD311, SD55MSG1, SD55MSG2, SD55MSG3
- Attributes: Non-executable
- Control Section: IEFSD311

IEFSD312: System Restart -- Message Module

This routine contains the messages required by the system restart routines.

- Entry: IEFSD312, SD304MG1, SD304MG2, SD305MG1
- Attributes: Non-executable
- Control Section: IEFSD312

IEFSD41Q: I/O Device Allocation -- Allocation Exit Routine

This routine provides an interface for exit from the I/O device allocation routine operating in a multiprogramming environment.

- Entry: IEFW41SD, IEFW1FAK, IEFW2FAK
- Exits: To IEFVM, or return to caller
- Tables/Work Areas: JCT, ICT, SCT, SMB, QMPA
- Attributes: Read-only, reenterable
- Control Section: IEFW41SD

IEFSD42Q: Termination Routine -- Termination Entry Routine

This routine provides an interface for entry to the termination routine operating in a multiprogramming environment.

- Entry: IEFW42SD
- Exit: To IEFYN
- Tables/Work Areas: JCT, SCT, SMB, LCT, TIOT
- Attributes: Read-only, reenterable
- Control Section: IEFW42SD

IEFSD510: Initiator -- Job Selection Routine

This routine selects a system or problem program job. This module executes only in a large (scheduler-size) partition.

- Entry: IEFSD510
- Exits: Branch to IEFSD511 or IEFSD515, XCTL to IEE596SD or IEFSD531
- Tables/Work Areas: LOT block, CSCB, SPIL, CVT, TCB, PIB
- Attributes: Read-only, reenterable
- Control Section: IEFSD510
- External References: IEFQMDQQ, IEFQMUNC
- Page Reference: 57

IEFSD511: Initiator -- Job Initiation Routine

This routine initializes information pertaining to a job.

- Entry: IEFSD511
- Exit: Branch to IEFSD541
- Tables/Work Areas: Life-of-Task Block, CSCB, JCT, SCT, SCD, PIB, IOB2
- Attributes: Read-only, Reenterable
- Control Section: IEFSD511
- External References: IEFQMRAW

- Page Reference: 62

IEFSD512: Initiator -- Step Initiation Routine

This routine passes control to allocation as a closed subroutine via a LINK macro instruction. If an allocation error occurs, it passes control to the Alternate Step Deletion routine. Otherwise, it continues normally and schedules a job step.

- Entry: IEFSD512
- Exits: Branch to IEFSD513 or IEFSD516
- Tables/Work Areas: LOT Block, JCT, SCT, APL, TIOT, CSCB, IOB1, IOB2
- Attributes: Read-only, reenterable
- Control Section: IEFSD512
- External References: IEFQMRAW, IEFSD556, IEFSD514
- Page Reference: 62

IEFSD513: Initiator -- Problem Program Interface

This routine prepares the partition for problem program execution by moving the TIOT to the highest available storage area.

The routine also opens JOBLIB and FETCH DCBs, if required. A final check is made to determine if a CANCEL command has been received for the job before the problem program is brought into the partition and given control. If scheduling was performed for a small partition, IEFSD513 communicates with the small partition.

- Entry: IEFSD513
- Exits: XCTL to problem program, ABEND, or Branch to IEFSD510
- Tables/Work Areas: LOT Block, Transfer Parameter List, TIOT, User's Parameter List, TCB, CVT, PIB, CSCB, SPIL, APL, JCT, SCT, DCB
- Attributes: Read-only, reenterable
- Control Section: IEFSD513
- Page Reference: 63

IEFSD514: Queue Management -- Table Breakup Routine

This routine reads and writes tables which may be required by the job scheduler. The routine breaks the tables into 176-byte records, writes the records on disk, and retrieves the records from disk to reconstruct the tables in main storage.

- Entry: IEFSD514
- Exit: Return to caller
- Tables/Work Areas: QMPA, TBR Parameter List
- Attributes: Read-only, reenterable
- Control Section: IEFSD514
- External References: IEFQASGN, IEFQMRAW
- Page Reference: 54

IEFSD515: Initiator -- Step Deletion Routine

This routine retrieves the TIOT and Life-of-Task Block from disk, reads in the JCT and SCT, and branches to termination, which is used as a closed subroutine. It also reads in the SCT for the next step to be scheduled, if required.

- Entry: IEFSD515, SMALTERM, or GO
- Exits: XCTL to IEFSD512 or Branch to IEFSD517 or IEFSD510
- Tables/Work Areas: Life-of-Task Block, Terminate Parameter List, CVT, TCB, PIB, IOB, CSCB, DCB, JCT, SCT, SPIL
- Attributes: Read-only, reenterable
- Control Section: IEFSD515
- External References: IEFQMRAW, IEFSD514, IEFSD42Q, IEFSD598
- Page Reference: 64

IEFSD516: Initiator -- Alternate Step Deletion Routine

This routine provides an interface with termination when an allocation error occurs during step initiation. Termination is used as a closed subroutine. If required, this routine reads the SCT of the next step to support job flushing.

- Entry: IEFSD516
- Exits: Branch to IEFSD512 or IEFSD517
- Tables/Work Areas: Life-of-Task block, CSCB, Terminate Parameter List, SCT
- Attributes: Read-only, reenterable
- Control Section: IEFSD516

- External References: IEFQMRAW, IEFSD42Q
- Page Reference: 64

IEFSD517: Initiator -- Job Deletion Routine

This routine deletes the disk queue entry for a terminated job and unchains and deletes the CSCB for the job.

- Entry: IEFSD517
- Exit: Branch to IEFSD510
- Tables/Work Areas: CSCB, Life-of-Task block, SPIL
- Attributes: Read-only, reenterable
- Control Section: IEFSD517
- External References: IEFQDELE, IEFSD598
- Page Reference: 64

IEFSD530: Interpreter -- Transient Reader Suspend Routine

This routine closes the reader input data set and procedure library, and saves data required to restore the reader.

- Entry: IEFSD530
- Exit: Return to caller
- Tables/Work Areas: IWA, TIOT, LWA, QMPA, CVT, UCB, MSRC, PIB, CSCB
- Attributes: Read-only, reenterable
- Control Section: IEFSD530
- External References: IEFSD514, IEFQMRAW, IEFQASNM, IEFQASGN
- Page Reference: 56

IEFSD531: Interpreter -- Transient Reader Restore Routine

This routine restores the information required to "restart" a transient reader after it has been suspended. It reopens the reader input data set and procedure library.

- Entry: IEFSD531
- Exit: XCTL to IEFVHCB
- Tables/Work Areas: IWA, TIOT, QMPA, CVT, UCB, MSRC, PIB, CSCB
- Attributes: Read-only, reenterable
- Control Sections: IEFSD531, IEFPH2
- External References: IEFSD514, IEFQMRAW, IEFQASNM, IEFQASGN
- Page Reference: 56

IEFSD532: Interpreter -- Transient Reader Suspend Tests

This routine determines the status of a transient reader. IEFSD532 receives control from IEFVHH after a job has been enqueued.

- Entry: IEFSD532
- Exits: Branches to IEFVHN or IEFSD530, or IEFVHHB
- Tables/Work Areas: IWA, LWA, QMPA, PIB, CVT
- Attributes: Read-only, reenterable
- Control Section: IEFSD532

IEFSD533: Interpreter -- Interface Routine

This routine provides an interface between the reader/interpreter and system task control.

- Entry: IEFIRC
- Exits: LINK to IEFVH1 and RETURN to IEESD591
- Tables/Work Areas: CSCB, CVT, QMPA
- Attributes: Reenterable
- Control Section: IEFSD534

IEFSD534: System Task Control -- LPSW Routine

This routine places system task control in problem program mode by loading a PSW.

- Entry: IEFSD534
- Exit: IEFVSTRT
- Tables/Work Areas: None
- Attributes: Reenterable
- Control Section: IEFSD534

IEFSD535: System Task Control -- Problem Program Mode Routine

This routine puts system task control in problem program mode for ABEND.

- Entry: IEFSD535
- Exit: IEEVTCTL
- Tables/Work Areas: None
- Attributes: Reenterable
- Control Section: IEFSD535

IEFSD536: Interpreter -- Operator Message Routine

This routine writes a message to the operator when an I/O error or CPO full condition has occurred. The routine also sets proper indicators to cause a cleanup of the current job.

- Entry: IEFVHR
- Exits: Return to caller, XCTL to IEFVHN, or LINK to IEFSD308
- Tables/Work Areas: IWA, JCT, LWA, UCB, CVT, PIB, CSCB, Master Scheduler resident data area
- Attributes: Read-only, reenterable
- Control Section: IEFVHR
- Page Reference: 56

IEFSD537: Interpreter -- Linkage Module

This routine provides an interface between system task control and a reader. It also frees the interpreter entrance list (NEL) and associated areas if a reader is being terminated or suspended.

- Entry: IEFSD537
- Exits: LINK to IEFVH1, or IEFSD531, or Return to system task control
- Tables/Work Areas: NEL
- Attributes: Read-only, reenterable
- Control Section: IEFSD537

IEFSD540: Initiator -- Linkage to IEFSD541

This routine provides an interface linkage to IEFSD541 via an XCTL macro instruction.

- Entry: IEFSD540
- Exit: XCTL to IEFSD541
- Tables/Work Areas: Same as caller
- Attributes: Read-only, reenterable
- Control Section: IEFSD540

IEFSD541: Initiator -- Data Set Integrity

This routine enqueues on explicit data sets and thus prevents concurrent, and impairing, access between tasks.

- Entry: IEFSD541
- Exit: Branch to IEFSD512
- Tables/Work Areas: LOT Block, IOB1, IOB2, JCT, SCT, CSCB, SPIL, DSENQ Table, Minor Name List, ENQ supervisor list.
- Attributes: Read-only
- Control Section: IEFSD541
- External References: IEFQMRAW
- Page Reference: 62

IEFSD551: I/O Device Allocation -- Linkage to IEFXJIMP

This routine provides an interface linkage to IEFXJIMP via an XCTL macro instruction in the 30K design package.

- Entry: IEFV15XL
- Exit: XCTL to IEFXJIMP
- Tables/Work Areas: Same as caller
- Attributes: Read-only, reenterable
- Control Section: IEFV15XL

IEFSD552: I/O Device Allocation -- Linkage to IEFXJIMP

This routine provides an interface linkage to IEFXJIMP via an XCTL macro instruction in the 30K design package.

- Entry: IEFXJX5A
- Exit: XCTL to IEFXJIMP
- Tables/Work Areas: Same as caller
- Attributes: Read-only, reenterable
- Control Section: IEFXJX5A

IEFSD553: Initiator -- Linkage to IEFSD512

This routine provides a linkage to IEFSD512 via an XCTL macro instruction.

- Entry: IEFSD512
- Exit: XCTL to IEFSD512

- Tables/Work Areas: Same as caller
- Attributes: Read-only, reenterable
- Control Section: IEFSD512

IEFSD554: Initiator -- Linkage to IEFSD516

This routine provides a linkage to IEFSD516 via an XCTL macro instruction.

- Entry: IEFSD554
- Exit: XCTL to IEFSD516
- Tables/Work Areas: Same as caller
- Attributes: Read-only, reenterable
- Control Section: IEFSD554

IEFSD555: Initiator -- Linkage to IEFSD510

This routine provides linkage to IEFSD510 via an XCTL macro instruction.

- Entry: IEFSD555
- Exit: XCTL to IEFSD510
- Tables/Work Areas: Same as caller.
- Attributes: Read-only, reenterable
- Control Section: IEFSD555

IEFSD556: Initiator -- Set Problem Program State Return

This routine establishes the allocation routine in a problem program state, upon entry.

- Entry: IEFSD556
- Exit: LPSW to IEFW21SD
- Tables/Work Areas: Same as caller.
- Attributes: Read-only, reenterable
- Control Section: IEFSD556

IEFSD557: I/O Device Allocation -- Interface Routine

This routine provides an interface between system task control and allocation.

- Entry: IEFW21SD
- Exit: IEFWSD21
- Tables/Work Areas: ECB, IOB
- Attributes: Reenterable
- Control Section: IEFSD557

IEFSD558: Initiator -- Linkage to IEFSD511

This routine provides a linkage to IEFSD511 via an XCTL macro instruction.

- Entry: IEFSD558
- Exit: IEFSD511
- Attributes: Read-only, reenterable
- Control Section: IEFSD558

IEFSD559: Initiator -- Linkage to IEFSD515

This routine provides a linkage to IEFSD515 via an XCTL macro instruction.

- Entry: SMALTERM
- Exit: IEFSD515
- Attributes: Read-only, reenterable
- Control Section: IEFSD559

IEFSD567: Nucleus -- Device-End Interrupt Handler Routine

This routine handles unsolicited device-end interrupts from a disk storage unit.

- Entry: IEFSD567
- Exit: Return to caller
- Tables/Work Areas: None
- Attributes: Reenterable
- Control Section: IEFSD567
- External Reference: Communications Task TCB

IEFSD569: Master Scheduler -- Initialization Routine

This routine issues the READY message and formats the job queue, as well as typing out the automatic commands and invoking processing of the automatic commands. This routine establishes partitioning of main storage at system initialization and readies the partitions for the START command. This routine is called out at system generation by the macro, SGIEE0VV.

- Entry: IEFSD569
- Exit: Branch to dispatcher
- Attributes: Read-only, reenterable

- Control Section: IEFSD569
- Page Reference: 43

IEFSD572: Queue Management -- Interpreter/ Queue Manager Interlock Routine

This routine determines if a possible interlock condition exists between the queue manager and the reader. The routine issues a message requesting the operator to reply with either WAIT, to wait for space to be freed, or CANCEL, to cancel the job.

- Entry: IEFSD572
- Exits: ABEND, or return to caller
- Attributes: Read-only, reenterable
- Control Section: IEFSD572, IEFSD573
- External Reference: IEFQDELQ
- Page Reference: 53

IEFSD587: System Task Control -- Linkage to IEFSD535

This routine provides a linkage to IEFSD535 via a LINK macro instruction.

- Entry: IEFSD587
- Exit: IEFSD535
- Attributes: Read-only, reenterable
- Control Section: IEFSD587

IEFSD588: System Task Control -- Linkage to IEE534SD

This routine links to IEE534SD to bring the suspended reader into the assigned partition so that upon return, the initiator will be in supervisor state.

- Entry: IEFSD588
- Exit: LINK to IEE534SD
- Tables/Work Areas: Same as caller
- Attributes: Read-only, reenterable
- Control Section: IEFSD588

IEFSD589: Initiator -- Linkage to IEESD534

This routine links to system task control so that upon return, the initiator will be in supervisor state.

- Entry: IEFSD589
- Exit: LINK to IEFSD534
- Tables/Work Areas: Same as caller
- Attributes: Read-only, reenterable
- Control Section: IEFSD589

IEFSD597: Initiator -- Shared DASD ENQ/DEQ Purge Routine

This routine is the purge routine for systems that include the shared DASD feature. In addition to purging all resources enqueued by a job step, but not dequeued, IEFSD597 also releases reserved devices.

- Entry: IEFSD597
- Exit: Return to caller
- Tables/Work Areas: Major QCB, Minor QCB, QEL, TCB, SVRB, CVT, ABTERM
- Attributes: Read-only, reenterable, disabled
- Control Section: IEFSD597

IEFSD598: Initiator -- ENQ/DEQ Purge Routine

This routine purges all resources enqueued by a job step, but not dequeued.

- Entry: IEFSD598
- Exit: Return to caller
- Tables/Work Areas: Major QCB, Minor QCB, QEL, TCB, SVRB, CVT, ABTERM
- Attributes: Read-only, Reenterable, disabled
- Control Section: IEFSD598
- Page Reference: 64

IEFSD599: Initiator -- Small Partition Module

This routine provides an interface with the scheduler in a large partition to initiate and terminate small partitions.

- Entry: IEFSD599,SMALLGO
- Exits: ABEND, or XCTL to problem program or writer
- Tables/Work Areas: SPIL, allocate parameter list (APL)
- Attributes: Read-only, reenterable
- Control Section: IEFSD599
- External Reference: IEFQMUNC

- Page Reference: 60

IEFVDA: Interpreter -- DD Statement Processor

This routine constructs and adds entries to a JFCB and SIOT from the complete logical DD statement in the internal text buffer.

- Entry: IEFVDA
- Exit: To IEFVHF
- Tables/Work Areas: IWA, LWA, SIOT, JFCB, JCB, SCT
- Attributes: Read-only, reenterable
- Control Section: IEFVDA

IEFVDBSD: Interpreter -- Data Set Name Table Construction Routine

This routine creates a data set name table.

- Entry: IEFVDBSD
- Exit: To IEFVDA
- Attributes: Reenterable
- Control Section: IEFVDBSD

IEFVEA: Interpreter -- EXEC Statement Processor

This routine constructs or updates an SCT, and, if necessary, a joblib JFCB and SIOT from the complete logical EXEC statement in the internal text buffer.

- Entry: IEFVEA, from IEFVFA
- Exit: To IEFVHF
- Tables/Work Areas: IWA, EXEC work area, interpreter key table, JCT, SCT, SIOT, QMPA, procedure override table.
- Attributes: Read-only, reenterable
- Control Section: IEFVEA

IEFVFA: Interpreter -- Scan Routine

This routine scans the card image of a JOB, EXEC, or DD statement, performs error checking of JCL syntax, builds internal text, and, when a complete logical statement (including continuations and overrides) has been scanned, passes control to the appropriate statement processor.

- Entry: IEFVFA
- Exits: To IEFVGM, IEFVHQ, IEFVHF, IEFVJA, IEFVDA, IEFVEA
- Tables/Work Areas: IWA, scan routine work area, interpreter key table, QMPA, internal text buffer, scan dictionary.
- Attributes: Read-only, reenterable
- Control Section: IEFVFA

IEFVFB: Interpreter -- Symbolic Parameter Processing Routine

This routine processes symbolic parameters by creating symbolic parameter table buffer entries to assign values to symbolic parameters, and extracts those values and places them in the intermediate text buffer when a symbolic parameter is used.

- Entry: IEFVFB
- Exit: Return to caller
- Tables/Work Areas: IWA, LWA SYMBUF, Intermediate Text Buffer, QMPA
- Attributes: Read-only, reenterable
- Control Section: IEFVFB

IEFVGI: Interpreter -- Dictionary Entry Routine

This routine constructs entries for the refer-back dictionary.

- Entry: IEFVGI
- Exit: Return to caller
- Tables/Work Areas: Refer-back dictionary, auxiliary work area, IWA, QMPA
- Control Section: IEFVGI

IEFVGK: Interpreter -- Get Parameter Routine

This routine searches the internal text buffer for the next parameter, performs basic error checking, and passes control to the appropriate keyword routine.

- Entry: IEFVGK
- Exit: Return to caller
- Tables/Work Areas: Local work area, IWA, internal text buffer, KBT, PDT.
- Control Section: IEFVGK

IEFVGM: Interpreter -- Message Processing Routine

This routine constructs SMBs containing interpreter error messages and JCL statement images, assigns space for these SMBs in the message class output queue entry, and writes the SMBs into the entry.

- Entry: IEFVGM
- Exit: Return to caller
- Tables/Work Areas: QMPA, SMB, SCD, IWA, JCT
- Attributes: Reenterable, character dependence type C
- Control Section: IEFVGM

IEFVGM1: Interpreter -- Message Module

This routine contains interpreter messages 01-07.

- Attributes: Non-executable
- Control Section: IEFVGM1

IEFVGM2: Interpreter -- Message Module

This routine contains interpreter messages 08-0F.

- Attributes: Non-executable
- Control Section: IEFVGM2

IEFVGM3: Interpreter -- Message Module

This routine contains interpreter messages 10-17.

- Attributes: Non-executable
- Control Section: IEFVGM3

IEFVGM4: Interpreter -- Message Module

This routine contains interpreter messages 18-1F.

- Attributes: Non-executable
- Control Section: IEFVGM4

IEFVGM5: Interpreter -- Message Module

This routine contains interpreter messages 20-27.

- Attributes: Non-executable
- Control Section: IEFVGM5

IEFVGM6: Interpreter -- Message Module

This routine contains interpreter messages 28-2F.

- Attributes: Non-executable
- Control Section: IEFVGM6

IEFVGM7: Interpreter -- Message Module

This routine contains interpreter messages 30-37.

- Attributes: Non-executable
- Control Section: IEFVGM7

IEFVGM8: Interpreter -- Message Module

This routine contains interpreter messages 50-57.

- Attributes: Non-executable
- Control Section: IEFVGM8

IEFVGM9: Interpreter -- Message Module

This routine contains interpreter messages 58-5F.

- Attributes: Non-executable
- Control Section: IEFVGM9

IEFVGM10: Interpreter -- Message Module

This routine contains interpreter messages 60-67.

- Attributes: Non-executable
- Control Section: IEFVGM10

IEFVGM11: Interpreter -- Message Module

This routine contains interpreter messages 68-6F.

- Attributes: Non-executable
- Control Section: IEFVGM11

IEFVGM12: Interpreter -- Message Module

This routine contains interpreter messages 70-77.

- Attributes: Non-executable
- Control Section: IEFVGM12

IEFVGM13: Interpreter -- Message Module

This routine contains interpreter messages 78-7F.

- Attributes: Non-executable
- Control Section: IEFVGM13

IEFVGM14: Interpreter -- Message Module

This routine contains interpreter messages 88-8F.

- Attributes: Non-executable
- Control Section: IEFVGM14

IEFVGM15: Interpreter Message -- Module

This routine contains interpreter messages 90-97.

- Attributes: Non-executable
- Control Section: IEFVGM15

IEFVGM16: Interpreter -- Message Module

This routine contains interpreter messages A0-A7.

- Attributes: Non-executable
- Control Section: IEFVGM16

IEFVGM17: Interpreter -- Message Module

This routine contains interpreter messages 56-5D.

- Attributes: Non-executable
- Control Section: IEFVGM17

IEFVGM18: Interpreter -- Message Module

This routine contains interpreter messages 80-87.

- Attributes: Non-executable
- Control Section: IEFVGM18

IEFVGM19: Interpreter -- Message Module

This routine contains interpreter messages 3E-45.

- Attributes: Non-executable
- Control Section: IEFVGM19

IEFVGM70: Interpreter -- Message Module

This routine contains interpreter messages 38-3F.

- Attributes: Non-executable
- Control Section: IEFVGM70

IEFVGM78: Interpreter -- Message Module

This routine contains interpreter messages 08-0D.

- Attributes: Non-executable
- Control Section: IEFVGM78

IEFVGS: Interpreter -- Dictionary Search Routine

This routine searches the refer-back dictionary for the address of a previously-defined SCT, SIOT, or JFCB.

- Entry: IEFVGS
- Exit: Return to caller
- Tables/Work Areas: Auxiliary work area, IWA, QMPA, refer-back dictionary
- Control Section: IEFVGS

IEFVGT: Interpreter -- Test and Store Routine

This routine performs operations on a parameter as indicated in the appropriate parameter descriptor table entry.

- Entry: IEFVGT
- Exit: Return to keyword routine
- Tables/Work Areas: Internal text buffer, PDT, local work area, IWA
- Control Section: IEFVGT

IEFVHA: Interpreter -- Get Routine

This routine reads statements from the input stream and the procedure library.

- Entry: IEFVHA
- Exits: IEFVHC, IEFVHB, IEFVHAA, IEFSD536, IEFVGM
- Tables/Work Areas: IWA, JCT, DCB
- Attributes: Read-only, reenterable
- Control Section: IEFVHA

IEFVHAA: Interpreter -- End-of-File Routine

This routine determines the conditions under which an end-of-file condition has occurred, and sets switches and passes control accordingly.

- Entry: IEFVHAA

- Exits: IEFVHC or IEFVHN
- Tables/Work Areas: IWA, JCT
- Attributes: Read-only, reenterable
- Control Section: IEFVHAA

IEFVHB: Interpreter -- DD * Statement Generator Routine

This routine generates a "SYSIN DD *" statement for data in the input stream, when no such statement was included.

- Entry: IEFVHB
- Exits: To IEFVHC, IEFVHA, IEFVGM
- Tables/Work Areas: IWA, JCT
- Attributes: Read-only, reenterable
- Control Section: IEFVHB

IEFVHC: Interpreter -- Continuation Statement Routine

This routine determines whether the current statement should be a continuation, and, if so, determines whether it is a valid continuation statement.

- Entry: IEFVHC
- Exits: To IEFVHEB, IEFVHCB, IEFVGM
- Tables/Work Areas: IWA, JCT
- Attributes: Read-only, reenterable
- Control Section: IEFVHC

IEFVHCB: Interpreter -- Verb Identification Routine

This routine identifies the verb in a control statement.

- Entry: IEFVHCB
- Exits: To IEFVHE, IEFVHM, IEFVHA, IEFVGM, IEFVHL
- Tables/Work Areas: IWA, JCT
- Attributes: Read-only, reenterable
- Control Section: IEFVHCB

IEFVHE: Interpreter -- Router

This routine determines the conditions under which it was entered, and passes control to the appropriate routine.

- Entry: IEFVHE
- Exits: To IEFVHEB, IEFVHH, IEFVHEC
- Tables/Work Areas: IWA
- Attributes: Read-only, reenterable
- Control Section: IEFVHE

IEFVHEB: Interpreter -- Pre-Scan Preparation Routine

This routine determines whether a message is required or additional work queue space is required before a statement is scanned. If so, it causes the message to be written or the work queue space to be assigned.

- Entry: IEFVHEB
- Exits: To IEFVHQ, IEFVGM, IEFVHG, IEFVFA
- Tables/Work Areas: IWA, JCT, SCT, QMPA
- Attributes: Read-only, reenterable
- Control Section: IEFVHEB

IEFVHEC: Interpreter -- Job Validity Check Routine

This routine determines whether an SCT has been built for the current job; if not, the routine constructs an SCT.

- Entry: IEFVHEC
- Exits: To IEFVGM, IEFVHH
- Tables/Work Areas: IWA, JCT, SCT
- Attributes: Read-only, reenterable
- Control Section: IEFVHEC

IEFVHF: Interpreter -- Post-Scan Routine

This routine determines the conditions under which it was entered, and passes control accordingly.

- Entry: IEFVHF
- Exits: To IEFVHG, IEFVHEB, IEFVHCB, IEFVHA
- Tables/Work Areas: IWA, CWA
- Attributes: Read-only, reenterable
- Control Section: IEFVHF

IEFVHG: Interpreter -- CPO Routine

This routine writes system input data sets on a direct-access device. If IEFVHG is unable to obtain enough space to com-

plete writing a data set, control passes to IEFVHR. If the input reaches end-of-file, control passes to IEFVHAA. If a /* is found following DD DATA, control passes to IEFVHA to read the next record. If a // is found, control passes to IEFVHC to identify the verb.

- Entry: IEFVHG
- Exits: To IEFSD536, IEFVGM, IEFVHQ, IEFVHAA, IEFVHA, IEFVHC, or IEFVHB
- Tables/Work Areas: IWA, JCT, SIOT, VOLT, TIOT, LWA, SCT, JFCB, UCB, QMPA, CWA
- Attributes: Read-only, reenterable
- Control Section: IEFVHG
- Page Reference: 53

IEFVHH: Interpreter -- Job and Step Enqueue Routine

This routine places the SCT, DSNT, VOLT, and JCT in the job's queue entry, and determines whether the interpreter is to enqueue jobs.

- Entry: IEFVHH
- Exits: To IEFKG, IEFVHQ, IEFSD532, IEFVHHB, IEFVHN
- Tables/Work Areas: IWA, JCT, SCT, QMPA, NEL
- Attributes: Read-only, reenterable
- Control Section: IEFVHH

IEFVHHB: Interpreter -- Housekeeping Routine

This routine initializes for merging a cataloged procedure.

- Entry: IEFVHHB
- Exits: IEFVHA, IEFVHEB
- Tables/Work Areas: IWA
- Attributes: Read-only, reenterable
- Control Section: IEFVHHB

IEFVHL: Interpreter -- Null Statement Routine

This routine determines the conditions under which the null statement was encountered, and passes control to the proper routine.

- Entry: IEFVHL
- Exits: To IEFVHCB, IEFHEC, IEFVHE, IEFVHA
- Tables/Work Areas: IWA, JCT
- Attributes: Read-only, reenterable
- Control Section: IEFVHL

IEFVHM: Interpreter -- Command Statement Routine

This routine tests for valid command verbs, and, if the verb is valid, issues SVC 34 to schedule execution of the command.

- Entry: IEFVHM
- Exits: To IEFVHA, IEFVGM
- Tables/Work Areas: IWA, JCT
- Attributes: Read-only, reenterable
- Control Section: IEFVHM

IEFVHN: Interpreter -- Termination Routine

This routine closes the input stream and procedure library data sets, frees main storage used by the interpreter, and builds the interpreter exit list.

- Entry: IEFVHN
- Exit: Return to caller
- Tables/Work Areas: IWA, JCT, CSCB, QMPA
- Attributes: Read-only, reenterable
- Control Section: IEFVHN
- Page Reference: 56

IEFVHQ: Interpreter -- Queue Management Interface Routine

This routine is a common interface between the queue management routines and the interpreter. If an I/O error occurs, IEFVHR receives control. Queue management may be unable to allocate space for a job's input data. If, in this case, the operator replies CANCEL to the message which is issued, IEFVHG receives control.

- Entry: IEFVHQ
- Exits: Return to caller, IEFSD536, or IEFVHG
- Tables/Work Areas: IWA, JCT, QMPA, CSCB

- Attributes: Read-only, reenterable
- Control Section: IEFVHQ

IEFVH1: Interpreter -- Initialization Routine

This routine initializes the interpreter; it obtains main storage for and initializes the IWA, local work areas, and DCBs.

- Entry: IEFVH1
- Exit: To IEFVH2
- Tables/Work Areas: UCB, CSCB, IWA, DCB, local work area
- Attributes: Not reusable
- Control Section: IEFVH1

IEFVH2: Interpreter -- Initialization Routine

This routine opens the input stream data set and the procedure library data set, and obtains main storage for a buffer for procedure library records.

- Entry: IEFVH2
- Exit: To IEFVHA
- Tables/Work Areas: IWA, UCB, TIOT
- Control Section: IEFVH2
- Attributes: Not reusable

IEFVJA: Interpreter -- Job Statement Processor

This routine initializes a JCT and job ACT from the complete logical job statement in the internal text buffer.

- Entry: IEFVJA
- Exit: To IEFVHF
- Tables/Work Areas: IWA, job work area, interpreter key table, JCT, ACT, QMPA
- Attributes: Read-only, reenterable
- Control Section: IEFVJA

IEFVJIMP: Termination -- JOB Statement Condition Code Processor

This routine tests the condition codes specified in the JOB statement to determine whether the remaining steps in the job are to be run.

- Entry: IEFVJ
- Exits: To IEFVK or IEFZA
- Tables/Work Areas: LCT, JCT, SCT
- Control Section: IEFVJ

IEFVJMSG: Termination -- JOB Statement Condition Code Processor Messages

This routine contains the messages issued to the programmer by the JOB statement condition code processor.

- Entry: IEFVJMSG
- Attributes: Non-executable
- Control Section: IEFVJMSG

IEFVKIMP: I/O Device Allocation -- EXEC Statement Condition Code Processor

This routine tests the condition codes specified in the EXEC statement to determine whether the next step of the job is to be run.

- Entry: IEFVK
- Exits: IEFVS, IEFLB
- Tables/Work Areas: JCT, LCT, SCT
- Control Section: IEFVK

IEFVKMSG: I/O Device Allocation -- EXEC Statement Condition Code Processor Messages

This routine contains the messages issued to the programmer by the EXEC -- statement condition -- code processor.

- Entry: IEFVKMJ1
- Attributes: Non-executable
- Control Section: IEFVKMSG

IEFVMFAK: I/O Device Allocation -- Linkage to IEFVMLS1

This routine passes control to entry point IEFVMCVL of the JFCB housekeeping module IEFVMLS1 via the XCTL macro instruction.

- Entry: IEFVMCVL
- Exit: To IEFVMCVL
- Control Section: IEFVMCVL

IEFVMLS1: I/O Device Allocation -- JFCB Housekeeping Control Routine and Allocate Processing Routines

The control routine obtains the required SIOTs, determines the processing required for each, and passes control to the appropriate routine. The allocate processing routine performs the processing required in certain refer-back situations, when the data set is cataloged or passed, and when unit name is specified.

- Entry: IEFVM, VM7000, VM7060, VM7090, VM7055, VM7070, VM7065

- Exits: To IEFVM2LS, IEFVM3LS, IEFVM4LS, IEFVM5LS, IEFVM6LS, and IEFXCSSS
- Tables/Work Areas: LCT, JCT, PDQ, SIOT, JFCB, QMPA
- Control Section: IEFVM1

IEFVMLS6: I/O Device Allocation -- JFCB Housekeeping Error Message Processing Routine

This routine prepares error messages for the JFCB housekeeping routines.

- Entry: IEFVMSGR
- Exit: Return to caller
- Tables/Work Areas: JCT, LCT
- Control Section: IEFVM6

IEFVMLS7: I/O Device Allocation -- JFCB Housekeeping Error Messages

This routine contains the messages issued by the JFCB housekeeping routines.

- Entry: IEFVM7
- Attributes: Non-executable
- Control Section: IEFVM7

IEFVMMS1: I/O Device Allocation -- Linkage to JFCB Housekeeping

This routine provides a linkage to the JFCB housekeeping routines for the step flush function.

- Entry: IEFVM1
- Exit: To IEFVM1
- Attributes: Read-only, reenterable
- Control Section: IEFVM1

IEFVPOST: I/O Device Allocation -- Unsolicited Device Interrupt Handler

This routine handles the posting of unsolicited device interruptions for I/O device allocation operating in a multiprogramming environment.

- Entry: IEFDPOST
- Exits: To IEAOPT01 or Return to caller
- Tables/Work Areas: CSCB, ECB, TCB
- Attributes: Read-only, reenterable, disabled, resident
- Control Section: IEFDPOST

IEFVM2LS: I/O Device Allocation -- JFCB Housekeeping Fetch DCB Processing Routine

This routine updates the SIOT, SCT, JFCB and VOLT with information required for the allocation of devices for the fetch DCB.

- Entry: VM7100
- Exit: To IEFVMLS1
- Tables/Work Areas: LCT, SCT, SIOT, JFCB, VOLT
- Control Section: IEFVM2

IEFVM3LS: I/O Device Allocation -- JFCB Housekeeping GDG Single Processing Routine

This routine obtains the fully qualified name of a member of a GDG, and completes the required information in the JFCB, VOLT, and SIOT for that member.

- Entry: VM7150
- Exit: To IEFVMLS1
- Tables/Work Areas: LCT, SIOT, GDG Bias Count table, JFCB
- Control Section: IEFVM3

IEFVM4LS: I/O Device Allocation -- JFCB Housekeeping GDG All Processing Routine

This routine builds an SIOT, JFCB, and VOLT, and PDQ entries for each member of the GDG.

- Entry: VM7200
- Exit: To IEFVMLS1
- Tables/Work Areas: LCT, SCT, VOLT, PDQ, SIOT, JFCB
- Control Section: IEFVM4

IEFVM5LS: I/O Device Allocation -- JFCB Housekeeping Patterning DSCB Routine

This routine establishes DCB control information within a JFCB.

- Entry: VM7300
- Exit: To IEFVMLS1
- Tables/Work Areas: LCT, SCT, SIOT, DSCB, JFCB
- Control Section: IEFVM5

IEFVM76: I/O Device Allocation -- JFCB Housekeeping Unique Volume ID Routine

This routine creates unique volume serials for unlabeled tape data sets, when the disposition is "PASS".

- Entry: VM7600
- Exit: Return to caller
- Tables/Work Areas: SIOT, JFCB, JFCBX
- Control Section: IEFVM76

IEFVSD12: Interpreter -- CPO Allocation Subroutine

This routine sets up a JFCB and allocates space on a direct-access device for a system input data set.

- Entry: IEFSD012
- Exit: Return to caller
- Attributes: Reenterable
- Tables/Work Areas: IWA, QMPA, LWA, SIOT, TIOT, UCB, JFCB, JCT, CSCB
- Control Section: IEFSD012
- External References: IEFVHQ

IEFVSD13: Interpreter -- SCD Construction Routine

This routine constructs an SCD entry for each system output class defined for a job, and assigns space for all DSBs that will be required.

- Entry: IEFSD090
- Exit: Return to caller
- Tables/Work Areas: IWA, QMPA, DD work area, SCD, SCT, SIOT, JCT, JFCB
- Control Section: IEFSD090

IEFWA000: I/O Device Allocation -- Demand Allocation Routine

This routine establishes data set device requirements, and allocates in response to specific unit requests.

- Entry: IEFWA000, IEFUCBL
- Exits: To IEFWD000, IEFX3000, IEFX5000
- Tables/Work Areas: UCB Address List, DMT, UCB, LCT, SCT, SIOT, VOLT, AWT
- Control Sections: IEFWA7, IEFWA002

IEFWCFAK: I/O Device Allocation -- Linkage Module

This routine passes control to the TIOT construction routine.

- Entry: IEFWC000
- Exit: To IEFWCIMP
- Control Section: IEFWC000

IEFWCIMP: I/O Device Allocation -- TIOT Construction Routine

This routine calculates the main storage required for the TIOT, builds the TIOT, and processes requests for direct-access space.

- Entry: IEFWC000
- Exits: To IEFXJIMP, IEFWDIMP
- Tables/Work Areas: JCT, SCT, LCT, SIOT, VOLT, AWT, TIOT
- Control Section: IEFWC000

IEFWDFAK: I/O Device Allocation -- Linkage Module

This routine passes control to the external action routine.

- Entry: IEFWD000
- Exit: To IEFWD000
- Control Section: IEFWD000

IEFWD000: I/O Device Allocation -- External Action Routine

This routine causes the correct volumes for the step to be mounted on the appropriate units.

- Entry: IEFWD000
- Exits: To IEFXT000, IEFW41SD, IEFXK000
- Tables/Work Areas: SCT, LCT, TIOT, UCB
- Control Section: IEFWD000

IEFWD001: I/O Device Allocation -- External Action Messages

This routine contains a directory and the messages used in the external action routine.

- Entry: IEFWD001
- Attributes: Non-executable
- Control Section: IEFWD001

IEFWSTRT: I/O Device Allocation -- Message Module

This routine contains the message issued to the operator when a job is started and the messages issued to the operator when a job is terminated due to ABEND, condition codes, or JCL errors.

- Entry: IEFWSTRT
- Attributes: Non-executable
- Control Section: IEFWSTRT

IEFWSWIN: I/O Device Allocation -- Linkage Module

This routine passes control to the decision allocation routine.

- Entry: IEFWSWIT
- Exit: To IEFX5000
- Control Section: IEFWSWIT

IEFWTERM: Termination -- Message Module

This routine contains the message issued to the operator when a job is terminated normally, or when it is terminated because of a JCL error found in the interpreter or initiator.

- Entry: IEFWTERM
- Attributes: Non-executable
- Control Section: IEFWTERM

IEFXAMSG: I/O Device Allocation -- Message Module

This routine contains the messages issued by the allocation control routine.

- Entry: IEFXAMSG
- Attributes: Non-executable
- Control Section: IEFXAMSG

IEFXCSSS: I/O Device Allocation -- Allocation Control Routine

This routine calculates table space requirements and obtains the main storage for the tables used or built during allocation.

- Entry: IEFXA
- Exits: To IEFXJ, IEFWA, IEFWC
- Tables/Work Areas: JCT, SCT, LCT, UCB, SIOT, VOLT, AWT
- Control Section: IEFXA

IEFXH000: I/O Device Allocation -- Separation Strikeout Routine

This routine strikes from AWT entries, the bits corresponding to devices that would violate separation or affinity requests.

- Entry: IEFXH000
- Exit: Return to caller
- Tables/Work Areas: LCT, AWT, AVT, UCB
- Control Section: IEFXH000

IEFXJFAK: I/O Device Allocation -- Linkage Module

This routine passes control to the allocation recovery routine.

- Entry: IEFXJ000
- Exit: To IEFXJIMP
- Control Section: IEFXJ000

IEFXJIMP: I/O Device Allocation -- Allocation Recovery Routine

This routine informs the operator of the allocation recovery options available, and passes control to the proper routine to comply with his request.

- Entry: IEFXJ000, IEFV15XL, IEFXJX5A
- Exits: To IEFXCSSS, IEFSD095, IEFW41SD
- Tables/Work Areas: LCT, AWT, JCT, CVT, UCB, SCT, SIOT
- Control Section: IEFXJ000

IEFXJMSG: I/O Device Allocation -- Allocation Recovery Messages

This routine contains the messages used by the allocation recovery routine.

- Entry: MSRCV, MSSYS, MSOFF
- Attributes: Non-executable
- Control Section: IEFXJMSG

IEFXKIMP: I/O Device Allocation -- Non-Recovery Error Routine

This routine cancels the step when a lack of available devices has been discovered after the TIOT is constructed.

- Entry: IEFXK000

- Exit: To IEFW41SD
- Tables/Work Areas: LCT, SCT, UCB, TIOT
- Control Section: IEFXK000

IEFXKMSG: I/O Device Allocation -- Non-Recovery Error Routine Messages

This routine contains the messages used by the non-recovery error routine.

- Entry: IEFXKMSG
- Attributes: Non-executable
- Control Section: IEFXKMSG

IEFXT00D: I/O Device Allocation -- Space Request Routine

This routine obtains space on direct-access devices for requesting data sets.

- Entry: IEFXT000
- Exits: To IEFW41SD, IEFXK000, IEFWD000
- Tables/Work Areas: LCT, TIOT, UCB, JCT, SIOT, JFCB, PDQ
- Control Section: XTTP00, IEFXT000

IEFXT002: I/O Device Allocation -- TIOT Compression Routine

This routine reduces the TIOT to its final size.

- Entry: IEFXT002, XTTRDJ, XTTEB3, XTTEA1, XTTEA0
- Exits: to IEFW41SD, IEFXKIMP, IEFXT003
- Tables/Work Areas: LCT, TIOT, UCB, JCT, SIOT, JFCB
- Control Section: IEFXT002

IEFXT003: I/O Device Allocation -- DADSM Error Recovery Routine

This routine determines what action should be taken when the request for space on a particular volume fails.

- Entry: IEFXT003, XUUH06, XUUB00
- Exits: To IEFXT00D, IEFXT002
- Tables/Work Areas: LCT, TIOT, UCB, JCT, SIOT, JFCB
- Control Section: IEFXT003

IEFXVMSG: I/O Device Allocation -- Automatic Volume Recognition Messages

This routine contains the messages used by the automatic volume recognition (AVR) routine.

- Entry: IEFXVMSG
- Attributes: Non-executable
- Control Section: IEFXVMSG

IEFXVNSL: I/O Device Allocation -- Automatic Volume Recognition -- Non-Standard Label Routine

This routine processes non-standard labels for the AVR routine.

- Entry: IEFXVNSL
- Exit: Return to caller
- Control Section: IEFXVNSL

IEFXV001: I/O Device Allocation -- Automatic Volume Recognition Routine

This routine finds and allocates volumes pre-mounted by the operator.

- Entry: IEFXV001
- Exits: IEFWC000, IEFX5000, IEFXJ000
- Tables/Work Areas: JCT, SCT, AWT, AVT, VOLT, SIOT, LCT, UCB
- Control Section: IEFXV001

IEFX300A: I/O Device Allocation -- Device Strikeout Routine

This routine modifies the primary and secondary bit patterns in AWT entries to complete the allocation to a data set.

- Entry: IEFX3000, X33B42
- Exit: Return to caller
- Tables/Work Areas: AWT, AVT, UCB, LCT
- Control Section: IEFX3000

IEFX5000: I/O Device Allocation -- Decision Allocation Routine

This routine selects devices for data sets with multiple unit possibilities.

- Entry: IEFX5000, XIIB32, X55C86, X55D3G
- Exits: To IEFWC000, IEFXJ000

- Tables/Work Areas: LCT, AWT, AVT, UCB
- Control Section: IEFX5000

IEFYNIMP: Termination -- Step Termination Control Routine

This routine passes control among the modules of the step termination routine and, when required, passes control to the job termination routine.

- Entry: IEFYN
- Exits: To IEFW22SD, IEFYPJB3, IEFVJIMP, IEFZAJB3
- Tables/Work Areas: JCT, SCT, LCT
- Control Section: IEFYN

IEFYNMSG: Termination -- Step Termination Control Routine Messages

This routine contains the messages required for the step termination control routine.

- Entry: IEFYNMSG, STRMSG01
- Attributes: Non-executable
- Control Section: IEFYNMSG

IEFYPJB3: Termination -- Step Termination Data Set Driver Routine

This routine obtains SIOTs and to pass control to the disposition and unallocation routine.

- Entry: IEFYP
- Exits: To IEFZG, IEFYN
- Tables/Work Areas: LCT, TIOT, UCB, QMPA, SIOT, TCB
- Control Section: IEFYP

IEFYPMSG: Termination -- Step Termination Messages

This routine contains the messages required by the step termination routine.

- Entry: IEFYPMSG, YPPMSG1, YPPMSG2
- Attributes: Non-executable
- Control Section: IEFYPMSG

IEFYSVMS: Termination -- Message Blocking Routine

This routine blocks system messages into SMBs, and places SMBs into the message class queue entry.

- Entry: IEFYS
- Exit: Return to caller
- Tables/Work Areas: LCT, SCT, SMB
- Attributes: Reenterable
- Control Section: IEFYS

IEFYTVMS: Termination -- DSB Processing Routine

This routine places data set blocks in the space reserved for them in the output queue entries.

- Entry: IEFYT
- Exit: Return to caller
- Tables/Work Areas: JCT, SCT, TIOT, SIOT, QMPA, DSCB, LCT, CVT, JFCB
- Attributes: Reenterable
- Control Section: IEFYT

IEFZAJB3: Termination -- Job Termination Control Routine

This routine provides entry to the job termination routine, obtains PDQ blocks, and passes control to the disposition and unallocation routine.

- Entry: IEFZA
- Exits: To IEFZG, IEFW31SD
- Tables/Work Areas: LCT, JCT, PDQ, UCB, QMPA
- Control Section: IEFZA

IEFZGJB1: Termination -- Disposition and Deallocation Routine

This routine directs the disposition and deallocation of those data sets that remain to be processed at job termination: passed data sets that were not received, and retained data sets that were not referred to.

- Entry: IEFZG, ZP0QM
- Exit: Return to caller
- Tables/Work Areas: JCT, PDQ, JFCB, LCT, QMPA, UCB
- Control Section: IEFZGJ

IEFZGMSG: Termination -- Disposition and Deallocation Messages

This routine contains the messages required for the disposition and deallocation routine (IEFZGJB1).

- Entry: IEFZGMSG
- Attributes: Non-executable
- Control Section: IEFZGMSG

IEFZGST1: Termination -- Disposition and Deallocation Routine

This routine directs the disposition of data sets as specified in the DISP field of the DD statement, and makes the associated units available for allocation to other data sets.

- Entry: IEFZG, ZP0QMGR1
- Exit: Return to caller
- Tables/Work Areas: LCT, PDQ, SIOT, TIOT, UCB, JFCB, QMPA
- Control Section: IEFZG

IEFZHMSG: Termination -- Disposition and Deallocation Message Routine

This routine prepares messages to the programmer and to the operator for the disposition and allocation routines.

- Entry: IEFZH, ZG0E60, ZK0D1, ZK0E1, XPS631
- Exit: Return to caller
- Tables/Work Areas: LCT, QMPA, SMB
- Control Section: IEFZH

IEF078SD: System Output Writer -- Linkage Module

This routine transfers control to module IEFSD078.

- Entry: IEFSD078
- Exit: To IEFSD078
- Attributes: Reenterable

IEF079SD: System Output Writer -- Linkage Module

This routine transfers control to IEFSD079.

- Entry: IEFSD079
- Exit: To IEFSD079
- Attributes: Reenterable

IEF082SD: System Output Writer -- Linkage Module

This routine passes control to the system output writer main processing routine.

- Entry: IEFSD082
- Exit: To IEFSD082
- Control Section: IEFSD082

IEF083SD: System Output Writer -- Linkage Module

This routine passes control to the system output writer command processing routine.

- Entry: IEFSD083
- Exit: IEFSD083
- Control Section: IEFSD083

IEF300SD: System Restart -- Linkage Module

This routine provides a linkage to the system restart initialization routine.

- Entry: IEFSD300
- Exits: To IEFSD300, IEFSD055
- Attributes: Reenterable

IEF304SD: System Restart -- Linkage Module

This routine provides a linkage to the system restart scratch data sets routine.

- Entry: IEFSD304
- Exits: To IEFSD304, IEFSD055
- Attributes: Reenterable
- Control Section: IEFSD304

IEF41FAK: I/O Device Allocation -- Linkage Module

This routine provides a linkage to the allocation exit routine during step flush.

- Entry: IEFW41SD, IEFW1FAK, IEFW2FAK
- Exit: To IEFW41SD
- Attributes: Read-only, reenterable
- Control Section: IEFW41SD

This appendix includes the MFT flowcharts that are different from MVT. For the flowcharts on allocation, termination, and system restart, see IBM System/360 Operating System: MVT Job Management, Program Logic Manual, Form Y28-6660.

Chart 01. Nucleus Initialization Program

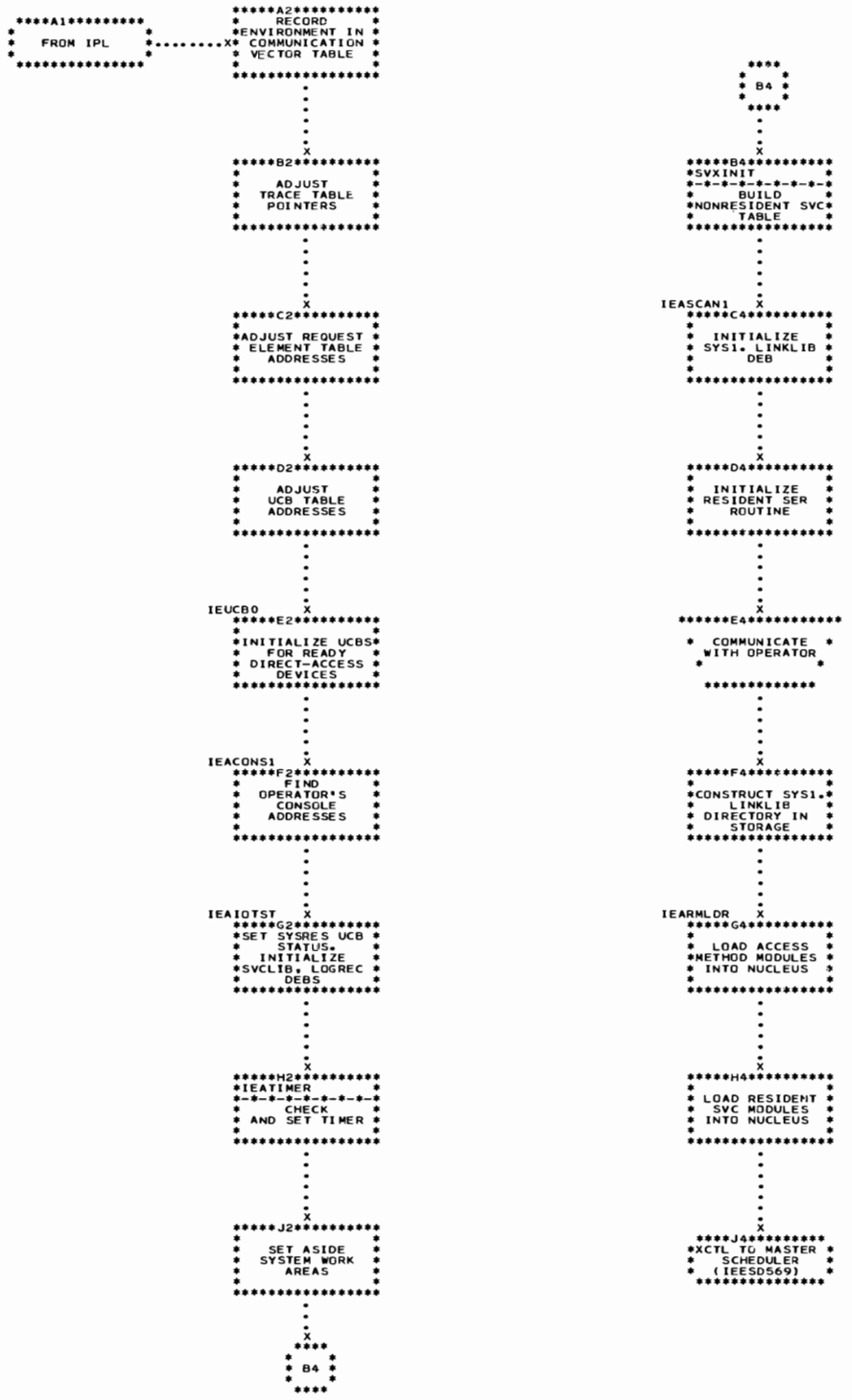


Chart 02. Task Dispatcher (without Time Slicing)

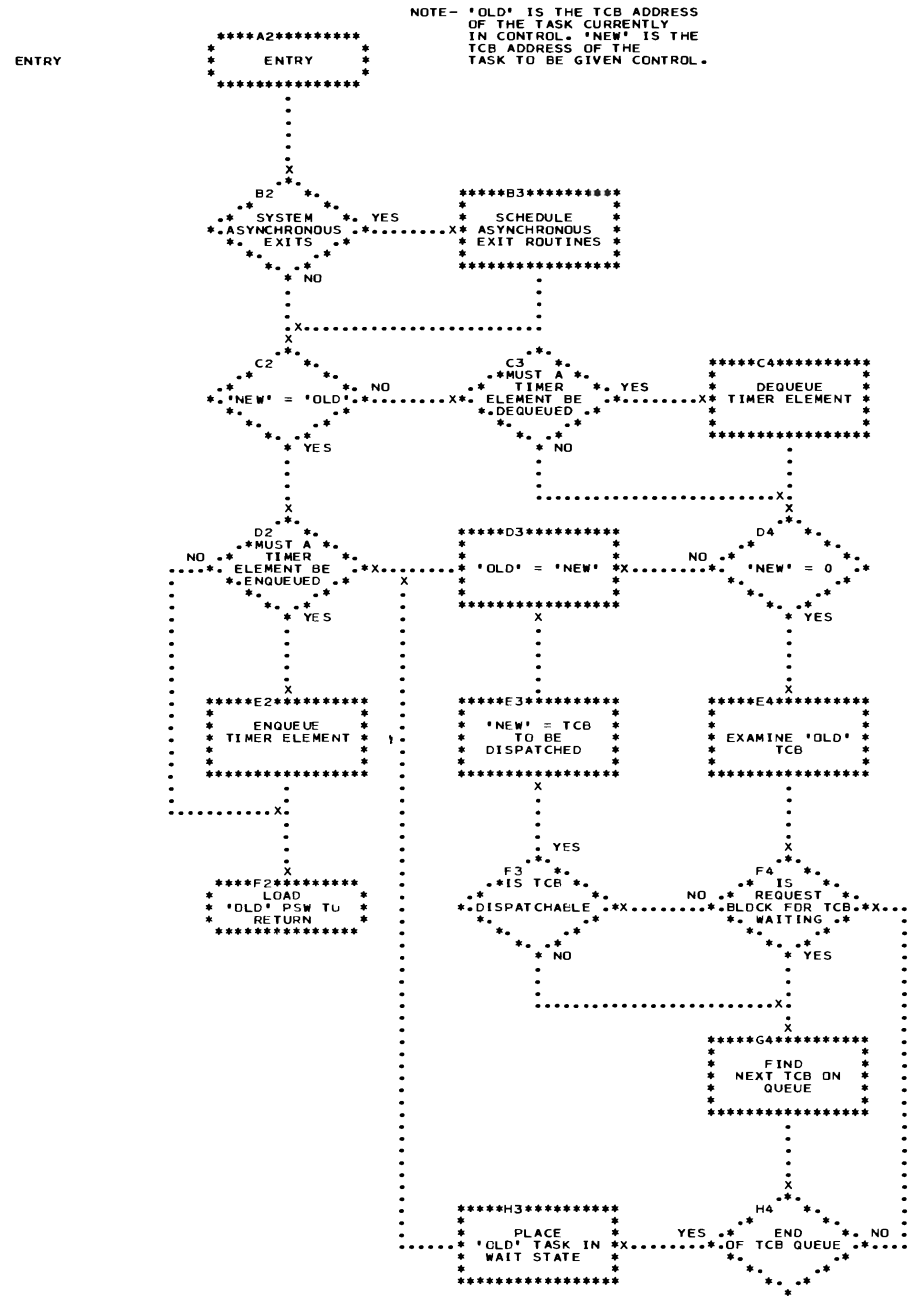
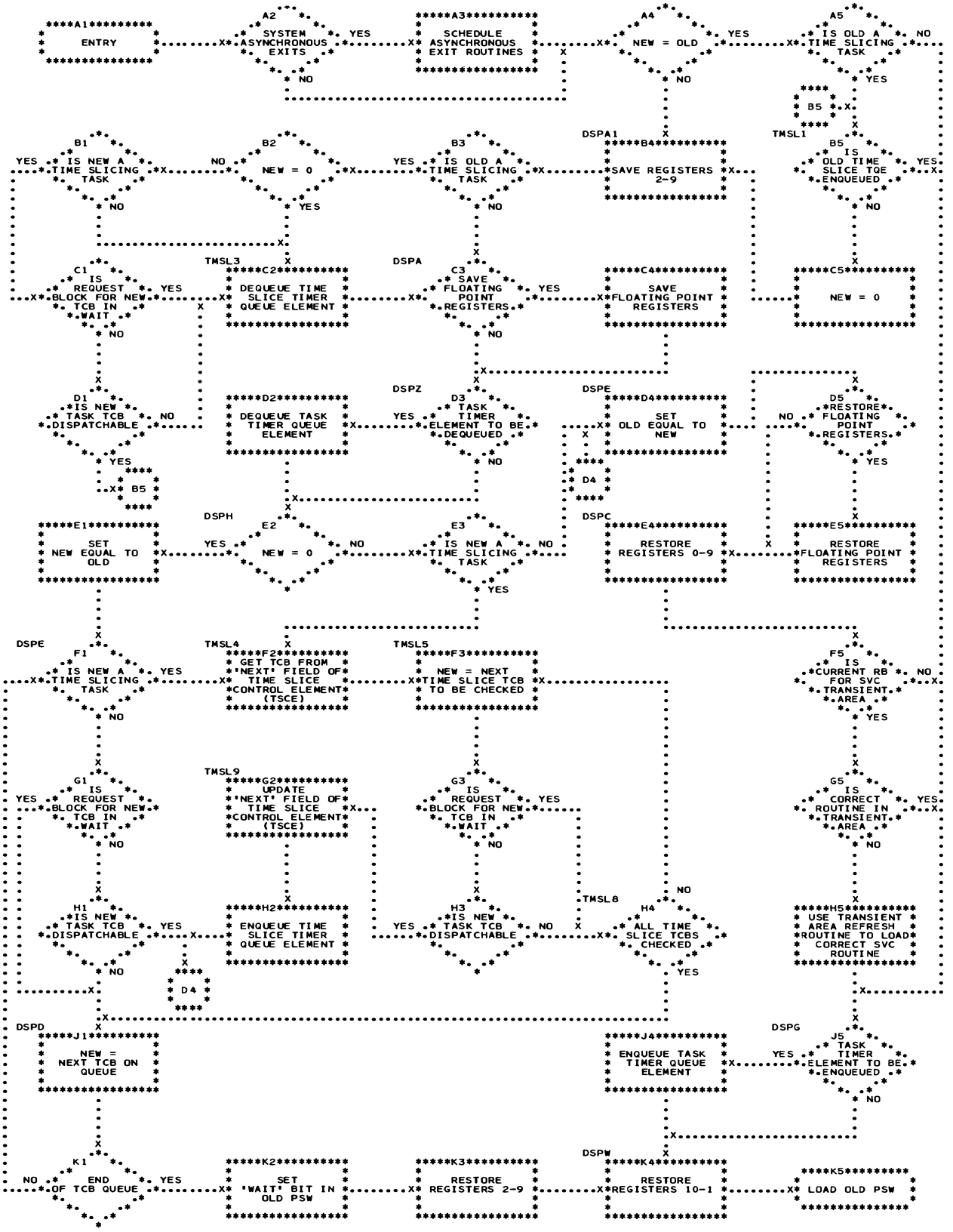
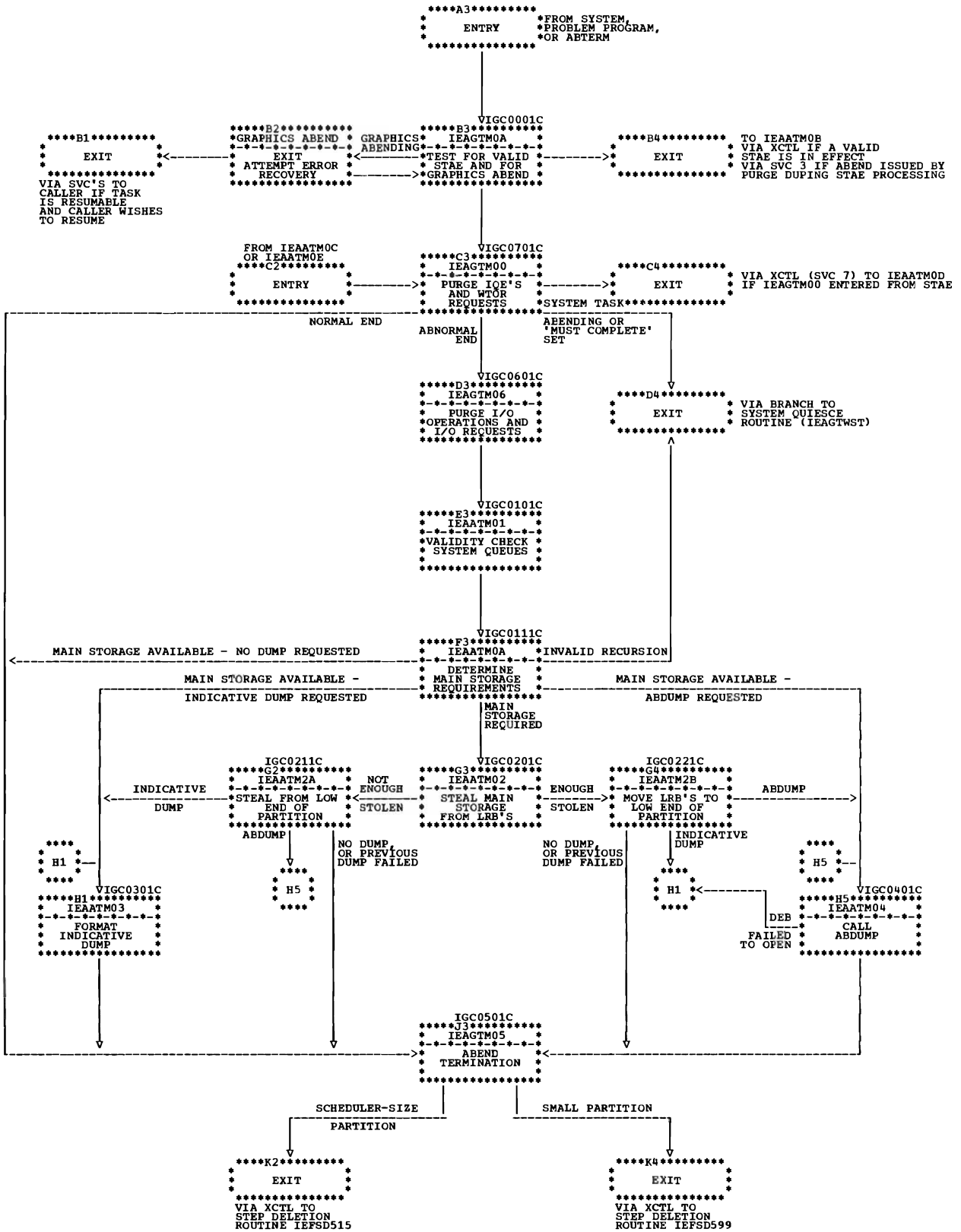


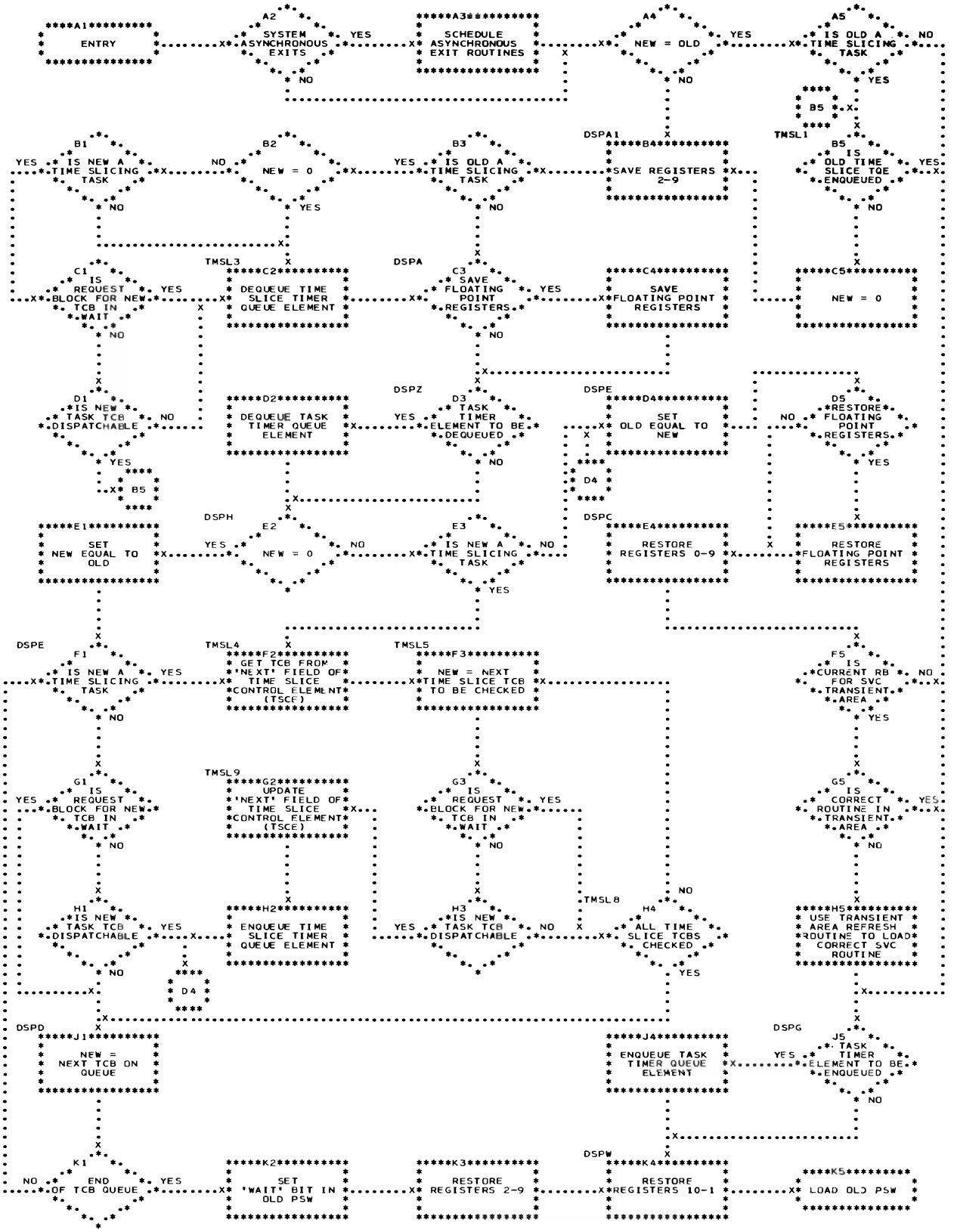
Chart 03. Task Dispatcher (with Time Slicing)



• Chart 04. ABEND Control Flow



- Chart 03. Task Dispatcher (with Time Slicing)



• Chart 04. ABEND Control Flow

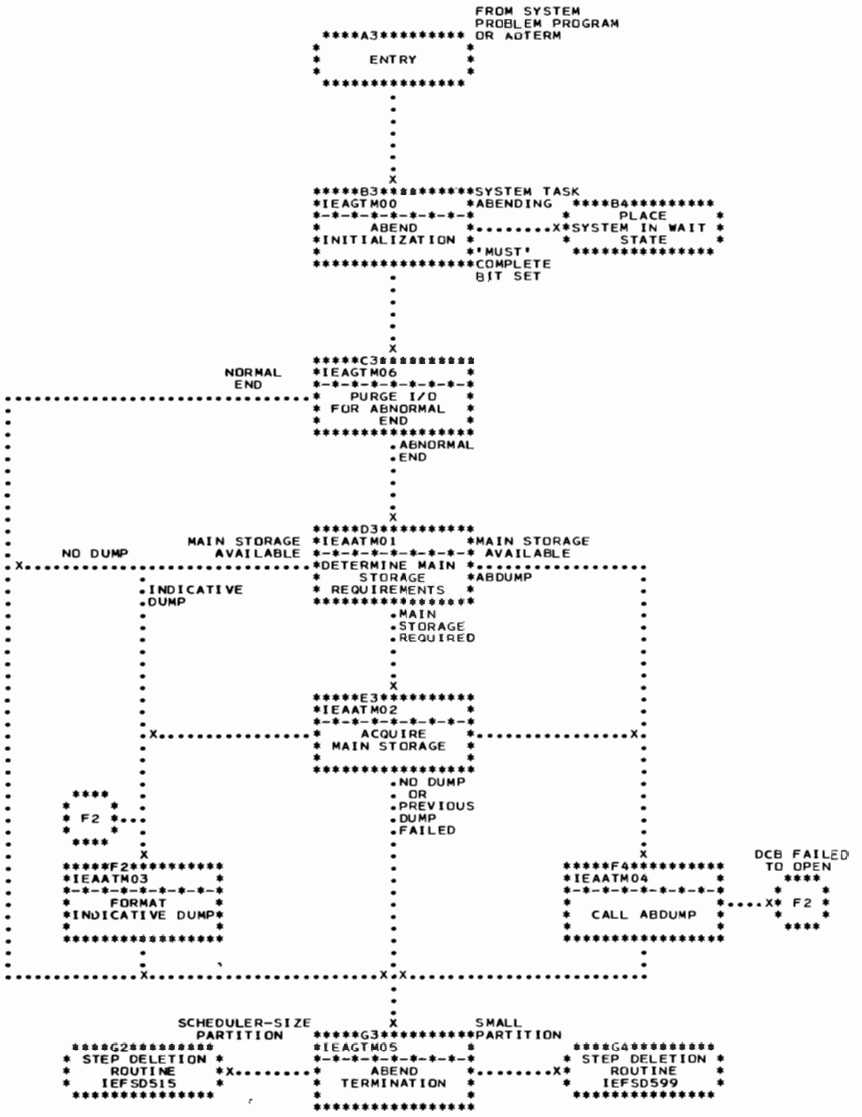
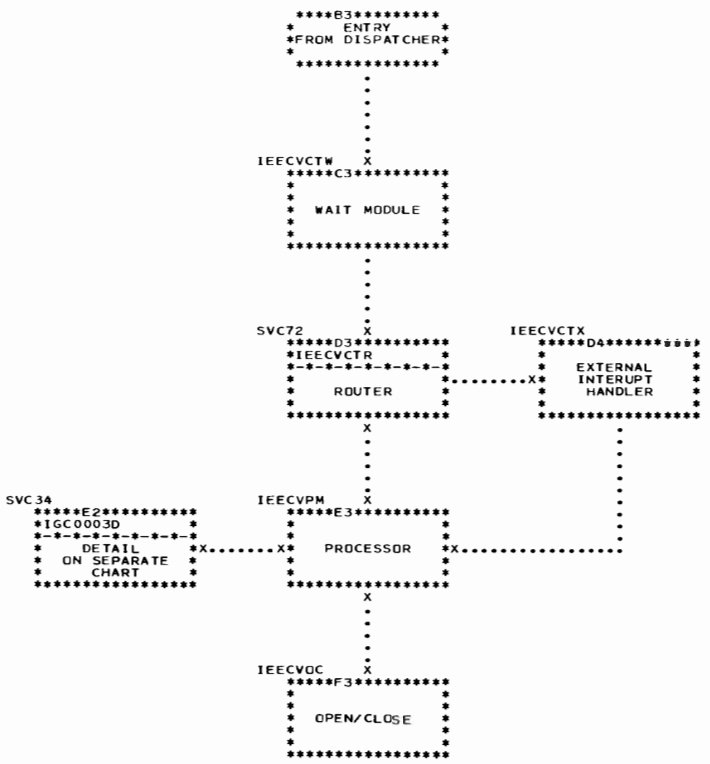


Chart 05. Communications Task



IEECVPM AND IEECVOC ARE CSECTS OF DEVICE DEPENDENT MODULES. THE CONSOLE DEVICE(S) WILL DETERMINE WHICH MODULES ARE USED.

Chart 06. SVC 34 Command Processing

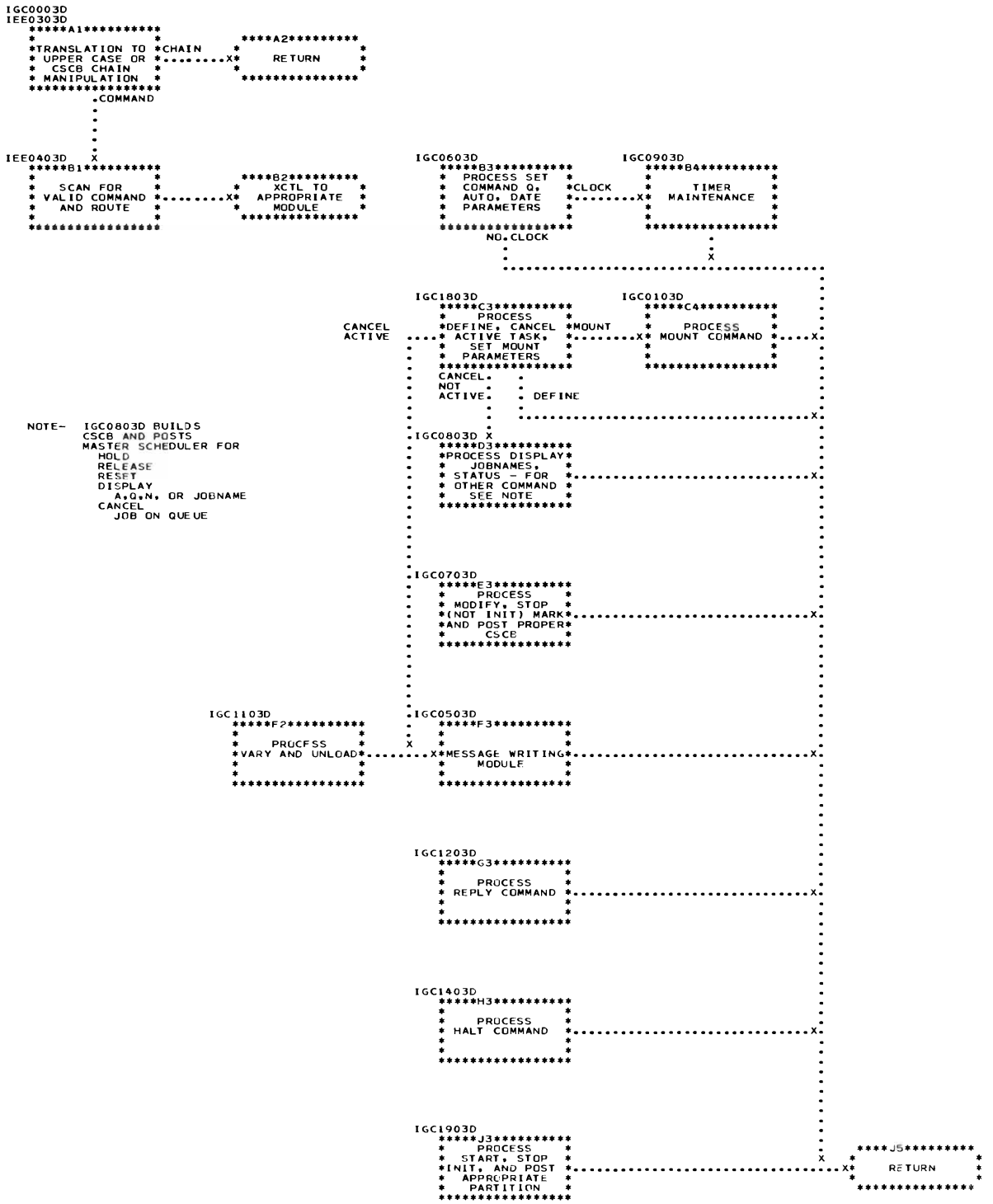


Chart 07. Master Scheduler Task

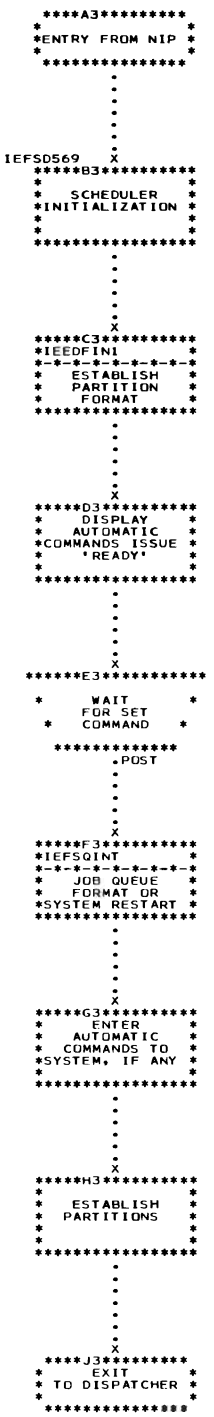


Chart 08. Master Scheduler Resident Command Processor

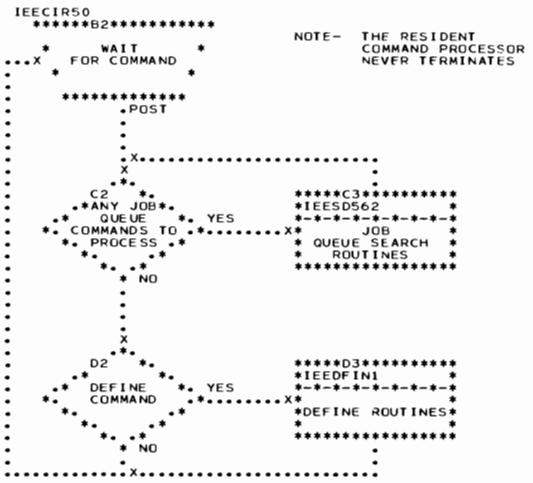
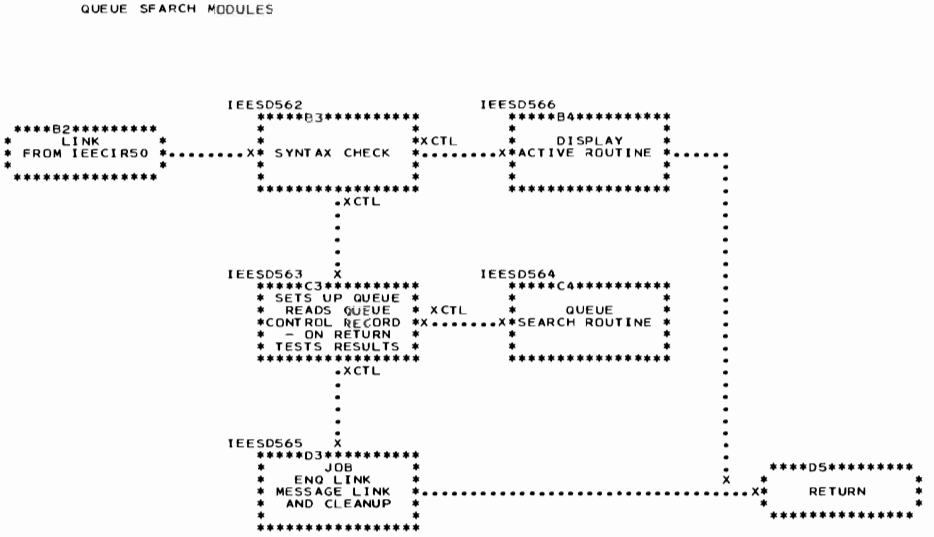


Chart 09. Queue Search



WILL DETERMINE WHICH MODULES ARE USED.

Chart 10. Queue Manager Table Breakup Routine

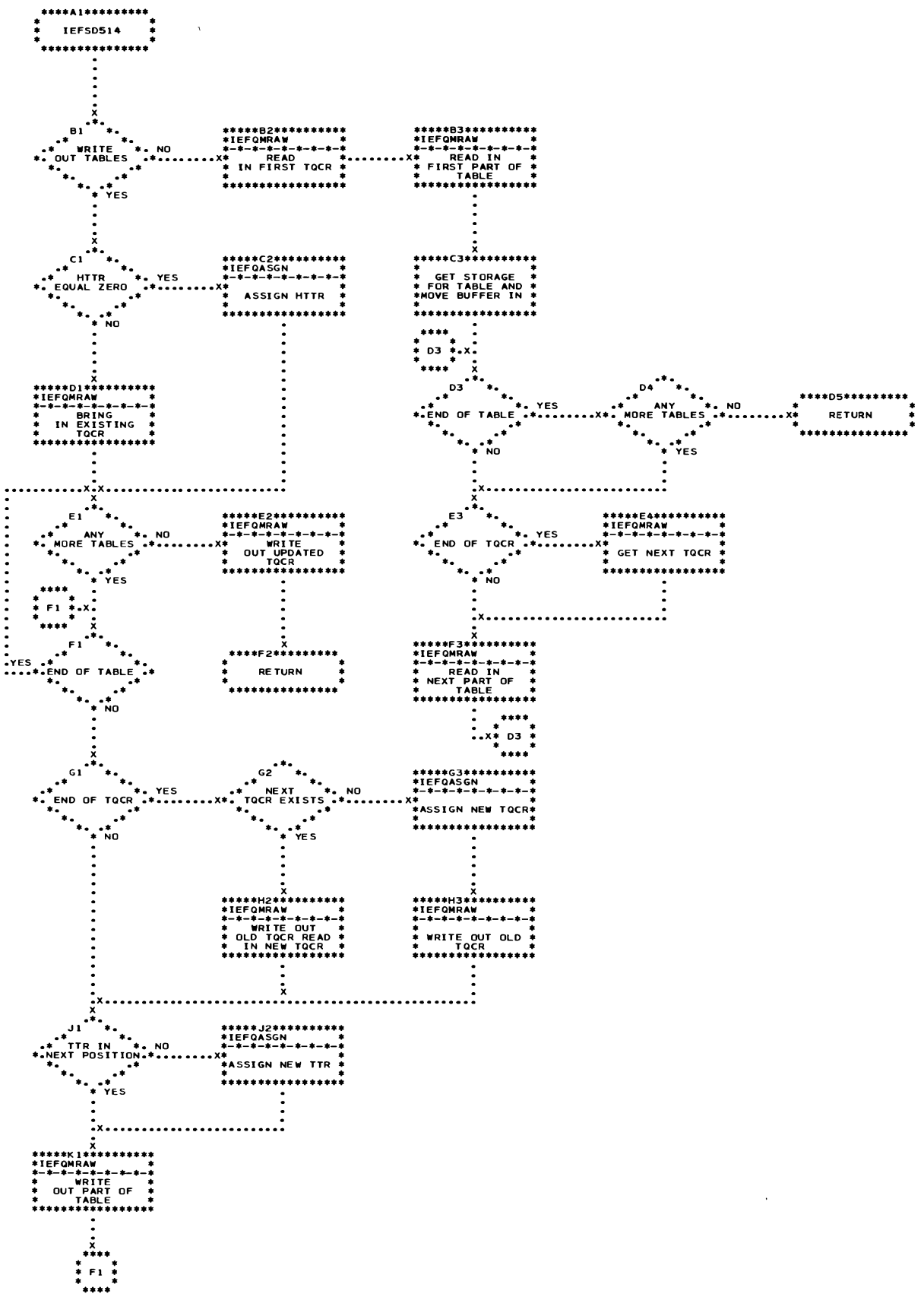


Chart 11. Reader/Interpreter (Sheet 1 of 3)

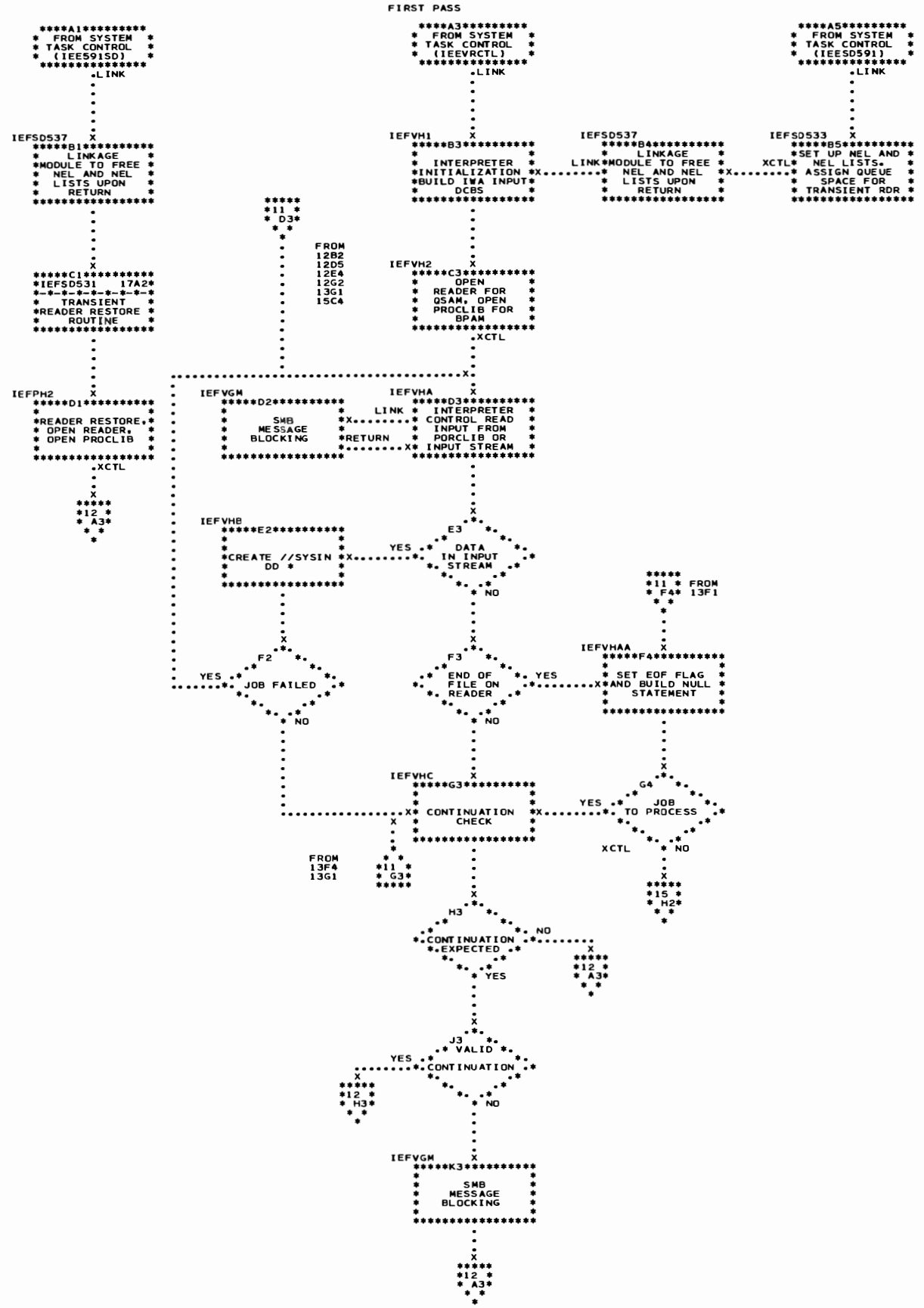


Chart 12. Reader/Interpreter (Sheet 2 of 3)

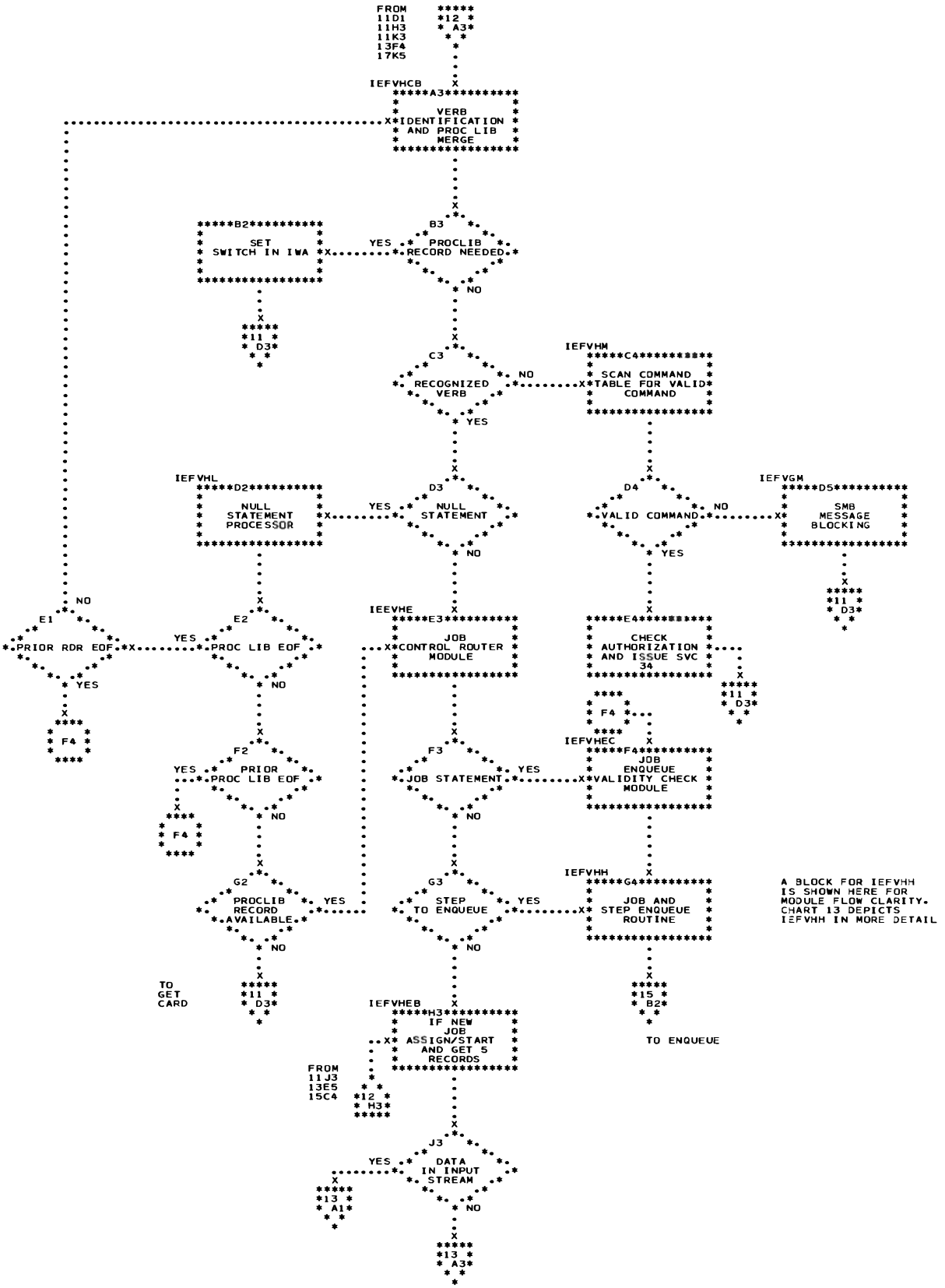


Chart 13. Reader/Interpreter (Sheet 3 of 3)

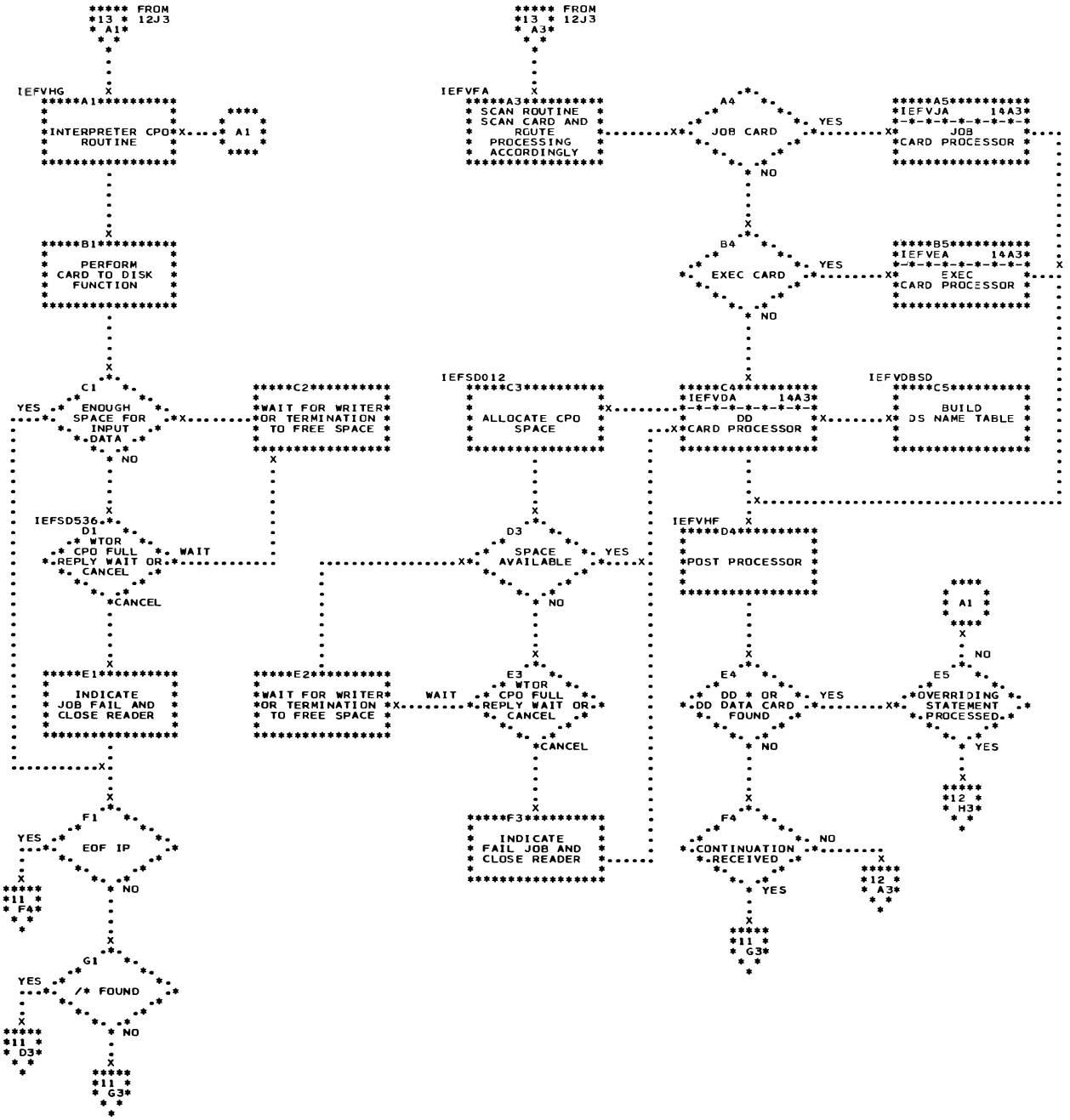


Chart 14. JCL Statement Processors

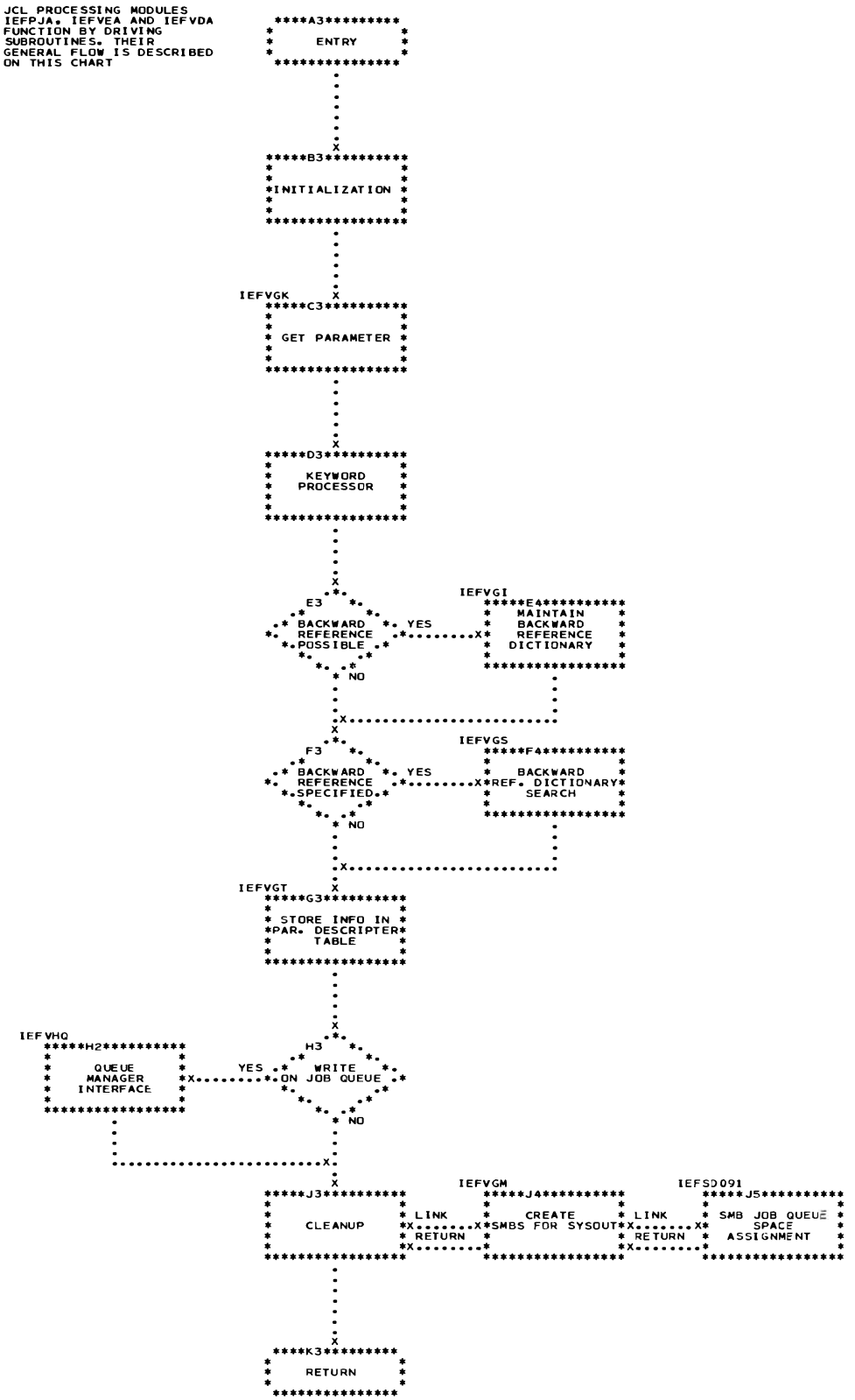


Chart 15. Job and Step Enqueue Routine

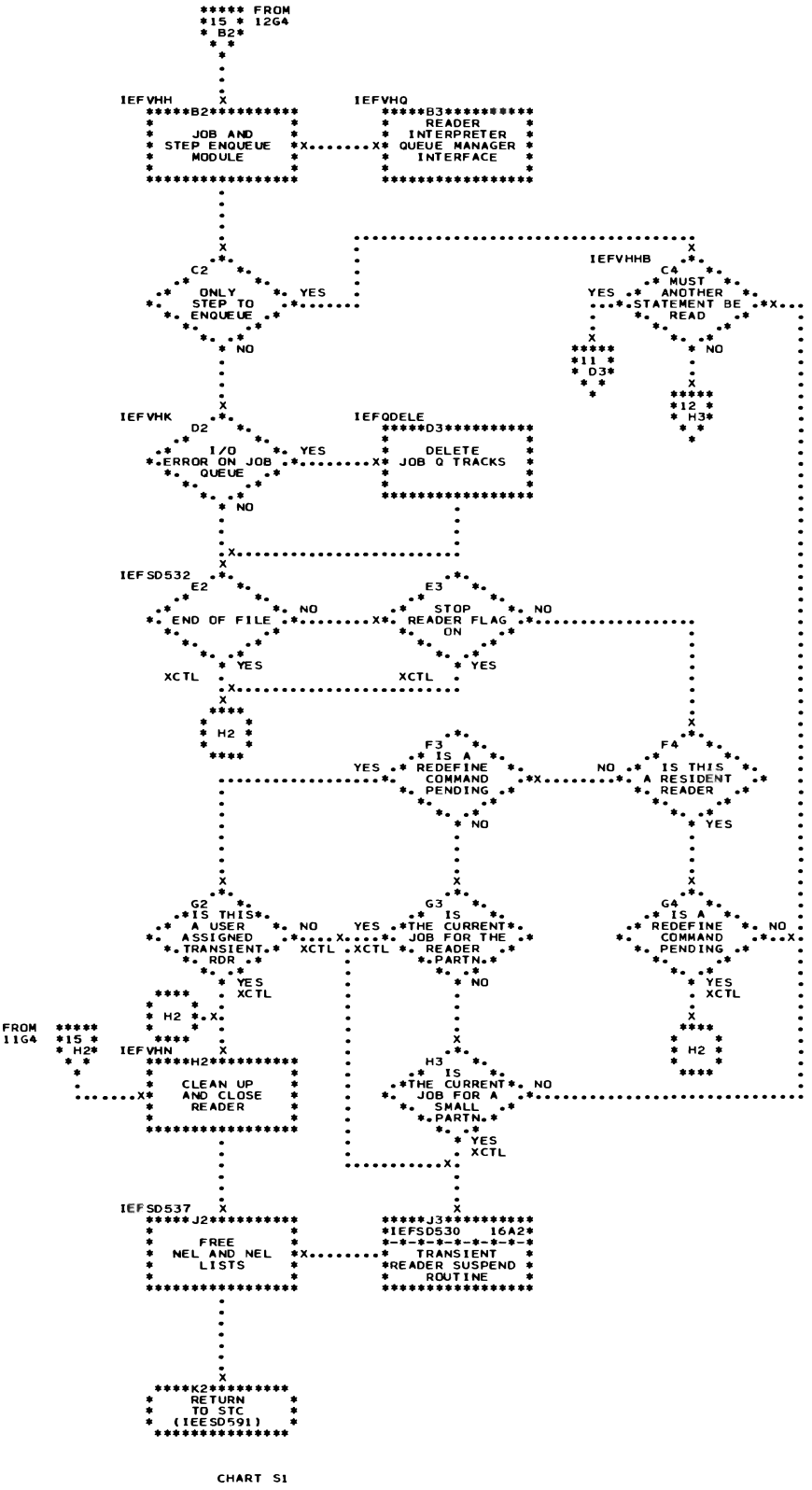


Chart 16. Transient Reader Suspend Routine

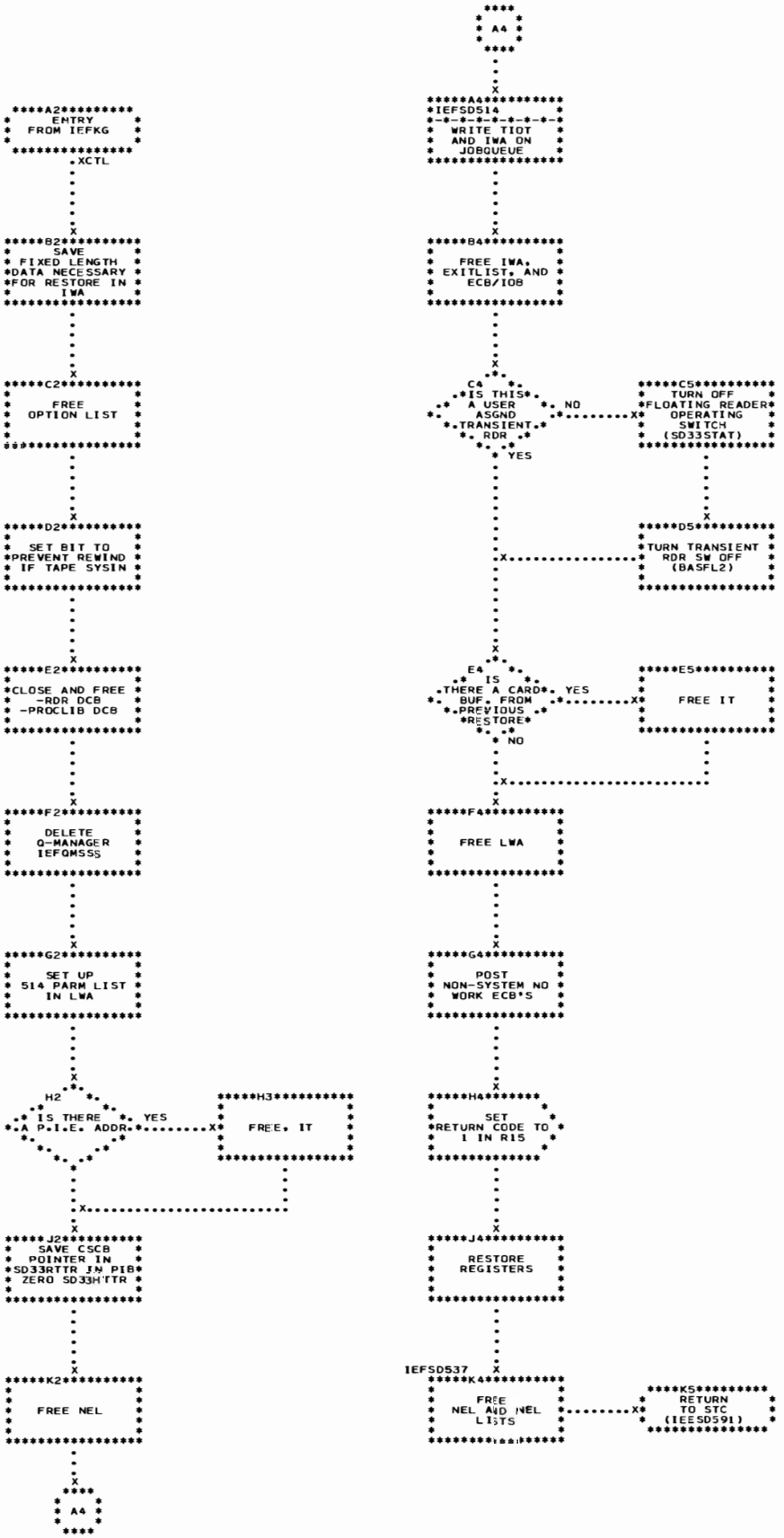
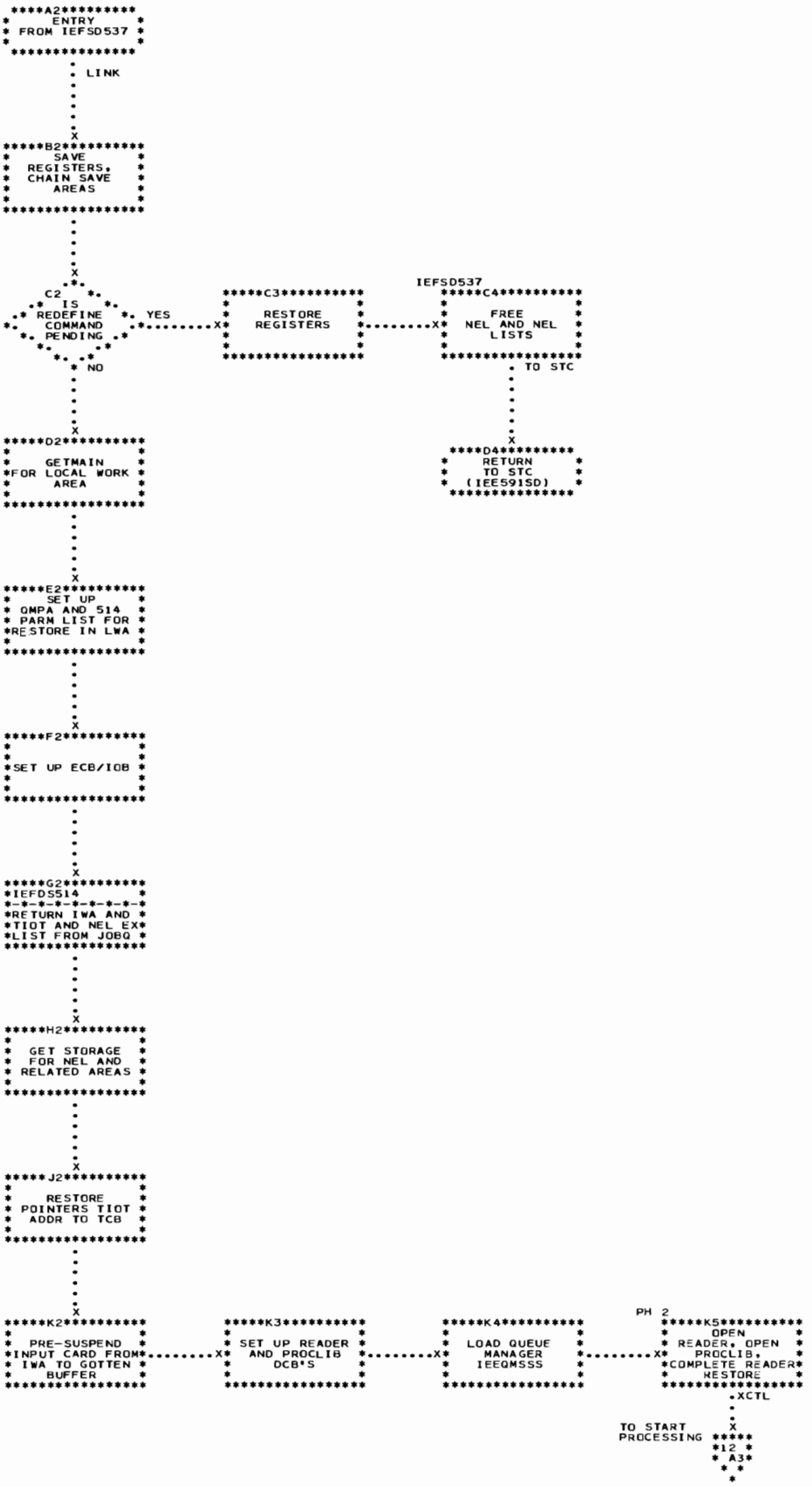


Chart 17. Transient Reader Restore Routine



• Chart 18. Initiator Control Flow

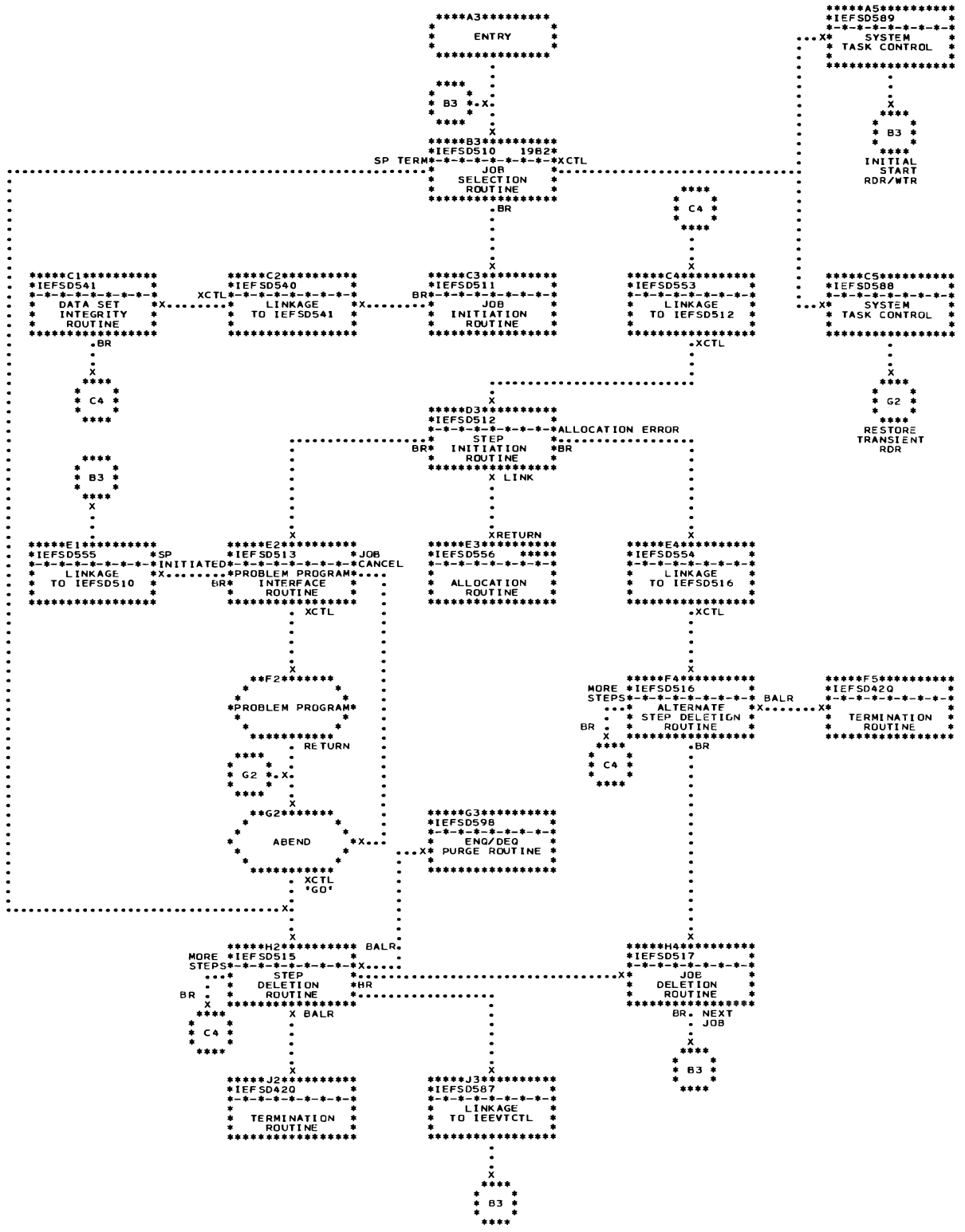


Chart 19. Job Selection Routine (Sheet 1 of 5)

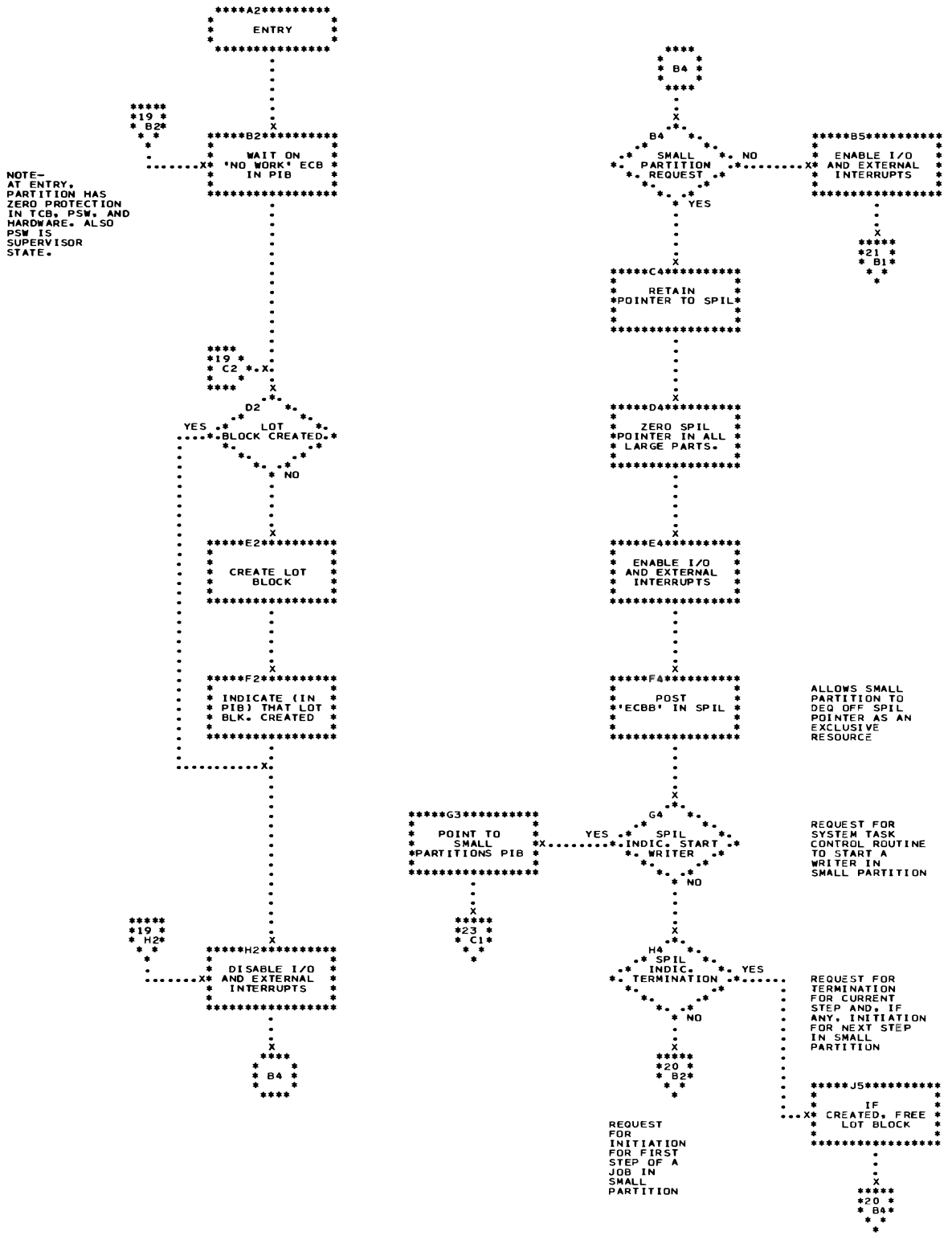


Chart 20. Job Selection Routine (Sheet 2 of 5)

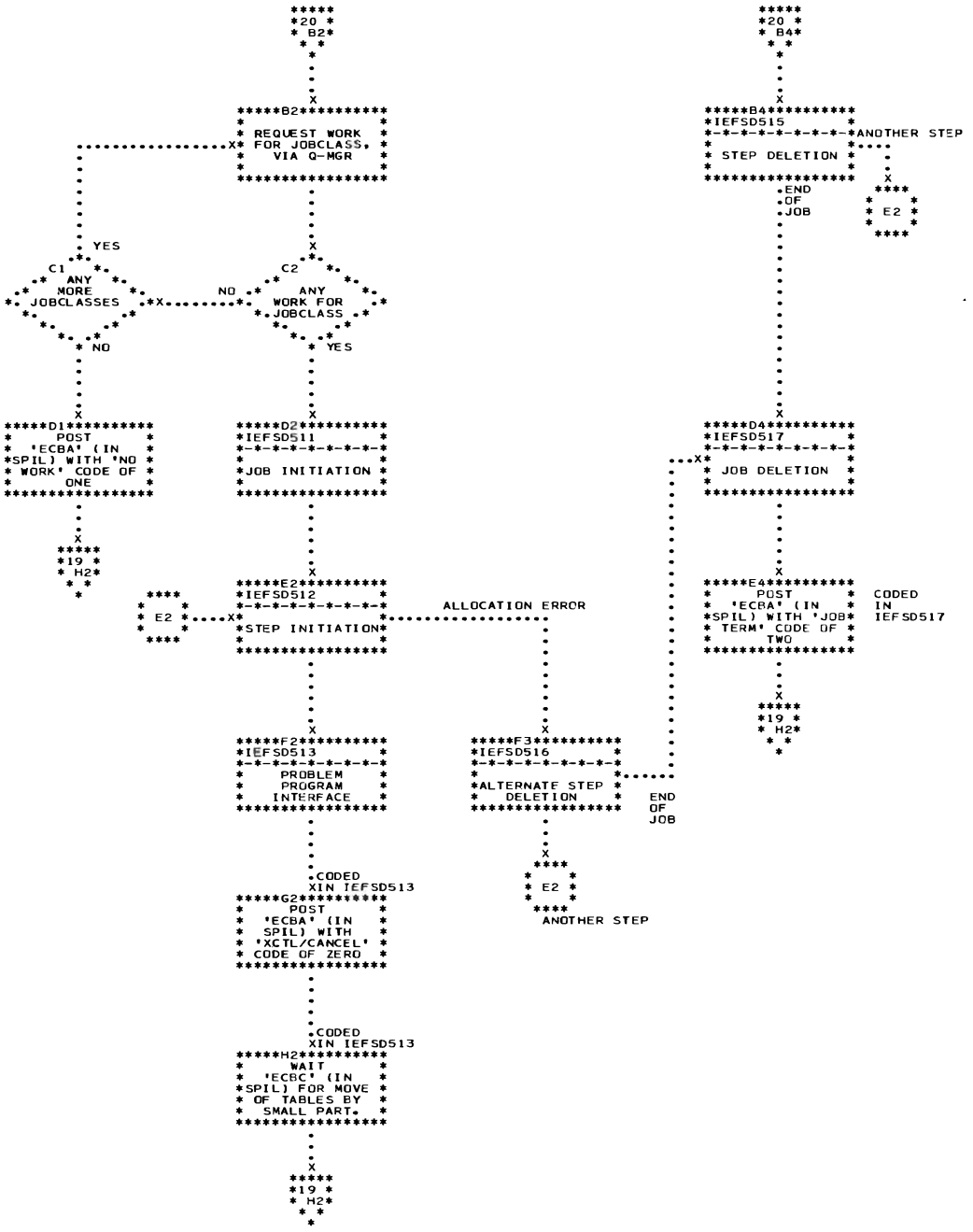


Chart 21. Job Selection Routine (Sheet 3 of 5)

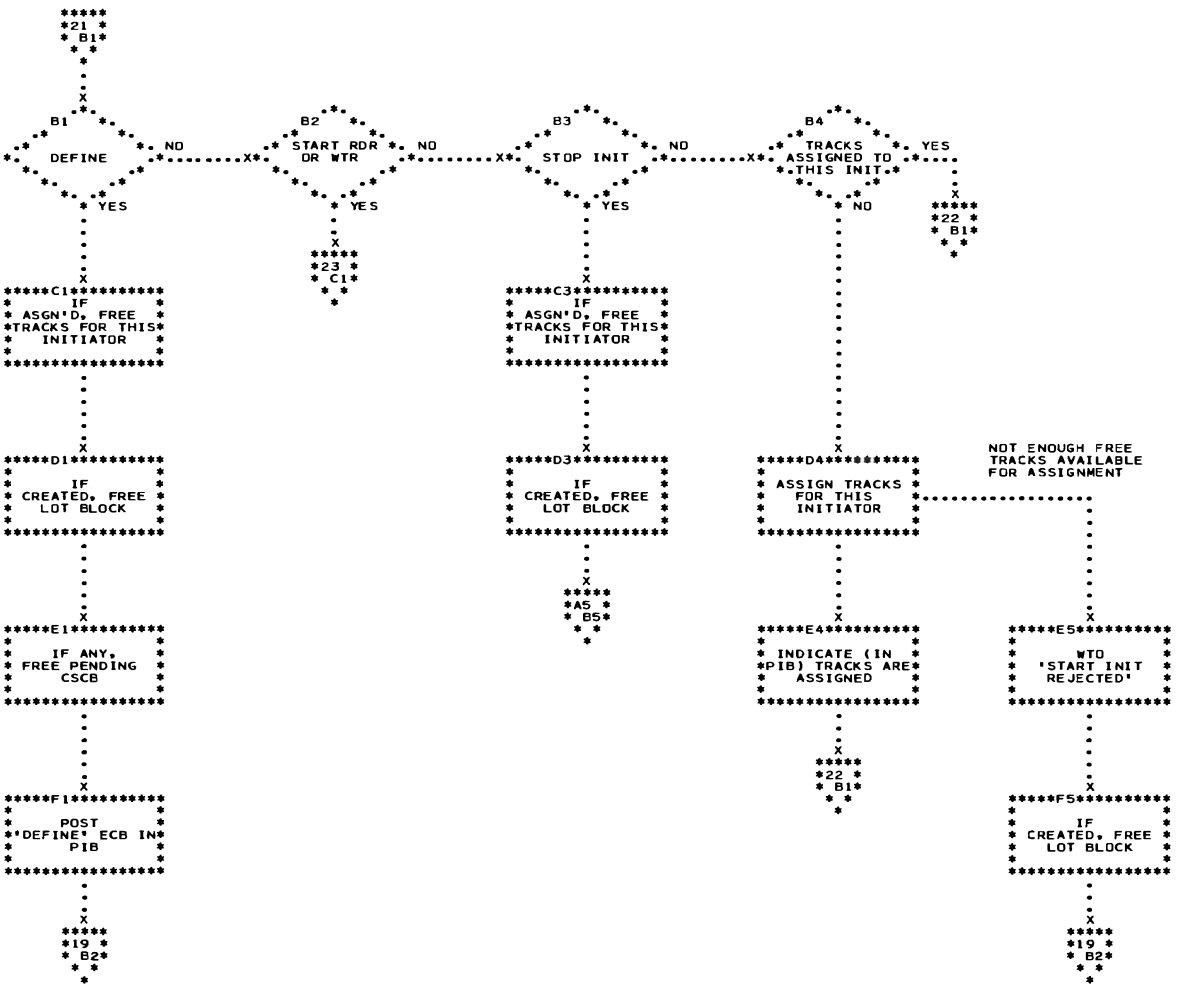
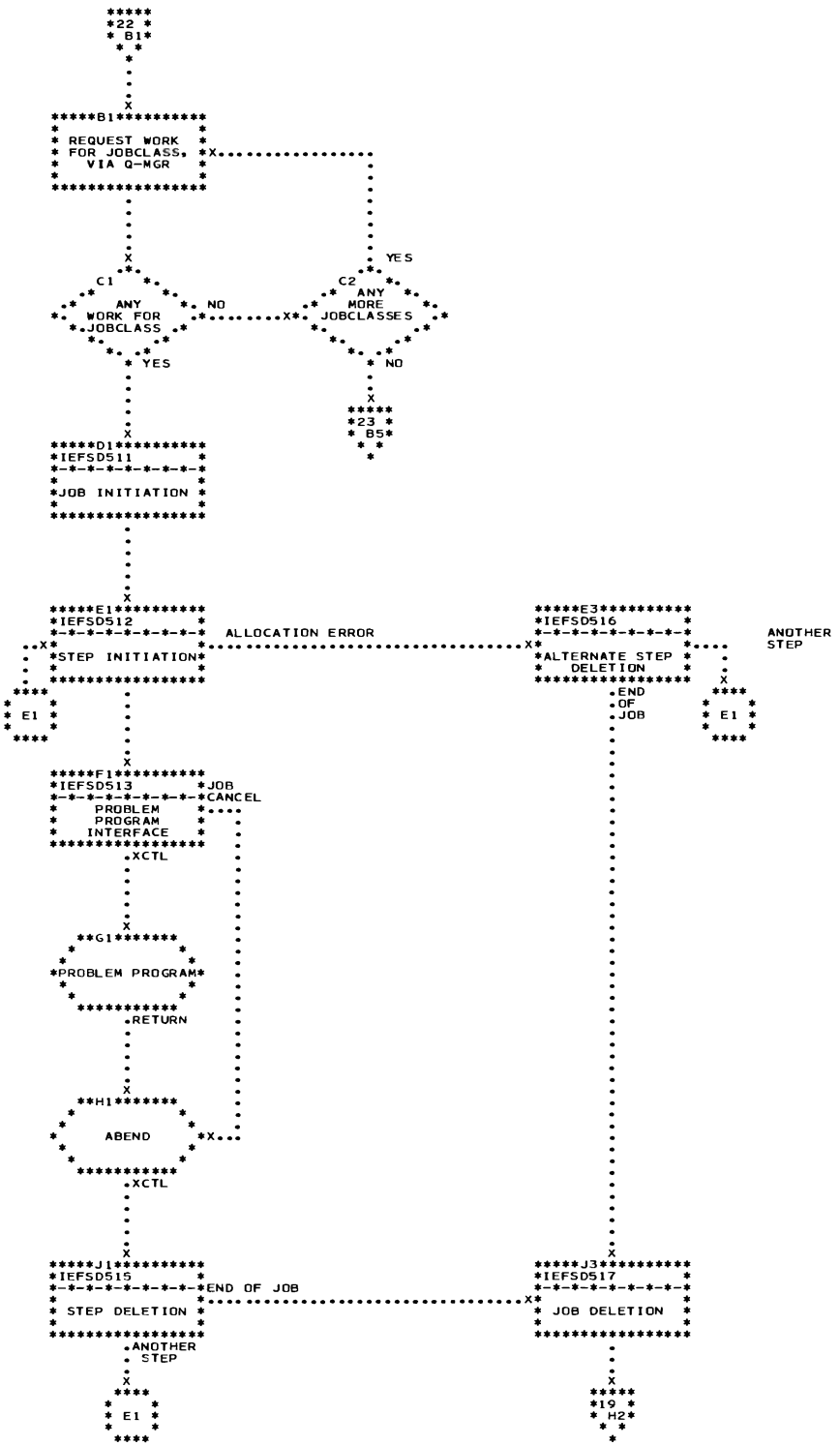


Chart 22. Job Selection Routine (Sheet 4 of 5)



• Chart 23. Job Selection Routine (Sheet 5 of 5)

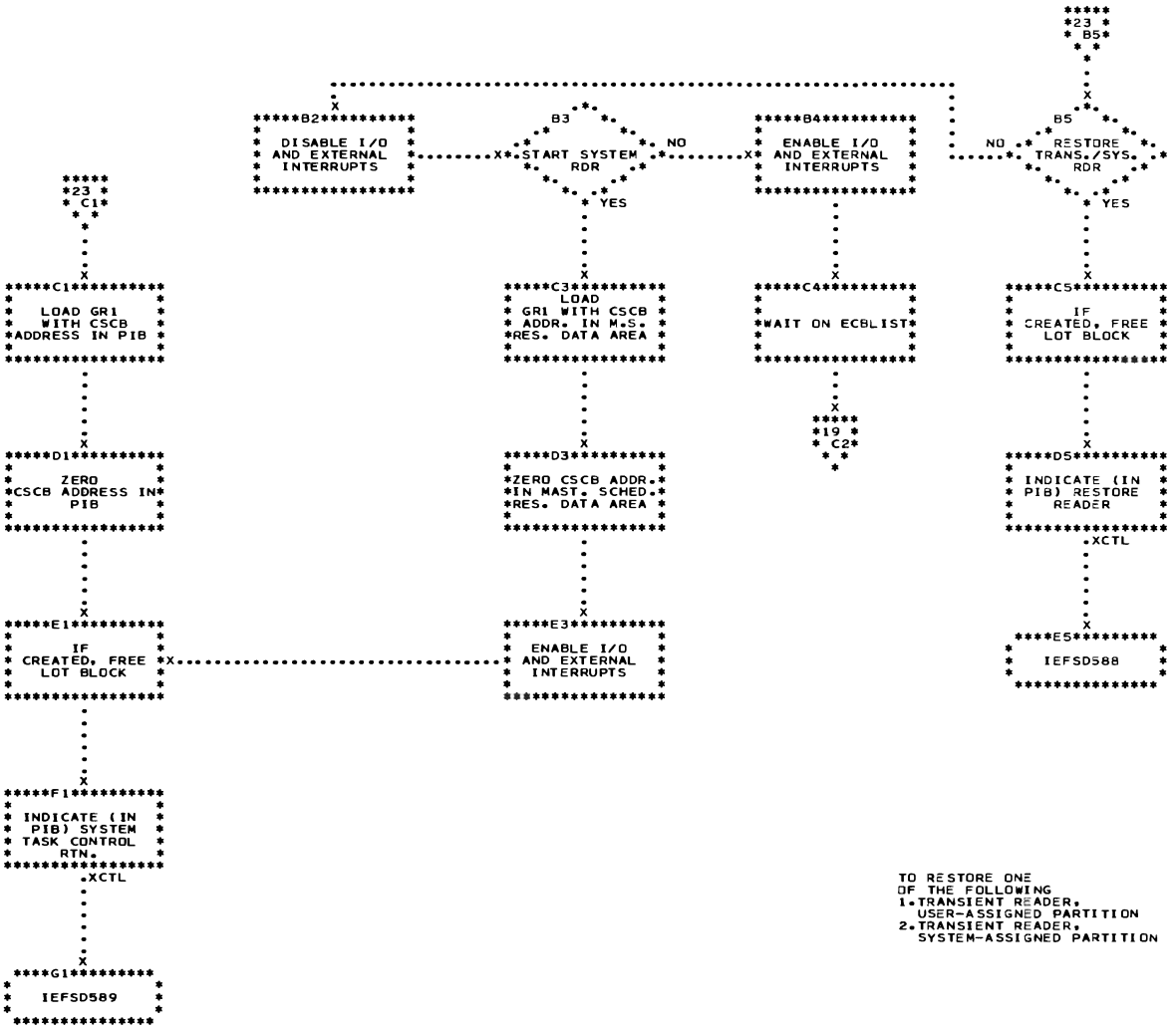
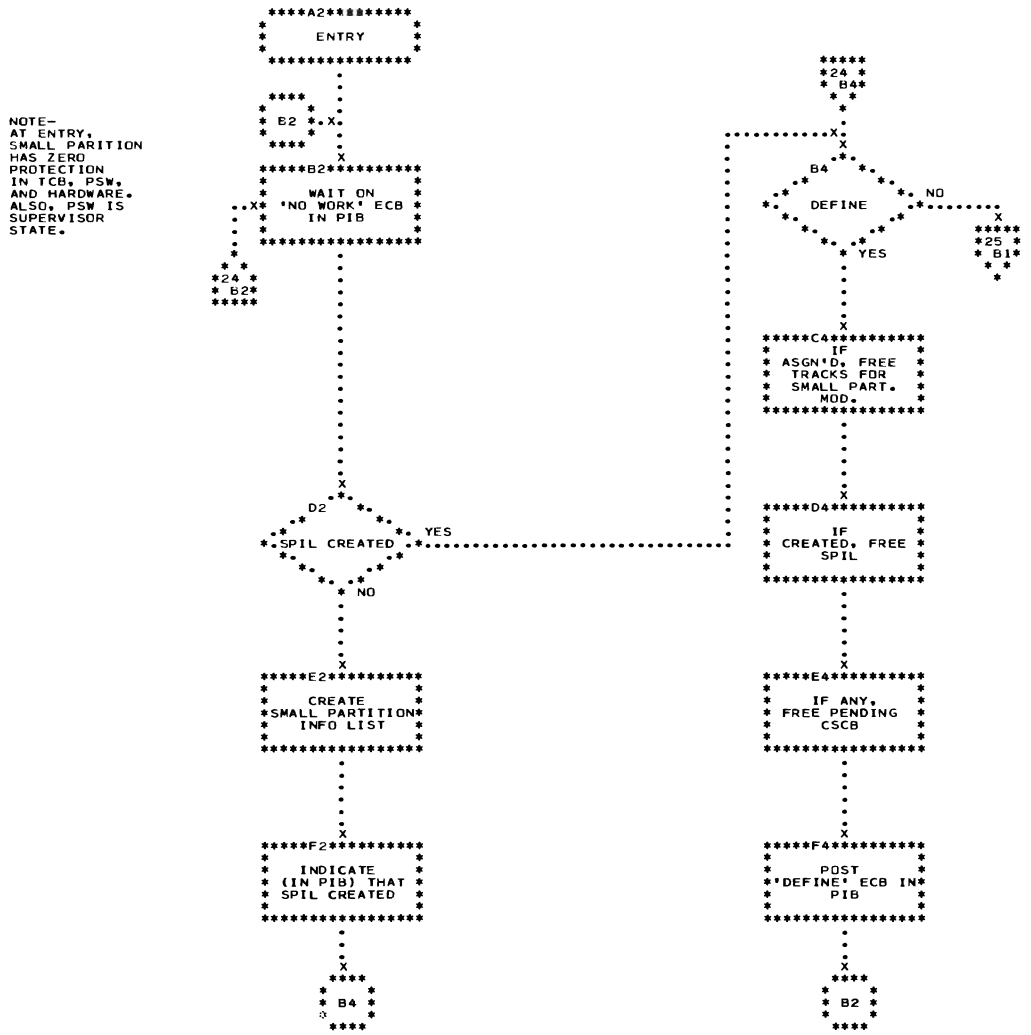


Chart 24. Small Partition Routine (Sheet 1 of 4)



• Chart 25. Small Partition Routine (Sheet 2 of 4)

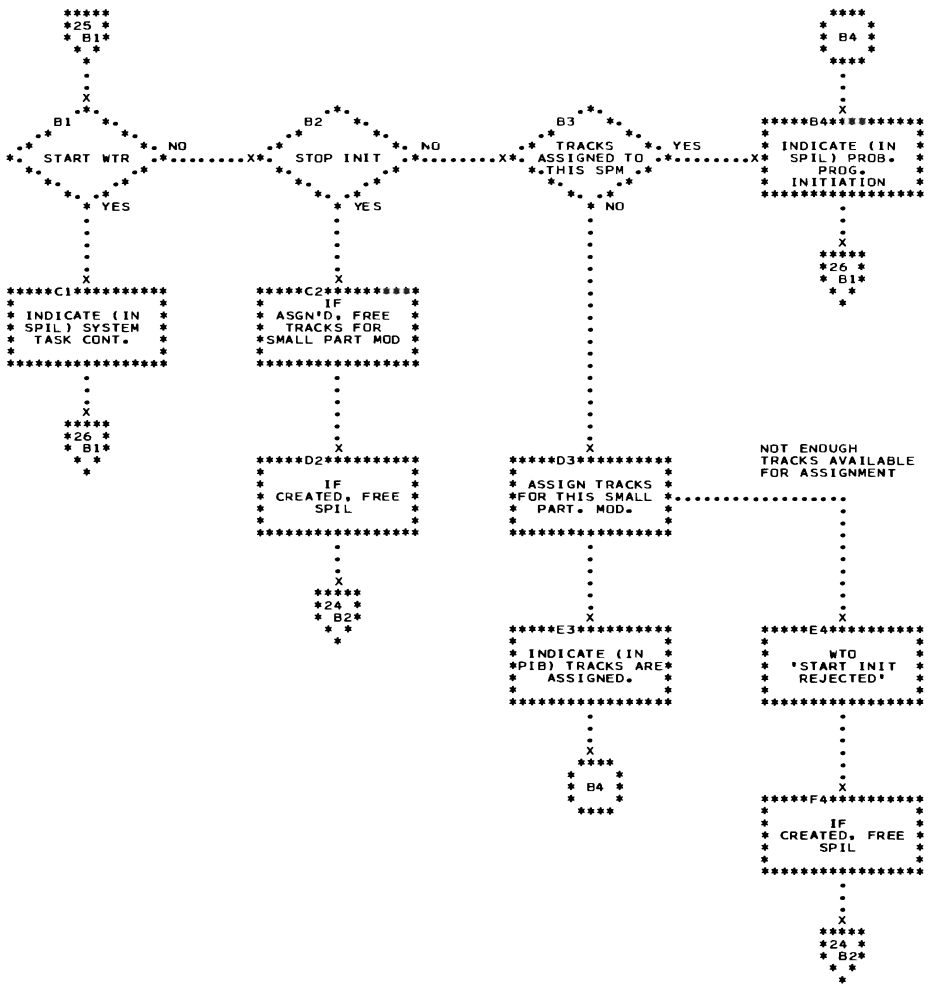


Chart 26. Small Partition Routine (Sheet 3 of 4)

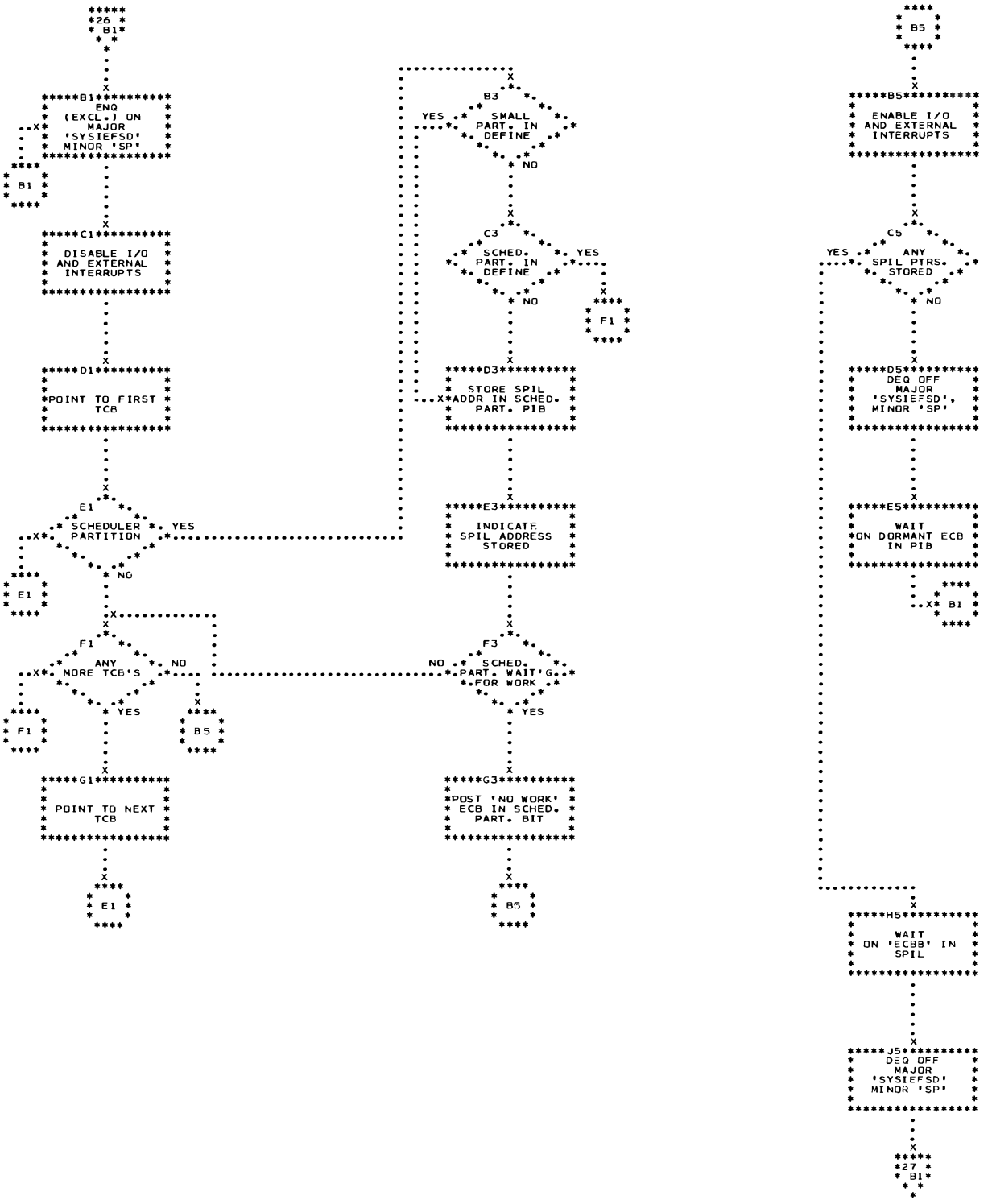
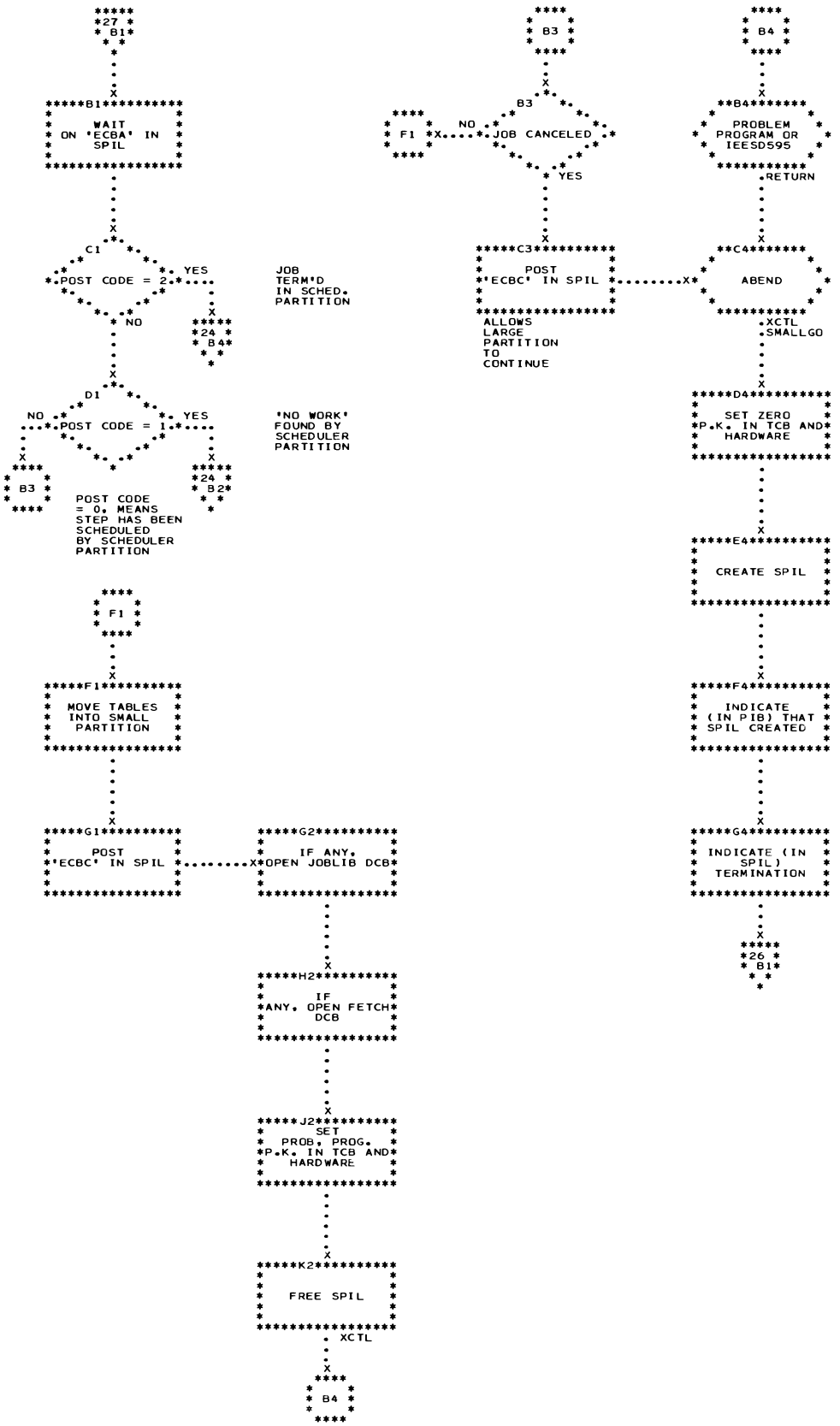
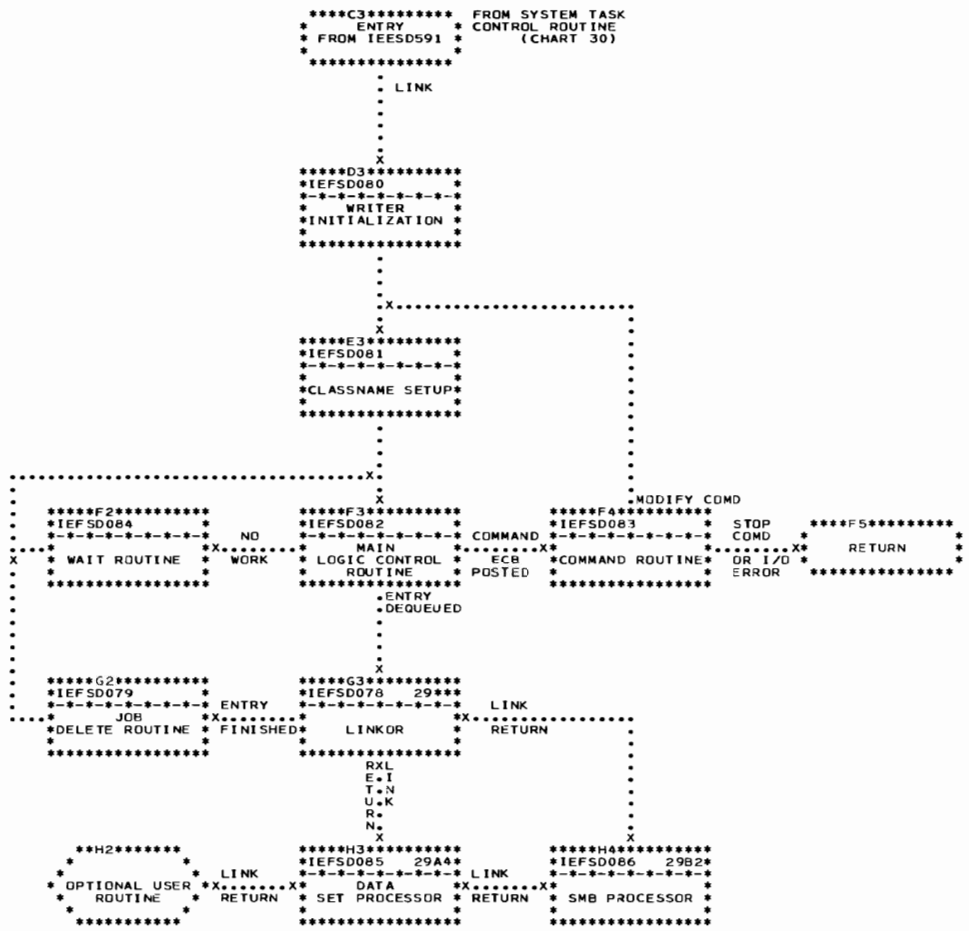


Chart 27. Small Partition Routine (Sheet 4 of 4)



• Chart 28. System Output Writer Control Flow



• Chart 29. System Output Writer

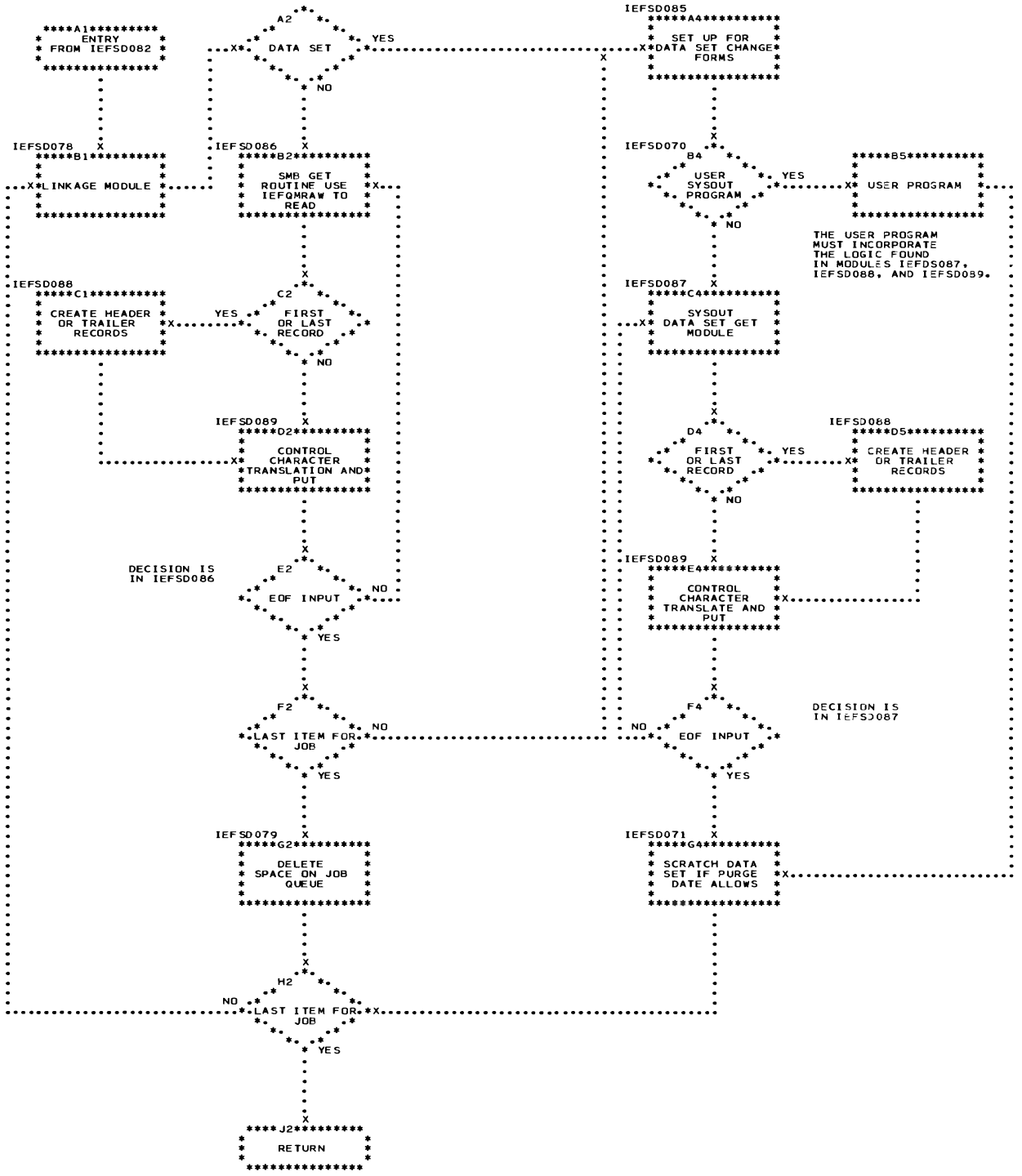
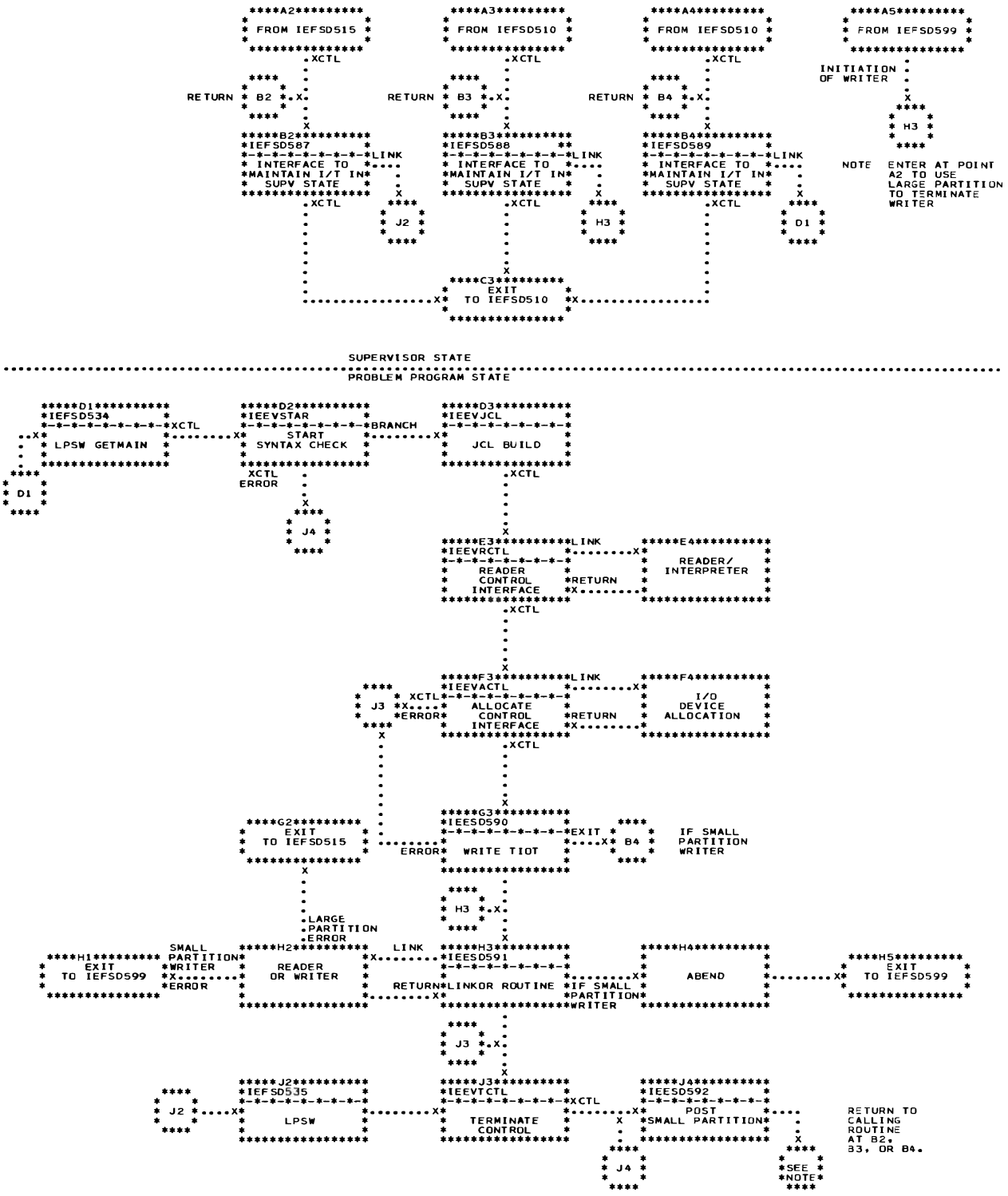
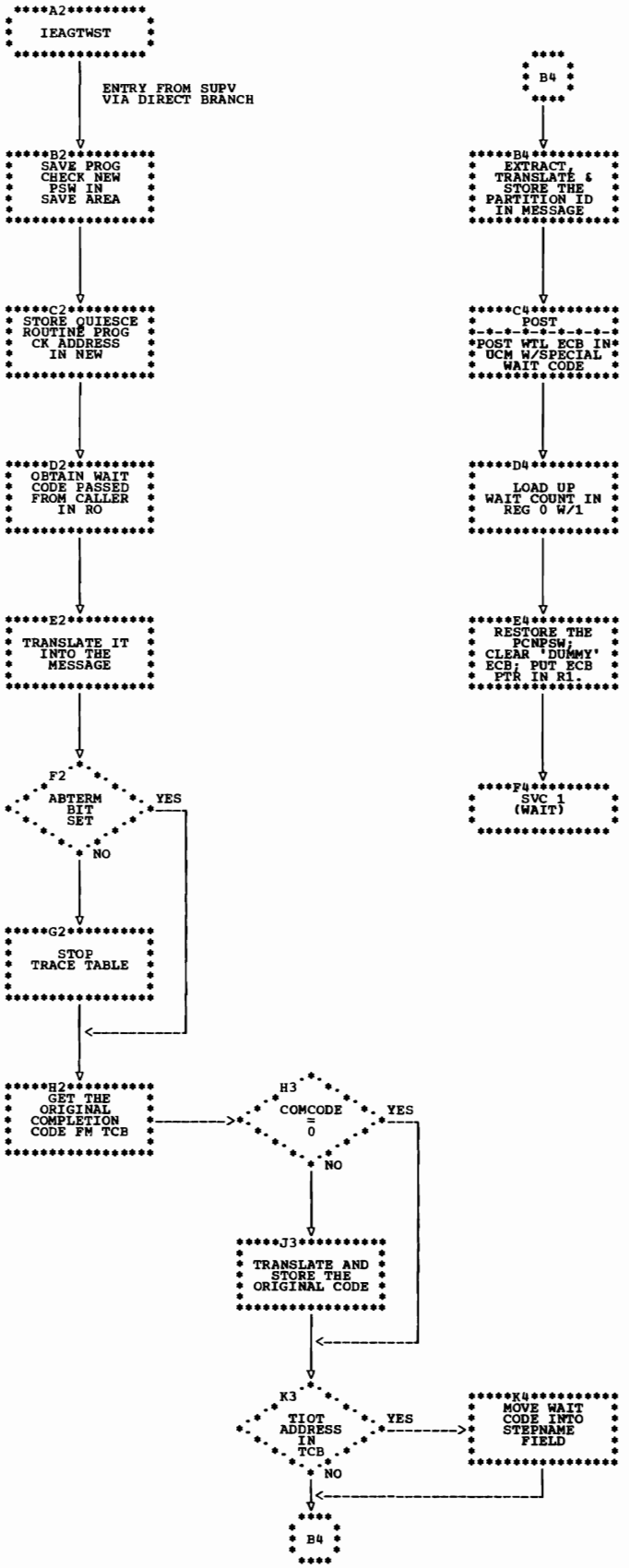


Chart 30. System Task Control



• Chart 31. System Quiesce Routine



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

READER'S COMMENT FORM

IBM System/360 Operating System; Control Program With MFT
Program Numbers 360S-CI-505, 360S-CI-535

Y27-7128-3

Please check or fill in the items below, adding 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:

Thank you for your cooperation. No postage necessary if mailed in the U.S.A.

YOUR COMMENTS PLEASE . . .

This manual is one of a series which serves as reference sources for systems analysts, programmers and operators of IBM systems. Your answers to the questions on the back of this form, together with your comments, will help us produce better publications for your use. Each reply will be carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

Please note: Requests for copies of publications and for assistance in utilizing your IBM system should be directed to your IBM representative or to the IBM sales office serving your locality.

FOLD FOLD

FIRST CLASS
PERMIT NO. 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

Printed in U.S.A. Y27-7128-3

IBM®

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

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

IBM S/360 Printed in U.S.A. Y28-7128-3

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

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

READER'S COMMENT FORM

IBM System/360 Operating System
Control Program with MFT
Program Logic Manual

Form Y27-7128-3

- Is the material:

	Yes	No
Easy to read?	☐	☐
Well organized?	☐	☐
Complete?	☐	☐
Well illustrated?	☐	☐
Accurate?	☐	☐
Suitable for its intended audience?	☐	☐

- How did you use this publication?
  - ☐ As an introduction to the subject
  - ☐ For additional knowledge
  - Other

- Please check the items that describe your position:
  - ☐ Customer personnel
  - ☐ IBM personnel
  - ☐ Manager
  - ☐ Systems Analyst
  - ☐ Operator
  - ☐ Programmer
  - ☐ Customer Engineer
  - ☐ Instructor
  - ☐ Sales Representative
  - ☐ Systems Engineer
  - ☐ Trainee
  - Other

- Please check specific criticism(s), give page number(s), and explain below:
  - ☐ Clarification on page(s)
  - ☐ Addition on page(s)
  - ☐ Deletion on page(s)
  - ☐ Error on page(s)

Explanation:

- Thank you for your cooperation. No postage necessary if mailed in the U.S.A.

YOUR COMMENTS PLEASE . . .

This publication is one of a series which serves as reference for systems analysts, programmers and operators of IBM systems. Your answers to the questions on the back of this form, together with your comments, will help us produce better publications for your use. Each reply will be carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

Please note: Requests for copies of publications and for assistance in utilizing your IBM system should be directed to your IBM representative or to the IBM sales office serving your locality.

Fold Fold

FIRST CLASS
PERMIT NO. 81
POUGHKEEPSIE, N.Y.

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

POSTAGE WILL BE PAID BY

IBM Corporation
P.O. Box 390
Poughkeepsie, N.Y. 12602

Attention: Programming Systems Publications
Department D58

Fold Fold

IBM S/360 Printed in U.S.A. Y28-7128-3

IBM®

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

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

IBM Technical Newsletter

File Number S360-36

Re: Form No. Y27-7128-3

This Newsletter No. Y28-2349

Date November 15, 1968

Previous Newsletter Nos. None

IBM SYSTEM/360 OPERATING SYSTEM
CONTROL PROGRAM WITH MFT
PROGRAM LOGIC MANUAL

This Technical Newsletter, a part of release 17 of IBM System/360 Operating System, provides replacement pages for the Control Program with MFT, Program Logic Manual, Form Y27-7128-3. These replacement pages remain in effect unless specifically altered. Pages to be inserted and/or removed are listed below.

Contents
Illustrations
17,18,18.1
23,24,24.1
29-34,34.1-.3
45-52
57,58
63,64
75,76,76.1
83-96,96.1-.2
99-102,102.1
117,118,118.1
123,124
127,128
154.1
155-157
Business Reply

A change to the text or a small change to an illustration is indicated by a vertical line to the left of the change; a changed or added illustration is denoted by the symbol • to the left of the caption.

Summary of Amendments

Changes have been made in the discussions of the DEFINE command processing and the ABEND service routine. New items discussed include: recording/recovery routines, main storage hierarchy support, job/step CPU timing, the system quiesce routine, and the STAE macro instruction. A revised Business Reply form is provided.

File this cover letter at the back of the manual to provide a record of changes.

IBM Corporation, Programming Systems Publications, P.O. Box 390, Poughkeepsie, N.Y. 12602

PRINTED IN U.S.A.