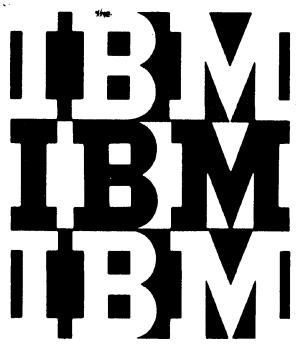


File No. S360-36 (OS)
GY27-7128-5

Program Logic

# IBM System/360 Operating System:

# Control Program With MFT,

# Program Logic Manual,

## Program Number 360S-CI-505

This publication describes the internal logic of the IBM System/360 Operating System Control Program with MFT. The publication provides an introduction to control program logic and describes the components of the program. It also describes the initialization of the operating system, the functions of the supervisor that differ from those of the PCP and MVT supervisors and the functions of job management that differ from those of PCP and MVT job management.

The appendix contains a description of all routines, major tables, and work areas used by MFT, and flowcharts of the routines of MFT that differ from those of either of the other control programs.

This manual is intended for persons involved in program maintenance, and system programmers who are altering the program design. Program logic information is not necessary for use and operation of the program.

Sixth Edition (June, 1970)

This is a major revision of, and obsoletes, Y27-7128-4. The text and illustrations have been changed to reflect the addition of the following:

- System management facilities.
- Direct system output processing.
- The multitasking capability and a revision of the ABEND and DAR routines.
- Write-to-programmer facility.

In addition, the text has been revised to include descriptions of: 7094 emulator support for the model 85; device-independent, operator-display console support; input/output recovery management support; channel-check handler dynamic loading; the express cancel facility; instream procedures; unit status display command; initiator modifications; changes to the UCB and the UCME; the resolution of the transient area contention problem; and revisions to the define command processing routines.

Other changes to the text, and small changes to illustrations, are indicated by a vertical line to the left of the change; changed or added illustrations are denoted by the symbol • to the left of the caption.

This edition applies to release 19 of the IBM System/360 Operating System, and to all subsequent releases until otherwise indicated in new editions or Technical Newsletters. Changes are continually made to the information herein; before using this publication in connection with the operation of IBM systems, consult the latest IBM System/360 SRL Newsletter, Order No. GN20-0360, for the editions that are applicable and current.

Requests for copies of IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A form for readers' comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, Programming Systems Publications, Department D58, PO Box 390, Poughkeepsie, N. Y. 12602

# Preface

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 Initialization of the Operating System 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 Operating System:

Principles of Operation, GA22-6821

Introduction to Control Program Logic, Program Logic Manual, GY28-6605

PCP Supervisor, Program Logic Manual, GY28-6612

MVT Job Management, Program Logic Manual, GY28-6660

Initial Program Loader and Nucleus Initialization Program, GY28-6661

MFT Guide, GY27-6939

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

IBM System/360 Operating System:

Concepts and Facilities, GC28-6535

Linkage Editor, GC28-6538

System Programmer's Guide, GC28-6550

System Generation, GC28-6554

MVT Control Program Logic Summary, GC28-6658

Input/Output Supervisor, Program Logic Manual, GY28-6616

MVT Supervisor, Program Logic Manual, GY28-6659

# Contents

# Illustrations

## Figures

## Tables



## Charts

# Summary of Major Changes--Release 19

| Name of Item | Description | Area of Publication Affected (Areas correspond to entries in the Table of Contents) |
|---|---|---|
| System Management Facilities (1) Follow-On | A set of routines and exits for user-supplied routines that gather information on system operation and place the information in special data sets. | Supervisor<br>Interruption Supervision -- The Dispatcher (Macro IEAAPS)<br><br>Interruption Supervision -- The Dispatcher (Macro IEAAPS): Dispatching a Task<br><br>Main Storage supervision<br><br>Task Supervision -- The Wait Routine<br><br>Timer Supervision -- Timer Second Level Interruption Handler (IEA0TI00): SMF Processing<br><br>System Management Facility<br><br>Job Management<br>Master Scheduler Task -- System Initialization<br><br>Master Scheduler Task -- Partition Definition by the Master Scheduler: Command Final Processor Routine (IEEDFIN9), MFT Storage Configuration Record Creation Routine (IEEDFINA)<br><br>Reader/Interpreter -- Reader Control Flow: Transient Reader Suspend Routine (IEFSD530), Transient Reader Restore Routine (IEFSD531)<br><br>Initiator/Terminator (Scheduler) -- Initiator/Terminator Control Flow: Step Initiation Routine (IEFSD512), SMF User Initiation Exit Routine (IEFSMFIE), Problem Program Interface Routine (IEFSD513), SMF TCTIOT Construction Routine (IEFSMFAT)<br><br>System Management Facility |
| Job Step Timing | A facility that provides for the accumulation of the CPU time used by a job step, and prevents a step from entering a wait state for more than 30 minutes. | Supervisor<br>Interruption Supervision -- The Dispatcher (Macro IEAAPS): Dispatching a Job, Handling Job Step Timing When a Task Switch is to Occur<br><br>Task Supervision -- The Wait Routine (Macro IEAAWT)<br><br>Task Supervision -- The Post Routine (Macro IEAAPT) |

(Part 1 of 6)

| Name of Item | Description | Area of Publication Affected (Areas correspond to entries in the Table of Contents) |
|---|---|---|
| Job Step Timing (Continued) | | Timer Supervision -- Timer Second Level Interruption Handler (IEA0TI00)<br><br>Job Management<br>Job Processing<br><br>Initiator/Terminator (Scheduler) -- Small Partition Scheduling: Terminating the Small Partition, Small Partition Module (IEFSD599)<br><br>Initiator/Terminator (Scheduler) -- Initator/Terminator Control Flow: Step Initiation Routine (IEFSD512), SMF User Exit Initiation Routine (IEFSMFIE) Problem Program Interface Routine (IEFSD513), Step Deletion Routine (IEFSD515) |
| System Management Facilities (2) | An extension to SMF (1) that provides the user with the information required to keep track of the installation's data set activity and status. | Supervisor<br>System Management Facility<br><br>Job Management<br>Initiator/Terminator (Scheduler) -- Initiator/Terminator Control Flow: SMF User Exit Initiation Routine (IEFSMFIE)<br><br>System Management Facility -- Comparison of SMF in MFT and MVT: SMF Initialization |
| 7094 Emulator Support for Model 85 | A facility that decodes 7094 instructions and passes control to the emulator for simulation. | Supervisor<br>Interruption Supervision -- The Dispatcher (Macro IEAAPS): Dispatching the 7094 Emulator Program for the Model 85 |
| Device Independent Display Operator Console Support (DIDOCS) | A facility that provides uniform operator console capabilities across a range of CRT devices. | Job Management<br>Communications Task with Multiple Console Support |
| Input/Output Recovery Management Support | Alternate Path Retry (APR) allows an I/O operation that has developed an error on one channel to be retried on another channel. Dynamic Device Reconfiguration (DDR) allows a demountable volume to be moved to another device without the occurrence of abnormal termination or the necessity of reperforming IPL. | Supervisor<br>Contents Supervision -- Contents Supervision in an MFT System without Subtasking: FINCH Service Routine (IEAATC00)<br><br>MFT Recording/Recovery Routines -- Alternate Path Retry Routine<br><br>MFT Recording/Recovery Routines -- Dynamic Device Reconfiguration Routine<br><br>MFT Recording/Recovery Routines -- Entry to Recording/Recovery Routines |

| Name of Item | Description | Area of Publication Affected (Areas correspond to entries in the Table of Contents) |
|---|---|---|
| Input/Output Recovery Management Support (continued) | | Job Management<br>Master Scheduler Task Functions<br>Command Processing |
| Channel Check Handler Dynamic Loading | A modification that consists of new internal CCH interfaces that enable the main part of CCH (channel and model independent) to link to various channel dependent analysis routines. | Supervisor<br>MFT Recording/Recovery Routines -- Machine Check Routines<br><br>MFT Recording/Recovery Routines -- Entry to Recording/Recovery Routines |
| Direct System Output Facility | A facility that provides the capability of writing system output directly to an output device. | Introduction<br>Theory of Operation<br><br>Job Management<br>Job Scheduler Functions<br><br>Job Management Control Flow<br><br>Job Processing<br><br>Initiator/Terminator (Scheduler) -- Job Selection (IEFSD510)<br><br>Initiator/Terminator (Scheduler) -- Initiator/Terminator Control Flow: Job Initiation Routine (IEFSD511), Step Initiation Routine (IEFSD512), Step Deletion Routine (IEFSD515), Partition Recovery Routine (IEFSD518)<br><br>System Output Writers -- Direct System Output Processing |
| Express CANCEL | A modification to the CANCEL processing that provides the facility for complete removal of a job from the system by the deletion of all of the job's queue entries, and the scratching of all of its data sets. | Job Management<br>Master Scheduler Task -- SVC 34 Functions: DEFINE and MOUNT Routine (IEESD571), CANCEL Command Routine IEE2803D)<br><br>Master Scheduler Task -- Master Scheduler Service Routines: Queue Search Setup Routine (IEESD563), Queue Scratch Setup Routine (IEESD575), Queue Alter Delete Routine (IEESD576), Queue Restart Enqueue Routine (IEESD577), Queue Message Class Setup Routine (IEESD578), Queue SMB Routine (IEESD579), Specific Cancel Message Routine (IEESD580), Queue Scratch Routine (IEESD581) |
| Unit Status Display | A facility that allows the operator, via the DISPLAY U command, to request a display of detailed information about the input/output devices specified for system generation. | Job Management<br>Master Scheduler Task Functions<br><br>Master Scheduler -- Master Scheduler Service Routine: Syntax Check Routine (IEESD562), DISPLAY U Routines (IEEUNIT1,IEEUNIT2,IEEUNIT3,IEEUNIT4) |

(Part 3 of 6)

| Name of Item | Description | Area of Publication Affected (Areas correspond to entries in the Table of Contents) |
|---|---|---|
| ATTACH in MFT | A facility that allows the user to have, within a partition, multitasking capabilities compatible with the current MVT ATTACH function. | Introduction<br><br>Supervisor<br>Interruption Supervision<br><br>Interruption Supervision -- The Dispatcher (Macro IEAAPS)<br><br>Interruption Supervision -- The Dispatcher (Macro IEAAPS): Dispatching a Task with Time-Slicing<br><br>Interruption Supervision -- SVC Second Level Interruption Handler<br><br>Interruption Supervision -- EXIT (Macro IEAATA)<br><br>Interruption Supervision -- ABEND Service Routine<br><br>*Interruption Supervision -- ABEND Service Routine: Normal Termination, Abnormal Termination<br><br>Interruption Supervision -- Damage Assessment Routines: DAR Core Image Dump Routine (IEADTM22), DAR Task Reinstatement Routine (IEADTM23)<br><br>Task Supervision -- The ATTACH Routine (MFT with subtasking) (Macro IEAQAT00)<br><br>Task Supervision -- The CHAP Routine (MFT with Subtasking)<br><br>Task Supervision -- The Detach Routine (MFT with Subtasking)<br><br>Task Supervision -- The Extract Routine (MFT with Subtasking)<br><br>Task Supervision -- The ENQ/DEQ Routine (IEAGENQ1)<br><br>Contents Supervision -- Contents Supervision in an MFT System with Subtasking<br><br>Main Storage Supervision<br>Checkpoint/Restart Routines<br><br>Job Management<br>Master Scheduler Task -- Master Scheduler Service Routines: DISPLAY A Routine (IEESD566)<br><br>*All of the ABEND routines described in this section are affected by "ATTACH in MFT" except those specified as being used by MFT without subtasking. |

(Part 4 of 6)

| Name of Item | Description | Area of Publication Affected (Areas correspond to entries in the Table of Contents) |
|---|---|---|
| New ABEND and DAR routines | The ABEND and DAR routines for MVT have been rewritten to support MFT with the subtasking option. | Supervisor<br>Interruption Supervision<br><br>Interruption Supervision -- STAE Service Routine<br><br>*Interruption Supervision -- ABEND Service Routine<br><br>Interruption Supervision -- Damage Assessment Routines: DAR Core Image Dump Routine (IEADTM22), DAR Task Reinstatement Routine (IEADTM23)<br><br>*This section of the publication has been completely rewritten to describe the new ABEND processing. |
| Write-to-Programmer | A facility that allows the system and/or a problem program to write messages to the programmer via a macro instruction. | Supervisor<br>Main Storage Supervision<br><br>Job Management<br>Communications Task -- WTO/WTOR Macro Instruction Processing<br><br>Communication Task Modules -- Write-Routines (IEECVWTO and IEEVWTOR)<br><br>Master Scheduler Task -- Write-to-Programmer Message Processing Routines (IEFWTP00, IEFWTP01, and IEFWTP02)<br><br>Queue Manager<br><br>Queue Manager -- Queue Manager Modules: Transient Queue Manager Routines (IEFXQM00, IEFXQM01, IEFXQM02)<br><br>Initiator/Terminator (Scheduler) -- Small Partition Scheduling: Small Partition Module (IEFSD599)<br><br>Initiator/Terminator (Scheduler) Initiator/Terminator Control Flow: Problem Program Interface Routine (IEFSD513), Step Deletion Routine (IEFSD515)<br><br>System Task Control -- Initiating System Tasks: START Syntax Check Routine (IEEVSTAR) |
| L-Shape/Initiator Merge | The combination of System Task Control and the Initiator into one basic set of routines. | Job Management<br>System Task Control -- Initiating System Tasks: Allocation Interface Control Routine (IEEVACTL), QMPA Builder Routine (IEEVSMBA), Termination Interface Control Routine (IEEVTCTL) |

(Part 5 of 6)

| Name of Item | Description | Area of Publication Affected (Areas correspond to entries in the Table of Contents) |
|---|---|---|
| Resolution of the Transient Area Contention Problem | A modification that places SVRBs representing requests for SVC transient area loading in a wait state, and provides for the loading task to operate under control of a system TCB. | Supervisor<br>Interruption Supervision -- The Dispatcher (Macro IEAAPS)<br><br>Interruption Supervision -- The Dispatcher (Macro IEAAPS): Dispatching a Task<br><br>Contents Supervision -- Contents Supervision in an MFT System Without Subtasking: FINCH Service Routine (IEAATC00) |
| Separation of a Module of the Command Scheduling Routines (SVC 34) | A modification that separates the STOP INIT and START Commands Routine into the STOP INIT and START Commands Syntax Check Routine and the STOP INIT and START Commands Processor Routine. | Job Management<br>Master Scheduler Task -- SVC 34 Functions: STOP INIT and START Commands Processing Routines (IEESD561) and (IEE3903D) |
| Separation of two Modules of the Master Scheduler Partition Definition Routines | A modification that separates the System Reinitialization Routine into the System Reinitialization Routine (Part 1) and the System Reinitialization Routine (Part 2) and also separates the DEFINE Command Validity Check Routine into the DEFINE Command Validity Check Routine (Processor Storage) and the DEFINE Command Validity Check Routine (Core Storage). | Job Management<br>Master Scheduler Task -- Partition Definition by the Master Scheduler: Validity Check Routine-Processor Storage (IEEDFIN3), Validity Check Routine-Core Storage (IEEDFINC), System Reinitialization Routine-1 (IEEDFIN8), System Reinitialization Routine-2 (IEEDFINB) |
| Resolution of Design-Point Problem in Termination | For a 30K scheduler, load module IEFSD515 has been split into three separate load modules; IEFSD515, IEFSD517 and IEFSD168. IEFSD517 now performs the job termination function and IEFSD168 performs the job suspension function. | Job Management<br>Initiator/Terminator (Schedule) -- Initiator/Terminator Control Flow: Step Deletion Routine (IEFSD515) |
| Resolution of Design-Point Problem in Allocation | Step initiation routine IEFSD515 now passes control to Allocation via an XCTL macro instruction rather than a LINK macro instruction. | Job Management<br>Initiator/Terminator (Scheduler) -- Initiator/Terminator Control Flow: Step Initiation Routine IEFSD512 |
| Tables and work areas in Appendix A, module descriptions in Appendix B, and flowcharts in Appendix C have been changed to reflect the revisions.<br><br>The Command Scheduling Control Block and the Interpreter Work Area in Appendix A have been redrawn to include the symbolic field names and more detailed descriptions. Prose descriptions of the bit settings in the switch fields have also been included. The remaining tables and work areas in Appendix A will be described in this manner in future revisions of the publication. | | |

(Part 6 of 6)

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 fixed 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 fixed area depends on the number of partitions established by the user, and the control program options selected at system generation.

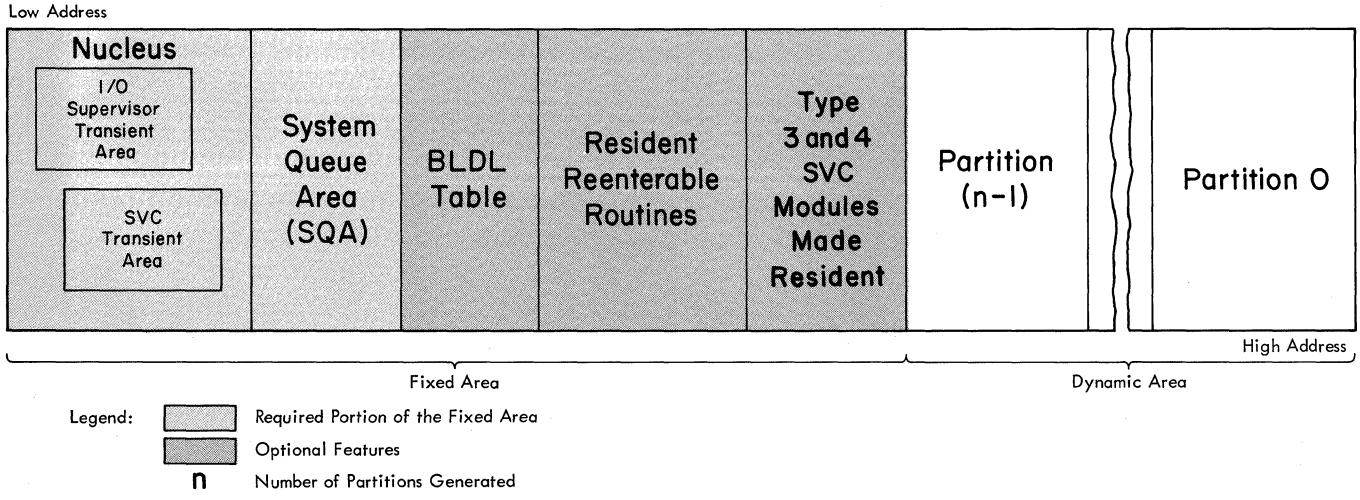
Partitions are defined within the dynamic area, located in the upper portion of main storage, at system generation. The number of partitions may be varied within the number specified at system generation, and the sizes and job classes of partitions may be redefined at system initialization or during operation. See the MFT Guide SRL. 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).

Provided the total number of partitions does not exceed 52 and enough computing system resources are available, 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. 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 task control block (TCB) on the dispatching queue. (The dispatching queue is the chain of TCBs indicated by the TCBTCB fields. If the MFT system does not have the subtasking option, all TCBs are established in the nucleus at system generation. These are ordered to provide a dispatching priority starting with resident system task TCBs, through the job-step task TCB of the highest priority partition (P0), to the successively lower priority partitions' TCBs (P1-P51). Control of the CPU is given to the program represented by the highest-priority ready TCB.

If the MFT system has the subtasking option, TCBs established at system generation in the nucleus represent the resident system tasks and the job-step task of each




• Figure 1. Main Storage Organization in MFT

partition. However, each job-step task can attach subtasks, each of which will have TCB located in the system queue area. The dispatching priority is initially the same as the partition priority. The dispatching priority differs from the partition priority when a job-step task issues a CHAP (change priority) macro instruction to change its dispatching priority. If dispatching priorities are changed, each partition job-step task is dispatched before its subtasks, which are then dispatched in the order in which they were attached. When all of a job-step's subtasks have been dispatched, the job-step task of the next lower partition can be dispatched.

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

PCP Supervisor, GY28-6612

MVT Supervisor, GY28-6659

MVT Job Management, GY28-6660

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

The Initialization of the Operating System section describes how the dynamic area of main storage is prepared by the master scheduler task after completion of the Nucleus Initialization Program.

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, direct system output processing, and system task control routines.

## Functions of the Control Program with MF

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, GY28-6616

Sequential Access Methods, GY28-6604

Indexed Sequential Access Methods, GY28-6618

Basic Direct Access Method, GY28-6617

Graphics Access Method, GY27-7113

Catalog Management, GY28-6606

Direct Access Device Space Management, GY28-6607

Input/Output Support (OPEN/CLOSE/EOV), GY28-6609

## Control Program Organization

The control program is 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 is 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 the Supervisor section of this publication, and in the PCP Supervisor PLM.

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, GY28-6616

Sequential Access Method, GY28-6604

The user may also select resident reenterable routines, which are access method routines from SYS1.SVCLIB, and other reenterable routines from SYS1.LINKLIB. At system generation, the user specifies that he wants such routines resident in main storage. At IPL, he identifies the specific routines desired in the RAM=entry. The selected routines are loaded during system initialization and reside adjacent to the higher end of the system queue area unless the BLDL table is also 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. NIP loads these routines adjacent to the higher end of the resident reenterable routines. If there is no resident BLDL table or resident reenterable routines, the routines are loaded adjacent

to the higher end of the system queue area. (See Figure 1.)

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 the optional resident areas are adjacent to the nucleus.

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 offline, 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.

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.

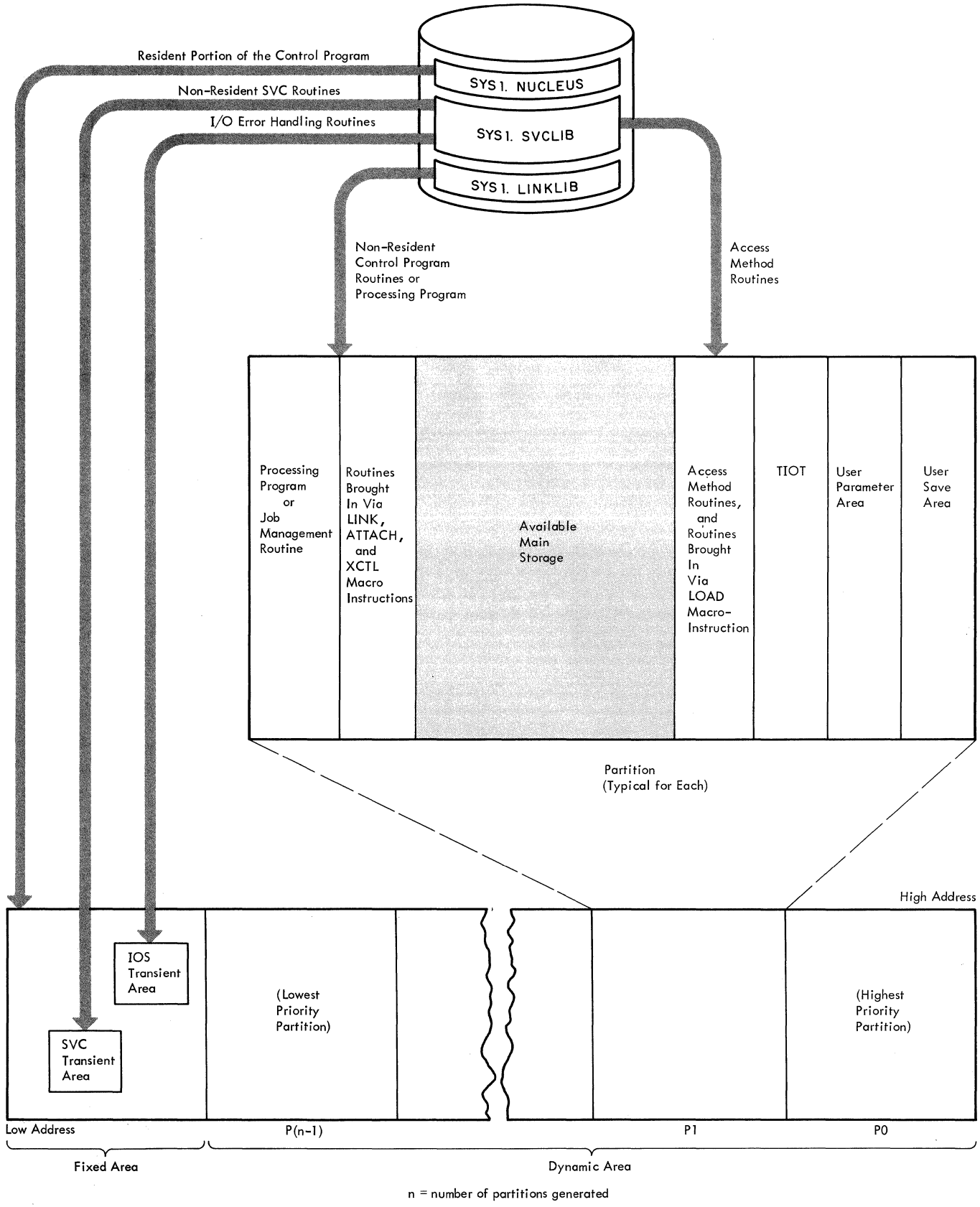
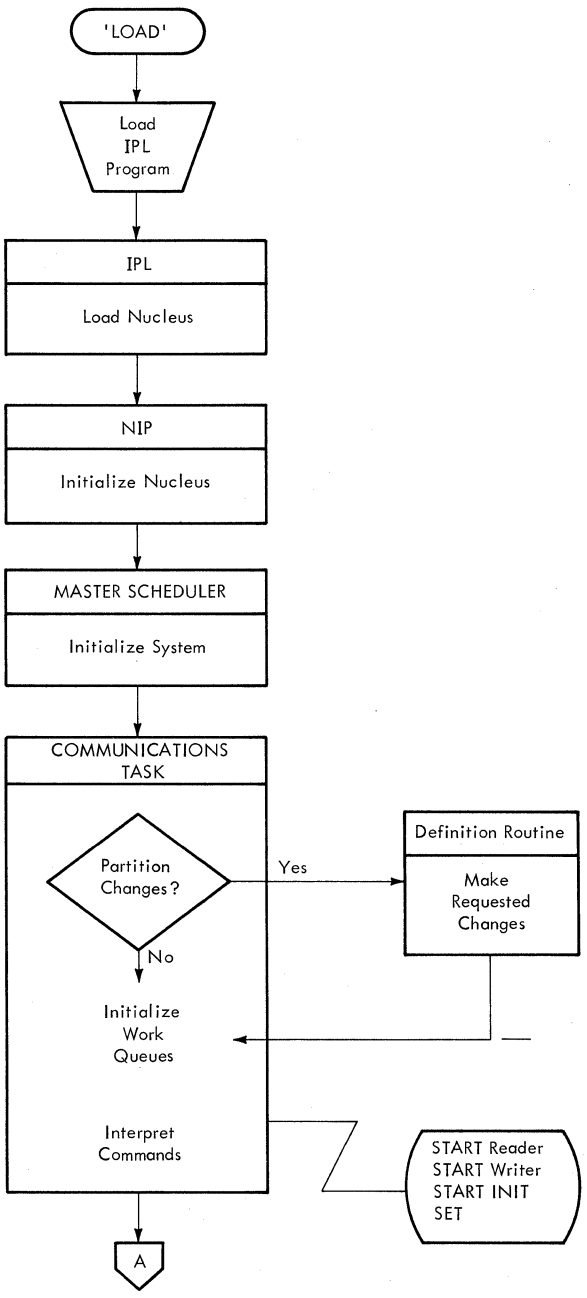



Figure 2. Division of Main Storage

## 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.

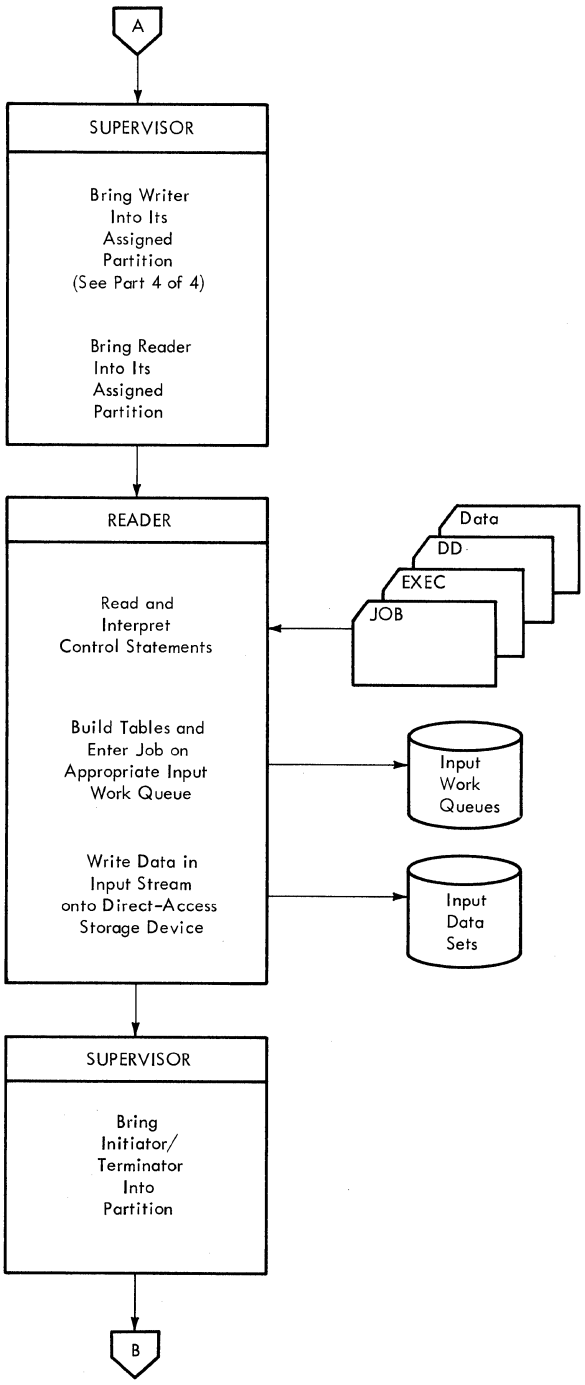



To load the nucleus, the operator sets the LOAD UNIT switches to the device on which the system residence volume is mounted, and presses the LOAD button on the operator control panel. This causes an IPL record to be read and to be given control. This record causes the second IPL record to be read, which 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 command is entered, the communications task calls the master scheduler command scheduling routine to have the command executed. An automatic START reader command or a subsequent operator entered START reader command causes a copy of the Reader/Interpreter (reader) to be 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:

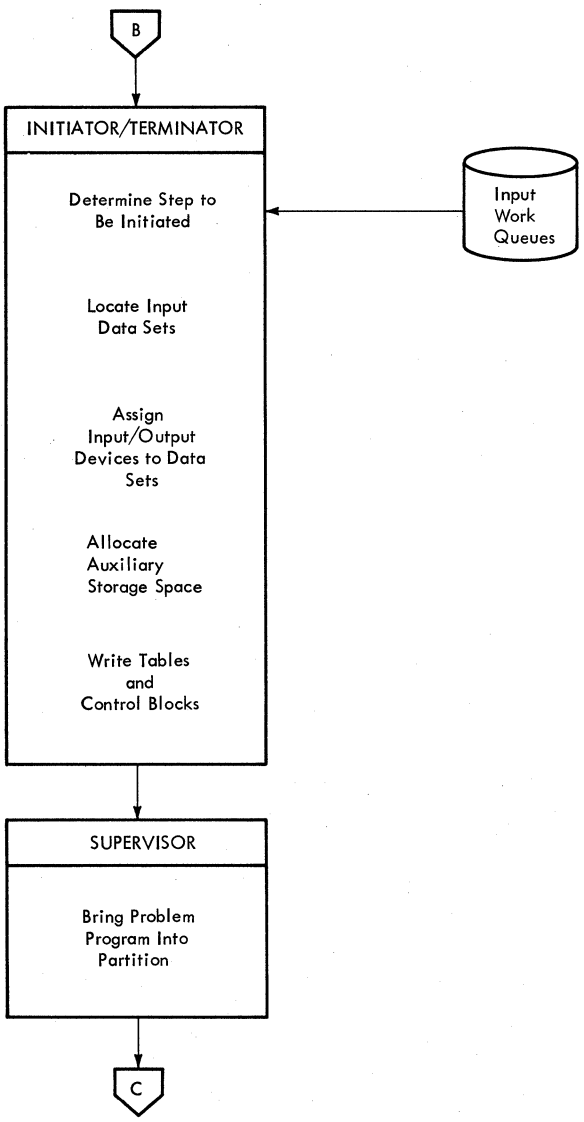
- Job control table (JCT) for the job being read.
- Step control table (SCT) for the step being read.
- Data set enqueue table (DSENQ) for the job being read.
- Job file control block (JFCB) and step input/output table (SIOT) for each data set being used or created by the job step.
- Volume table (VOLT) 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, GY28-6606.)

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.

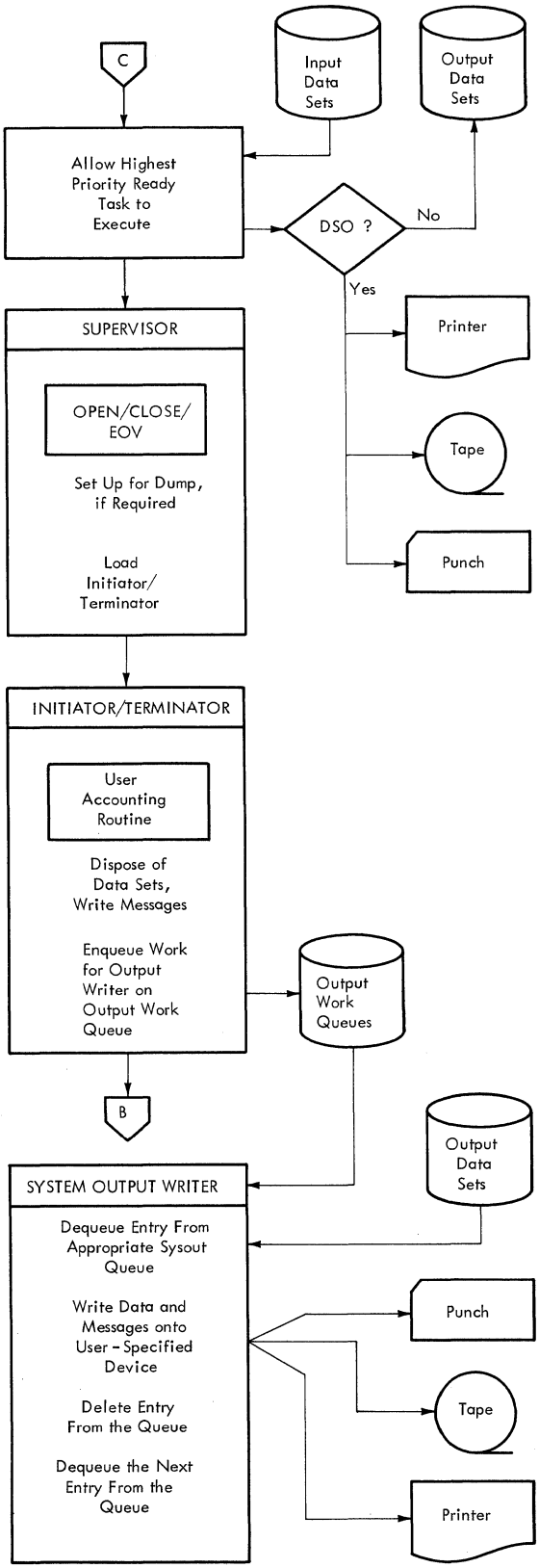
Allocates Auxiliary Storage Space: Direct access volume space required for output data sets of a job step not using direct system output (DSO) processing is acquired by the allocation routine, which uses the Direct Access Device Space Management (DADSM) routines. (A description of the operation of the DADSM routines is given in the publication IBM System/360 Operating System: Direct Access Device Space Management, Program Logic Manual, GY28-6607.)

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)




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, GY28-6609.)

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 termination, of a job not using direct system output processing, 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.

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

# Initialization of the Operating System

When the system is loaded, routines perform required and optional initialization of functions needed for control program operation. (These routines are described in the Initial Program Loader and Nucleus Initialization Program, Program Logic Manual.) When the Nucleus Initialization Program (NIP) has defined the fixed area, it then assigns the rest of main storage to the master scheduler task to be prepared as the dynamic area for control program operation.

## 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. 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 contrcl 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.

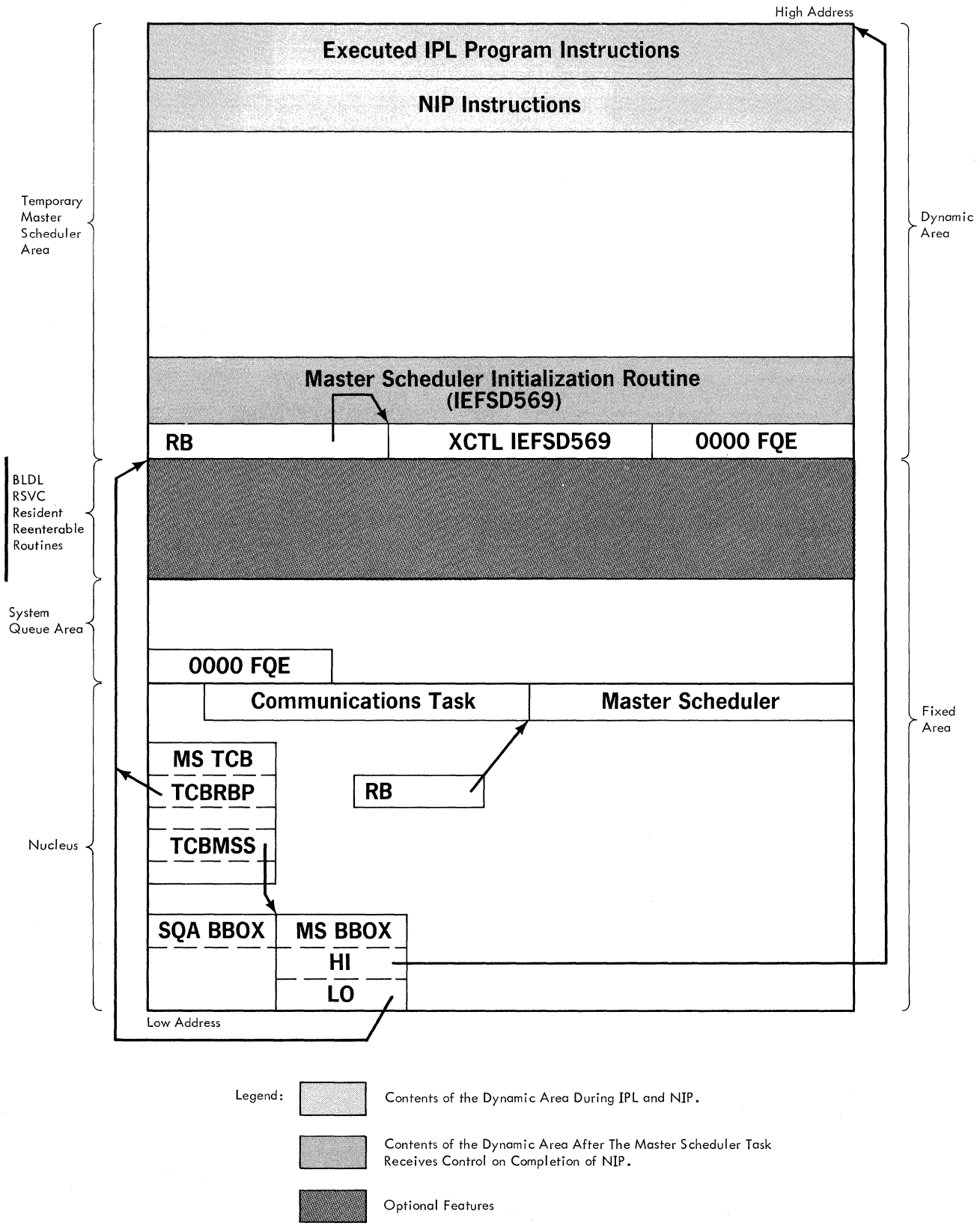



Figure 4. Main Storage During Execution of NIP

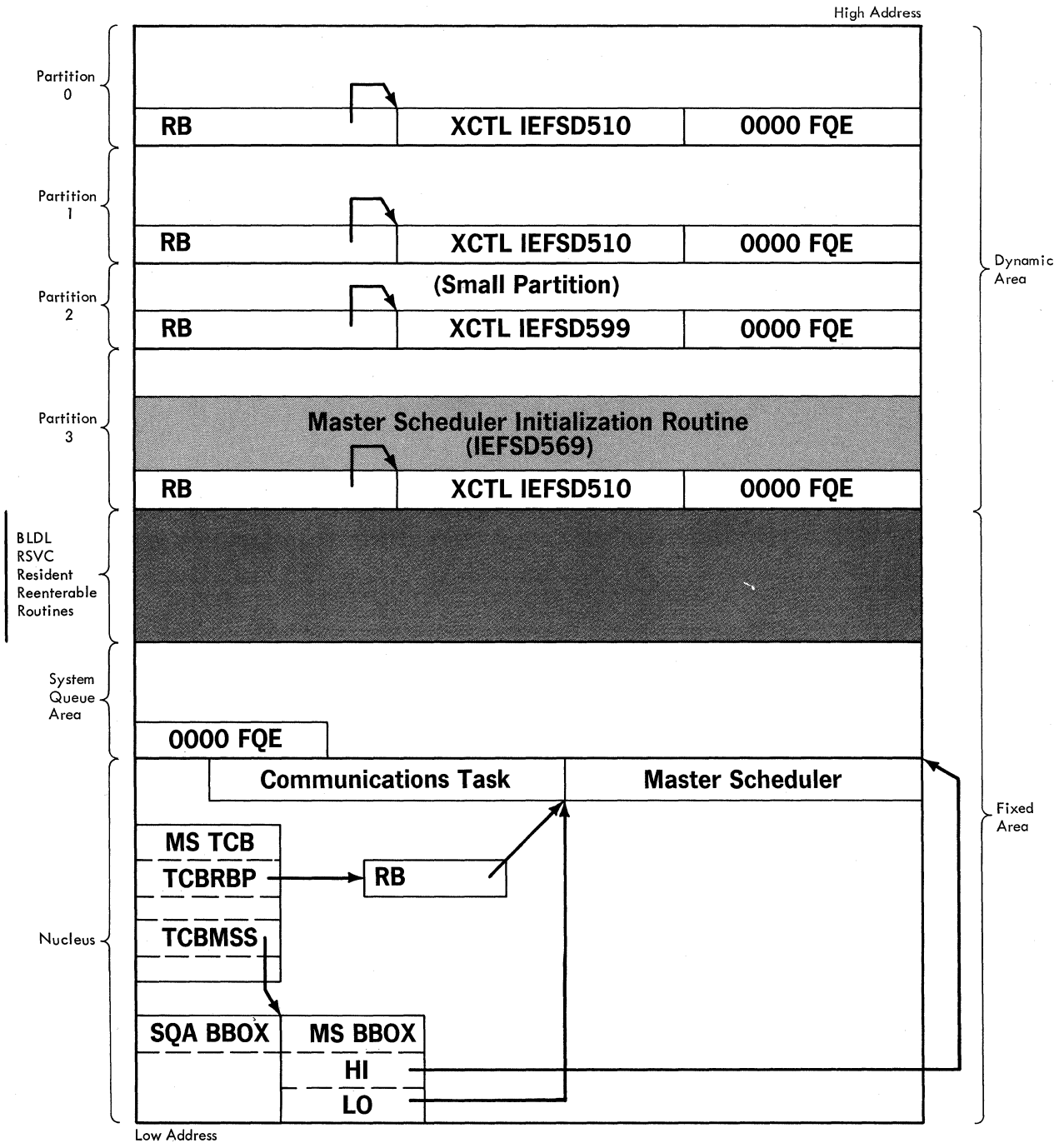



Figure 5. Main Storage at Termination of Master Scheduler Initialization

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. In MFT systems with subtasking, the supervisor also records the main storage allocated by the system to attached subtasks.

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

The interruption supervision routines in MFT function in the manner described in the PCP Supervisor PLM with the exception of:

- The Dispatcher
- The supervisor call second level interruption handler (SVC SLIH).
- The Exit routine.
- The ABEND routines.
- The timer second level interruption handler (TSLIH).

The timer second level interruption handler is described in the "Timer Supervision" section of this publication. The other routines are described below.

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.

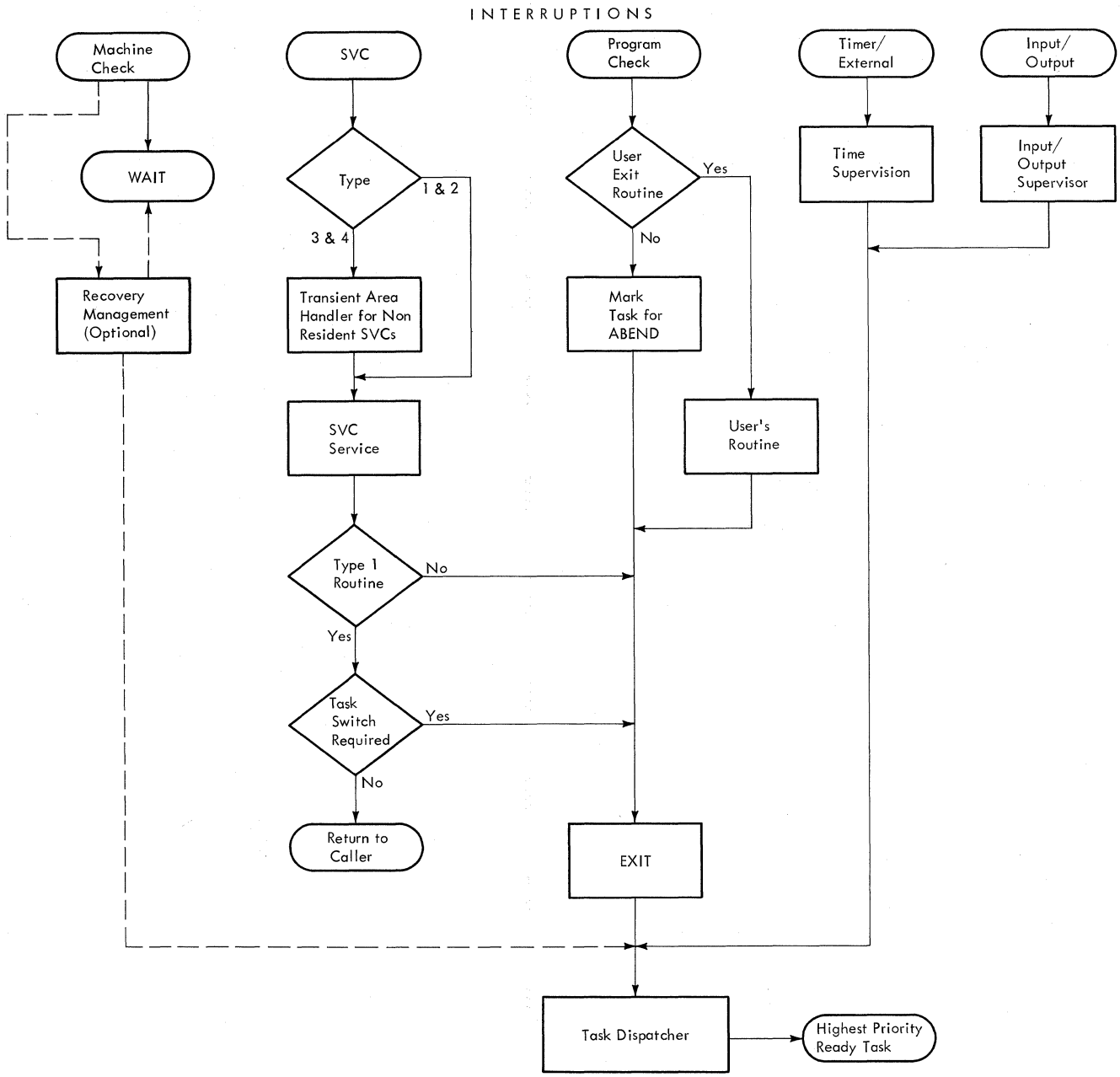



Figure 6. MFT Supervisor

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 TCBs for the system tasks have the highest priority and are arranged in the following dispatching sequence. When optional system tasks are selected, their TCBs always appear in the order indicated.

1. The transient area loading task TCB.
2. The system error TCB, when the subtasking option is specified.
3. The optional LOG task TCB.
4. The optional dynamic device reconfiguration task TCB.
5. The communication task TCB.
6. The master scheduler task TCB.
7. The optional system management facility task TCB.

These system TCBs are followed by TCBs for each task in the dynamic area arranged in descending order based on the value in dispatching priority field TCBDSP. (If the system does not have the subtasking option, the dispatching priority values are always

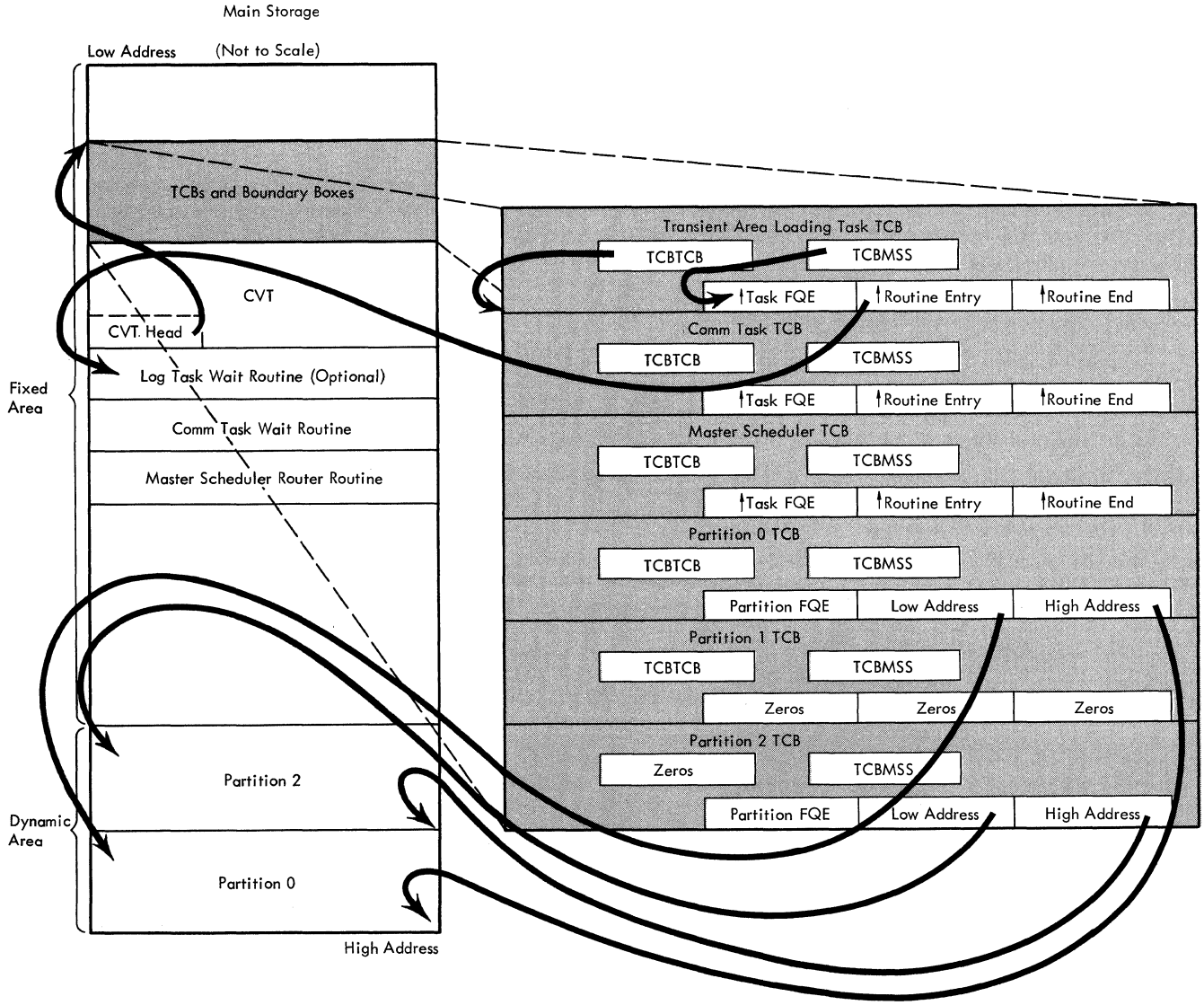
ordered by partition number, P0 through P51.) Figure 7 illustrates a TCB queue in a system without any options.

Note: If MFT with subtasking is included in the system, the SIRB for the error recovery procedures is queued to the system error TCB, not the TCB for the failing task.

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 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.




• Figure 7. TCB Queue

NEW can be set by any of the supervisory routines associated with task switching:

- WAIT, POST, ENQ/DEQ, and Manual Purge and in addition;
- For systems with subtasking, ATTACH, CHAP, LOAD, EXIT and Stage 3 Exit Effector.

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.

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 01). 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. The contents of registers 2 through 9 and the floating point registers are stored in OLD's TCB. If job-step CPU timing is included in the system, and OLD'S TCB has an address in its TCBPIB field, (indicating that OLD is not a system task), the dispatcher enques the job-step timer queue element. If necessary, the dispatcher enqueues 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 of the task to be given control. The dispatcher sets OLD equal to NEW, goes to the trace routine (if present) to trace the save areas, and restores the floating point registers (if present). If job-step CPU timing is included in the system, the dispatcher determines if the TCB to receive control has a PIB pointer in its TCBPIB field. If there is a pointer, the task is not a system task, and the dispatcher enqueues the job-step timer queue element. The dispatcher then branches to a transient area refresh subroutine. Upon return, the dispatcher branches to the task pointed to by NEW.

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). Control then passes to the new task.

The dispatcher does not pass control directly to the new task when the TCBRBP field for the task to be given control addresses an SVRB for a loaded transient SVC routine which is no longer present in the SVC transient area. In this case, the SVC routine was overlaid (before its execution completed) by another SVC routine. The dispatcher determines if the SVC routine currently in the transient area is the one required for the new task by comparing the contents of the doubleword XSNTCC with the XRENM field of the SVRB. (XSNTCC is located in the SVC Second Level Interruption Handler-IEAATA00, and contains the name of the SVC routine currently in the transient area.) If the fields are identical, the SVC routine currently in the transient area is the one required for the new task, and the dispatcher passes control to the new task. If they are not identical, the dispatcher prepares to reload the SVC transient area with the SVC routine indicated in the XRBNM field. It issues a branch and link to the SVC SLIH for creation and initialization of a new SVRB. This SVRB represents the refresh request for the SVC routine that was overlaid before its execution completed. The dispatcher then branches to the FINCH routine requesting the loading of the SVC transient area with the desired SVC routine. (See the discussion of the transient area loading task in the "Contents Supervision" section of this publication for an explanation of the processing required to bring the appropriate SVC routine into the SVC transient area.)

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 iden-

tify 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.

If the system management facility is included in the system, the Dispatcher records the beginning of a system wait before loading the RB old PSW. It reads the interval timer and stores its value in the first word of a special save area, SYSWSAVE. This value is later used by the SMF wait time collection routine to calculate elapsed system wait time.

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).

Handling Job Step Timing When a Task Switch is to Occur

Job step timing is requested by a job step's Initiator via a STIMER macro instruction that specifies the TASK operand. The Dispatcher handles job step timing if two conditions are met:

- A task switch is needed.
- The job step timing option was specified at system generation.

If these conditions are met, the Dispatcher suspends timing of the job-step task that is giving up control by branching to the timer dequeue routine (IEAQTD01) to dequeue the "old" task's TQE. It then restarts timing of the job step whose task is next to be dispatched by branching to the timer enqueue routine (IEAQTE00) to add the "new" task's TQE to the timer queue.

REMOVING FROM THE TIMER QUEUE THE TQE FOR THE JOB STEP ASSOCIATED WITH THE TASK THAT IS GIVING UP CONTROL: The Dispatcher performs two tests to determine if job step timing is being performed for the job step associated with the task that is giving up control:

- It tests the TCBPIB field of the TCB for the address of a PIB. If this field contains zeros, a system task is giving up control and therefore has no job step timing associated with it.
- If the TCBPIB field contains a PIB address, the Dispatcher tests the high-order bit in the job step timing status bits field of the PIB. If this bit is off, an initiator, a reader, or a writer is executing in the partition and also has no job step timing associated with it.

If the bit is on, the Initiator has requested job step timing for a problem program and a job step TQE is on the timer queue. The Dispatcher therefore examines the TQEFLGS field to determine the TQE type. If the TQE is a REAL type, it will not be removed from the timer queue because it represents a wait time limit interval. It was established as a wait time limit TQE and enqueued by the WAIT routine (IEAAWT) and it will be dequeued by the POST routine (IEAAPT).

If the TQE is a TASK type, the Dispatcher branches to the timer dequeue routine (IEAQTD01) in the timer second level interruption handler, to suspend job step timing for the "old" task.

PLACING ON THE TIMER QUEUE THE JOB STEP TQE FOR THE JOB STEP ASSOCIATED WITH THE TASK TO BE DISPATCHED: The Dispatcher performs two tests to determine if job step timing is being performed for the job step associated with the task that is about to be dispatched:

- It tests the TCBPIB field of the TCB for the address of a PIB. If this field contains zeros, a system task is receiving control and therefore has no job step timing associated with it.
- If the TCBPIB field contains a PIB address, the Dispatcher tests the high-order bit in the job step timing status bits field of the PIB. If this bit is off, a reader, a writer, or an initiator is receiving control and also has no job step timing associated with it.

If the bit is on, job step timing has been requested for the problem program in the partition via a STIMER macro instruction issued by the Initiator. The Dispatcher therefore branches to the timer enqueue routine (IEAQTE00) in the timer second level interruption handler to add the TQE to the timer queue and restart job step timing for the task that is about to be dispatched.

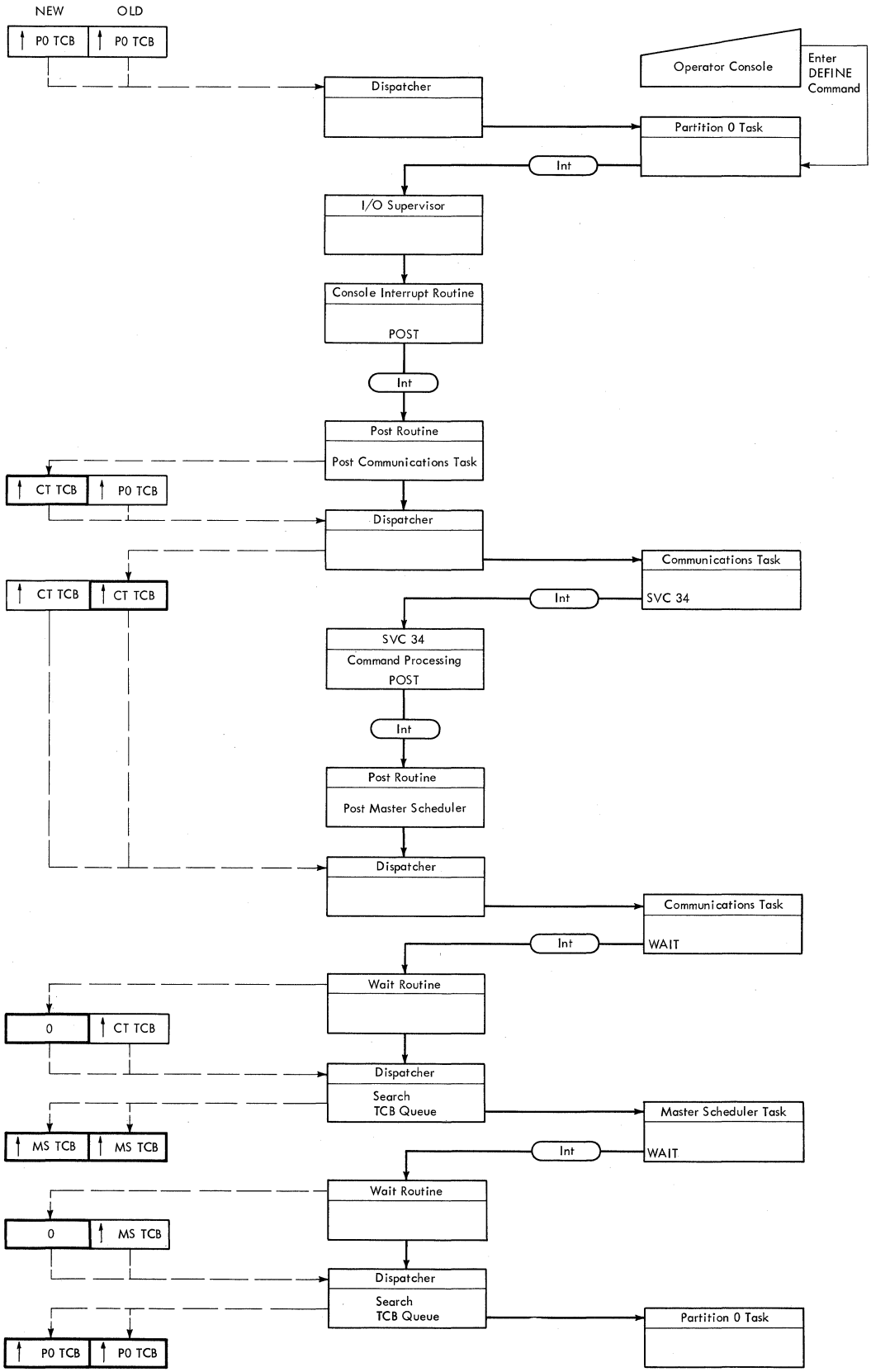



Figure 8. Dispatching Communications and Master Scheduler Tasks

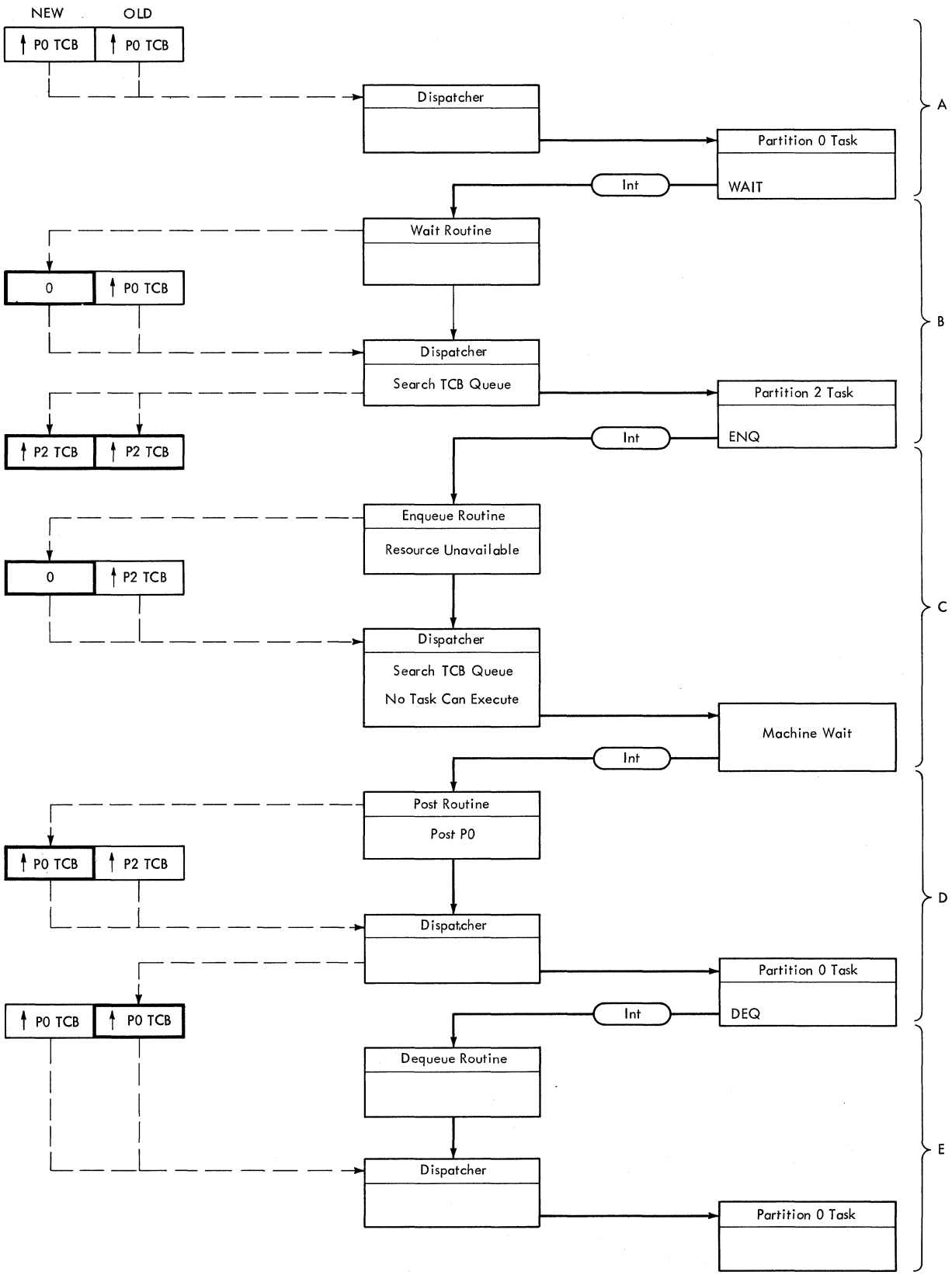



Figure 9. Task Switching

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 passes control to resident device-support routines or issues SVC 72 for transient device-support routines. The device-support routines read the console's command and then issue 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 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 the time-slicing option is selected at system generation in an MFT system without subtasking, one or more contiguous partitions will contain tasks that share CPU time. The number of time-slicing partitions, and the maximum amount of time each task will have, are determined by the time-slice parameters set at system generation. Both may be modified by the DEFINE command.

When a TCB of a partition within the time-slice partitions is the highest ready TCB, the dispatcher will share the CPU time among all the ready TCBs within the time-slice partitions. These TCBs will share the CPU until none of the TCBs are dispatchable or a higher priority task is ready, at which time the dispatcher will no longer be limited to the time-slicing tasks.

The dispatcher determines whether the tasks are to share the CPU by testing the time-slice control element (TSCE). The TCB of the highest priority partition within the time-slice group is addressed by the field "FIRST"; the lowest priority partition within the group is addressed by the field "LAST". The time-slice dispatching queue is that portion of the dispatching queue starting with the FIRST TCB and ending with the LAST TCB. The dispatcher stores the address of the TCB to next receive control in the field "NEXT", and allows each task to run for the interval given in the field "LENGTH".

| | |
|---|---|
| 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 |

If time-slicing is selected in a system with subtasking, two additional fields are added to the TSCE as shown below. In addition to the address of the FIRST and LAST TCBs, the dispatching priorities given to the time-slicing group are included. The value for first is the limit priority of the job-step TCB in the highest priority partition in the time-slice group; the value for LAST is one more than the limit priority of the partition below the LAST partition. Any TCB whose dispatching priority falls within the range of values from FIRST to LAST will be dispatched as a time-slice task.

| | |
|---|---|
| Highest Dispatching Priority | FIRST - Address of the first time-slice TCB |
| Lowest Dispatching Priority | LAST - Address of the last time-slice TCB |
| NEXT - Address of the next time-slice TCB | |
| LENGTH - Time-slice length (in milliseconds) | |

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 02.

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.

### Dispatching the 7094 Emulator Program for the Model 85

If the 7094 Emulator option is specified at system generation for a Model 85, the dispatcher uses the Diagnose instruction to enter or leave the emulator mode. Each time the dispatcher receives control, it checks bit 3 of the TCBTRN field of the TCBs addressed by NEW and/or OLD to determine if either TCB is that of the 7094 emulator program. If it is, the dispatcher issues a Diagnose instruction to enter or leave the emulator mode.

## SVC SECOND LEVEL INTERRUPTION HANDLER

The SVC second level interruption handler (SVC SLIH) differs from that described in the PCP Supervisor PLM in the following manner:

If main storage space is not available for the construction of an SVRB for the abnormal termination routines, the SVC SLIH determines from the TCB if the task requesting abnormal termination is a job-step task. If the requesting task is a job-step task and if the system log option is included, the SVC SLIH determines whether the job-step task is the optional system log task. If the job-step task is the system log task, the SVC SLIH branches to the system log routine.

If the job-step task is not the system log task, the SVC SLIH increments the wait count (if the wait count is less than 255); turns on the wait flags in the TCBWAT field; sets the DAR bytes in the TCBDAR field; enables the resume PSW for interruptions; and indicates that a task switch is needed. The SVC SLIH than branches to the dispatcher.

If the system has subtasking, and the requesting task is a subtask, the SVC SLIH schedules the abnormal termination of the job-step task unless the job-step task is already terminating, and indicates that a task switch is needed. The SVC SLIH then branches to the dispatcher.

## EXIT (MACRO IEAATA)

In MFT without subtasking, the Exit routine operates as described in the PCP Supervisor PLM. In MFT with subtasking, the Exit routine differs from that described in the PCP Supervisor PLM in that an end-of-task routine is included and special handling of subtask exits is provided. The differences are described below:

- If the exiting program's RB is the only RB on the RB queue and subtask end is not indicated in the TCB, the Exit routine determines whether the TCB has any active subtasks. If there is an active subtask, the Exit routine schedules the exiting program for abnormal termination.

- If the exiting program is a subtask, the Exit routine determines whether there are any outstanding enqueue requests (that is, requests for which there are no corresponding dequeue requests). If there are outstanding requests, the Exit routine schedules the subtask for abnormal termination.

The end-of-task (EOT) routine is included as a subroutine of the exit routine in MFT with subtasking. The EOT routine is entered only if the exiting program's RB is the only RB on a subtask's RB queue and the subtask's TCB is flagged for subtask end. The EOT routine functions as follows:

- The EOT routine moves the contents of the task's registers to the TCB from the SVC save area and zeros the RB queue pointer in the TCB.
- If the ETXR (End-of-Task Exit Routine) operand was specified in the ATTACH macro instruction, the EOT routine branches to the Stage 2 Exit Effector to schedule the interruption queue element (IQE).
- The EOT routine removes the subtask's TCB from the dispatching queue and changes the address in location OLD to the address of the TCB immediately preceding the subtask TCB on the dispatching queue. The EOT routine then tests the address in location NEW to determine whether it points to the exiting subtask's TCB. If it does, the EOT routine changes the address in location NEW to the address of the TCB immediately preceding the subtask TCB.
- The EOT routine removes the subtask address from the TSCE if it is first, next, or last and updates the TSCE to reflect the removal of the subtask.
- If an ECB was specified in the ATTACH macro instruction, the EOT routine branches to the Post routine to post the ECB.
- The EOT routines dequeues the IQEs for job-step timing and time-slicing if either or both are enqueued.
- The EOT routine loads the registers with the contents stored in OLD's TCB and branches to the dispatcher.

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 the System Programmer's Guide SRL for further explanation of the STAE macro instruction.

The five routines that perform the functions of the STAE macro instruction are the STAE routine (IEAAST00) and the four ABEND/STAE interface routines (IEASTM11, IEASTM12, IEASTM13, and IEASTM14). These routines perform the same functions as the STAE routines described in the MVT Supervisor PLM with the exception that both IEASTM12 and IEASTM14 pass control to the ABEND routine (IEANTM0B) to purge WTOR requests before passing control to the next STAE routine (IEASTM13). If MCS is included in the system, STAE uses IEACTM0B to purge WTOR requests.

ABEND SERVICE ROUTINE

ABEND is a type 4 SVC routine that performs both normal and abnormal task termination by freeing the resources and control blocks associated with the terminating task. The freed resources thus become available for use by other tasks in the system. The control blocks affected are:

- Task Control Blocks (TCBs).
- Data Extent Blocks (DEBs).
- Free Queue Elements (FQEs).
- Request Blocks (RBs).
- Interrupt Queue Elements (IQEs).
- Timer Queue Elements (TQEs).
- Gotten Area Subtask Queue Elements (GQEs).
- Program Interrupt Elements (PIEs).

ABEND can be entered directly from a problem program or system task via an ABEND macro instruction, or indirectly through the ABTERM service routine. (In cases where a system routine detects an error but cannot issue an ABEND macro instruction, ABTERM schedules the execution of ABEND.) The SVC second level interruption handler (SLIH) causes the first load module of ABEND (IEANTM00) to be brought into the SVC transient area and passes control to it. Control is passed between ABEND modules via an XCTL macro instruction (SVC 7).

To return control to the job management step deletion routines IEFSD515 or

IEFSD599, ABEND constructs a dummy PRB and a dummy program at the beginning of the partition. This dummy program contains an LPSW instruction followed by an XCTL macro instruction specifying the name of the job management routine that is to receive control. ABEND adds the PRB to the beginning of the TCB/RB queue and branches to the dummy program.

ABEND processing will vary depending on the type of termination (normal or abnormal), the type of task terminating (job step or subtask), options included in the system, conditions causing the termination, and conditions arising within ABEND during processing (macro failures, etc). Charts 4 and 34 and the following paragraphs describe the normal and abnormal task termination functions of ABEND. For a more detailed description of ABEND processing, refer to the module descriptions.

Normal Termination

ABEND stores the normal completion code of the terminating task in the TCBCMP field in the TCB of the terminating task. If the system has an interval timer, ABEND clears the IQE address in the TQE. If the TQE is in use, it issues a TTIMER macro instruction to cancel the timer. If a job step task is terminating, ABEND closes all data sets whose DCBs and main storage areas are queued to the TCB, purges any WTOR requests, and dequeues all IQEs associated with the task. It frees all main storage within the partition, except that for the dummy program, and uses the created dummy program to branch to the job management routines IEFSD515 or IEFSD599.

If the terminating task is a subtask, ABEND purges the IQEs for the task, closes open data sets, dequeues all GQEs and frees the area they describe, and removes from the active RB queue all request blocks except the SVRB for ABEND. ABEND sets the current task nondispatchable and passes control to the SVC EXIT routine IEAATA.

Abnormal Termination

ABEND first tests for a DAR recursion and, if found, passes control to the DAR service routines. If a subtask is terminating and an invalid recursion has occurred, ABTERM is used to terminate the job step. ABEND then issues a WAIT macro instruction.

If a CLOSE, OPEN, ABDUMP, or message recursion (WTO failure) has occurred, ABEND halts all I/O operations for this task and dequeues all request blocks created as a result of the recursion. Processing continues with the test for dump described below.

If this is a nonrecursive termination, ABEND determines if STAE processing should be initiated or, if purge failed in STAE, resumed. If so, ABEND passes control to the STAE interface routines. If no STAE processing is indicated, ABEND purges IQEs, active TQEs, SPIE requests, STAE requests, and transient area requests for the terminating task and any subtasks. It also purges WTOR requests, halts I/O, and validity checks control blocks.

ABEND determines if a dump is requested and what type of dump will be presented. It ensures that sufficient main storage is available for the dump by issuing a GETMAIN macro instruction. If sufficient main storage is not available, it frees the programs described by the load list to obtain this main storage. ABEND searches the TIOT for a SYSABEND or a SYSUDUMP DDNAME. If neither is found and the terminating task is a job step task, ABEND stores pertinent information in main storage for the eventual printing of an indicative dump by the job management routines. If a SYSABEND entry is found in the TIOT, ABEND opens a DCB for the dump data set if it has not been opened already, and calls ABDUMP to dump the nucleus, the system queue area, and that area within the task's partition that is not free. If a SYSUDUMP entry is found, ABEND opens a DCB for the dump data set if it has not been opened already, and calls ABDUMP to dump only that area within the task's partition that is not free.

Upon completion of the dump, or if no dump was provided, ABEND determines the type of task that is terminating. If the task is a job step task with no subtasks, ABEND closes all data sets associated with the task, dequeues IQEs, and clears the pointers to the ABEND appendages and transient area queues. ABEND passes control to the job management routines to print the indicative dump if one will be provided, and to initiate the next task.

If the terminating task has subtasks, or is itself a subtask, ABEND frees IQEs for the terminating task and any subtasks, chains all subtask data sets to the TCB of the current task, dequeues subtask TCBs from the ready queue and frees the associated main storage. If the terminating task is a job step task with subtasks, ABEND purges resources for its subtasks if any resources are enqueued, and processing continues as for a job step task without subtasks described above. If the terminating task is itself a subtask, ABEND frees main storage for partially loaded programs, minor LPRBs, and GQEs and the area they describe. It closes all data sets queued to the TCB of the terminating task, purges subtask resources, and truncates the TCB/RB queue, leaving only the SVRB for ABEND.

ABEND sets the subtask nondispatchable and passes control to the SVC EXIT routine IEAATA.

Detailed descriptions of the 17 ABEND load modules follow. In the "condition-action" tables, execution within the module is shown sequentially from top to bottom in the table. If the condition slot corresponding to an action is blank, the action is always taken. An asterisk (*) denotes a condition and action that will occur only if MFT with the subtasking option is included in the system.

## ABEND Normal Termination Processing and Abnormal Termination Router Routine (IEANTM00)

IEANTM00 is entered from the SVC SLIH to process normal or abnormal task or subtask terminations. It may also be reentered from IEANTM0B if normal termination required purging of WTOR requests or if MCS is included in the system. The following tables show the possible conditions within the routine and the actions taken.

Normal Termination

| CONDITON | ACTION |
|---|---|
| | Store completion code in TCBCMP |
| Timer included | Set IQE pointer in TQE to zero |
| TQE in use | Issue TTIMER CANCEL |
| | Close DCBs, dequeue DEBs for task |
| MCS in system | Pass control to IEACTM0B |
| WTOR requests to purge | Pass control to IEANTM0B |
| * Subtask terminating | Pass control to IEANTM0E |
| Entry from IEANTM0B | Dequeue IQEs for terminating task |
| * Limit priority does not equal dispatching priority | Issue CHAP macro instruction |
| | Initialize boundary box and FQE for hierarchy 0 (and hierarchy 1 if applicable), clear pointers to TIOT and problem program save area |
| Small partition | Pass control to IEFSD599 |
| | Pass control to IEFSD515 |

Abnormal Termination

| CONDITION | ACTION |
|---|---|
| ABEND in progress | This is a recursion - test for type |
| Primary DAR recursion | Pass control to IEADTM22 |
| Secondary DAR recursion | Pass control to IEADTM23 |
| * Invalid subtask recursion | Use ABTERM to terminate job step, issue a WAIT macro instruction |
| OPEN, CLOSE, ABDUMP, or message recursion | Pass control to IEANTM02 |
| Non recursive termination | Pass control to IEANTM01 |

Before issuing the XCTL macro instruction, IEANTM00 determines if there is sufficient main storage available for the XCTL SVRB. If not, and if the task to be terminated is a subtask, it uses ABTERM to terminate the job step and then issues a WAIT macro instruction. If the task being terminated is a job step, IEANTM00 obtains main storage from the partition for creation of the XCTL SVRB.

## ABEND/STAE Graphics Linkage Routine (IEANTM01)

IEANTM01 processes nonrecursive terminations for job step tasks, for failures when the terminating program has issued a STAE macro instruction, or for failures when GJP is included in the system. The following table shows the possible conditions within the routine and actions taken.

| CONDITION | ACTION |
|---|---|
| Purge failed in STAE | Issue SVC EXIT to pass control to STAE |
| Valid STAE issued, ABEND not caused by operator's cancel, timer expiration, invalid detach, or STAE recursion | Free PIE, indicate ABTERM entered, indicate first ABEND entry from STAE, pass control to IEASTM11 |
| Graphics Job processor, no graphics recursion, user wishes to resume, and user has issued ABEND | Issue SVC EXIT to pass control to user routine. |
| | Indicate task as highest terminating task |
| * Terminating task has subtasks | Purge task and subtasks of IQEs, active TQEs, SPIE requests, transient area requests, and STAE requests, and pass control to IEANTM0B (IEACTM0B for MCS) |
| | Purge task of IQEs, active TQEs, SPIE requests, transient area requests, and STAE requests; stop trace table; pass control to IEANTM0B (IEACTM0B for MCS) |

## ABEND I/O Purge Routine (IEANTM02)

For a nonrecursive ABEND, IEANTM02 purges I/O for the terminating task and any subtasks. For a recursive ABEND, it purges I/O for the terminating task only. For either a recursive or nonrecursive ABEND, if the task is a system task or is in "must complete" status, IEANTM02 passes control to IEADTM22. Otherwise, IEANTM02 passes control to IEANTM03 for a nonrecursive ABEND, or to IEANTM09 for a recursive ABEND.

## ABEND Control Block Validity Check Routine (IEANTM03)

IEANTM03 tests the validity of the control blocks associated with a nonrecursive terminating task and, for MFT with subtasking,

its subtasks. This validity testing ensures system integrity by preventing an abnormal termination within ABEND. If a control block is found to be invalid, the routine truncates the queue on which it resides up to the last valid control block. The table below shows the various validity checks performed on each control block. A bullet (•) indicates that an FQE found to be invalid will be dequeued. Its queue will not be truncated. GQE and JPAQ control blocks exist only if MFT with subtasking is included in the system.

| VALIDITY REQUIREMENTS | FQE | LOAD LIST | DEB | ACTIVE RB | GQE | JPAQ |
|---|---|---|---|---|---|---|
| Alignment on a fullword boundary | | | X | | | |
| Alignment on a doubleword boundary | X | X | | X | X | X |
| Block not in free core | | X | X | X | X | X |
| Maximum of 300 blocks on queue | | X | X | X | X | X |
| Valid forward and backward pointers | | X | | | | X |
| Block size is a multiple of 8 | • | | | | X | |
| Block does not extend beyond partition boundary | • | | | | X | |
| Block does not describe an area described by an FQE | • | | | | X | |
| Block resides in the partition | X | X | X | X | X | X |
| Blocks are queued in descending order | • | | | | | |

## ABEND Dump Test Routine (IEANTM04)

IEANTM04 determines if a dump is required, and ensures that sufficient main storage is available for any type of dump, for an ENQ purge, or for a CLOSE. A dump will not be provided if the system writer is abnormally terminating, if a SYSABEND or SYSUDUMP DDNAME is not found in the TIOT, or if there is insufficient free main storage available for the dump. The following table shows the conditions possible within the routine and the exits taken.

| CONDITION | EXIT TAKEN |
|---|---|
| * Full dump requested | IEANTM05 |
| Full dump requested | IEAMTM05 |
| Indicative dump requested | IEANTM08 |
| Insufficient main storage available for providing dump | IEANTM0A |
| * ABEND is for subtask | IEANTM0C |
| * Subtasks need ENQ purge | IEANTM0D |
| * Subtasks do not need ENQ purge | IEANTM0E |
| Job step termination | IEANTM07 |
| * Job step, no subtasks | IEANTM07 |

## ABEND Open Dump Data Set Routine (IEAMTM05) (MFT without subtasking)

IEAMTM05 opens the dump data set for the terminating task and passes control to IEANTM06. If a SYSABEND or SYSUDUMP DDNAME is not found in the TIOT, or if the data set cannot be opened, IEAMTM05 passes control to IEANTM09.

## ABEND Open Dump Data Set Routine (IEANTM05) (MFT with subtasking)

IEANTM05 opens the dump data set, if it is not already open, for the terminating task. This data set will remain open until termination of the job step task, and will not be re-opened for subsequent ABEND processing. Any subtasks of the terminating task that are enqueued on the dump data set are dequeued via a branch entry to DEQ. (These subtasks are nondispatchable.) IEANTM05 issues an ENQ macro instruction specifying the dump data set to prevent other tasks from using it until the dump is finished. If the data set cannot be opened, or if a SYSABEND or SYSUDUMP entry is not found in the TIOT, IEANTM05 passes control to IEANTM09. If any task in the job step is in the process of opening the dump data set, the routine sets the current task nondispatchable, and sets its resume PSW to re-enter IEANTM05. It then passes control to the dispatcher. If the OPEN is successful, IEANTM05 resets dispatchable all tasks in the job step that were set nondispatchable waiting for the dump data set to open. It then passes control to IEANTM06.

## ABEND Dump Routine (IEANTM06)

IEANTM06 prints appropriate messages pertinent to the dump and issues a SNAP macro instruction (SVC 51) to call ABDUMP for the terminating task and, for MFT with subtasking, any subtasks. The following table shows the conditions within the routine and the exits taken.

| CONDITION | EXIT TAKEN |
|---|---|
| * Job step task has no subtasks | IEANTM07 |
| SNAP macro instruction failed | IEANTM09 |
| * Subtask is terminating | IEANTM0C |
| * Subtask needs ENQ purge | IEANTM0D |
| * Subtasks do not need ENQ purge | IEANTM0E |
| Job step termination | IEANTM07 |

## ABEND Termination Routine (IEANTM07)

IEANTM07 closes all data set chained to the TCB of the terminating task, dequeues IQEs,

and clears the pointers to the ABEND appendages and transient area queues. If an indicative dump has been created, the ABEND termination routine moves it to the upper part of the partition. If MFT with subtasking is included in the system, this routine is entered only if a job step task is terminating and its subtask's TCB has been freed. IEANTM07 passes control through the dummy program to IEFSD515 (GO) for a large partition, or to IEFSD599 (SMALLGO) for a small partition.

## ABEND Indicative Dump Routine (IEANTM08)

IEANTM08 creates an indicative dump area containing the following information:

- Register contents at entry to ABEND.
- Floating point register contents, if any.
- Completion code.
- Program check instruction.
- TCB flags.
- Program name.
- Entry point.
- Resume PSW.
- Active RBs.
- Loaded RBs.

IEANTM08 passes control to IEANTM07 for job step termination. If subtasking is included in the system, IEANTM08 passes control to IEANTM0D if subtasks need ENQ purge, or to IEANTM0E if subtasks do not need ENQ purge.

## ABEND Recursion Processing Routine (IEANTM09)

IEANTM09 dequeues all request blocks created as a result of the recursion and processes all OPEN, CLOSE, ABDUMP, and message recursions. The following table shows the conditions possible within the routines and the actions taken.

| CONDITION | ACTION TAKEN |
|---|---|
| Message recursion | Pass control to IEANTM04 |
| * OPEN recursion | Reestablish JPAQ Chain, reset dispatchable any tasks waiting for open of the dump data set |
| * Subtask ABEND | Reestablish GQE chain |
| OPEN recursion | Set switch for indicative dump, reset dispatchable any tasks waiting for open of the dump data set |
| CLOSE recursion | Indicate no dump |
| * ABDUMP recursion | Set switch for indicative dump, dequeue task from dump data set |
| ABDUMP recursion | Set switch for indicative dump |
| | Issue WTO macro instruction to print appropriate message and pass control to IEANTM04 |

## ABEND Steal Main Storage Routine (IEANTM0A)

IEANTM0A frees main storage for all LRBs, LPRBs for the job step task, and, for MFT with subtasking, all LPRBs for the subtasks queued on the load list. The following table shows further conditions possible within the routine and actions taken.

| CONDITION | ACTION |
|---|---|
| Dump requested | Test for sufficient main storage |
| Sufficient main storage available now | Pass control to IEANTM04 |
| Insufficient main storage available for providing dump | Turn off switch indicating dump, issue WTO macro instruction for "insufficient storage available" message |
| Dump cannot be provided or is not requested | Test for freeing main storage |
| * ABEND for job step task | Free main storage for LRBs on JPAQ chain |
| * SVRB for ABEND is in PRB area | Move SVRB to top of partition |
| * FQE is in PRB area | Dequeue FQE |
| | Free main storage for all PRBs for terminating task |
| * Terminating task has subtasks | Free main storage for all PRBs for subtasks |
| | Pass control to IEANTM04 |

## ABEND WTOR Purge Routine (IEANTM0B) (MFT Without MCS)

IEANTM0B purges outstanding WTOR requests for the terminating task and, for MFT with subtasking, its subtasks. IEANTM0B passes control to IEANTM00 for normal job step termination, to IEANTM02 for abnormal termination, to IEANTM0E for normal subtask termination (MFT with subtasking only), or to IEASTM13 if STAE requested its processing

## ABEND WTCR Purge Routine (IEACTM0B) (MFT With MCS)

IEACTM0B has the same functions and exits as IEANTM0B (above).

## ABEND Loading Program Purge Routine (IEANTM0C) (MFT With Subtasking Only)

IEANTM0C is entered only when the terminating task is a subtask. It frees partially loaded programs and FRBs for the terminating task and its subtasks and passes control to IEANTM0D.

ABEND Subtask ENQ Purge Routine (IEANTM0D) (MFT With Subtasking Only )

IEANTM0D performs an ENQ purge for all subtasks of the terminating task and for the terminating task itself if it is a subtask. Resources are therefore freed (e.g., data sets, devices, etc.) for use by other tasks. Outstanding ENQs for the job step task will be purged by the job management routines IEFSD597 or IEFSD598 at step termination. IEANTM0D passes control to IEANTM0E.

ABEND IQE Purge and Data Set Close Routine (IEANTM0E) (MFT With Subtasking Only)

IEANTM0E purges IQEs for the terminating task and any subtasks. If the task is terminating abnormally, IEANTM0E chains the data sets for all of its subtasks to the TCB for the terminating task. It also frees main storage for its subtask's TCBs, minor LPRBs, and GQEs. If the terminating task is a job step task, IEANTM0E passes control to IEANTM07. If the terminating task is a subtask terminating normally or abnormally, IEANTM0E closes all open data sets for the terminating task, dequeues all request blocks except the SVRB for ABEND from the active RB chain, and frees main storage for all GQEs and their associated areas. IEANTM0E sets the terminating task nondispatchable and passes control to the SVC EXIT routine (IEAATA).

DAMAGE ASSESSMENT ROUTINES

The damage assessment routines (DAR) process, and attempt to recover from the following failures:

- System tasks (log, communication, or master scheduler).
- Tasks in "must complete" status.
- Tasks experiencing invalid ABEND recursion.

A record of the failures is provided in a main storage dump. A primary DAR recursion results when a failure occurs while writing the main storage dump. A secondary DAR recursion results when a failure occurs during partition recovery. The damage assessment routines also advise the operator of the failure and subsequent reinstatement of the task.

DAR Core Image Dump Routine (IEADTM22)

The DAR core image dump routine IEADTM22 writes on the SYS1.DUMP data set an image of main storage at the time of failure. When entered, the routine sets all tasks except the failing task and the communications task nondispatchable. If MFT with subtasking is included in the system, any subtasks of the terminating task are indicated for ABEND processing. The routine then writes the image of main storage and passes control to the DAR task reinstatement routine IEADTM23.

If the SYS1.DUMP data set has not been allocated, the routine informs the operator via a WTO macro instruction. If the routine is entered as a result of a primary DAR recursion, which is caused by a failure to write the image of main storage, the routine does not try to rewrite but informs the operator of the failure via a WTO. In both cases the routine passes control to IEADTM23.

If the communications task is the failing task, messages are queued pending reinstatement of the communications task by IEADTM23.

DAR Task Reinstatement Routine (IEADTM23)

If the DAR task reinstatement routine IEADTM23 is entered as a result of a failing system task, the routine attempts to reinstate the task. It points the resume PSWs of all but the highest level RB of the task's TCB to an SVC 3 instruction in the CVT. It points the highest level RB to entry point IEECIR50 for the Master Scheduler task, entry point IEECIR45 for the Communications task, or entry point IEEVLIN for the Log task. The routine then passes control to the dispatcher via a branch instruction. If the system error task is failing, ABTERM is used to terminate the requesting task. The resume PSWs of all RBs except the two highest level RBs (the SIRB and the wait RB) are pointed to the SVC 3 instruction in the CVT. IEADTM23 points the resume PSW in the SIRB to the entry point IEAMSERB for the system error task and exits to the dispatcher.

If the routine is entered as a result of a secondary DAR recursion, which is caused by a failure to reinstate the failing task, the routine informs the operator via a WTO, sets all tasks dispatchable except the failing task, and passes control to the dispatcher via a branch instruction.

If the failing task is in "Must Complete" status, the task reinstatement routine issues a message to the operator listing the major and minor names of the enqueued resources that have caused the "Must Complete" condition and asking the operator to reply whether the resources are critical. If the reply indicates the resources are critical, processing is identical to the processing of a secondary DAR recursion described above. If the reply indicates the resources are not critical, the "Must Complete" status is removed, and the resources are designated as shareable.

The task is processed as a failing non-system task as described below.

If the routine is entered as a result of a failing non-system task, it sets indicators showing that a dump has been taken by DAR and issues a message to the operator indicating that the system has been reinstated. The routine then sets all tasks, except the subtasks, if any, of the terminating task, dispatchable and passes control to IEANTM07 if the terminating task is a job step task with no subtasks, to IEANTM0C if the terminating task is itself a subtask, to IEANTM0D if the terminating task has subtasks that need ENQ purge, or to IEANTM0E if the terminating task has subtasks that do not need ENQ purge.

## 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 (ECB).
- Maintain control levels indicated by request blocks.

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

In addition, if subtasking is included in the system, task supervision service routines:

- Attach subtasks when requested by the user.
- Detach subtasks previously created by the requester.
- Change the dispatching priority of a requesting task or of a task under the control of the requester.

The routines that accomplish these functions are ATTACH, DETACH, and CHAP.

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 in MFT without subtasking is similar to that described in the PCP Supervisor PLM. Task supervision in MFT systems with subtasking is similar to task supervision in an MVT system and is described in the MVT Supervisor PLM. Additional information applicable to MFT is presented in the following paragraphs.

### THE ATTACH ROUTINE (MFT WITHOUT SUBTASKING) (MACRO IEAAAT)

In MFT without subtasking, 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 the Supervisor and Data Management Services SRL, GC28-6646 for further explanation of the ATTACH macro instruction with MFT.

### THE ATTACH ROUTINE (MFT WITH SUBTASKING) (MACRO IEAQAT00)

A task issues an ATTACH macro instruction to cause the supervisor to schedule execution of a requested program as a subtask of the calling task. A subtask competes for CPU time and for resources allocated to the calling task. When the ATTACH macro instruction is issued, the Attach routine gains control and performs the following main functions:

- Obtains storage space in the system queue area for a new task control block (TCB).
- Places in the new TCB information needed for controlling the subtask (refer to MVT Supervisor PLM).
- Allocates to the subtask the main storage belonging to the calling task.
- Places the address of the new TCB on two TCB queues:
  The subtask queue of the calling task and the dispatching queue (according to dispatching priority).
- Places a dummy request block on the TCB's active request block queue and schedules linkage to contents supervision to locate the first program to be executed for the new subtask, fetch the program if necessary, and schedule its execution.

In addition to performing the main functions, the Attach routine also performs the following minor functions: If the ATTACH macro instruction contains the ETXR operands, storage space is obtained for and

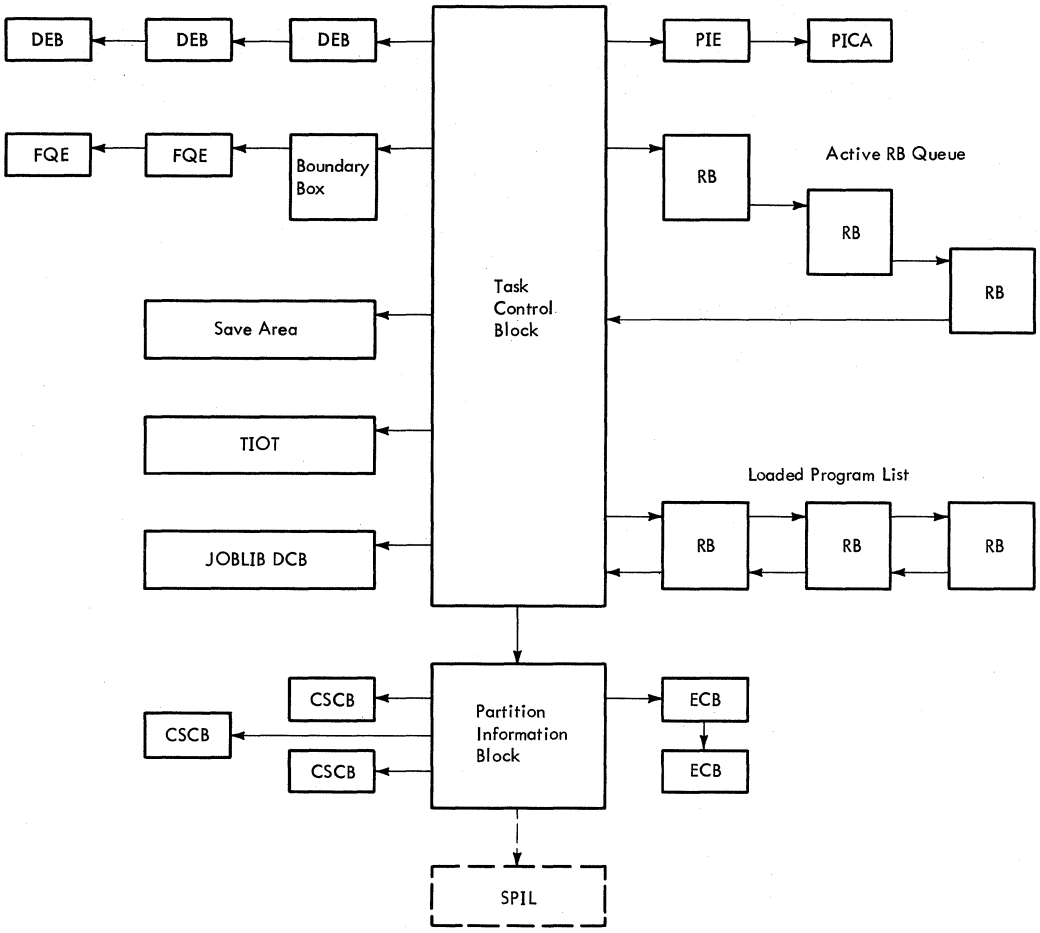



Figure 10. System Control Block Relationship

control information is placed in, either or both the interruption queue element (IQE) and the interruption request block (IRB) to be used for scheduling and controlling an end-of-task exit routine. (If an IQE or IRB already exists for the specified exit routine, the Attach routine does not build another one.)

The TCB that is created will be initialized by the Attach routine to contain status information and list origins for queues needed by programs being executed for the subtask. For example, this information includes the address of the Attach event control block (ECB), if specified, and the limit and dispatching priorities of the new TCB. The Attach routine also determines whether the calling routine or the newly created routine receives control by comparing the dispatching priority of the requesting task with that of the newly created subtask. The Attach routine places the address of the higher priority TCB in the "new" TCB pointer (IEATCBP) to be later tested by the dispatcher when it receives control during the exiting procedure.

## THE CHAP ROUTINE (MFT WITH SUBTASKING)

The CHAP (Change Priority) macro instruction allows a problem program to change its own dispatching priority, or that of any subtask created by the problem program. A CHAP macro instruction can change the dispatching priority of a task to any value from zero through the limit priority of the issuer. The CHAP macro instruction in MFT with subtasking is the same as that described in the MVT Supervisor PLM. A complete discussion of task priorities can be found in the Supervisor and Data Management Services SRL, GC28-6646.

## THE DETACH ROUTINE (MFT WITH SUBTASKING)

The operation of the Detach routine in MFT with subtasking is the same as the operation of the Detach routine as described in the MVT Supervisor PLM, with the following exceptions:

- In MFT with subtasking, validity checking is performed by the Detach routine. The Detach routine checks for boundary

alignment and determines whether the TCB address passed by the caller represents a valid subtask.

- The subtask's save area, freed in MFT with subtasking by the Detach routine, is not located in a subpool. The TCB, floating point register save area, and the timer queue element area (if present in storage) are freed from the system queue area.
- After freeing the subtask's main storage, the Detach routine decrements by one the subtask count stored in the PIB, and zeros the TCB table address entry that pointed to the detached subtask.
- The Detach routine does not validity check the user's ECB. However, it does issue a POST SVC instruction to post the ECB.

## THE EXTRACT ROUTINE (MFT WITH SUBTASKING)

The operation of the extract routine in MFT with subtasking is the same as that described in the MVT Supervisor PLM except that it is a type 3 SVC routine and returns control to the SVC second level interruption handler.

## THE EXTRACT ROUTINE (MFT WITHOUT SUBTASKING)

The operation of the Extract routine in MFT without subtasking is the same as that described in the PCP Supervisor PLM.

## THE WAIT ROUTINE (MACRO IEAAWT)

The MFT configuration of the operating system uses the WAIT routine described in the PCP Supervisor PLM, with the addition of imposing job step wait time limiting if the job step timing option was specified at system generation.

If the job step timing option was specified at system generation, the WAIT routine imposes a wait time limit on a job step that is being job step timed. This action is taken so that a job step will not wait on an ECB that, for some reason, is never posted. The POST routine removes an unexpired wait time limit imposition when the job step's wait is satisfied via a POST macro instruction or a branch entry to POST. If the wait time limit established in the TQE by the WAIT routine expires, the timer second level interruption handler abnormally terminates the task unless SMF is in the system and SMF user time limit expiration routine IEFUTL is specified.

The WAIT routine implements wait time limiting by converting a TASK type TQE (the job step TQE associated with job step timing) to a REAL type TQE containing a specified wait time limit.

After the WAIT routine places the wait count in the caller's RB, it performs two tests to determine if job step timing is in effect for the task under which the WAIT macro instruction was issued:

- It tests the TCBPIB field of the TCB for the address of a PIB. If this field contains zeros, the task is a system task and therefore has no job step timing associated with it. In this case the WAIT routine bypasses the processing for job step wait time limiting.
- If the TCBPIB field contains a PIB address, the WAIT routine tests the high-order bit in the job step timing status bits field of the PIB. If this bit is off, an initiator, a reader, or a writer is executing in the partition and also has no job step timing associated with it. Processing for job step wait time limiting is therefore bypassed.

If the bit is on, the Initiator has requested job step timing for a problem program via the STIMER macro instruction and a job step TQE is on the timer queue. The WAIT routine examines the TQEFLGS field to determine the TQE type. If the TQE is a TASK type, the WAIT routine branches to the timer dequeue routine (IEAQTD01) in the timer second level interruption handler to remove the TQE from the timer queue. It saves the CPU job step time remaining in the TQESAV field. It then stores a 30-minute wait time limit value in the TQEVAL field. (If SMF is supported, the WAIT routine obtains the wait time limit value from the TCTWLMT field of the timing control table (TCT), and stores this value in the TQEVAL field.) The WAIT routine then converts the job step TQE from a TASK type to a wait time limit TQE (REAL type). It branches to the timer enqueue routine (IEAQTE00) in the timer second level interruption handler to place the REAL type TQE on the timer queue, and then continues with normal WAIT processing.

If the TQE is a REAL type, processing for job step wait time limiting is bypassed.

Note: If the optional validity checking feature 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 the PCP Supervisor PLM, except for the addition of processing for job step wait time limiting, if the job step timing option was specified at system generation.

If the job step timing option was specified at system generation, the POST routine performs the additional function of restoring a REAL type TQE (established as a wait time limit TQE by the WAIT routine) that is on the timer queue, to a TASK type TQE. This action is taken so that the Dispatcher is able to continue job step timing for the task the next time that it is dispatched.

When the wait count in the caller's RB becomes zero, the POST routine performs two tests to determine if job step timing is in effect for the task whose top RB wait count has become zero. It bypasses the restoring processing if job step timing is not associated with the task.

- It tests the TCBPIB field of the TCB for the address of a PIB. If this field contains zeros, the task is a system task and therefore has no job step timing associated with it.

- If the TCBPIB field contains a PIB address, the POST routine tests the high-order bit in the job step timing status bits field of the PIB. If this bit is off, an initiator, a reader, or a writer is executing in the partition and also has no job step timing associated with it. If the bit is on, job step timing is in effect for a problem program, and a job step TQE is on the timer queue. The POST routine examines the TQEFLGS field to determine the TQE type. If the TQE is a TASK type, restoring processing is also bypassed.

If the TQE is a REAL type, it was established as a wait time limit TQE by the WAIT routine. In this case the POST routine must restore the TQE to a TASK type. It branches to the timer dequeue routine (IEAQTD01) in the timer second level interruption handler to remove the REAL type TQE from the timer queue. It then restores the TQEVAL field with the actual CPU job step time remaining. (The WAIT routine originally saved this value in the TQESAV field.) The POST routine then sets the TQEFLGS field to mark the TQE as a TASK type to allow the Dispatcher to once again begin job step timing for the task the next time it is dispatched. It then continues with normal POST processing.

Note: Validity checking applies to POST in the same way that 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 the MVT Supervisor PLM. ENQ/DEQ for MFT is identical to MVT except as described below.

In MFT, resource queues are located in the system queue area and are obtained via GETMAIN macro instructions specifying subpool 255. The address of the first queue control block in the queue can be found at location IEAOQCBO in the ENQ/DEQ routine.

In a system without the subtasking option, the "must complete" function of ENQ/DEQ applies only to system tasks, not to job steps. If "must complete" is specified for a system task, all other tasks are set non-dispatchable until the task completes its processing.

In a system with the subtasking option, the "step must complete" function may be specified for a non-system task as well as a system task. If "step must complete" is specified for a non-system task, all other tasks in the job step are set non-dispatchable until the specified task completes its processing.

## Contents Supervision

Contents supervision routines determine the type and location of requested modules and bring them into main storage within the requester's partition if necessary. Contents supervision also maintains records of all modules in main storage within each partition. The first part of this section

describes the records and routines used by contents supervision in an MFT system without subtasking. The second part of this section describes the additional records used by contents supervision and the routines that are changed in an MFT system with subtasking.

CONTENTS SUPERVISION IN AN MFT SYSTEM WITHOUT SUBTASKING

There are six types of request blocks in MFT systems without subtasking:

- 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 modules brought in by a LOAD macro instruction. LPRBs also control section of modules 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 because they may not follow the linkage conventions for the macro instructions. LRBs are created by the routines that perform the LOAD function.

There are two types of records of the modules that are in the partition's main storage or have been requested to be brought into the partition's main storage. The first is the requesting task's active request block queue. This queue is the chain of request blocks associated with the load modules and SVC routines being used by the task and pointed to by the TCB/RBP. The queue can contain any type of request block as long as it is in use by the requesting task. When the modules represented by the RB has completed exection, contents supervision will remove the RB from the queue.

The second record of modules in the partition's main storage is the loaded program list pointed to by TCBLLS. This list consists of LPRBs and LRBs (that is, modules requested by a LOAD macro instruction). These modules are assumed to be reusable and as such can be shared by other routines within the partition. The request blocks include a use count which indicates the number of tasks that have requested the module via a LOAD macro instruction and have not indicated completion via a DELETE macro instruction. A module's RB will not be deleted from the loaded program list until its request count is zero.

Contents supervision routines alter the active RB queue and the loaded program list, and bring nonresident programs into the problem program partitions in response to LINK, ATTACH, LOAD, and XCTL macro instructions. Additional contents supervision services are provided by the use of IDENTIFY, DELETE, and SYNCH macro instructions. IDENTIFY and DELETE alter the loaded program list. SYNCH alters the active request block queue. The routines that service these macro instructions are described below.

LINK Service Routine (Macro IEAATC)

The LINK service routine determines if the RB of the requested module 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), and the resident reenterable module option was selected at system generation, the routine searches the resident area. If the module is found in the area, a dummy LPRB for the module is placed on the loaded program list, and processing continues as if the module were originally found on the load list. If the module is not found, the LINK routine constructs an RB for the requested routine, places the RB on the active RB queue, and fetches the requested module 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 module is in the resident reenterable module area (if the resident module option was specified at system generation). If so, the entry point of the module is passed to the requesting routine in register zero. If the module is not resident, 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 module in register zero.

If the requested module is not found on the loaded program list, the LOAD routine branches to the FINCH routine to load the requested module into storage. On return from the FINCH routine, the LOAD routine initializes the requested module'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 transient SVC routine. It then determines if the resident SVC (RSVC) option was chosen at system generation and determines if the requested SVC module is an RSVC module . If it is, the module need not be brought into main storage. If the requested module is not an RSVC, the XCTL routine branches to the FINCH module to locate the routine on the SVC library and to bring it into the SVC transient area. In either case the XCTL routine initializes the module'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 it was brought in via a LOAD macro instruction. If the requested module 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 RB of the requested module was not found inactive on the loaded program list, and the resident reenterable module option was selected at system generation, the routine searches the resident area. If the module is found in the area, a dummy LPRB for the module is placed on the loaded program list, and processing continues as if the module were originally found on the load list. If the module is not found, the XCTL routine branches to the FINCH routine to bring in the module. 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.

IDENTIFY Service Routine (IEAAID00)

The IDENTIFY service routine builds and initializes a minor request block to describe a module 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 module that 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 module specified in the DELETE macro instruction is resident. If it is, the DELETE routine exits immediately. If the module is not resident, the DELETE module finds the module'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 module and its RB. On return from the FREEMAIN routine, the DELETE routine repeats the deleting process for each minor RB belonging to the specified module. 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.

FINCH Service Routine (IEAATC00)

The functions performed by the FINCH service routine are the same for PCP and MFT with the following exceptions:

- I/O error handling.
- SVC transient area loading.

These functions are described below.

I/O ERROR HANDLING: In PCP, when control is returned from program FETCH, FINCH tests for a permanent I/O error in the processing. If no error condition exists, control is returned to the mainline routine that requested the program fetch. If an error is detected, a branch is taken to abnormally terminate the requesting task.

In MFT, when control is returned from Program FETCH on a system fetch task, if FINCH detects a permanent I/O error, the FETCH operation is recycled five times. The recycle count is kept in FINCH's work area. If the error persists after five recycles of the operation, FINCH passes control to the Dynamic Device Reconfiguration (DDR) SYSRES Effector routine, if DDR SYSRES support is in the system. DDR SYSRES returns to FINCH with a return code of 0 or 4. If the return code is 0, FINCH again recycles the Program FETCH operation. If the return code is 4, permanent-error processing takes place. If the renewed attempt to recycle the operation results in another permanent I/O error, DDR SYSRES is not invoked again. Instead, permanent-error processing takes place.

SVC TRANSIENT AREA LOADING (IEA0FN00): In MFT there is only one SVC transient area. Because more than one request for the use of this transient area may be issued at the same time, a method for resolving contention for the use of this area must be established. Therefore, in order to control the loading of the SVC transient area, a separate transient area loading task TCB and supervisor request block (SVRB) are constructed in the nucleus when the system is generated. The transient area loading task's TCB, which is the highest priority system TCB, is marked dispatchable by setting the TCBFLGS field to X'00'. The SVRB is placed in an RB wait state by setting the XRBWT field to a wait count of 1. The resume PSW field (XRBPSW) and the XRBEP field of the SVRB are initialized to contain the address of the entry point in FINCH (IEAFNCH) where the code for execution of the transient area loading task is located.

There are three kinds of requests for the loading of an SVC module into the SVC transient area that require the services of FINCH:

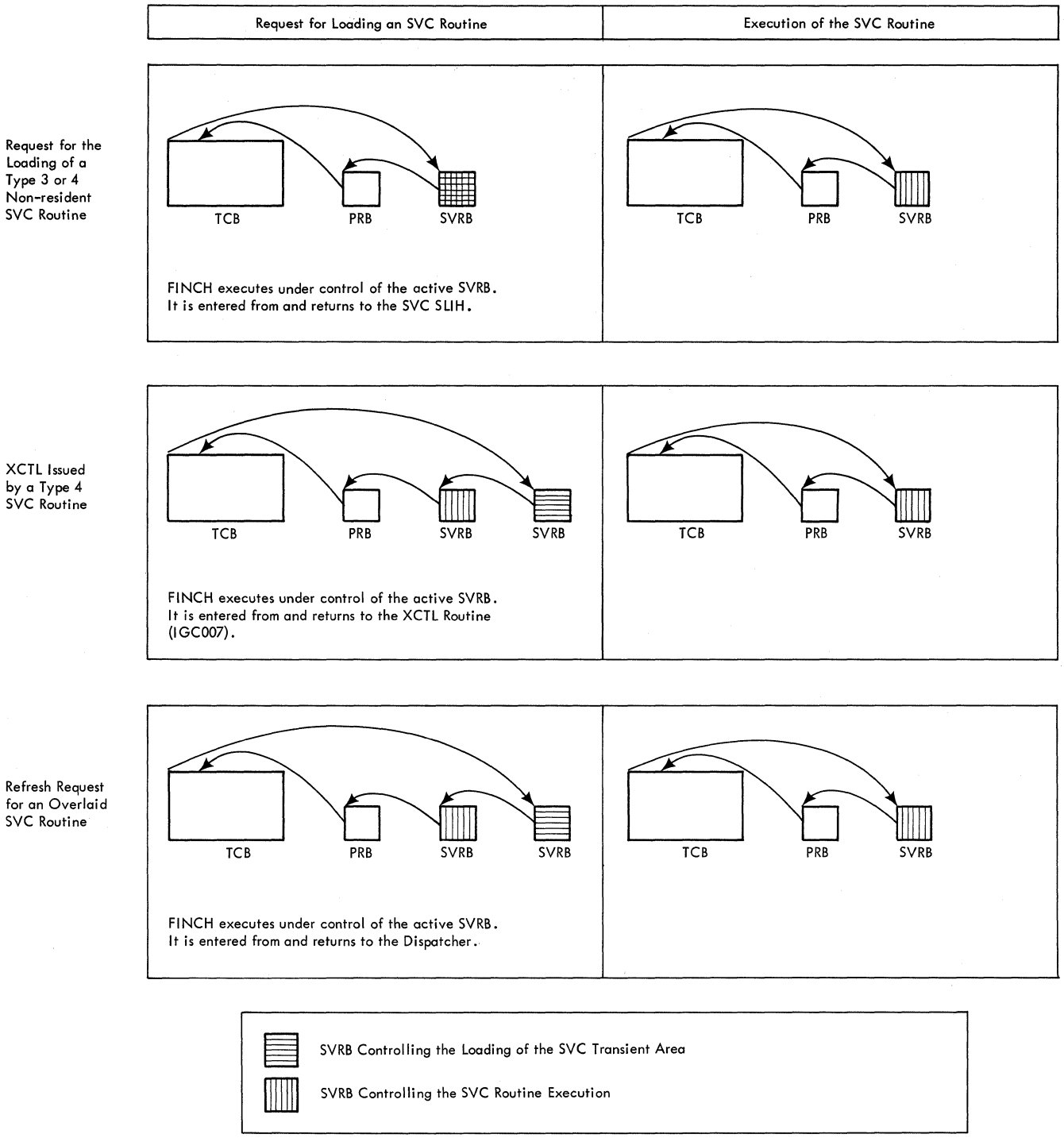
- A request for the loading of a nonresident type 3 or 4 SVC module.
- A request for loading the transient area resulting from the issuance of an XCTL macro instruction by a type 4 SVC routine for the load of a nonresident SVC module.
- A refresh request for loading an SVC module previously executing in the transient area but overlaid, before its execution completed, by another SVC module.

Each of these requests for the loading of an SVC module into the transient area is governed by an SVRB which may or may not be the same SVRB governing the execution of the SVC routine. (See Figure 11.)

When any task issues an SVC resulting in a request for the loading of a non-resident type 3 or 4 SVC module, the SVC second level interruption handler (SVC SLIH) creates and initializes an SVRB to govern the request, and adds it to the task's TCB/RB queue as the active RB. (This procedure is described in the "Interruption Supervision" section of the PCP Supervisor PLM.) This SVRB, which represents the loading request, also governs the execution of the SVC routine when it has been loaded into the transient area.

When a type 4 SVC routine issues an XCTL macro instruction, a new SVRB is created, initialized, and added to the controlling task's TCB/RB queue as the active RB by the SVC SLIH. (The expansion of the XCTL macro instruction contains a type 2 SVC which requires this creation of a new SVRB.) This new SVRB governs the execution of the XCTL routine and the request for the new SVC module load. It is removed by EXIT when the requested loading is complete. (The functions of EXIT are described in the "Interruption Supervision" section of the PCP Supervisor PLM.) FINCH reinitializes the SVRB originally created for the SVC routine to govern the execution of each succeeding load of the SVC module.

The dispatcher and the SVC SLIH set up the SVRB representing a refresh request for an SVC routine that has been overlaid before its execution completed. It is marked as an IRB by setting its XSTAB field to X'40'. (Because this SVRB is created for the sole purpose of controlling the request for an SVC routine load, it is marked as an IRB so that EXIT will not treat it as an SVRB when removing it from the TCB/RB queue.) This SVRB is removed by EXIT when the desired SVC routine is loaded into the transient area so that the SVRB controlling the initial load and execution of the SVC routine will once again control the SVC routine's execution.




• Figure 11. The SVRBs Controlling the Loading of the Transient Area and the Execution of the Loaded SVC Routines

FINCH is entered operating under control of the SVRB governing the request for the loading of the SVC routine into the transient area. It tests to determine if the desired SVC routine is already present in the transient area by comparing the transient area contents field (XSNTCC located in the SVC SLIH) with the name contained in the XRBNM field of the SVRB of the requesting task. If the contents of these areas match, FINCH sets the XSTAB field of the SVRB to X'D0' to indicate that the desired SVC routine is loaded and then exits to the routine indicated in Figure 11. If the desired SVC routine is not already present in the SVC transient area, FINCH checks to determine if that area is available by testing the load switch (LOADSW located in FINCH). A X'00' value indicates that the area is not already in use and may therefore be loaded.

In this case FINCH prepares for task switching from the requesting task to the transient area loading task. It zeros out the XSNTCC field and turns on the load switch (X'FF') to indicate that the transient area is no longer available for loading. It then takes the transient area loading task's SVRB out of the RB wait state by setting the XRBWT field to a wait count of 0. FINCH then adds the requesting task's SVRB to the transient area request queue and puts it in an RB wait state by setting the XRBWT field to a wait count of 1. It sets the address in the resume PSW field of the requesting SVRB to contain the address of a re-entry point in FINCH (labeled XSNTQUES) where the check is made to determine if the desired SVC routine is present in the transient area.

FINCH then sets NEW in the dispatcher's IEATCBP field to contain the address of the transient area loading task's TCB to indicate to the dispatcher that a task switch is required between the requesting task and the transient area loading task. (See the description of the dispatcher in this publication for an explanation of the OLD and NEW fields.) FINCH then branches to the dispatcher for the dispatching of the transient area loading task.

If the transient area is not available for loading, FINCH prepares for task switching from the requesting task to the highest priority ready task. It adds the SVRB of the requesting task to the transient area request queue and puts it in an RB wait state. (If an SVRB representing another request from the same task is already on the queue, it is removed and taken out of the RB wait state. This action is taken because the SVRB representing the previous request is no longer the active RB and will not regain control until the SVRB representing the current request completes its utilization of the SVC transient area.) FINCH then sets the address of the XSNTQUES re-entry point in the resume PSW field of the requesting task's SVRB. It sets NEW in the dispatcher's IEATCBP field to 0 to signal the dispatcher to search the ready queue for the highest priority dispatchable task. FINCH then branches to the dispatcher for dispatching of the highest priority ready task.

When the transient area loading task is dispatched, execution begins at entry point IEAFNCH in FINCH. The transient area loading task issues a branch and link instruction to the FETCH routine for bringing the desired SVC routine into the transient area. (The FETCH routine is described in the "Program Fetch" section of the PCP Supervisor PLM.) Upon return from FETCH, the transient area loading task tests to determine if an error occurred. If so, it branches to ABTERM to schedule the task requesting the loading of the transient area for ABEND. If no error is found, the transient area loading task performs the following operations:

- It moves the contents of the XRBNM field of the SVRB governing the loading request to the XSNTCC field so that the transient area contents field is initialized to contain the name of the loaded SVC module.

- It turns off the loading switch to indicate that the transient area is now available for loading other SVC modules.

- It removes all of the SVRBs on the transient area request queue by zeroing out the pointer fields and it takes them out of the RB wait state by setting their XRBWT fields to a wait count of 0.

- It places the address of the IEAFNCH entry point in FINCH in the resume PSW field of the transient area loading task's SVRB and it places the SVRB in an RB wait state.

- It sets NEW in the dispatcher's IEATCBP field to 0 to signal the dispatcher to search the ready queue for the highest priority dispatchable task. (This task may or may not have issued a request for loading an SVC module which is the same as the SVC module which has just been loaded.)

The transient area loading task then branches to the dispatcher for dispatching of the highest priority ready task.

The Transient Area Request Queue: The transient area request queue is necessary for keeping track of the SVRBs put in the RB wait state by FINCH while the transient area loading task is loading an SVC routine into the SVC transient area. Each SVRB representing a request for the loading of the transient area is put into an RB wait state and added to the queue as FINCH is entered. The SVRBs are therefore queued in 'last in first out' order. (See Figure 12.)

The transient area request queue consists of a doubleword located at displacement -16 from the address of the boundary box for each task, including tasks created by ATTACH. The first fullword contains the chain pointers. A zero value indicates that the SVRB is the last on the queue. The second fullword contains the address of the SVRB of the task requesting the load. A zero value in this field indicates that the task has not issued a request for the loading of the transient area. The head of the transient area request queue (TAQUE) is a fullword located in FINCH. It contains the address of the SVRB most recently added to the queue.

When the transient area loading task has completed loading the SVC transient area, it removes all of the SVRBs from the queue by zeroing out the pointer fields. It removes them from the RB wait state so that they may contend for use of the transient area.

## CONTENTS SUPERVISION IN AN MFT SYSTEM WITH SUBTASKING

This section describes the additional records used by contents supervision in an MFT system with subtasking and the changes in contents supervision routines caused by the additional records.

- FINCH Request Block (FRB) -- represents a request from the LOAD routine to the FINCH routine to bring a module into main storage within the requester's partition. LOAD places the FRB on the partition's job pack area queue (JPAQ), described below, and it remains there until FINCH has completed bringing the module into main storage. The FRB allows the LOAD service routine to recognize that the module is in the process of being loaded when subsequent requests for the same module are made.

The records of the modules that are in main storage or have been requested are changed in MFT systems with subtasking. Each task's active request block queue will contain all but FRBs for the task. The request represented by the FRB on the JPAQ is represented by an SVRB on the active request block queue. The active request block queue of each task or subtask continues to be pointed to by the TCBRBP field.

A separate loaded program list is pointed to by TCBLLS of each TCB, (job-step or subtask). The list contains the LRBs and LPRBs for each module in the partition's main storage that was requested by a LOAD macro instruction issued by tasks of that TCB.

A new record of modules in main storage is the job pack area queue (JPAQ). This queue has the same name as an MVT queue but it does contain the same information. The MFT job pack area queue is a queue of the LPRBs of all reenterable modules brought into the partition by LOAD macro instructions issued by any of the tasks within the partition. The queue also contains FRBs for all modules during the time they are being searched for and/or brought into the partition by the contents supervisor. The JPAQ is located by referring to the partition information block which points to the first RB on the queue.

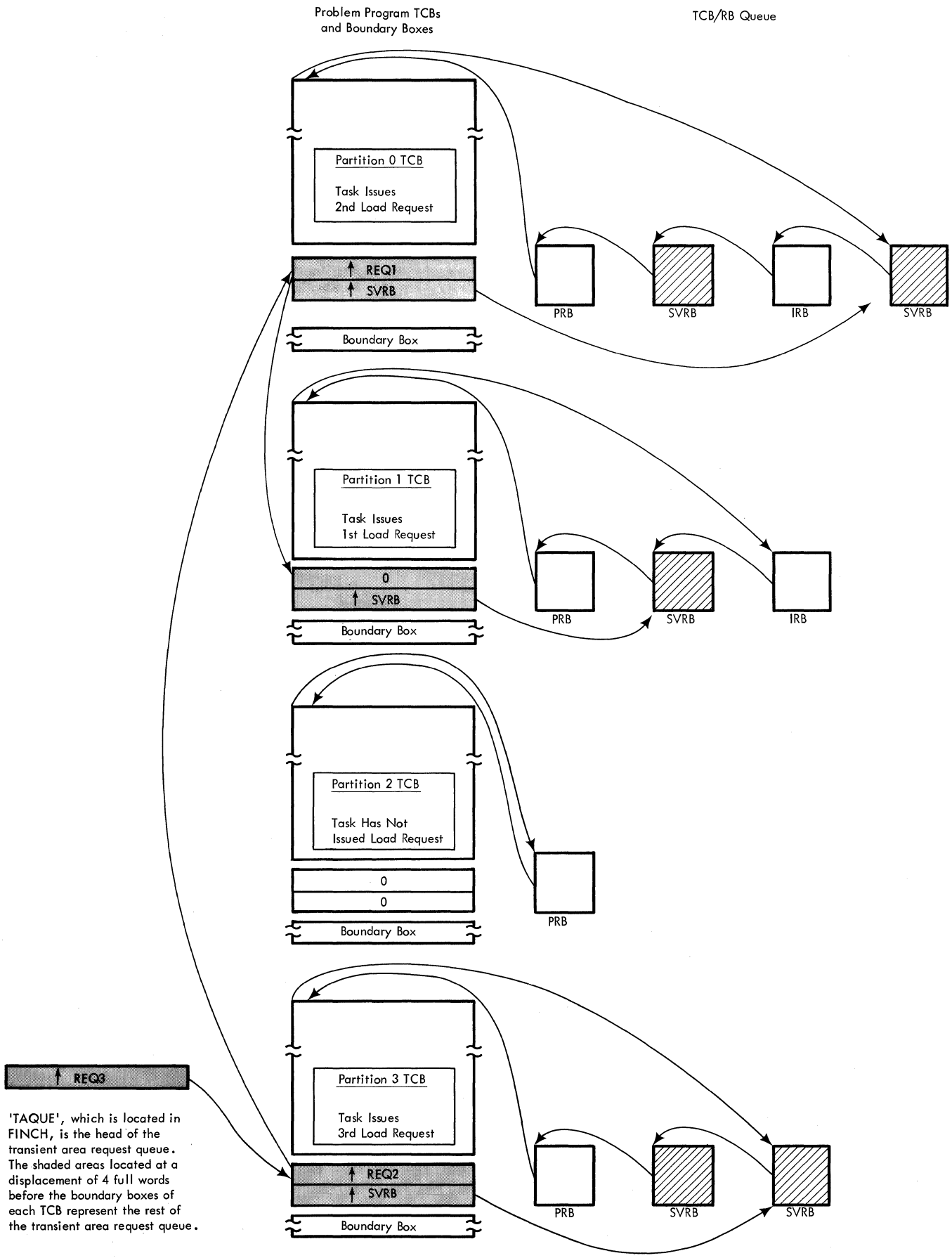
The contents supervision routines have been modified for MFT with subtasking to include the JPAQ and the FRB in their processing. The descriptions that follow describe the additional or changed functions of the service routines.

### LINK Service Routine (MFT With Subtasking) (Macro IEAATC)

The LINK service routine in MFT systems with subtasking differs from the Link service routine in MFT without subtasking as follows:

The Link routine searches the job pack area queue (JPAQ) before searching the loaded program list. If an LPRB for a module is found on the JPAQ, the Link routine queues the RB to the requester's active request block queue. If the requested module is in the process of being loaded (that is, an FRB for the module is found), the Link routine places the requester in a wait state until the module is loaded.

After finding a usable copy of a module or having FINCH bring one into main storage, the Link routine checks whether the routine was requested as the result of an Attach macro instruction. If an Attach macro instruction was the cause, the Link routine removes the dummy RB from the subtask's TCB active request block queue and frees the dummy RB's main storage space. The Link routine then constructs an RB and places it on the active request block queue.




• Figure 12. The Transient Area Request Queue and the TCB/RB Queue

LOAD Service Routine (MFT With Subtasking) (Macro IEAATC)

The LOAD service routine first determines whether an RB for the requested routine is queued on the job pack area queue (JPAQ). If the RB is queued on the JPAQ, and the entry is a Finch request block (FRB), then the LOAD service routine defers the new request by queueing a wait list element onto the FRB and placing the requesting routine in a wait state, pending completion of the original load request. If the entry on the JPAQ is an LRB or LPRB the LOAD service routine places the entry point of the requested routine in register zero and returns.

If the requested routine is not represented by an RB, the LOAD service routine builds an FRB for the requested routine and queues it on the JPAQ. The LOAD service routine then searches the resident reenterable routine area for the module. If the module is found in the resident reenterable routine area, LOAD dequeues the FRB, sets dispatchable all tasks waiting on the FRB, and returns the address of the entry point to the caller in register zero. If the module is not in the resident reenterable routine area, LOAD searches the task load list. If the module is found on the task load list, LOAD dequeues the FRB, sets dispatchable all programs waiting on the FRB, and returns the entry point address to the caller. Otherwise, LOAD passes control to FINCH to bring a new copy into main storage.

On return from FINCH, LOAD dequeues the FRB and then tests the RB created by FINCH to determine if the requested module is reenterable. If the module is reenterable, LOAD queues the RB on the job pack area queue. If the module is not reenterable, LOAD queues the RB on the loaded program list. LOAD then sets dispatchable any tasks waiting on the FRB and passes the entry point address of the requested routine to the caller.

DELETE Service Routine

The DELETE Service routine determines if the routine specified in the DELETE macro instruction is a resident reentrant load module (if the resident reentrant load module option is in the system). If the module resides in the resident area, DELETE exits immediately. If the routine specified does not reside in the resident area, the Delete routine searches the load list. If the routine is not on the load list and the subtasking option is included in the system, DELETE searches the job pack area queue (JPAQ). If the module is not on either queue, DELETE returns to the caller. If the module is found on either the load list or the job pack area queue, DELETE decrements the use count in the RB by one. If the use count reaches zero, DELETE dequeues the routines 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.

## 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 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 supervision 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 Attach routine for subtask TCBs, if the subtasking option is included in the system.
- 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.

- The CIB creation routine of SVC 34 and System Task Control for CIBs.

- The JSCB creation routine of System Task Control for JSCBs.

- The DISPLAY R routine of SVC 34 for the DISPLAY R WTO buffer.

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.

If SMF is supported, the FREEMAIN and GETMAIN routines, which are described in the PCP Supervisor PLM, are modified to pass control to SMF subroutines that maintain storage usage information in the timing control table (TCT).

If SMF is in the system, the FREEMAIN routine passes control to its SMF storage information subroutine (FMSMFCRE). The subroutine first tests the TCBTCT field of the TCB for the address of a TCT. If this field contains zeroes, there is no TCT and storage usage information can not be recorded. If the TCBTCT field contains a TCT address, the subroutine determines if the newly released storage causes a change in the low water mark (LWM) or the high water mark (HWM) for the partition. (The LWM is the value of the highest storage address allocated from the bottom of the partition, and the HWM is the value of the lowest storage address allocated from the top of the partition.) If either is changed, the SMF storage subroutine stores the new value in the appropriate TCT field (TCTLWM or TCTHWM). It then returns control to the FREEMAIN routine.

If SMF is in the system, the GETMAIN routine also passes control to its SMF storage information subroutine (GMSMFCRE). It tests the TCBTCT field of the TCB for the address of a TCT. If this field contains zeroes, there is no TCT and storage usage information cannot be recorded. If the TCBTCT field contains a TCT address, the subroutine determines if the newly allocated storage alters either the LWM or the HWM. If either is altered, it stores the new value in the appropriate TCT field. It also calculates the all-time minimum difference, in terms of 2048-byte blocks, between the LWM and the HWM. If the new allocation creates a new minimum difference, the SMF storage subroutine records the new difference in the TCTMINC field of the TCT. It then returns control to the GETMAIN routine.

If an unconditional GETMAIN macro instruction cannot be satisfied by the main storage supervisor, the requesting task is usually abnormally terminated. However, before scheduling the task for termination, the main storage supervisor tests the task's TCB for an "ABEND in progress" or "ABTERM scheduled" bit. If either bit is on, the ABEND is not scheduled, but an error code of four is placed in register 15 and control is returned to the requesting task. This avoids a loop caused by rescheduling an abnormal termination for the task.

Main storage supervision in MFT systems without subtasking is fully described in the PCP Supervisor PLM. Main storage supervision in MFT systems with subtasking differs from that described as follows:

- In MFT systems with subtasking, the Supervisor builds a gotten area subtask queue to describe the main storage obtained for a subtask by a system-issued GETMAIN macro instruction. The Supervisor adds eight bytes to the amount of requested main storage. These eight bytes precede the main storage available to the subtask and contain the gotten area subtask queue element (GQE). The gotten area subtask queue originates in the TCBMSS field of the subtask's TCB. This field contains the address of a four-byte field that contains the address of the first GQE.

- When a subtask terminates, the normal or abnormal termination routines refer to the GQEs chained to the subtask's TCB in order to free the main storage belonging to the terminating subtask. The GQEs are removed from the queue whenever the subtask storage space is freed.

## 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 second level interruption handling routine IEA0TI00 takes appropriate action, depending on the type of interval which has expired.

### TIMER SECOND LEVEL INTERRUPTION HANDLER (IEA0TI00)

The MFT configuration of the operating system utilizes the timer second level interruption handler (TSLIH) described in the PCP Supervisor PLM with additional processing for handling expired job step time or expired wait time if the job step timing option was specified at system generation.

When the TSLIH determines that the timer interruption occurred because of a job step or wait time limit expiration, it prepares to abnormally terminate the task whose time expired. In this case there is a PIB address in the TCBPIB field and the high-order bit in the job step timing status bits field of the PIB is on, indicating that job step timing for the problem program was originally requested by the Initiator via the STIMER macro instruction.

The TSLIH examines the TQEFLGS field to determine the TQE type. If the TQE is a REAL type, a wait time limit has expired. The TSLIH branches to ABTERM with the address of the TCB to schedule ABEND processing. It also passes to ABTERM an ABEND code of 522 indicating that the wait time limit expired. Upon return from ABEND, the TSLIH reinstates the TQE as a TASK type with the actual value of the CPU time remaining for the job step, by setting the TQEFLGS to indicate a TASK type TQE and moving the value of the CPU time remaining for the job step from the TQESAV field to the TQEVAL field.

If the TQE is a TASK type, a job step time limit has expired. The TSLIH branches to ABTERM with the address of the TCB to schedule ABEND processing. It also passes to ABTERM an ABEND code of 322 indicating that the job step time limit has expired.

#### SMF Processing

When the system management facility is supported, the timer second level interruption handler is required to perform additional processing for handling:

- An expired system wait time 10-minute TQE.
- Expired job step time.
- Expired wait time.

If the timer interruption is due to the expiration of a supervisor system wait time 10-minute TQE, the TSLIH obtains the accumulated system wait time for the preceding 10 minutes from the second word in the save area SYSWSAVE. It then adds this value to the SMCAWAIT+4 field of the system management control area (SMCA), and zeroes out the second word in SYSWSAVE. It places a value of 10 minutes in the 10-minute TQE and returns it to the timer queue. (Each time that step termination is entered, the SMCAWAIT field is checked. If it is non-zero, an SMF system 10-minute wait time record (type 1) is generated.)

If the timer interruption is due to the expiration of a job step or wait time limit TQE, the TSLIH checks the SMCAOPT field of the SMCA to determine if user exits are specified. If user exits are specified, the TSLIH initializes a compiled-in IRB/IQE to schedule the asynchronous SMF time/output limit expiration routine IEATLEXT.

If user exits are not specified, the TSLIH schedules the task for ABEND. (See the "Timer Second Level Interruption Handler" section above.)

### 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.

For a complete description of timer supervisor, see PCP 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 the PCP Supervisor PLM.

## MFT Recording/Recovery Routines

Operating System Recording/Recovery routines are optional control program routines which may be selected during system generation.

They handle the following types of equipment malfunctions.

- Malfunctions of the central processing unit (CPU).
- Malfunctions in a channel.
- Malfunctions of devices.

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

System Environment Recording includes:

- System Environment Recording 0 (SER0, described in the PCP Supervisor PLM.
- System Environment Recording 1 (SER1), also described in the PCP 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, GY27-7155.
- Channel-Check Handler (CCH), described in IBM System/360 Operating System: Input/Output Supervisor, Program Logic Manual, GY28-6616.
- Alternate Path Retry (APR), described in IBM System/360 Operating System: Input/Output Supervisor, Program Logic Manual, GY28-6616.
- Dynamic Device Reconfiguration (DDR), described in IBM System/360, Operating System: Input/Output Supervisor, Program Logic Manual, GY28-6616.

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 the System Generation SRL).

Channel-Check Handler: The channel-check handler (CCH) may be selected during system generation for System/360 Models 65, 75, and 91 using either the 2860 or 2870 Multiplexor Channel. CCH is standard for the Model 85.

CCH aids recovery from channel errors (channel control checks and interface control checks) by providing channel error information to IBM-supplied device-dependent error recovery procedures (ERP). CCH also builds a record entry which is later written on SYS.LOGREC by the outboard recorder (OBR) of the I/O supervisor.

ALTERNATE PATH RETRY ROUTINE

Alternate Path Retry (APR) allows an I/O operation that has developed an error on one channel to be retried on another channel (if another channel is assigned to the device performing the I/O operation). APR accomplishes this by causing channel-detected errors to be retried in a selective manner on the available paths to a device. As paths are found to be inoperative, they are marked offline, thus preventing unnecessary retry from being initiated to the failing paths.

APR also provides the capability to VARY a path to a device online or offline, using the VARY PATH command. The VARY PATH command processor is part of the Master Scheduler (SVC 34). The last path to a device will not be varied offline.

While it is not model dependent, APR only performs its function usefully in a system with alternate paths and CCH.

Four paths to each device are supported; teleprocessing paths are not supported.

DYNAMIC DEVICE RECONFIGURATION ROUTINE

Dynamic Device Reconfiguration (DDR) allows a demountable volume to be moved from one device to another, and repositioned if necessary, without abnormally terminating the affected job or reperforming IPL. A request to move a volume may be initiated by the operator with the SWAP command. The SWAP command processor is part of the Master Scheduler (SVC 34).

The system may request a SWAP after a permanent I/O error for non-SYSRES devices or after an error in a system fetch operation for SYSRES devices.

DDR is not model-dependent.

SYSTEMS WITHOUT RECORDING/RECOVERY ROUTINES

A machine check or I/O interruption caused by an equipment malfunction places in a wait state those IBM System/360 models that do not have Recording/Recovery routines. A message is issued on the console telling the operator to load the System Environment Recording, Editing, and Printing (SEREP) program. SEREP is a model-dependent, stand alone diagnostic program. Its use is described in IBM System/360 Operating System: Operators Guide, GC28-6540.

ENTRY TO RECORDING/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.

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.

If the Channel-Check Handler was selected during system generation, the I/O Supervisor enters the Channel-Check Handler Interface within the I/O Supervisor.

When a permanent I/O error is indicated (after retry by the device-dependent error recovery procedures of IOS), the OBR/SDR routine is entered to record the error.

If DDR was selected during system generation, the OBR/SDR routine enters DDR after recording the permanent I/O error. When the permanent I/O error occurs on a system fetch operation, the DASD ERP or FINCH enters DDR SYSRES before OBR/SDR receives control (if DDR SYSRES was selected during system generation).

## Checkpoint/Restart Routines

The checkpoint/restart routines used by MFT allow a job to restart after an abnormal termination. The checkpoint routine (SVC 63) is used by the programmer to create a

record of the job's main storage region at selected points during the execution of a job step. The routine is identical with the PCP checkpoint routine described in the PCP Supervisor PLM.

The restart routine (SVC 52) allows jobs to restart at a checkpoint. If the restart is automatic, it will occur at the last valid checkpoint taken by the job before it abnormally terminated. If the restart is deferred, it will occur at the checkpoint specified by the job statement. Processing of the restarting job is discussed in the Job Processing section of this manual. The restart routine is described in the PCP Supervisor PLM.

In MFT systems with subtasking, a CHKPT macro instruction must not be issued by a subtask or by a job step task that has active subtasks.

## System Management Facility

The Supervisor performs the following functions if the System Management Facility (SMF) has been specified at system generation:

- Maintains a record of system wait time over a 10-minute period.
- Assists in handling job step or wait time limit expirations.
- Records the number of references to user data sets.
- Performs an output limiting function for SYSOUT data sets.
- Records the number of 2048-byte blocks of storage assigned to a user program.

Whenever the Dispatcher puts the system in the wait state, it places the contents of the interval timer in the first word of a special save area, SYSWSAVE. When an external or input/output interruption ends the wait state, the appropriate interruption handler branches to SMF wait time collection routine IEAQWAIT. This routine reads the interval timer again and compares its value with the values stored by the Dispatcher to determine the elapsed system wait time. It then adds this elapsed time to the value in the second word of SYSWSAVE, to obtain the accumulated wait time for the system. When the supervisor 10-minute interval expires, the timer second level interruption handler reads out the value in the second word of SYSWSAVE, which represents the total system wait time for the preceding 10-minute interval. The TSLIH adds this value to the value in a field in the system management control area, SMCAWAIT+4. Each time the step termination routine of Job Management is entered, it checks the total wait time recorded in the SMCA. If it is nonzero, the termination routine generates an SMF 10-minute wait time record (type 1).

Whenever a job step or wait time limit TQE expires and user exits are specified, SMF time/output limit expiration routine IEATLEXT passes control to SMF user time limit expiration routine IEFUTL. The user routine determines whether or not to grant a time limit extension. If an extension is granted, IEATLEXT resets the TQE with the value of the extension. If no extension is granted, the routine prepares for abnormal termination of the task whose job step or wait time limit expired.

Whenever a reference is made to a user data set, SMF EXCP counting routine IEASMFEX, in the I/O Supervisor, records the references in an EXCP counter. There is a counter for each SYSOUT data set/device combinaton. The counters are part of the TCTIOTBL segment of the timing control table (TCT). IEASMFEX totals the EXCP count fields and compares the sum of the EXCP counts with the output limit value placed in the output limit field of the TCTIOT by the OPEN routine. If user exits are specified in the SMF options field of the SMCA (SMCAOPT), when the output limit is exceeded, the routine tests the compiled-in output limit IQE to determine if it is already in use. If it is being used, the routine increases the output limit value by one and returns to the I/O Supervisor. If the output limit IQE is not in use, IEASMFEX initializes the IRB/IQE for scheduling SMF time/output limit expiration routine IEATLEXT, and marks the IQE to indicate that it is now in use. The user exit routine IEFUS0 determines whether or not an additional number of EXCPs will be granted. If an additional number of EXCPs is granted, IEATLEXT increases the output limit and stores a new output limit in the TCTIOT. If no additional EXCPs are granted, the routine prepares for abnormal termination of the task whose output limit was exceeded. When IEATLEXT and the user routine complete processing, the IQE is once again marked available for use.

As the main storage supervision routines allocate or release storage within a partition assigned to a user program, the following information is recorded in the appropriate TCT fields.

TCTLWM -- the highest address currently allocated from the bottom of the partition - low water mark (LWM).

TCTHWM -- the lowest address currently allocated from the top of the partition - high water mark (HWM).

TCTMINC -- the smallest amount of space within the partition that has ever been unused at any one time (the all-time minimum difference between the LWM and HWM).

This information is maintained by the SMF storage usage information subroutines, GMSMFCRE and FMSMFCRE, which are part of the GETMAIN/FREEMAIN routine (IEAAMS). They are described in the "Main Storage Supervision" section of this publication.

SMF ROUTINES

The major SMF routines that perform the functions described above include:

- SMF wait time collection routine IEAQWAIT.
- SMF time/output limit expiration routine IEATLEXT.
- SMF EXCP counting routine IEASMFEX.

MFT and MVT use the same wait time collection routine. MFT and MVT also use the same EXCP counting routine except for modifications for MFT. In MFT, the EXCP counting routine schedules SMF time/output limit expiration routine IEATLEXT by initializing a compiled-in IRB/IQE. In MVT, storage is obtained for the IRB/IQE via a GETMAIN macro instruction. These routines are described in the "Special Features" section of the MVT Supervisor PLM. SMF time/output limit expiration routine IEATLEXT also performs the same functions in MFT and MVT. However, these functions are implemented differently. Two areas of difference that should be noted include:

- In MFT, IEATLEXT executes under a compiled-in IRB/IQE.

- In MFT, the user exit routines to which it passes control (IEFUTL and IEFUSO), are resident in the nucleus. These routines reside in the link pack area in MVT.

IEATLEXT is described below.

SMF Time/Output Limit Expiration Routine (IEATLEXT)

This routine, which is resident in the nucleus, is part of the timer second level interruption handler (IEAQTI00). It executes under a compiled-in IRB/IQE scheduled by the TSLIH or the SMF EXCP counting routine.

It receives control in two cases:

- When a job/step or wait time limit has expired.
- When the SYSOUT output limit has been exceeded.

When it is entered, IEATLEXT determines if its IRB/IQE was initialized by the TSLIH or the SMF EXCP counting routine. If the TSLIH requested its processing, a job/step or wait time limit has expired and IEATLEXT prepares to pass control to SMF user time limit expiration routine IEFUTL.

The routine passes control to IEFUTL, indicating the type of expiration in register 15. It also passes the address of a 72-byte save area and the contents of the user data field (TCTUDATA) in the TCT.

The user routine determines whether or not a time extension will be granted. It returns control to the SMF time/output limit expiration routine with a return code of 0 to indicate no time extension, and 4 to indicate a time extension has been granted.

IEATLEXT takes various actions depending on the return code and the type of time limit that expired. If the job step time limit expired and:

- If the user grants an extension, it is returned in register 1. The routine adds the value of the extension to the TCT field containing the total time allocated to the task. It also places the value of the extension into the expired TQE and branches to the timer enqueue routine (IEAQTE00) to add the TQE to the timer queue.

- If the user does not grant an extension, the routine abnormally terminates the task by branching to ABTERM with an ABEND code of 322.

If the wait time limit expired and:

- If the user grants an extension, it is returned in register 1. The routine places the value of the extension into the expired TQE and branches to the timer enqueue routine to add the TQE to the timer queue.

- If the user does not grant an extension, the routine abnormally terminates the task by branching to ABTERM with an ABEND code of 522.

If the SMF EXCP counting routine requested its processing, the output limit has been exceeded and IEATLEXT prepares to pass control to SMF SYSOUT limit user exit routine IEFUSO.

It passes control to IEFUSO with the address of a two-word parameter list in register 1. This parameter list contains

the address of the job name and time stamp from the JMR and the address of the DCB for the data set.

The user routine determines whether or not an additional number of EXCPs (a SYSOUT limit extension) will be granted. It returns control to IEATLEXT with a return code of 0 to indicate no extension, or 4 to indicate that an additional number of records are to be added to the output limit.

IEATLEXT again takes various actions depending on the return code:

- If the user grants an extension, it is returned in register 1. The routine adds the extension to the output limit in the TCTIOT.
- If the user does not grant an extension, the routine abnormally terminates the task by branching to ABTERM with an ABEND code of 722.

# Job Management

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 an 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: the reader/interpreter, the initiator/terminator, the system output writer, and direct system output (DSO) processing. The functions of the reader/interpreter are similar to the MVT reader; additional information can be found in the MVT Job Management PLM.

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 the Job Management section of this publication; for information on allocation and termination, refer to the MVT Job Management PLM.

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 and DSO was not used, 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 the MVT Job Management PLM 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 a console.
- 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/master console to its alternate console.
- If the system has Multiple Console Support, the communications task processes the delete operator message (DOM) macro instruction and provides buffer management for all console devices.

## MASTER SCHEDULER TASK FUNCTIONS

The master scheduler task consists of SVC 34 and the master scheduler resident command processor routines. The SVC 34 command scheduler routines process all commands initially. The job queue manipulation and partition definitions, which are not fully processed by SVC 34, are passed to the master scheduler resident command processor. Table 1 lists the commands used in MFT and indicates the routine which responds to the commands after initial processing.

•Table 1. Responders to Commands After Initial Processing

| Command | Responder |
|---|---|
| CANCEL (active jobs) | Initiator |
| CANCEL (job in queue) | Master Scheduler |
| DEFINE | Master Scheduler |
| DISPLAY STATUS, JOBNAMES, DSNAME | Initiator |
| DISPLAY A,N,Q,U,jobname, CONSOLES | Master Scheduler |
| DISPLAY R | Master Scheduler |
| DISPLAY SPACE | I/O Device Allocation |
| DISPLAY T | Timer Maintenance Routine * |
| HALT | Statistics Update Routine * |
| HOLD | Master Scheduler |
| LOG | System Log |
| MODE | Master Scheduler |
| MODIFY | Writer or DSO 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 DSO Writer | DSO Writer |
| START/STOP Initiator | Initiator |
| START/STOP Reader | Reader/Interpreter |
| START/STOP Writer | Writer |
| SWAP | Master Scheduler |
| UNLOAD | Initiator |
| VARY UNIT | Initiator |
| VARY CH, CPU, PATH, STOR | Master Scheduler |
| WRITELOG | System Log* |

*See the publication IBM System/360 Operating System: MVT Supervisor, Program Logic Manual, GY28-6659.

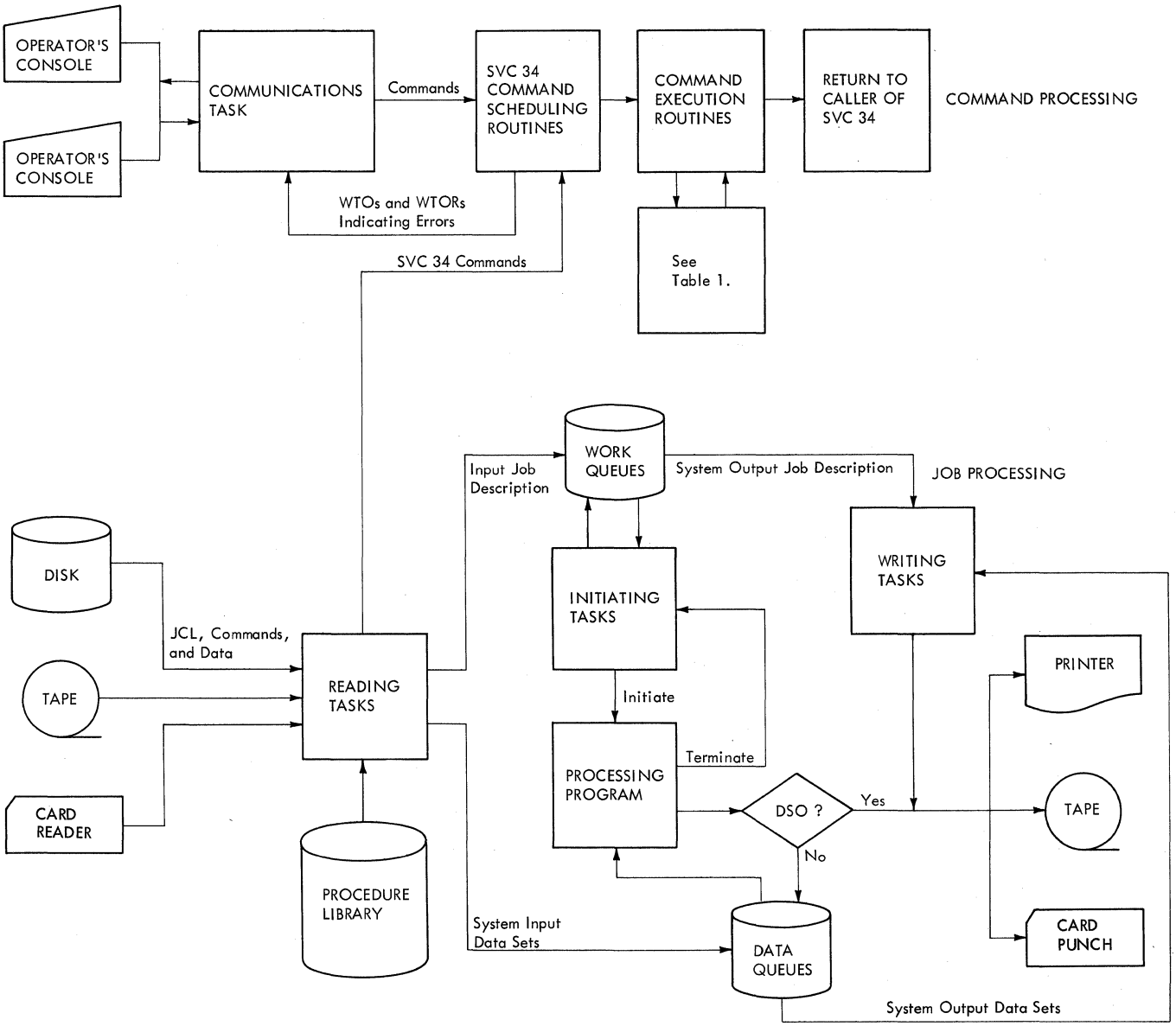
## JOB MANAGEMENT CONTROL FLOW

Figure 13 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 not, the initialization commands (SET, START reader, START writer, and START INIT) are issued either automatically by the master scheduler task or by the operator performing the IPL, and job processing begins.




• Figure 13. Job Management Data Flow

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:

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

References to the MVT Job Management PLM 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.

## 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, HALT, and SWAP, commands can be entered into the system through the console or the input job stream. The DEFINE, HALT, and SWAP commands 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 14). 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. It also deletes messages from the CRT display of the Model 85 operator console via the DOM macro instruction.

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.

The commands accepted and processed by MFT are the following:

CANCEL
DEFINE
DISPLAY
HALT
HOLD
LOG
MODE
MODIFY
MOUNT
RELEASE
REPLY
RESET
SET
START
STOP
SWAP
UNLOAD
VARY
WRITELOG

The format and syntax of these commands can be found in the Operator's Guide SRL.

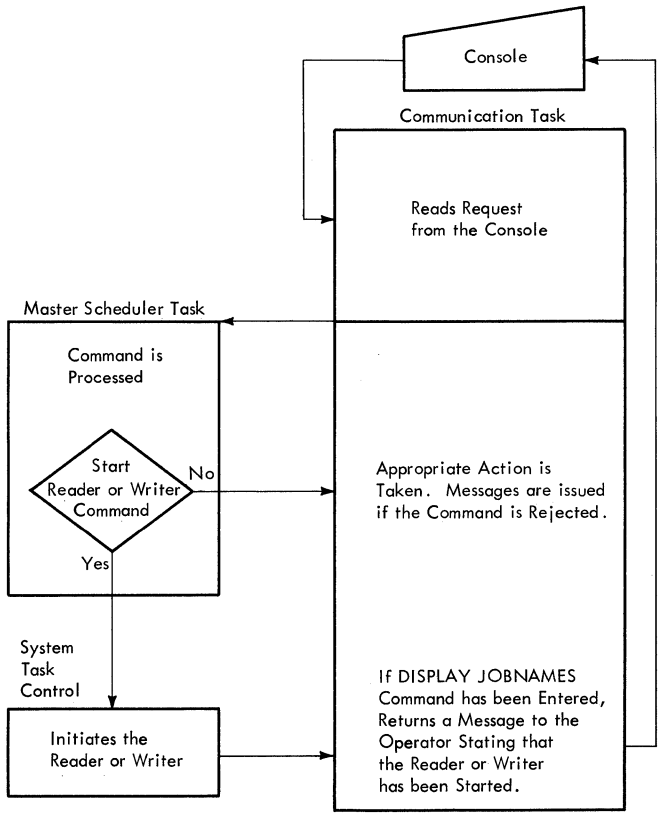



Figure 14. Command Processing Flow

## Communications Task

The routines that handle operator-system communication are contained in the communications task. Communication may take one of two forms: commands, which allow the

operator to change the status of the system or of a job or job step; and 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.

The WTO macro instruction processor also provides initial processing for write-to-programmer instructions.

## WTO/WTOR MACRO INSTRUCTION PROCESSING

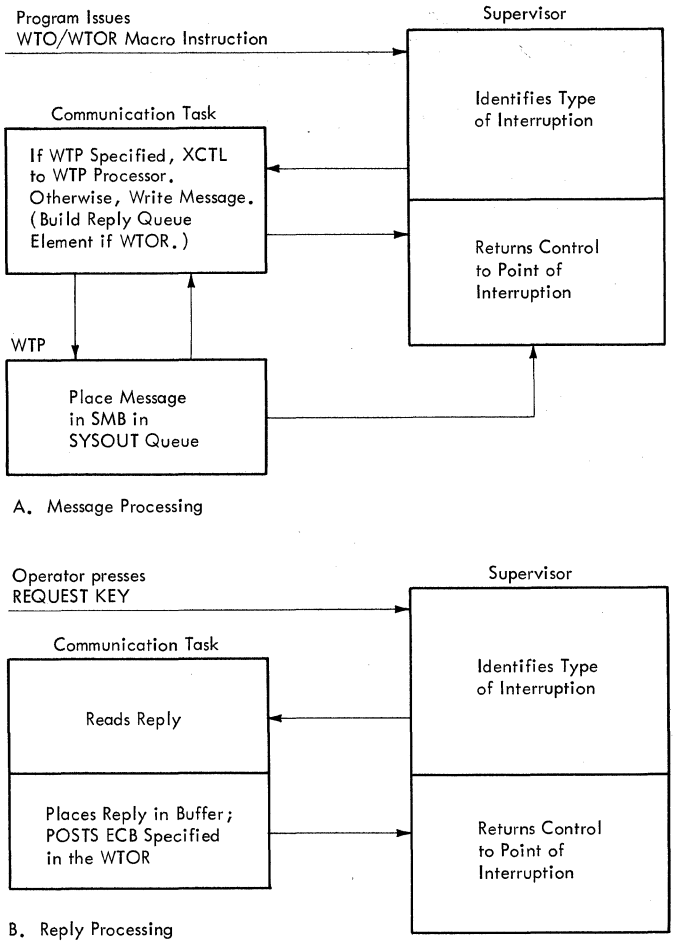
Whenever a WTO or WTOR macro instruction is issued, a supervisor call (SVC 35) interruption occurs. The supervisor identifies the type of interruption and passes control to the WTO routine. If a routing code of 11 is not specified, control is passed to the communications task to issue messages and/or to read replies.

If the WTO or WTOR macro instruction specifies a routing code of 11, then the message is a write-to-programmer, and the WTO routine passes control to the write-to-programmer routines. Upon completion of write-to-programmer processing, control is returned to the WTO routine which will process any WTO macro instruction with an additional routing code. All WTOR macro instructions are also processed by the WTOR routine, regardless of the routing code of 11. (See Figure 15.)

Write-to-programmer processing is described in the section "Master Scheduler Task."

## EXTERNAL INTERRUPTION PROCESSING

When the operator presses the INTERRUPT key, an external interruption occurs. The communications task then switches from the primary console/master console to its alternate device. (See Figure 16.)




• Figure 15. WTO/WTOR Macro Instruction Processing Flow

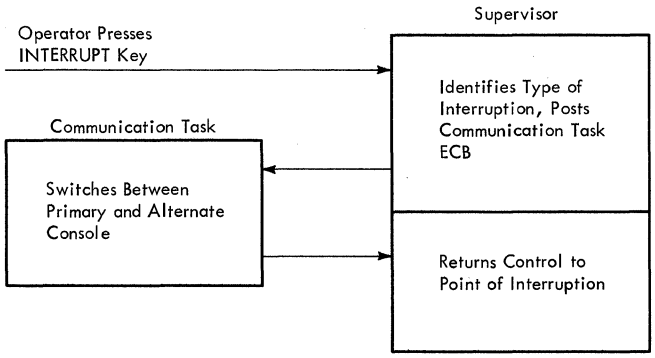



Figure 16. External Interruption Processing Flow

## Communications Task Modules

The communications task (Chart 16) 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 write operations and error checking.

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

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

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

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.

The write-to-operator routine tests the macro instruction for a routing code of 11. A routing code of 11 indicates a write-to-programmer message. If there is such a routing code, control is passed via an XCTL macro instruction to the write-to-programmer routines. After write-to-programmer processing is completed, control is returned to either the WTO routine or to the program that issued the macro instruction.

Control is returned to the WTO routine for processing as described below if any one of the following conditions is true:

- MCS is in the system and a specific console is designated to receive routing code 11 messages, or
- An additional routing code is specified in the macro instruction, or
- The macro instruction is a WTOR.

Otherwise, control is returned to the program that issued the macro instruction.

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.

## Communications Task with Multiple Console Support

The MFT communications task with Multiple Console Support (MCS) is similar to the MVT communications task except that MFT does not obtain buffers dynamically. The MCS communications task receives control as a result of an external interruption, an operator console attention, an I/O interruption for a console, or a WTO (R) or DOM macro instruction. The following paragraphs describe the communications task routines with MCS (for a detailed description of these modules see the MVT Supervisor PLM):

Communications Task Router Routine (IEECMAWR): waits for the posting of an external, attention, I/O, WTO(R), or DOM ECB. Control is passed to the appropriate routine to handle the posted ECB, to pro-

vide console switching, or to provide buffer management.

Communications Task Device Interface Routine (IEECMDSV): passes control to the device support routine for the device on which I/O is to be performed, or consolidates system and console output queues.

Communications Task Console Switch Routine (IEECMCSW): performs console switching as a result of an external interruption, an unrecoverable I/O error, or a VARY command. It also switches the hard copy log to the master console when both log data sets are full.

Communications Task WTO(R) Routine (IEECMWSV): marks WTO queue elements to appropriate console output queues.

Communications Task DOM Routine (IEECMDOM): marks WTO queue elements on the system output queue to be purged.

Console Device Support Routines: provide read and write functions for the associated console devices.

The following modules remain unchanged with MCS:

Write-to-operator (IEECVWTO)
Write-to-operator with reply (IEEVWTOR)
External Interrupt (IEECVCRX)
Console Interrupt (IEECVCRA)

Note: The routines that support the cathode ray tube (CRT) display operator consoles (that is, the 2260 Display Station, the 2250 Display Unit, and the Model 85 Operator Console) are identical with those used with MVT. For a complete description of these routines, see the MVT Supervisor PLM.

## 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 completely except CANCEL (inactive jobs), DEFINE, DISPLAY (A, Q, N, U, jobname), HOLD, RELEASE, RESET, START and WRITELOG. SVC 34 does preliminary processing of these commands and passes control to the resident command processor to complete the processing of all but the WRITELOG command. When a WRITELOG command is found, SVC 34 stores it and posts the System Log task ECB.

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 communicatcns 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.

MULTIPLE CONSOLE SUPPORT REQUIREMENTS

In systems that include Multiple Console Support (MCS), a hard copy of all operator and system messages is required when there is an active graphic console or more than one active non-graphic console. Because of this requirement, a system log function is provided which may be specified as the hard copy log. In MFT, the System Log operates under its own TCB created at system generation. The System Log task is the highest priority task in the operating system. The master scheduler routine IEFSD569 calls the log initialization routine IEEVLIN which initializes control blocks and obtains storage for the Log Control Area and the log buffer. The Log Support routines in an MFT environment function similarly to those in an MVT environment. For a further description of the system log and the Log Support routines with MCS, see the MVT Supervisor, Program Logic Manual.

SVC 34 FUNCTIONS

SVC 34 (Charts 13, 14, and 15) 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 with respect to the master scheduler are:

CANCEL (active jobs only)
DISPLAY (JOBNAMES, R, SPACE, DSNAME, T, or STATUS)
HALT
MODIFY
MODE
MOUNT
REPLY
STOP
SWAP
UNLOAD
VARY

For CANCEL (inactive jobs or with the IN or OUT parameter), DEFINE, DISPLAY (A, Q, N, U, jobname), HOLD, RELEASE, and RESET, SVC 34 does preliminary processing before passing control to the resident command processor. If the resident command processor is processing a DEFINE command, SVC 34 will queue all commands until the DEFINE command has been completely processed.

For the LOG command, SVC 34 issues a WTL (SVC 36) to have the LOG command processed in manner similar to a Write-to-log macro instruction issued from a problem program.

The SWAP command is accepted and processed only if Dynamic Device Reconfiguration (DDR) is in the system. The VARY PATH command is accepted and processed only if Alternate Path Retry (APR) is in the system.

With four exceptions, the routines used for MFT SVC 34 processing are those used for MVT SVC 34 processing. The four routines unique to MFT are routine IEESD571 (used for the DEFINE and MOUNT commands); routines IEESD561 and IEE3903D (used for the STOP INIT and START commands); and routine IEE2803D (used for the CANCEL command). These routines are described in the following paragraphs.

DEFINE and MOUNT 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 IEECIR50.

This routine processes the MOUNT command as that command is processed in PCP. It builds a parameter list for, and issues an XCTL macro instruction to, the PCP master command EXCP routine IGC0103D.

CANCEL Command Routine (IEE2803D)

This routine processes the CANCEL command by scanning the CSCBs for the job name given in the CANCEL command. If the job name is found, indicating that the job is active, and if the command did not have an IN or OUT parameter, the CSCB is checked to determine if it is cancelable, that is, if it represents a problem program. If it does, IEE2803D issues a BALR to ABTERM, passing the address of the job's TCB and indicating a completion code of 222 if no dump is to be taken, or 122 if a dump is to be taken.

If the CSCB is not cancelable, that is, if it represents a system task, the CSCB is marked canceled and is posted.

If the job is represented on the CSCB chain, but the command specified IN or OUT, the "Job Selected" message is written to the operator and control is returned to the caller.

If the job is not represented on the CSCB, indicating that the job is either in the input or output queue(s) or that it does not exist, IEE2803D passes control via an XCTL macro instruction to CSCB creation routine IEE0803D to build a CSCB for the CANCEL command. (See the MVT Job Management PLM for a description of IEE0803D.)

STOP INIT and START Commands Processing Routines (IEESD561 and IEE3903D)

These routines perform the initial processing for all the START commands and the STOP INIT command. When a START command is received, STOP INIT and START command syntax scan routine IEESD561 examines the command parameters. If anything other than a system reader or writer is to be started, the routine determines the number and status of the partition named in the command. If the command is a STOP INIT command, IEESD561 determines which partition contains the initiator to be stopped. The routine then passes control via an XCTL macro instruction to STOP INIT and START command processor routine IEE3903D.

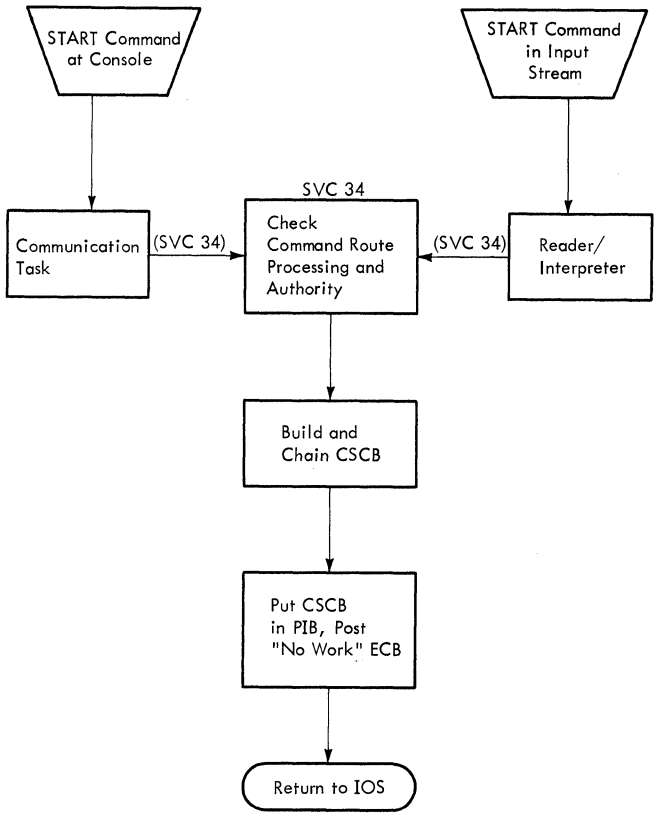



Figure 17. START Command Processing Flow

If the command is a START command, command processor routine IEE3903D builds and chains a CSCB, places the address of the CSCB in the partition's PIB, and posts the partition. If a system reader is to be started, the routine searches for a scheduler-size problem program partition which is inactive; if a system writer is to be started, the routine searches for any inactive problem program partition. If a partition is located, the routine builds and processes a CSCB as stated above. If a partition cannot be found, the routine issues a message to the operator stating that the command has failed. If the command is a STOP INIT command, the routine verifies that the partition contains an initiator and sets the STOP INIT indicator in the partition's PIB.

The section "System Task Control" describes the further processing of the START command CSCB. The processing of a STOP INIT indicator is completed by the Initiator/Terminator.

## WRITE-TO-PROGRAMMER MESSAGE PROCESSING ROUTINES (IEFWTP00, IEFWTP01, AND IEFWTP02)

A write-to-programmer (WTP) message is issued by including a routing code of 11 (ROUTCDE=11) with a WTO or WTOR macro instruction. The WTO routine will identify the macro instruction as containing a WTP message, and pass control via an XCTL macro instruction to write-to-programmer initialization routine IEFWTP00. The initialization routine passes control to message processing routine IEFWTP01, which uses the transient queue manager (SVC 90) to write the message in a system message block in the system message class data set.

If I/O errors are encountered by the transient queue manager, control is passed to error handling routine IEFWTP02.

Upon completion of WTP processing control is returned to the WTO routine for further processing if any one of the following conditions is true; otherwise, control is returned to the program that issued the macro instruction.

- MCS is in the system and a specific console is designated to receive routing code 11 messages, or
- An additional routing code is specified in the macro instruction, or
- The macro instruction is a WTOR.

## SYSTEM INITIALIZATION

The master scheduler task (Chart 09) 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. After the communications task is initialized, 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, and if the system log was requested, passes control to IEEVLIN to initialize the system log. It then 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 jcb queue and procedure library are created. To format the job queue, the routine passes control to queue initialization routine IEFSD055 via a LINK macro instruc-

tion 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.

If the system management facility is specified, the routine stores the SMF options in the first byte of the CVTSMCA field of the CVT. It then passes control via a LINK macro instruction to SMF initialization routine IEESMFIT to initialize the system management facility. (See the "SMF Initialization" section in this publication.)

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 SERVICE ROUTINES

### 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 12) 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 10). 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 CONSOLES command, control is passed to DISPLAY CONSOLES routine IEEXEDNA. If it was a DISPLAY U command, control is passed to DISPLAY U routine IEEUNIT1.

### Queue Search Setup Routine (IEESD563)

If the CANCEL command is being processed, queue search setup routine IEESD563 passes control to queue scratch setup routine IEESD575. Otherwise, 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 setup routine IEESD563.

### 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.

4. If another queue needs to be searched, it passes control to queue search set up routine IEESD563 via an XCTL macro instruction.

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 and, if subtasking is included, a count of the number of subtasks within the job step. The DISPLAY A routine returns control to the router routine.

DISPLAY CONSOLES Routine (IEEXEDNA)

DISPLAY CONSOLES routine IEEXEDNA receives control from the Syntax Check routine IEESD562 when the DISPLAY CONSOLES command is entered. This routine issues a header message that describes the status message. It then constructs and issues a message describing the status of the hard copy log (if one exists) and each console in the system, both active and inactive. When the message is issued, it returns to the Master Scheduler Router routine IEECIR50.

DISPLAY U Routines (IEEUNIT1, IEEUNIT2, IEEUNIT3, IEEUNIT4)

The DISPLAY U routines create a tabular display of unit status, as requested by the DISPLAY U command, based on information in the UCBs. They construct the WTO messages to report on the status of the devices specified by the operands of the DISPLAY U command. DISPLAY U routine (1) IEEUNIT1 receives control from syntax check routine IEESD562 when a DISPLAY U command is entered. When all messages have been issued to the console device, DISPLAY U routine (3) IEEUNIT3 returns control to the Master Scheduler router routine IEECIR50. For a description of each DISPLAY U routine, see the module descriptions in Appendix B of this publication.

Queue Scratch Setup Routine (IEESD575)

Queue scratch setup routine IEESD575 builds the parameter list for the SCRATCH macro instruction (SVC 29) according to whether the canceled job was found on the input or output queue(s). If the job was found on the input queue, IEESD575 determines whether there are SYSIN data sets to be scratched. If not, IEESD575 passes control to queue alter delete routine IEESD576. If the job was found on the input or output queue with data sets to be scratched, IEESD575 passes control to queue scratch routine IEESD581. When IEESD581 has scratched all data sets, IEESD575 passes control to queue alter delete routine IEESD576.

Queue Alter Delete Routine (IEESD576)

Queue alter delete routine IEESD576 passes control to queue manager delete routine IEFQDELE to delete the queue entries associated with the canceled job. For a job on the output queue, with more queues to be searched, control is passed to IEESD563. If the cancel command was issued for an output class other than the message class, control is passed to specific cancel message routine IEESD580, otherwise control is passed to queue message class setup routine IEESD578.

Queue Restart Enqueue Routine IEESD577

Queue restart enqueue routine IEESD577 passes control to the queue manager enqueue routine IEFQMNQQ to enqueue the SYSOUT data sets for canceled restarting jobs. Upon return from IEFQMNQQ, IEFSD577 passes control to IEESD579.

Queue Message Class Setup Routine (IEESD578)

Queue message class setup routine IEESD578 zeroes out the DSBs in the message class and sets up the queue manager parameter area for enqueuing the message class. If the job was a restarting job, IEESD578 passes control to IEESD577. Otherwise, control is passed to queue SMB routine IEESD579.

Queue SMB Routine (IEESD579)

Queue SMB routine IEESD579 places the appropriate cancel message into the first SMB and passes control to the queue manager enqueue routine IEFQMNQQ to enqueue the message class. IEESD579 issues the cancel message to the operator and returns control to master scheduler router routine IEECIR50.

Specific Cancel Message Routine (IEESD580)

Specific cancel message routine IEESD580 issues the cancel message to the operator if the cancel command specified an output

class other than the message class. This routine then returns control to master scheduler router routine IEECIR50.

### Queue Scratch Routine (IEESD581)

Queue scratch routine IEESD581 issues the SCRATCH macro instruction (SVC 29). Upon return from the Scratch service routine, IEESD581 issues a "data set not deleted" message if the return code is nonzero. IEESD581 returns control to queue scratch setup routine IEESD575.

## PARTITION DEFINITION BY THE MASTER SCHEDULER

The master scheduler uses the DEFINE command processing routines (shown in Figure 18) to initialize or change partition definitions in MFT. These routines handle:

- Commands from the operator via a console, issued after nucleus initialization, to change the size and description of any partition while processing continues in unaffected partitions.
- Commands from the system at IPL time to prepare the partition as it was described at system generation.

All transfers of control among the processing routines are accomplished via an XCTL macro instruction.

### 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 scheduler 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, the routine issues a DEFINE COMMAND BEING PROCESSED message to all active consoles. It then determines whether LIST was specified and if so, passes control to listing routine IEEDFIN4. If not, the routine passes control to message routine IEEDFIN5 for issuance of an ENTER DEFINITION message.

If the DEFINE command initialization routine was entered during the system initialization, the routine also issues a DEFINE COMMAND BEING PROCESSED message to all active consoles. It then determines whether partition redefinition or LIST was specified by the operator, and if not, passes control to validity check routine IEEDFIN3. If either LIST or partition redefinition was specified, the routine continues processing as if a DEFINE command had been entered by the operator.

### Syntax Check Routine (IEEDFIN2)

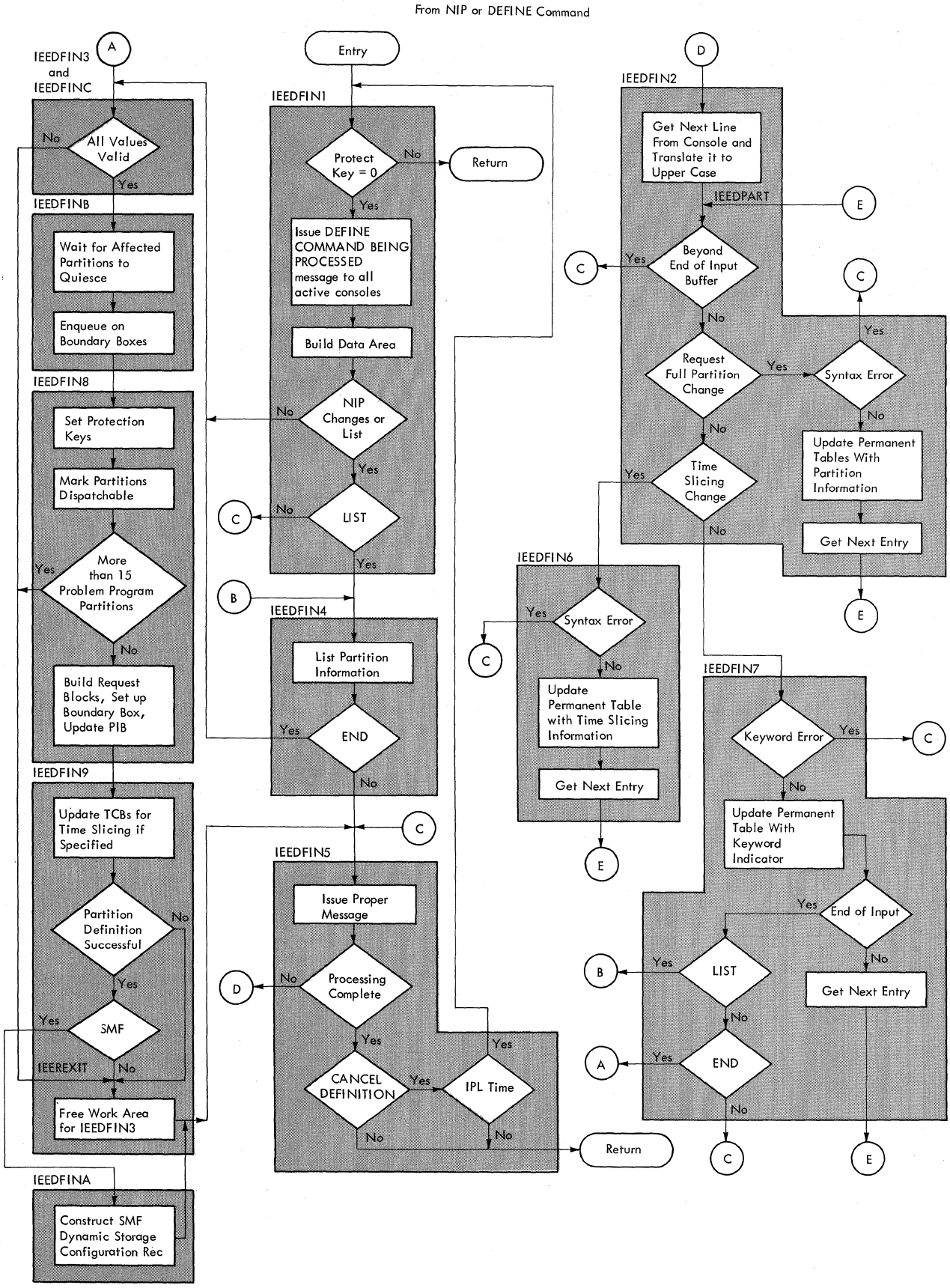
When syntax check routine IEEDFIN2 receives control at primary entry point IEEDFIN2, it translates the statements entered by the operator to upper case. When the routine receives control at secondary entry point IEEDPART, this operation is bypassed.

The statement is scanned and 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 for issuance of the appropriate syntax error message. The erroneous entry and all following entries are ignored. If the syntax is correct, IEEDFIN2 updates the DEFINE data area with the partition information and gets the next entry for processing.

If the entry is a time-slicing change, the routine passes control to time-slice check routine IEEDFIN6.

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.




• Figure 18. DEFINE Command Processing Flow

Validity Check Routine -- Processor Storage (IEEDFIN3)

Validity check routine IEEDFIN3 (for processor storage) makes final checks to determine whether the information entered by the operator is correct (e.g., that the definition changes which have been requested are within legal bounds or that the time-slicing specification is valid). If an error is detected, the routine passes control to IEEREXIT, a secondary entry point in command final processor routine IEEDFIN9. If the information is valid, the routine determines the partitions affected by the DEFINE command constructs a list of PIB pointers (one for each affected active partition) and passes control to validity check routine IEEDFINC.

Validity Check Routine -- Core Storage (IEEDFINC)

Validity check routine IEEDFINC (for core storage) determines whether Main Storage Hierarchy Support is in the system. If it is not, control is passed to system reinitialization routine IEEDFINB. If it is, IEEDFINC determines whether a partition has been defined in two segments. If both H0 and H1 size have been reduced to zero, the routine marks the partition inactive in the DEFINE data area. It also checks to determine if a partition has been specified for excess bytes resulting from a redefinition in either H0 or H1 of an adjacent partition. If no partition has been specified, the routine passes control to secondary entry point IEEREXIT in command final processor routine IEEDFIN9. Otherwise, it sets up a message indicating the number of excess bytes, the partition, and the hierarchy to which they have been added. It then passes control to IEEREXIT.

If the information is valid, IEEDFINC passes control to system reinitialization routine IEEDFINB.

Listing Routine (IEEDFIN4)

Listing routine IEEDFIN4 lists partition definitions and job classes. 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 IEEDFIN3. If not, it passes control to message routine IEEDFIN5.

Message Routine (IEEDFIN5)

Message routine IEEDFIN5 handles the messages required by the DEFINE command processing routines. These messages, which are written to the operator, are concerned with:

- Entering and continuing the definition of partitions.
- Syntax, parameter, and time-slicing errors.
- Illegal number of partitions or oversize partitions.
- Completing the definition of partitions.

After issuing the appropriate message, the routine determines whether processing is complete and if so, issues a DEFINITION COMPLETED message to all active consoles. It then determines if a DEFINITION CANCELLED message has previously been issued and if so, tests to see if the system is being initialized. If the message has been issued and it is IPL time, IEEDFIN5 passes control to command initialization routine IEEDFIN1 to repeat the DEFINE command processing. If the DEFINITION CANCELLED message has not been issued, or if it has been issued at other than IPL time, the routine returns control to the caller.

If processing is not complete, IEEDFIN5 passes control to syntax check routine IEEDFIN2.

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 for issuance of a PARAMETER ERROR message. It ignores the erroneous entry and all following entries. If there are no syntax errors, 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 secondary entry point IEEDPART in syntax check routine IEEDFIN2.

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 for issuance of a PARAMETER ERROR message. It ignores 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 secondary entry point IEEDPART in syntax check routine IEEDFIN2.

If there are no more entries to be processed (end of input), the routine determines whether a LIST keyword has been entered and if so, passes control to listing routine IEEDFIN4. 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. If an END entry is not found, the routine passes control to message routine IEEDFIN5 for issuance of a CONTINUE DEFINITION message.

System Reinitialization Routine 1 (IEEDFIN8)

After the partitions have quiesced, IEEDFIN8 assigns protection keys (if the system is protected) and marks dispatchable partitions not of zero size. It makes one final check to determine that no more than 15 problem program partitions have been defined. If an error is found, the routine passes control to secondary entry point IEEREXIT in command final processor routine IEEDFIN9.

If no error is found, IEEDFIN8 uses the information in the DEFINE data area to build request blocks and boundary boxes and to update the TCBPIB field and the PIB for the defined partition. The routine then passes control to IEEDFIN9 at its primary entry point, IEEDFIN9.

Before passing control to IEEDFIN9 at either entry point, IEEDFIN8 issues the DEQUEUE macro instruction specifying the boundary boxes.

Command Final Processor Routine (IEEDFIN9)

Command final processor routine IEEDFIN9 updates the time-slice control element and the task control blocks affected by time-slicing if this feature is specified.

It then tests to determine if successful partition definition has taken place. If so, it tests the CVTSMCA field of the CVT for the address of the system management control area (SMCA). If this field contains zeroes, one of two possible situations exists:

- SMF is not supported.
- SMF is supported, but has not been completely initialized at this time.

In either of these cases, or if partition definition has not completed successfully, IEEDFIN9 issues a FREEMAIN macro instruction to free the work area previously obtained by IEEDFIN3. It then passes control to IEEDFIN5 for issuance of the appropriate message specified by its caller (IEEDFIN3 or IEEDFIN8).

If the CVTSMCA field contains the address of the SMCA, SMF is supported and its initialization is complete. Partition definition has also completed successfully. Therefore, IEEDFIN9 passes control to IEEDFINA for creation of the SMF storage configuration record (type 13). Upon return from IEEDFINA, the command final processor routine passes control to IEEDFIN5 for issuance of the appropriate message.

MFT Storage Configuration Record Creation Routine (IEEDFINA)

MFT storage configuration record creation routine IEEDFINA creates the SMF storage configuration record (type 13). It receives control from SMF initialization routine IEESMFI2 during SMF initialization, and from command final processor routine IEEDFIN9 whenever a DEFINE command is issued. It creates the SMF storage configuration record and issues an SVC 83 to have it transferred to the SMF buffer. It then returns control to its caller.

System Reinitialization Routine 2 (IEEDFINB)

System reinitialization routine IEEDFINB places the ECB that must be posted by the affected partition in the PIB of the partition. If a partition has been marked inactive (i.e., no H0 or H1 size is contained in the DEFINE data area), IEEDFINB sets the partition's TCB nondispatchable. If any partition being redefined contains a system writer, the routine posts the STOP ECB in the Start Parameter List to stop the writer as if a "Stop Writer" command had been issued from the console. Therefore the operator must issue a "Start Writer" command for any writer partition involved in the redefinition.

The routine then issues the WAIT macro instruction for the posting of the ECB list. After the ECB is posted, IEEDFINB issues the ENQUEUE macro instruction specifying the boundary boxes.

## 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 so long as the total does not exceed 52. 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 by system output writers 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. In addition, each problem program may use direct system output (DSO) processing. DSO is stated by entering a START DSO command for a partition naming a system output class and a device. DSO processing is limited only by the number of available devices. (See the Operator's Guide SRL, GC28-6540).

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

## 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. The discussion of the queue manager includes descriptions of some MVT modules to provide a more complete explanation of the relationship of these modules to the entire system. A discussion of the transient queue manager (SVC 90) is also included.

### WORK QUEUES

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

- Automatic SYSIN blocking 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 extent 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 19 for the format of the master QCR.) Control then passes to queue formatting routine IEFORMAT.

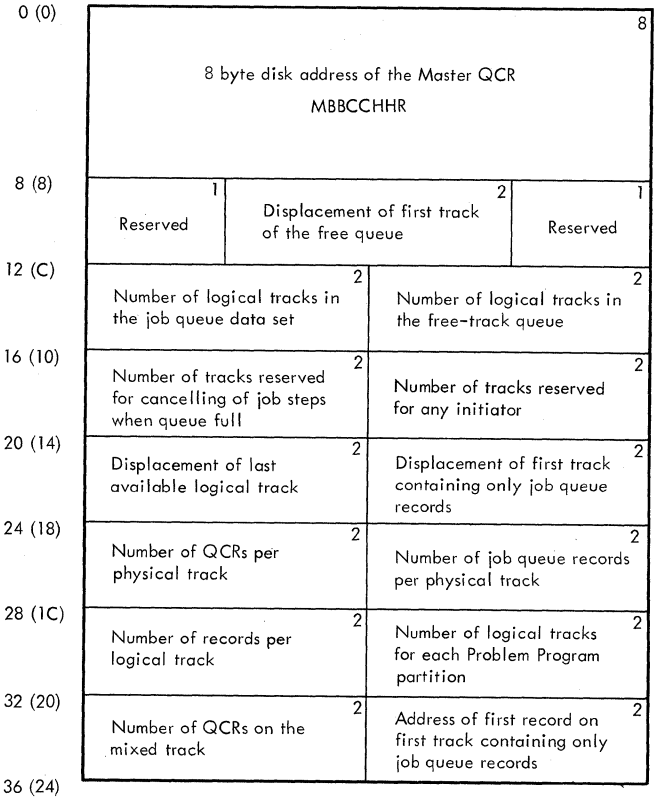



Figure 19. Master Queue Control Record (Master QCR) Format

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 automatic SYSIN batching (ASB) 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 20 for the format of an input queue control record.)

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 (Hexadecimal) |
|---|---|
| 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 (LTH) (as shown in Figure 21) 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 and may range from 10 to 255 records. The number may be modified within this range at IPL. 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 22. Figure 23 illustrates typical input and output work queues. Each input and output QCR contains the address of the last entry in each priority queue.

## 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.

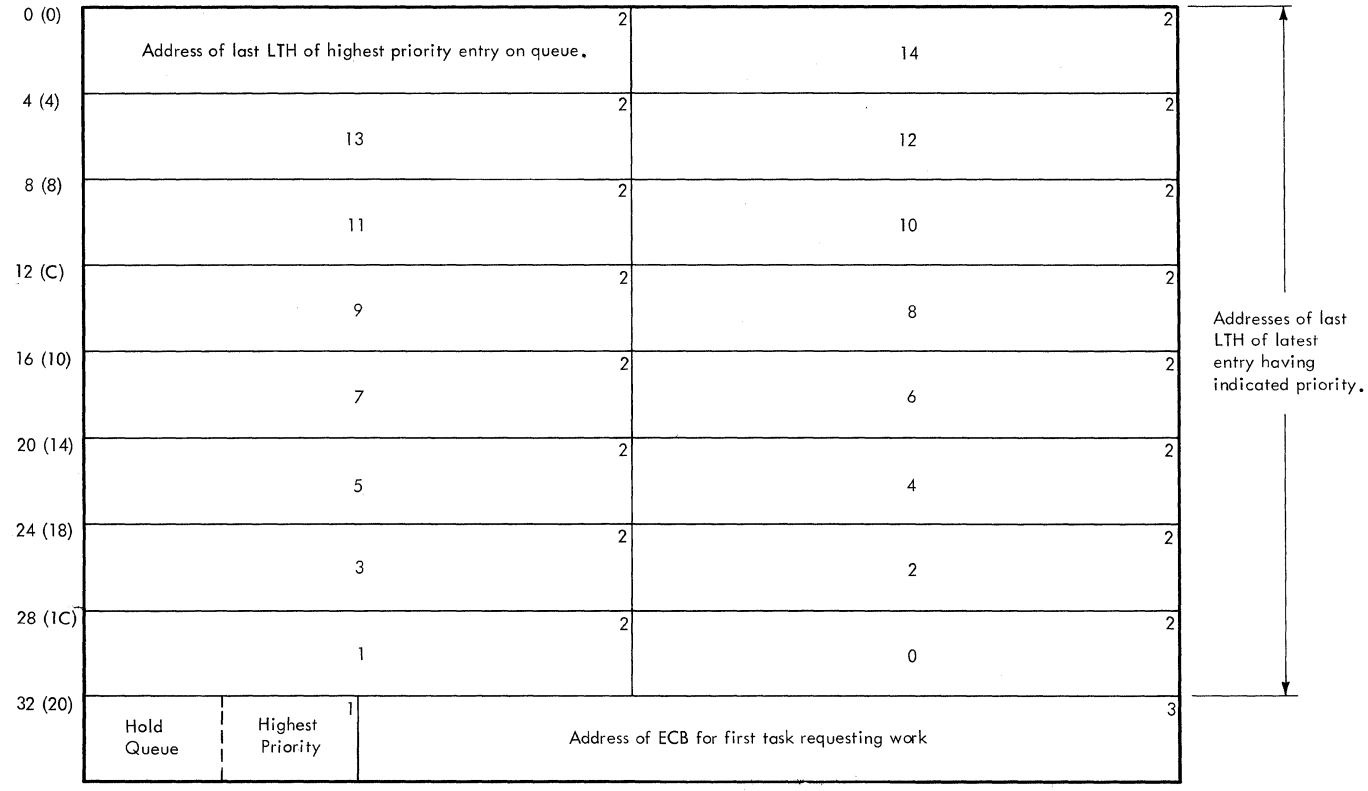
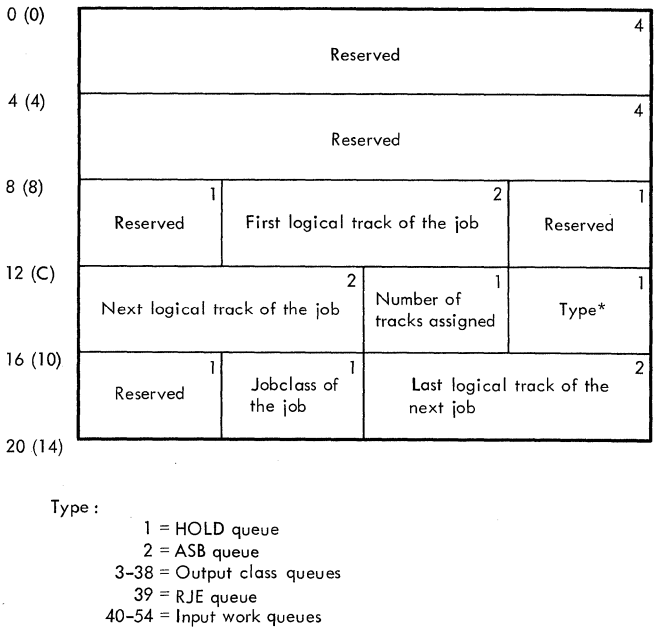



Figure 20. Job Queue Control Record (QCR)




Type:
1 = HOLD queue
2 = ASB queue
3-38 = Output class queues
39 = RJE queue
40-54 = Input work queues

Figure 21. Logical Track Header (LTH) Record Format

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 the MVT Job Management PLM 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 jcb 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 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

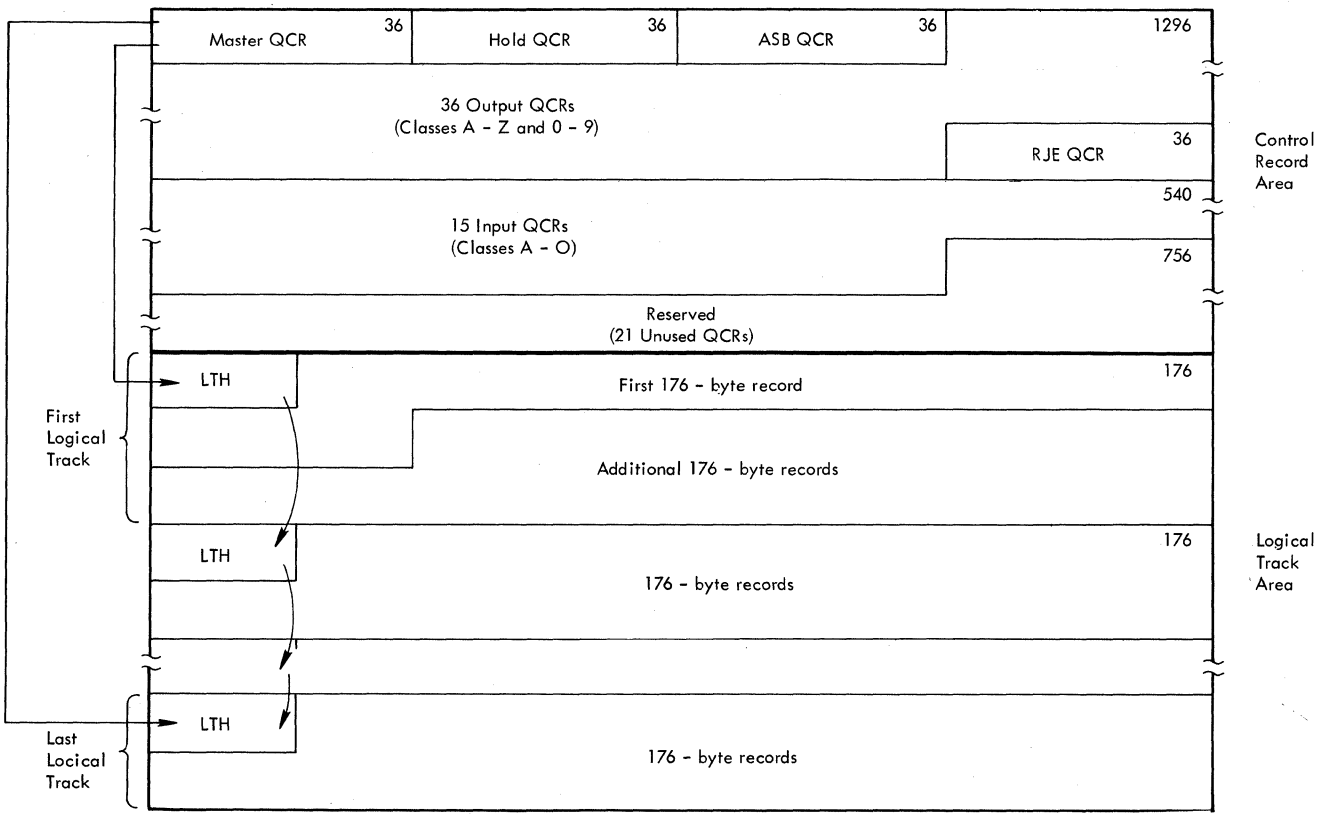



Figure 22. Sample Job Queue (SYS1.SYSJOBQE) Format After Initialization

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, and the requesting routine is a reader/interpreter, the assign routine passes control to queue manager/interpreter interlock routine IEFSD572. If the reader/interpreter is resident, control returns to the assign routine to wait for tracks to become available. If the reader/interpreter is transient, IEFSD572 issues a message to the operator requesting him to reply "WAIT" or "CANCEL". If the reply is WAIT, control returns to the assign routine, otherwise control is passed to the ABEND routines to cancel the reader/interpreter.

If there are no available logical tracks and the requesting routine is an initiator/terminator, the assign routine issues a message to the operator stating that queue space has been exceeded and passes control back to the initiator/terminator to cancel the jcb.

When the requesting routine is assigned the 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 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.

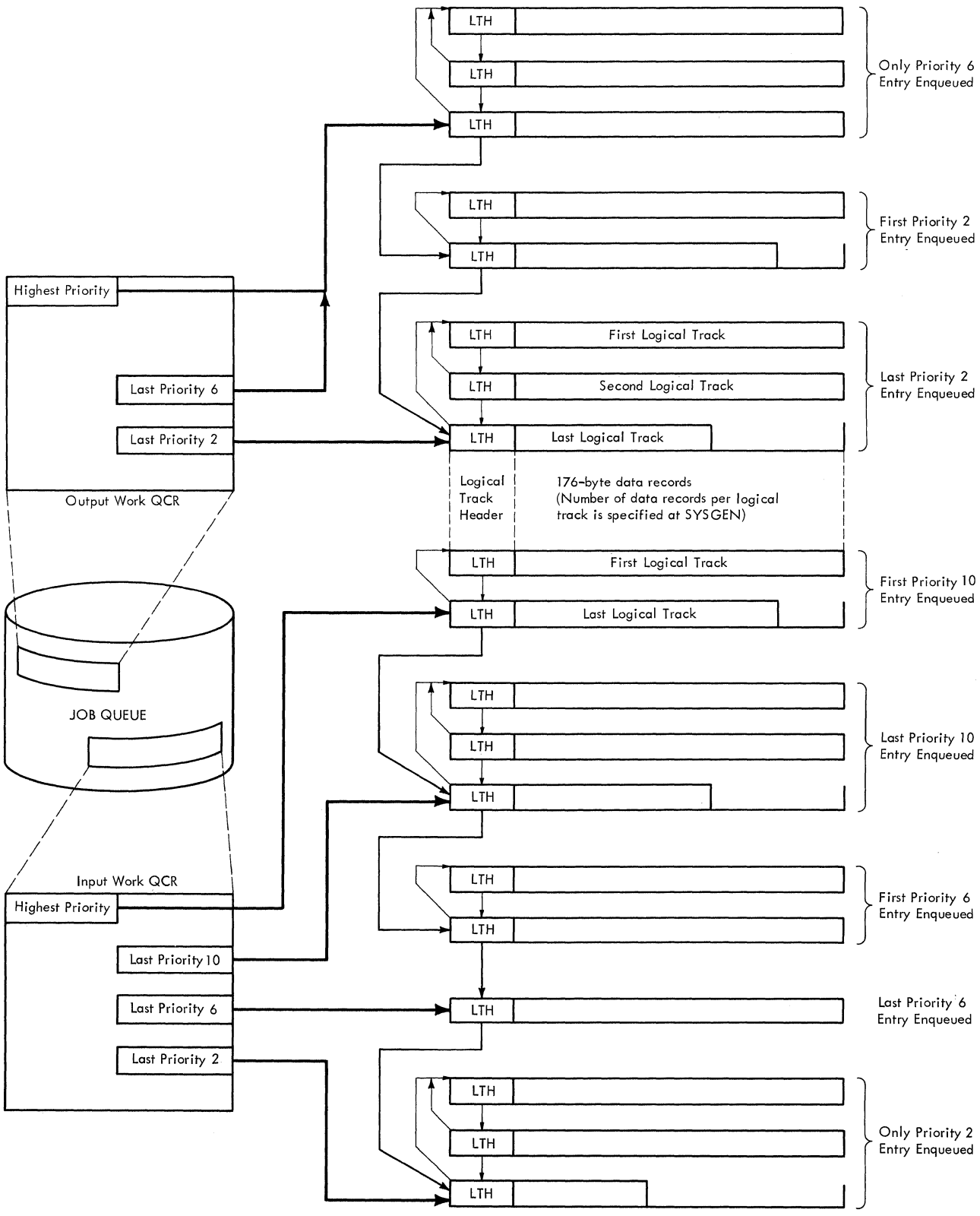



Figure 23. Input and Output Queue Entries

Interpreter/Queue Manager Interlock Routine (IEFSD572)

When the reader/interpreter requests tracks for the job it is processing, and no 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 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 23. 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 24) and place the address of the list in general register 1 before calling the TBR.

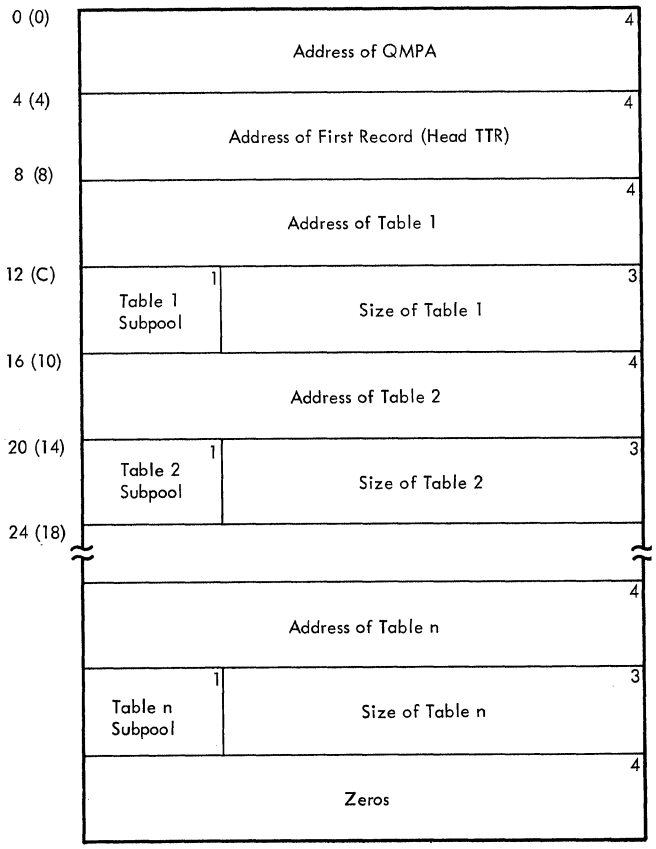



Figure 24. Table Breakup Parameter List

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.

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.

Transient Queue Manager Routines (IEFXQM00, IEFXQM01, and IEFXQM02)

The transient queue manager consists of initialization and read/write routine IEFXQM00, track assignment routine IEFXQM01, and record assignment routine IEFXQM02. These routines provide the services of assign/start routine IEFQAGST, assign routine IEFQASGQ, and read/write routine IEFQMRAW. The transient queue manager is in SYS1.SVCLIB and operates in the transient SVC area.

When the transient queue manager initialization and read/write routine IEFXQM00 receives control it first initializes an ECB/IOB and prepares the QMPA. If the queue manager was requested to provide a track or record, IEFXQM00 branches via an XCTL macro instruction to track assignment routine IEFXQM01 or record assignment routine IEFXQM02. If the queue manager was requested to read or write a record onto the job queue, IEFXQM00 performs the read or write. IEFXQM00 returns control to the caller upon completion, as does IEFXQM01 and IEFXQM02.

## 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. Each reader is assigned to an input device 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 routine IEFSD537 which issues a LINK macro instruction to reader initialization routine IEFVH1 to begin reading the input job stream (Chart 24-26).

When reader initialization routine IEFVH1 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).

For systems that include Multiple Console Support (MCS), the PARM field on an EXEC statement includes a command authority code. This code is included in the option list created by interface routine IEFSD533, and placed in the interpreter work area (IWA) by reader initialization routine IEFVH1. This code is passed by the reader when it issues an SVC 34 due to a command read in the input stream.

After the reader has completed reading a job, control passes to 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 CMPA associated with that input queue entry.

If data is encountered in the input stream, control is passed to 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 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 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 termination routine IEFVHN. If the reader is to be suspended, control passes to transient reader suspend routine IEFSD530. Otherwise, control returns to 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 29) 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 the JCTJMR field of the JCT is tested to determine if SMF is supported. If this field contains zeroes, there is no JMR and SMF is not supported. If SMF is supported, the user's SMF exit routine IEFUJV (whose address is contained in the JMRUJVP field of the JMR) is deleted if it is present in main storage. Storage for the JMR is also freed via the FREEMAIN macro instruction.

The transient reader suspend routine then 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 30) 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.

If SMF is in the system and if SMF options are specified, a GETMAIN macro instruction is issued to obtain main storage for the job management record (JMR). The JMR is then initialized with the SMF options and the RDR device type and name. If SMF exits are specified, the name of SMF user exit routine IEFUJV is placed in the NEL. The routine is then loaded and its address is placed in the JMRUJVP field of the JMR. 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 routine 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 partition, 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. Scheduling 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 jcb 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 for any pending STOP DSO or MODIFY DSO commands. These are processed by passing control to stop and modify command processing routine IEFDSOSM.

Upon return from IEFDSOSM, the PIB is checked to determine if the partition is involved in partition redefinition; if the partition is to be changed, the PIB is checked further. If a job is queued on the checkpoint/restart internal queue it is processed; if a restart reader is pending, it is started. If neither exists, any DSO processing is stopped, no further scheduling is allowed in the partition and the partition can be redefined. (See "Master Scheduler Task." )

If the partition in which the initiator is operating is not part of a partition redefinition, a test is made for a pending Restart Reader command. If no command is pending, a test is made to determine if a system-task reader or writer is to be started. If a restart reader or a system-task reader or writer is to be started, control passes to system task control which initiates readers and writers. If a restart reader is being started, and a user-assigned reader had been rolled out of the partition, the PIB is marked accordingly.

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 25). If a STOP INIT command has been issued, the attempt to dequeue a job is bypassed.

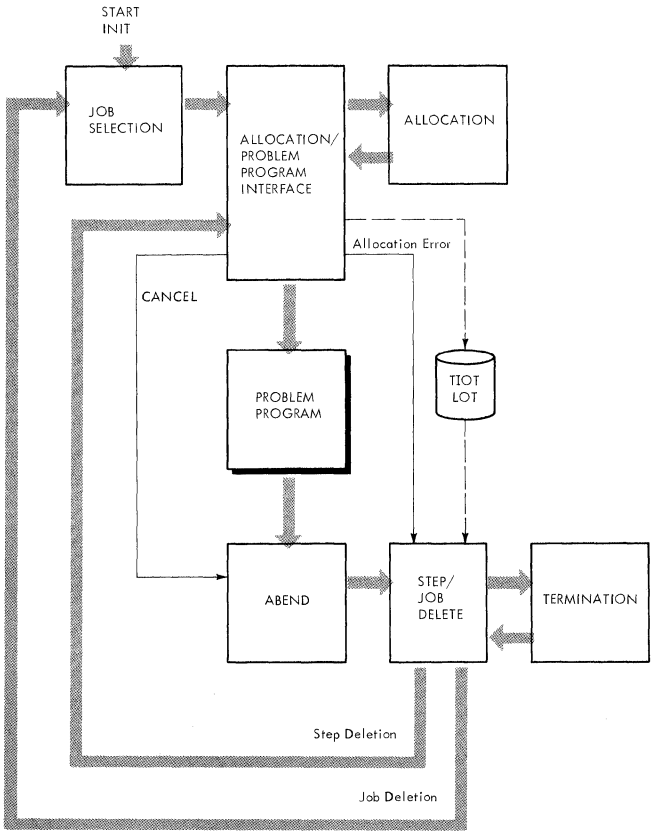



Figure 25. Scheduling a Problem Program in a Large Partition

A threshold check is then made to determine if enough logical tracks are available on SYS1.SYSJOBQE to start the initiator. If not, message IEF427I COMD REJECTED FOR INITIATOR 'ident' - INSUFFICIENT QUEUE SPACE 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 checks to determine if a job is queued on the checkpoint/restart internal queue. If a job exists, dequeue by jobname routine (IEFLOCDQ) is used to remove it from the hold queue for processing. If no job is on the internal queue, the routine 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 all four fields of the timer work area in the LCT 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.

## Command Processing Services

In response to system commands entered in the input stream or from a 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 a 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 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 nonzero 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 26, 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 27, 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 program 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, IEFSD599 determines if job step timing is being performed by testing the high-order bit in the job step timing status bits field of the PIB. If the bit is on, the TQE is being used for job step timing and the routine issues a TTIMER macro instruction to stop the timing and to obtain the step time remaining for use in

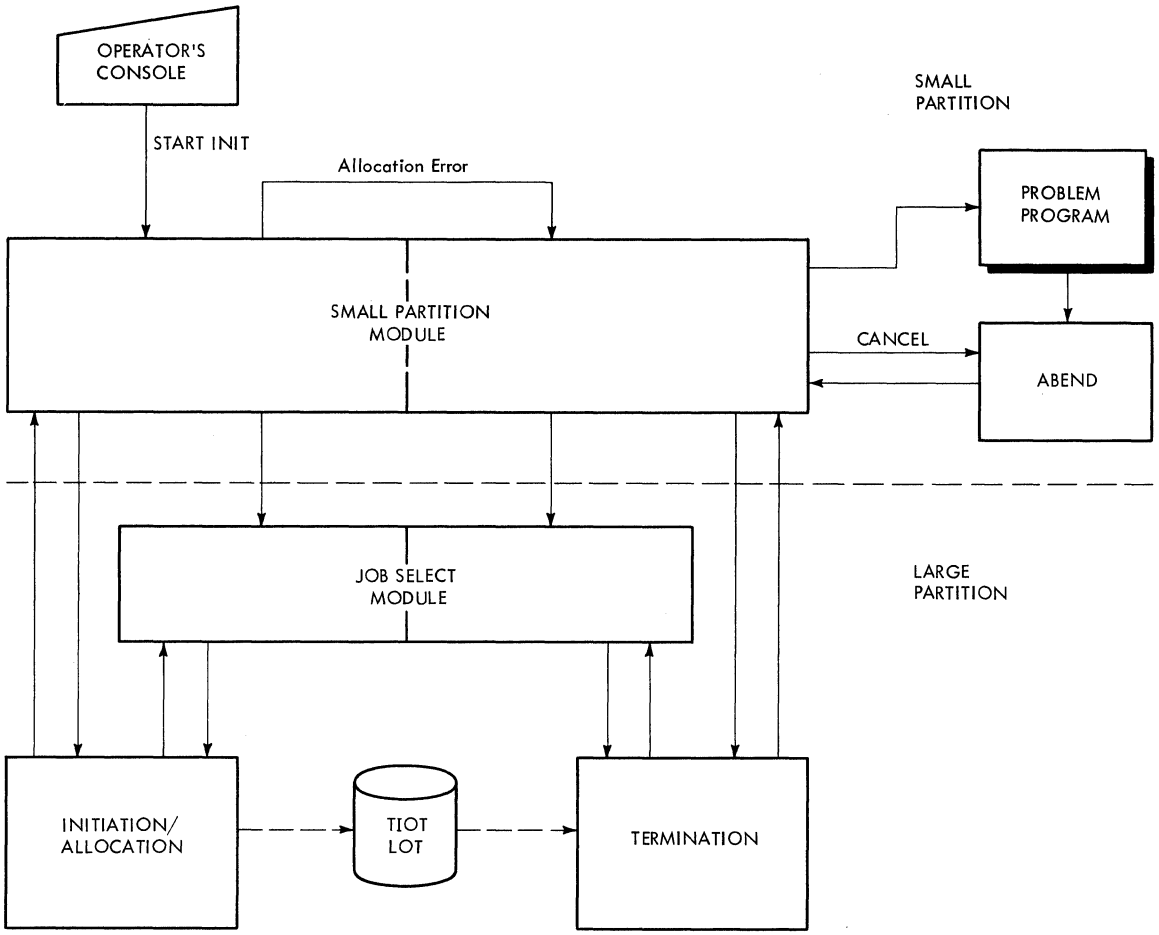



Figure 26. Scheduling a Problem Program in a Small Partition

updating the SPIL. It then turns off the bit and saves the step time remaining in a register until the SPIL is created. When the SPIL is created, the routine updates it with the step time remaining and sets the status bit indicating that termination services are requested. The small partition module then begins a search for a large partition to perform the job termination required.

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 05-08) is entered from the redefinition routines at system initialization or when a DEFINE command is issued or from the master scheduler. The module is entered at special entry point SMALLGO from 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 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 then checked for pending STOP DSO or MODIFY DSO commands. IEFSD599 passes control to stop and modify command processing routine IEFDSOSM to process any such pending DSO commands.

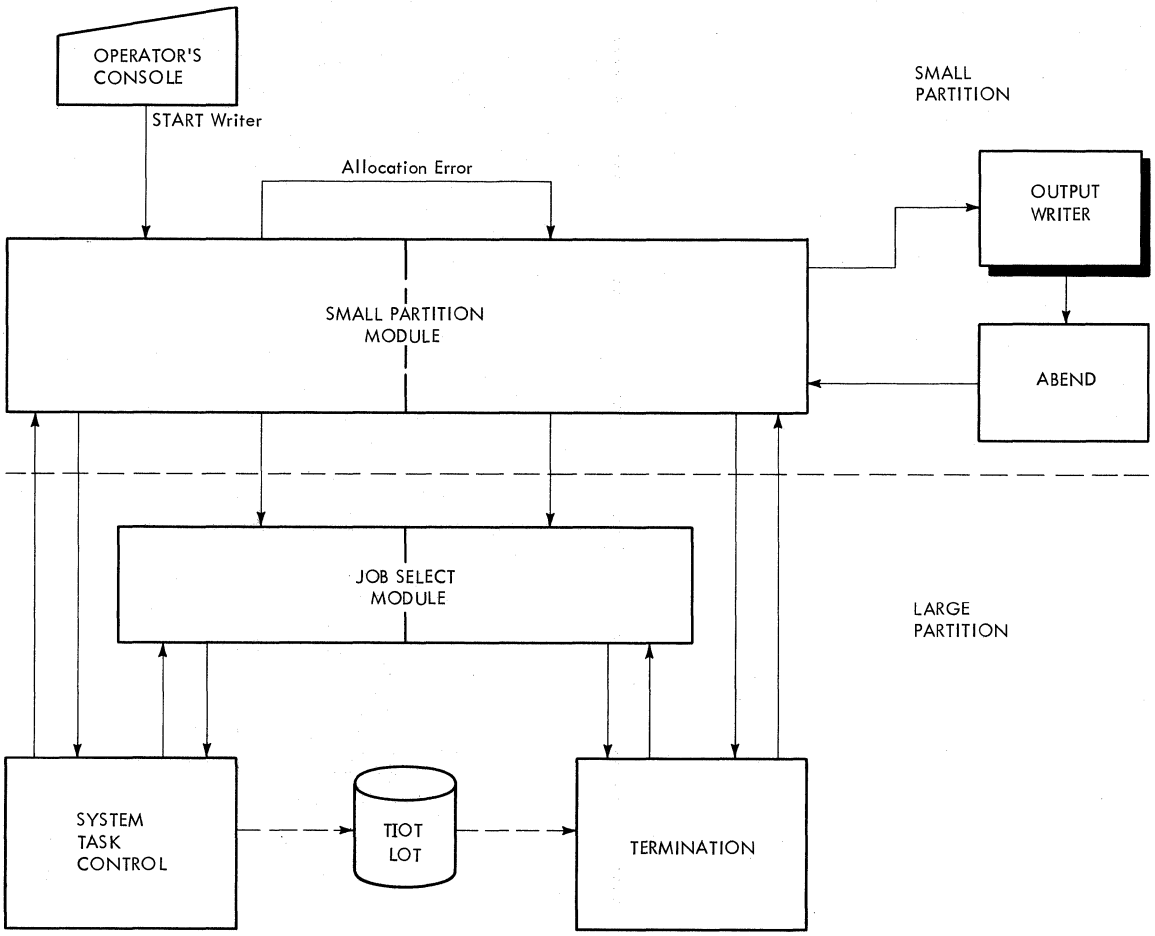



Figure 27. Scheduling a Writer in a Small Partition

Upon return from IEFDSOSM, IEFSD599 checks the PIB to determine if the partition is involved in a redefinition operation. If a redefinition is pending, the internal job queue of checkpoint/restart jobs is checked and any jobs on the queue are processed before the partition redefinition. If there is nothing on the internal job queue and redefinition is pending, assigned tracks are deleted, the SPIL is freed, any DSO processing is stopped, 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), the large partition is posted and the large partition routine clears the SPIL addresses in the other large partition PIBs. 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. To process write-to-programmer messages during problem program execution, IEFSD599 puts the address of the SYSOUT QMPA into the WTPCB, which is located in the CSCB.

The routine then determines if job step timing will be performed by testing the step time limit in the timer work area of the LOT block. If this value is equal to 24 hours, the job step will not be timed. If the job step is to be timed, IEFSD599 issues the STIMER macro instruction to set up the step time interval. The routine then sets bit zero of the job step timing status bits field of the PIB to one indicate that the job step TQE is being used by the Initiator. It also sets bit one to one to indicate to step deletion routine IEFSDS15 that the STIMER macro instruction was issued specifying the TQE addressed in the PIB.

The small partition routine establishes the partition in the problem program protection mode and frees the SPIL. If the program to be initiated is the DSDR processing step of a checkpoint restart, IEFSD599 uses the APL to move the TIOT and user parameter area into the small partition, and posts ECBC. The routine moves the job QMPA and the SYSOUT QMPA from the LOT to the CSCB, and bypasses opening the JOBLIB and FETCH DCBs. The routine also bypasses setting the storage protection key but frees the SPIL.

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 the small partition routine and enters at special entry point SMALLGO. The routine changes the small partition protection key to zero. If job step timing is being performed, it issues the TTIMER macro instruction to stop the timing and to obtain the step time remaining for use in updating the SPIL. It sets bit zero of the job step timing status bits field in the PIB to zero to indicate that the job step TQE is no longer active. After it it creates the SPIL, the routine updates it with the step time remaining and turns on the status bit indicating that termination services are requested. IEFSD599 then begins the search for a large partition to service the request.

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

## INITIATOR/TERMINATOR CONTROL FLOW

There are no terminator routines that are unique to MFT; the modules used in MFT task

termination are described in the MVT Job Management PLM.

In addition to IEFSD510 and IEFSD599, several other initiator routines are unique to MFT. These are described in the following paragraphs. Descriptions of the MVT allocation and step initiation routines that have not been modified by MFT can be found in the MVT Job Management PLM.

## 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.

If direct system output (DSO) processing is available in the partition, job initiation uses the SCD to build a table of all classes of SYSOUT, including the message class, contained in the job stream. Job initiation uses this table to determine if DSO is available for the job; if so, it selects DSOCBs for the job. Selection of a DSOCB is indicated by placing the problem program's protection key into the DSOCB and flagging the job's JCT.

## 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 nonzero setting. If the setting is nonzero, the associated DSNAME (minor name) is identified to the operator as unavailable. The operator is 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 macro instruction to obtain storage for a 72-byte register save area for SMF user initiation exit routine IEFSMFIE and branches to IEFSMFIE. Upon return, it frees the register save area and tests to determine if job step timing will be performed. If the job time limit in the JCT is equal to 24 hours, the job step will not be timed. In this case the step

initiation routine moves the 24 hour limit to the timer work area in the life-of-task (LOT) block, and bypasses the procedure for setting up the step time limit.

If the job time limit in the JCT is equal to any value other than 24 hours, IEFSD512 determines the value to be used as the step time limit in the timer work area of the LOT block. For each step of the job, the routine determines if allowing the step to use the full amount of time specified for it would cause the job time limit to be exceeded: IEFSD512 calculates the amount of job time remaining by subtracting the job time used from the job time limit and compares this figure with the step time limit. It establishes the step time limit by placing the smaller of the two figures in the step time limit field of the timer work area. If the smaller of the two figures is the job time remaining, the routine turns on the high order bit in the step time remaining field of the timer work area to indicate that the job time remaining is being used as the step time limit.

IEFSD512 then 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 28) is initialized containing addresses of the LOT, JCT, and SCT, and two words of zeros.

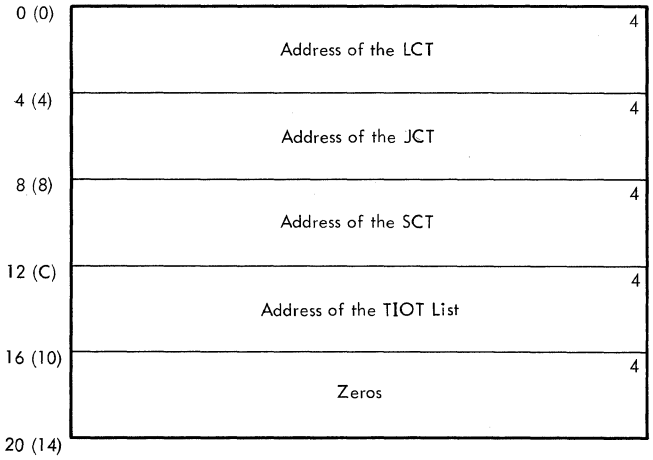



Figure 28. Allocate/Terminate Parameter List

The step initiation routine checks the current step to determine if it is either the checkpoint/restart data set descriptor record (DSDR) processing step or the restart step. If the step is a DSDR processing step being scheduled for a small partition containing less than 12K bytes, the PIB of the partition containing the step initiation routine will be tagged to indicate that the DSDR step is to execute in that partition. The step initiation routine will place the address of its TCB and PIB in the LOT and pass control to allocation via an XCTL macro instruction. If the DSDR step is to be processed in a large partition, normal processing is continued.

If the step is the restart step, the step initiation routine will pass control to partition recovery routine IEFSD518 via a LINK macro instruction. If the return code from IEFSD518 is a zero, normal processing is continued; if the return code from IEFSD518 is a four, the address of the LOT is placed in register 1 and control is passed to job selection IEFSD510 via an XCTL macro instruction.

If the job is using DSO, a message to that effect is placed in the first SMB. Step initiation then passes control to Allocation via an XCTL 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 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). If the job is using DSO, and if job separator and/or system message processing is required, step initiation links to system message and job separator writer routine IEFDSOWR. If IEFDSOWR is unable to process due to a job queue I/O error, the initiator will ABEND with an error code of OBO; if IEFDSOWR is unable to process due to I/O errors, step initiation will set the job failed bit (which is tested later by IEFSD513).

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 LCT 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.

SMF User Initiation Exit Routine (IEFSMFIE)

SMF user initiation exit routine IEFSMFIE receives control from step initiation routine IEFSD512. It first determines if SMF is supported by testing the JCTJMROP field of the JCT for a zero value. A zero value indicates that SMF is not supported. In this case the routine immediately returns control to the caller (IEFSD512). If SMF is supported, the SMF user initiation exit routine performs the following functions:

- It initializes and updates the timing control table (TCT).
- It updates the job log portion of the job management record (JMR).
- It passes control to the user's job initiation exit routine, IEFUJI, or step initiation exit routine, IEFUSI.
- It constructs the SMF Job Commencement Record (type 20).

When it is entered, IEFSMFIE issues the TIME BIN macro instruction and stores the job initiation start time and date in the JCT. It then determines if the step being initated is the first step of the job. If so, it issues a GETMAIN macro instruction specifying the system queue area to obtain main storage for the TCT and for the first 40 bytes of the JMR. The routine initializes the TCT and stores its address in the TCBTCT field of the TCB. It then uses the Queue Management Read/Write routine to bring the JMR into main storage. It copies the first 40 bytes of the JMR into the area reserved for it and updates it with the job initiation start time and date. If user exits are specified, the routine brings the job account control table (ACT) into main storage and then passes control to user job initiation exit routine IEFUJI.

If the step being initated is not the first step of the job, the TCT and JMR are already in main storage. IEFSMFIE stores the step initiation start time and date in the JMR. If user exits are specified, the routine brings the job ACT into main storage and then passes control to user step initiation routine IEFUSI.

Upon return from the user exit routine, IEFSMFIE inspects the return code. If the return code specifies that the job is to be canceled, the routine sets the job-failed bit in the JCT.

For each job, the SMF user initiation exit routine also determines if the data set accounting option is specified by testing the SMCAOPT field in the SMCA. If the option is not specified, or if the job was cancelled, the routine bypasses construction of a Job Commencement Record (type 20). Otherwise, IEFSMFIE builds the record using the accounting information in the job ACT and issues an SVC 83 to have the record transferred to the SMF buffer.

When processing is complete, IEFSMFIE returns control to the caller (IEFSD512).

Note: For the format and description of the JMR and TCT, see "Appendix A" in the MVT Job Management PLM.

Problem Program Interface Routine (IEFSD513)

The problem program interface routine prepares the partition for execution of the job step. It first passes control to SMF TCTIOT construction routine IEFSMFAT. Upon return the routine determines if SMF is supported by testing register 15. A zero value indicates that SMF is not supported and in this case IEFSD513 bypasses the procedures for updating the TCT with the job wait time limit.

If SMF is supported, register 15 contains the address of the TCT and register 0 contains the job wait time limit obtained from the system management control area (SMCA) by IEFSMFAT. In this case IEFSD513 places the job wait time limit in the TCTWLMT field of the TCT. It also initializes bit zero of the TCTSW field to correspond with the bit set in the time remaining field of the timer work area by IEFSD512 indicating whether the job time remaining or the step time limit was established as the time limit for the step about to receive control.

The problem program interface routine then tests to determine if scheduling was performed for a small partition. If so, this routine tests its partition's PIB to determine whether a checkpoint/restart data set descriptor record (DSDR) is to be processed. If the DSDR step is to be processed, the SPIL pointer in the LCT is ignored; otherwise 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 LCT block, which is retained. The address of the LCT 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 job OMPA and the SYSOUT QMPA are moved from the LOT to the CSCB, 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 29) 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.

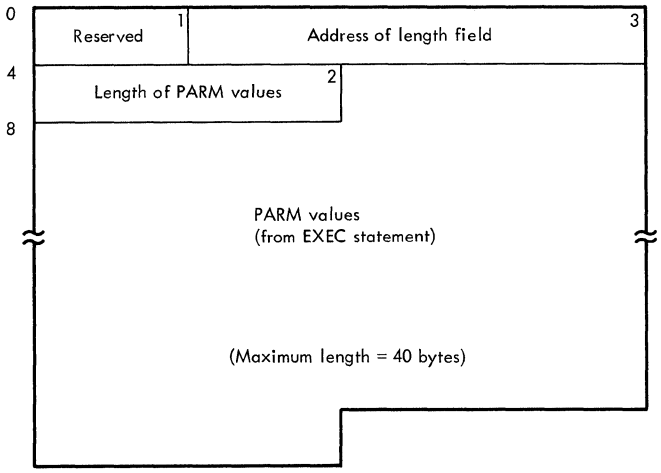



Figure 29. User's Parameter List

If the job being started in the partition is a checkpoint/restart data set descriptor record (DSDR) processing job, the routine bypasses opening the STEPLIB, JOBLIB, and FETCH DCBs and also bypasses setting the storage protection key.

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.

The routine then sets the PSW to the problem program mode. IEFSD513 then tests to determine if job step timing will be performed. If the step time limit in the timer work area of the LOT block is equal to 24 hours, the job step will not be timed. If the step time limit is equal to any value other than a 24 hours, the problem program interface routine issues the STIMER macro instruction to set up the step time interval. It then sets bit zero of the job step timing status bits field in the PIB to one to indicate that the job step TQE is being used by the Initiator. It also sets bit one to one to indicate to step deletion routine IEFSD515 that the STIMER macro instruction was issued specifying the TQE addressed in the PIB.

Whether or not job step timing is performed, IEFSD513 frees main storage for the LOT block, moves the TIOT to the highest available position within the partition, updates the TCB, and passes control to the problem program via an XCTL macro instruction.

## SMF TCTIOT Construction Routine (IEFSMFAT)

If SMF is in the system and if the user accounting option is specified, the SMF TCTIOT construction routine IEFSMFAT builds and initializes a timing control task input/output table (TCTIOT). The routine first determines if SMF is supported by testing the TCBTCT field of the TCB for a zero value. A zero value indicates that SMF is not supported. In this case the routine places a return code of zero in register 15 and returns control to the caller (IEFSD513).

If SMF is in the system, the TCBTCT field contains the address of a TCT built by SMF user initiation exit routine IEFSMFIE. In this case the routine obtains the job wait time limit from the system management control area (SMCA) for return in register 0 to IEFSD513 for updating the TCT. If user exits are specified, IEFSMFAT places the address of SMF user time limit expiration routine IEFUTL in the TCT.

The routine next determines if the user step option is specified by testing the SMCA options field for a X'40'. For any other value the option is not specified and the TCTIOT construction is bypassed. If the user step option is specified, IEFSMFAT constructs a TCTIOT to contain the information necessary for the SMF termination record. The routine issues a GETMAIN macro instruction specifying the system queue area to obtain storage for the TCTIOT and initializes the TCT EXCP counter lookup table.

Whether or not the routine constructed a TCTIOT, it initializes the TCT core map for both hierarchies (0 and 1). It utilizes the boundary box describing the partition to determine the lowest addresses allocated at the high end of hierarchies 0 and 1, and the highest addresses allocated at the low end of hierarchies 0 and 1. It then calculates the amount of storage unused and stores these figures in the TCT.

Finally, IEFSMFAT places the TCT address in register 15 and returns control to the caller (IEFSD513).

Note: For the format and description of the SMCA and the TCTIOT, see "Appendix A" in the MVT Job Management PLM.

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 stating that the scheduler has ABENDed is issued to the operator and all CSCBs are removed from the CSCB chain. DSO processing, if any, in the partition is marked for stopping. Control passes to job selection routine IEFSD510 which passes control to DSO stop and modify command processing routine IEFDSOSM.

If the scheduler ABENDs again while trying to stop DSO, the DSOCB will be marked as being no longer available for selection. The DSOCB I/O device will remain allocated to DSO, and the device will not be available until the system is reinitialized.

If an ABEND did not occur, the step deletion routine prepares to calculate the amount of time used by the step and the job when the last step completed execution. It determines if job step timing is being performed by testing the high-order bit in the job step timing status bits field of the PIB. If the bit is off, the following processing is bypassed. If it is on, the TQE is being used for job step timing and IEFSD515 issues a TTIMER macro instruction to stop the timing started by problem program interface routine IEFSD513. It also obtains the step time remaining for use in updating the timer work area when the LOT block is read back in. It then turns off the high-order bit in the job step timing status bits field of the PIB.

Whether or not job step timing is being performed, 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.

IEFSD515 updates the step time remaining field of the timer work area in the LOT block with the step time remaining value obtained from the TTIMER macro instruction. It restores the addresses in the TIOT and LOT blocks, and frees the temporary work areas.

It returns the job QMPA and the SYSOUT QMPA to the LOT block from the CSCB to reflect any activity that occurred during problem program execution.

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. The JCT is updated with the address of the next SCT and written back on the job queue.

A test is made to determine if job step timing is being performed. If the step time limit in the timer work area is equal to 24 hours and if bit one of the job step timing status bits field in the PIB is set to zero, neither the job nor the step has been timed and the routine bypasses the following processing and obtains storage for the terminate register save area and parameter list. If the step has been timed, the values in the timer work area must be updated to reflect the time used by the step that just completed execution. If SMF is supported (determined by a nonzero value in the TCBTCT field of the TCB), the time extension specified is calculated and added to the step time limit.

The information in the timer work area is then used to calculate the new values for job time used, job time remaining, step time used, and step time remaining, and the timer work area is updated with these calculations.

The Queue Management Read/Write routine is used to read in the job and step ACTs. They are updated with the new values for the jcb and step time, and then written back out.

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 reinitializes the WTPCB for the next step of the job or the next job. If the partition was executing the DSDR step for a small partition, step deletion places the addresses of the small partition's TCB and PIB into the LOT block. Step deletion then checks the return code from termination.

If the return code indicates that the job is to be suspended, step deletion loads the address of the LOT block in register 1. In MFT systems with the 44K scheduler, step deletion then passes control to IEFSD168 via a BALR instruction. In MFT systems with a 30K scheduler, however, step deletion branches to linkage routine IEFSD167 to pass control to IEFSD168 via an XCTL macro instruction. If the return code indicates that job termination was entered, step deletion branches to job deletion routine IEFSD517 and, in MFT systems with the 44K scheduler, receives control again. In MFT systems with the 30K scheduler, however, control does not return to step deletion. It is passed immediately to 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. When the routine is entered at this point, it performs the following functions before invoking ENQ/DEQ purge routine IEFSD598. It obtains the step time remaining for the step which executed in the small partition from the SPIL and saves this value for updating the step time remaining field of the timer work area in the LOT block, when the block is read back in. IEFSD515 also establishes pointers to the SPIL and the small partition's TCB.

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.

Partition Recovery Routine (IEFSD518)

Partition recovery routine IEFSD518 determines the location of main storage required for a checkpoint restart. If the partition being scheduled for the job to be restarted contains the required main storage, the JCT is checked to determine if the job used DSO. If it did, the job's SIOTs are checked to determine which types of I/O devices were used. If any needed type is not available, a message informing the

operator of the missing devices is sent and the job is placed on the hold queue. If all devices are available, the routine returns to the step initiation routine for normal processing. If the nucleus has expanded past the lower boundary of the partition containing the required main storage, the routine sets the job fail bit in the JCT, issues a message stating that main storage is not available for the job, and returns to the step initiation routine IEFSD512 with a return code of zero.

If the partition being scheduled does not contain the required main storage, the routine places the job on the hold queue, updates the SCD and places the SCD back on the job queue. The job's CSCB is unchained and the space containing the CSCB and the ECB/IOBs is freed. If the job used DSO, the routine links to release DSOCB routine IEFDSOFB to release any DSO processor allocated to the job. The routine then branches to ENQ/DEQ purge routine IEFSD598.

Upon return from ENQ/DEQ purge routine, if a problem program partition exists that contains the required main storage, this routine will create an internal queue element and chain it to the partition's PIB. The partition's "no work" ECB will be posted and a message will be issued stating that the job will start in the partition. If an existing partition contains the required main storage and is defined as a reader or writer partition, this routine issues a message indicating that the partition must be redefined to accept the desired jobclass. If no partition contains the required main storage or the partition that contains the required main storage is about to be redefined, this routine issues a message stating the length and displacement of the required main storage. If the partition being scheduled was a large partition its no-work ECB is posted; if it was a small partition, the SPIL is posted indicating job termination. The partition recovery routine frees the JCT and SCT areas of the partition and returns control to step initiation routine IEFSD512 with a return code of four.

### Dequeue by Jobname Interface Routine (IEFSD519)

Dequeue by jobname interface routine (IEFSD519) builds a parameter list used by dequeue by jobname routine IEFLOCDQ to locate a job named on the checkpoint/restart internal job queue. When a checkpoint/restart job is indicated by an entry in the internal job queue pointer in the PIB being processed by job selection routine IEFSD510, job selection branches to IEFSD519 which builds the seven-word parameter list required by IEFLOCDQ. When the job is dequeued, IEFLOCDQ returns control to IEFSD519.

The interface routine marks the job as ready and returns to job selection with a code of zero in register 15, indicating that the job has been found, and a pointer to the LOT in register 1. If the job is not found by IEFLOCDQ, a return code of four is returned in register 15 to job selection. (A description of IEFLOCDQ is in the MVT Job Management PLM.)

## System Output Writers

MFT uses the MVT system output writer (Charts 31-32) 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 nonresident. A resident writer operates in its own partition. A nonresident writer operates in any problem program partition large enough to accommodate it.

### RESIDENT WRITERS

Resident output writer partitions are designated in the TCB by a setting of '10' in the first two bits of the pointer to the partition information block (PIB). This designation is made at system generation by assigning W to the partition in place of the jcb 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.

### NONRESIDENT WRITERS

A nonresident 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 nonresident writer.

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 - Data Set Block (DSB) Handler Routine.
- IEFSD087 - Standard Writer Routine.

Descriptions of all other system output writer modules can be found in the MVT Job Management PLM.

### 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.

## DIRECT SYSTEM OUTPUT PROCESSING

Direct system output (DSO) processing operates in MFT in the same manner as in MVT. The main difference between DSO in MFT and MVT is that DSO started in an MFT partition can only process output from jobs

within that partition whereas DSO started in an MVT system is not restricted by partition boundary.

## System Task Control

System task control (STC) (Chart 33) 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 30, 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.

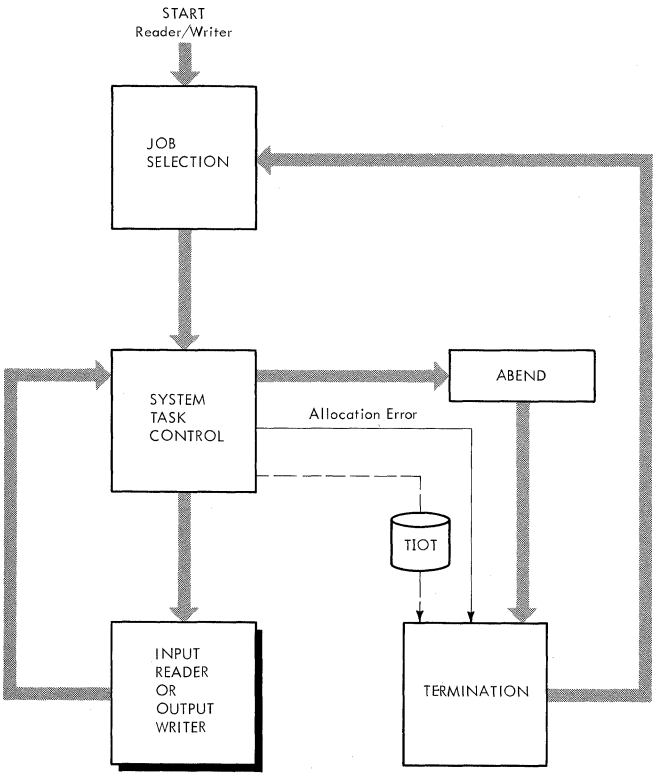



Figure 30. Scheduling a Writer in a Large Partition

The job description information for a reader or writer comes from three sources: the procedure library, Job Control Language (JCL) statements, and the operator. The procedure library contains standard descriptions of a reader and 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 routine IEESD510 determines that a START command for a reader or writer has been entered, it passes control to START syntax check routine IEEVSTAR via an XCTL macro instruction.

### START Syntax Check Routine (IEEVSTAR)

The START syntax check routine gets main storage for, and builds, the start descriptor table (SDT) (see Figure 31). Seven entries are provided in the SDT: the first contains the JOB statement, the second contains the EXEC statement that calls the procedure specified in the START command, the remaining entries are provided for a DD statement and continuations of the EXEC and DD statements. Each entry contains a one-byte identification flags field, whose bits, when set to one, have the following meanings:

- Bit 0 indicates a JOB statement.
- Bit 1 indicates an EXEC statement.
- Bit 2 indicates a DD statement.
- Bit 3 indicates a DD statement continuation.
- Bit 4 indicates an EXEC statement continuation.
- Bits 5 through 7 are reserved.

The routine generates the JOB, EXEC, and DD statements that are placed in the SDT. The keyword parameters in the START command are compared with a list of keyword parameters that are allowable in a DD statement; they are not compared with DD subparameters. If the keyword corresponds to a member of the list, the routine stores it in the DD statement in the SDT. This DD statement overrides the IEFRDER DD statement in the procedure specified in the START command. If the keyword does not correspond to a member of the list, it is assumed to be a symbolic parameter keyword and is placed in the EXEC statement in the SDT. The routine then constructs the

required allocate parameter list, and a JSCB and WTPCB in subpool 255 (SQA). It then stores the JSCB address in the TCB and CSCB.

Finally, the Syntax Check routine passes control to the JCL Edit routine (module IEEVJCL), which builds the job control language set (JCLS). Using the information in the SDT, the JCL Edit routine puts the JCL in the form appropriate for the interpreter. Each statement is built in an 88-character buffer (obtained with a GETMAIN macro instruction). A pointer to the first buffer is placed in the CSCB associated with the START command. Each buffer contains a pointer to the next buffer, 4 bytes of reserved space, and a "card image" of the statement in the last 80 bytes.

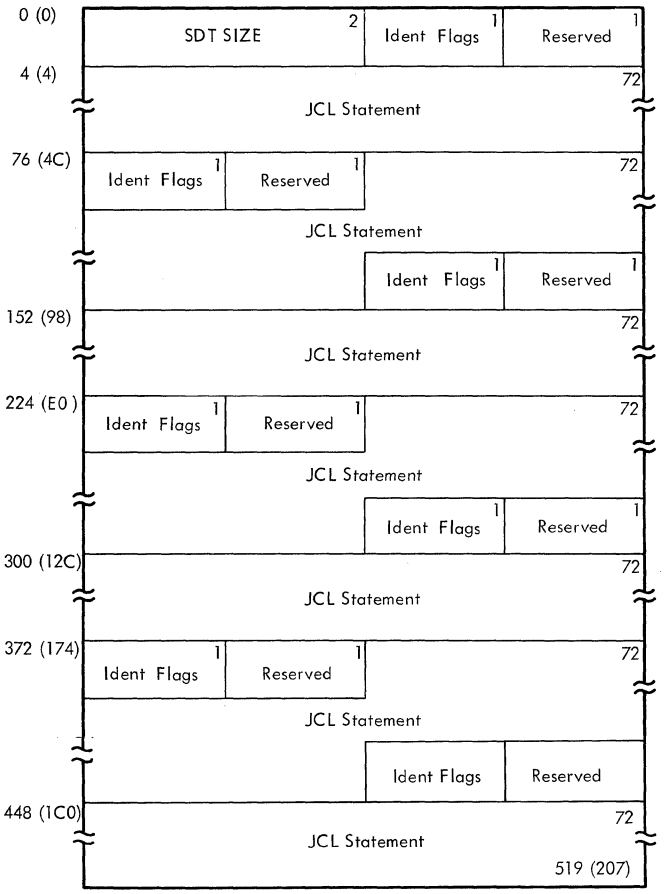



Figure 31. START Descriptor Table (SDT)

## 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 nonzero 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 message interface routine IEEVMSG1 to set up the necessary parameter list for IEEVSMSG to issue the WTO macro instruction to inform the operator of any errors that have been found. Finally it 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 routine to be given control is an authorized routine and

then transfers control to Write TIOT routine IEESD590.

Note: A list of "authorized" routines is contained in a table in link-table routine IEEVLNKT.

QMPA Builder Routine (IEEVSMBA)

The QMPA builder routine obtains main storage for and builds:

- The message class queue manager parameter area (QMPA).
- Its associated ECB/IOB.

It uses the Queue Management Read/Write Routine to read in the SYSOUT class directory (SCD) for the task from the job queue. It uses information in the SCD to initialize the QMPA. IEEVSMBA then builds the ECB/IOB for the QMPA and stores the address of the QMPA in the LCT.

The QMPA builder routine then returns control to its caller.

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.

Termination Interface Control Routine (IEEVTCTL)

The termination interface control routine sets up the necessary tables and parameters for termination processing and passes control to termination entry routine IEFSD42Q via a LINK macro instruction. Upon return from termination, it frees main storage used for the various tables and passes control to POST routine IEESD592 via an XCTL macro instruction.

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 the MVT Job Management PLM.

## System Management Facility

The system management facility is an optional feature of the control program that provides a means for gathering and recording the type of information that can be used to evaluate system usage. It consists of routines that collect data, routines that record the collected data in a data set, and routines that provide interfaces for exits to user-supplied data collection routines. The system management facility may include either one tape data set (SYS1.MANX) or two direct access data sets (SYS1.MANX and SYS1.MANY). In the latter case, the two data sets are filled alternately: while one is in use, the other may be dumped via the SMF dump routine (IFASMFDP).

SMF data collection routines and exits for user-supplied data collection routines are in the Supervisor, the Interpreter, the Initiator, and the Termination routines. IBM-supplied data collection routines are in the Supervisor, the System Output Writer and the Vary Command Processor.

The routines that record the data in the SMF data set include the SMF SVC routines (SVC 83) and the SMF writer routine (IEESMFWT).

COMPARISON OF SMF IN MFT AND MVT

The areas of SMF processing that are the same in MFT and MVT include:

- The SMF SVC routines (SVC 83).
- The SMF dump routine.
- The SMF processing in the Interpreter, the System Output Writer, the VARY Command Processor, and the Termination routines.

This processing is described in the MVT Job Management PLM.

Additional SMF processing in MFT differs from MVT only in the manner in which it is implemented. The areas of difference in implementation occur in:

- The SMF initialization procedure.
- The SMF writer routine.
- The SMF processing in the Supervisor and the Initiator.

SMF processing in the Supervisor and the Initiator is described in the "Supervisor" and "Job Management" sections of this publication. The SMF writer routine and the processing for SMF initialization is described below.

The SMF records are the same for MFT and MVT, except for the addition of an SMF storage configuration record (type 13) in MFT. A description of the SMF records, showing the data elements that occur in each record type and the source from which each data element is obtained, is found in the "Common Elements of Job Management" section for the MVT Job Management PLM.

SMF Initialization

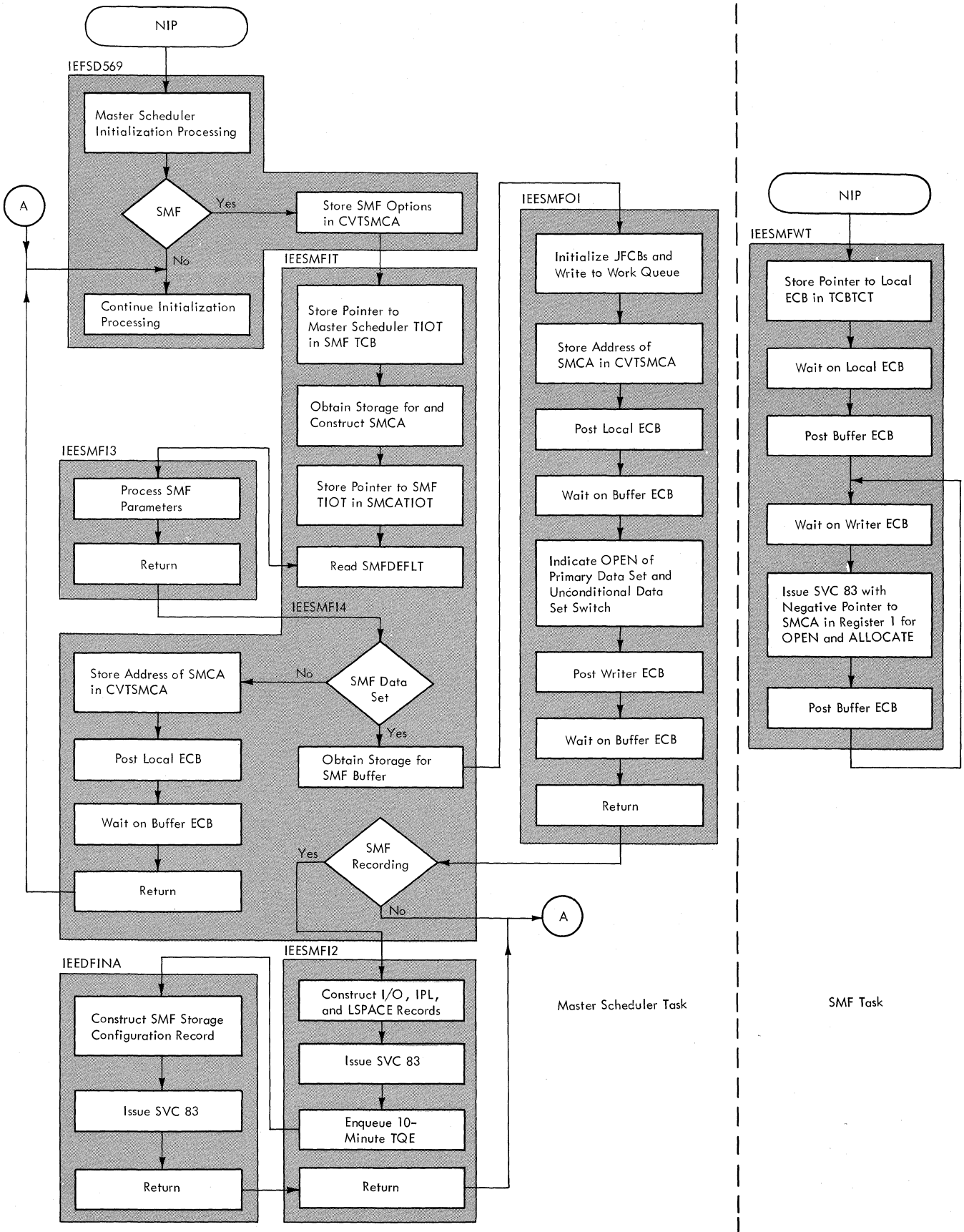
In MFT the routines initializing the system management facility operate under the SMF TCB and the master scheduler TCB. (See Figure 32.) SMF writer routine IEESMFWT executes under the SMF TCB. It is initially involved in a WAIT/POST interchange with the SMF initialization routines operating under the master scheduler TCB. (This is to ensure that the system management control area (SMCA) is initialized and that IEESMFWT has access to it before the writer routine begins initialization processing.) The SMF writer routine's primary initialization function is to issue an SVC 83 for the opening and allocation of the SMF data sets.

The other SMF initialization routines execute under the master scheduler TCB. SMF initialization routine IEESMFIT adds the SMF DD names to the master scheduler task input/output table (TIOT) and reads the SMF parameter member SMFDEFLT from SYS1.PARMLIB. SMF parameter processing routine IEESMFI3 processes the SMFDEFLT parameters and/or the SMF parameters entered from the console. SMF initialization routine IEESMFI2 initializes the 10-minute timer queue element (TQE) and constructs the SMF IPL and online device records. SMF open initialization routine IEESMFOI initializes the job file control blocks (JFCBs) for the SMF data sets and posts the SMF writer routine for the opening and allocation of these data sets. SMF storage configuration record creation routine IEEDFINA constructs the SMF storage configuration record (type 13).

When the SMF writer routine is dispatched, it stores the address of a local ECB in the TCBTCT field of the SMF TCB. (This local ECB provides communication between the master scheduler and SMF tasks. It is posted by IEESMFIT or by IEESMFOI after the address of the SMCA has been stored in the CVTSMCA field of the CVT.) The SMF writer routine then issues a WAIT macro instruction specifying the local ECB.

SMF initialization routine IEESMFIT receives control from master scheduler initialization routine IEFSD569 via a LINK macro instruction. When it is entered, it adds the compiled-in DD names SMFMANX and SMFMANY to the master scheduler TIOT and stores the address of the master scheduler TIOT in the SMCATIOT field of the SMCA. (For the format and description of the SMCA, see "Appendix A" in the MVT Job Management PLM.)

The SMF initialization routine then checks SYS1.PARMLIB for the existence of the SMF parameter member, SMFDEFLT. If it does not exist, the routine passes control to IEESMFI3 at entry point IEESMFMS. Otherwise, it opens the SYS1.PARMLIB data set and reads SMFDEFLT into main storage. If an I/O error occurs during the reading of the data set, IEESMFIT passes control to IEESMFI3 at entry point IEESMFI0.




• **Figure 32. SMF Initialization Processing Flow**

When it has completed reading in the SMF parameter member, IEESMFIT passes control to IEESMFI3 to determine if all of the parameters have been entered correctly. If so, IEESMFI3 stores the SMF parameter values in the SMCA. If any required parameters are missing or incorrectly specified, or if IEESMFI3 was entered at IEESMFMS or IEESMFIO, the routine issues a WTOR macro instruction to request the operator to enter the SMF parameters to make the appropriate corrections. When he has entered the parameters correctly, IEESMFI3 stores the values in the SMCA. The routine then issues a WTO macro instruction to display the parameters. It then inspects the parameter associated with the OPI keyword. If the parameter is "YES", operator intervention is allowed. IEESMFI3 therefore issues a WTOR macro instruction to allow the operator to make changes. When the changes, if any, have been stored, IEESMFI3 returns control to IEESMFIT at entry point IEESMFI4.

IEESMFIT then determines if an SMF data set is specified by testing the SMCAMAN bit in the SMCAMISC field of the SMCA. If the bit is off, an SMF data set is not specified. In this case, IEESMFIT prepares to return to the master scheduler initialization routine, because SMF recording will not be possible. It stores the address of the SMCA in the CVTSMCA field of the CVT. It then issues a POST macro instruction specifying the local ECB to indicate to the SMF writer routine that the SMCA is initialized and its address stored in the CVT. IEESMFIT then issues a WAIT macro instruction specifying the buffer ECB (SMCABECB).

When the local ECB is posted the SMF writer routine has access to the SMCA. It, therefore issues a POST macro instruction specifying the buffer ECB, to indicate that the writer is ready and waiting for work. The SMF writer routine will now wait on the writer ECB (SMCAWECB).

When the buffer ECB is posted, the SMF initialization routine is activated. It returns control to master scheduler initialization routine IEFSD569.

If an SMF data set is specified, IEESMFIT continues with initialization processing. It issues a GETMAIN macro instruction specifying the system queue area to obtain main storage for the SMF buffer. It then passes control to SMF open initialization routine IEESFMOI via a LINK macro instruction.

The SMF open initializer prepares to have the SMF data sets opened and allocated. It compiles the JFCBs for the data sets (SYS1.MANX and SYS1.MANY) and uses the Queue Management Assign/Start and Read/Write routines to obtain space in the work queue data set and to write the JFCBs.[1] It stores the JFCB addresses in the master scheduler TIOT and stores the address of the SMCA in the CVTSMCA field of the CVT. It indicates completion of this action to IEESMFWT by issuing a POST macro instruction specifying the local ECB. It then issues a WAIT macro instruction specifying the buffer ECB.

When the local ECB is posted the SMF writer routine has access to the SMCA. It therefore issues a POST macro instruction specifying the buffer ECB, to indicate that the writer is ready and waiting for work. The SMF writer routine stores pointers to the DCBs for the SMF data sets in the SMCA and then waits on the writer ECB.

When the buffer ECB is posted the SMF open initialization routine is activated. It sets the "first time switch" (SMCAFIRT) in the SMCA miscellaneous indicators field to specify that the primary data set is to be opened. It also sets the "unconditional switch bit" in the SMCASWA switches field to specify an unconditional data set switch. (This bit is set to indicate to the SMF writer routine that register 1 must be negative when the SVC 83 for opening and allocation of the SMF data sets is issued.) IEESMFOI then issues a POST macro instruction specifying the writer ECB to activate the SMF writer routine.

When the SMF writer routine determines that the "unconditional data set switch bit" is set, it loads register 1 with the address of the SMCA. It then sets register 1 negative and issues an SVC 83 for the opening and allocation of the SMF data sets. (The SVC processing for SMF initialization is described in the "Initialization and Restart" section of the MVT Job Management PLM.)

Upon return from the SMF SVC routines, the SMF writer routine issues a POST macro instruction specifying the buffer ECB, to indicate to IEESMFOI that opening and allocation are completed. It will now wait on the writer ECB until its services are again requested.

---

[1]The JFCB records appear as an incomplete queue entry. In the event of a system restart, the system restart routines return the space occupied by the JFCB records to the free-track queue, and the SMF open initialization routine replaces them when it is executed again.

When the buffer ECB is posted, IEESMFOI is activated and it returns control to IEESMFIT. The SMF initialization routine tests the SMCA miscellaneous indicators field to determine if SMF recording will be performed. If not, IEESMFIT returns control to master scheduler initialization routine IEFSD569.

If SMF recording is implemented, IEESMFIT passes control to SMF initialization routine IEESMFI2. This routine constructs the SMF IPL record (type 0) and the SMF online devices records (type 8). If volume accounting is specified, it also issues an SVC 78 for construction of the SMF LSPACE records (type 19). It issues an SVC 83 to have the records transferred to the SMF buffer.

IEESMFI2 then branches to the timer enqueue routine (IEAQTE00) in the timer second level interruption handler to add a compiled-in timer queue element (TQE), requesting 10-minute time intervals, to the timer queue. It then passes control to IEEDFINA via a LINK macro instruction.

IEEDFINA constructs the SMF storage configuration record (type 13). It issues an SVC 83 to have the record transferred to the SMF buffer. Upon return from the SVC routines, IEEDFINA returns to IEESMFI2, which returns to IEESMFIT, which returns to master scheduler initialization routine IEFSD569.

## The SMF Writer Routine (IEESMFWT)

The SMF writer routine is the resident wait routine of the SMF task. It executes under a system TCB (the SMF TCB). It is initially involved in a WAIT/POST interchange with the SMF initialization routines executing under the master scheduler TCB. This is to ensure the SMF writer routine that the SMCA is initialized and its address stored in the CVT, before the writer routine performs its SMF processing.

When the SMCA is initialized, IEESMFWT issues a WAIT macro instruction specifying the writer ECB. The SMF writer routine regains control each time the ECB is posted, and performs one of the following functions:

- It requests opening of the SMF data sets.
- It requests data set switching.
- It writes the contents of the SMF buffer in the SMF data set.
- It determines if the SMF data set contains enough storage space for a record that must be written in segments.

These functions are implemented in the same manner for MFT and MVT. They are described in the "Common Elements of Job Management" section of the MVT Job Management PLM.

# Appendix A: Tables and Work Areas

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)
- Write-to-Programmer Control Block (WTPCB)

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 33) is an area for communications between the command scheduling routine (SVC 34) and the command execution routines. Input CSCBs are created by several system routines. When an input 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. An input 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.
- 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 cancelation (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 control CSCB is updated (and changed to the control format if necessary) by the command scheduling routine when a CANCEL jobname (jcb 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.

| Name | Bit | Setting | Meaning |
|---|---|---|---|
| CHAP | 0 | 1 | Assignment pending |
| CHAC | 1 | | Reserved |
| CHHIAR | 2 | 0 | H0 specified START command |
| | | 1 | H1 specified on START command |
| CHDEF | 3 | 0 | Use hierarchy specified by bit 2 |
| | | 1 | Default to H0 |
| CHAD | 4 | 1 | Add this CSCB to the chain |
| CHDL | 5 | 1 | Delete this CSCB from the chain |
| CHFC | 6 | 1 | Free this CSCB's core |
| CHABTERM | 7 | 1 | Execute branch entry to ABTERM |

- Activity Flags: This byte indicates the type of activity with which the CSCB is associated.

| Name | Bit | Setting | Meaning |
|---|---|---|---|
| CHPROC | 0 | | Reserved |
| CHRSV | 1 | | Reserved |
| CHIN | 2 | 1 | Initiator waiting for work |
| CHSP | 3 | 1 | Special |
| CHCL | 4 | 1 | Cancelable job step |
| CHCLD | 5 | 1 | Cancel communication switch |
| CHAIFX | 6 | 1 | Cancelable (MFT only) |
| CHIFY | 7 | 1 | System assigned procedure (MFT only) |

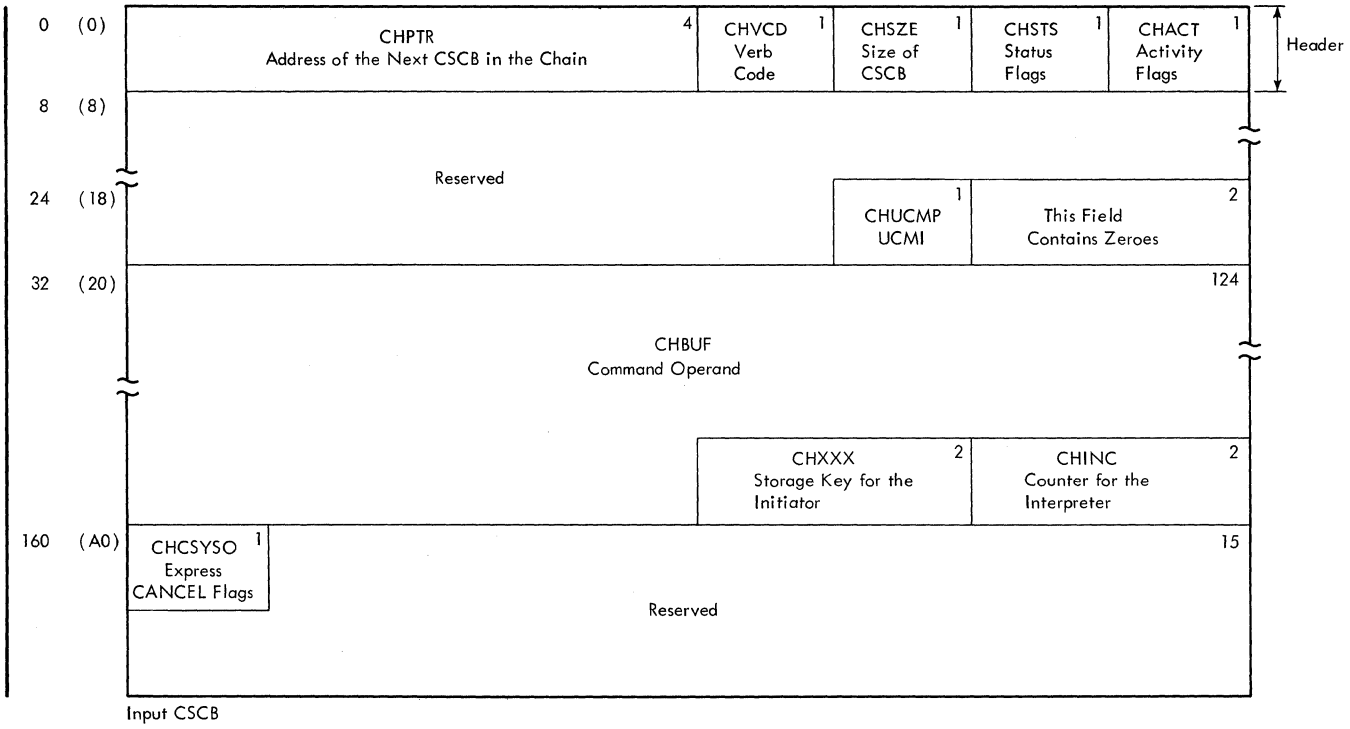
- Communication Flags: This byte indicates the function to be performed by the command processing routine.

| Name | Bit | Setting | Meaning |
|---|---|---|---|
| CHSTP | 0 | 1 | Stop |
| CHJCT | 1 | 1 | Reader return with in-core JCT |
| CHPSD | 2 | 1 | Writer pause data set |
| CHPSF | 3 | 1 | Writer pause forms |
| CHSYS | 4 | 1 | System task |
| CHPPF | 5 | | Reserved |
| CHSWY | 6 | | Reserved |
| CHSWZ | 7 | | Reserved |

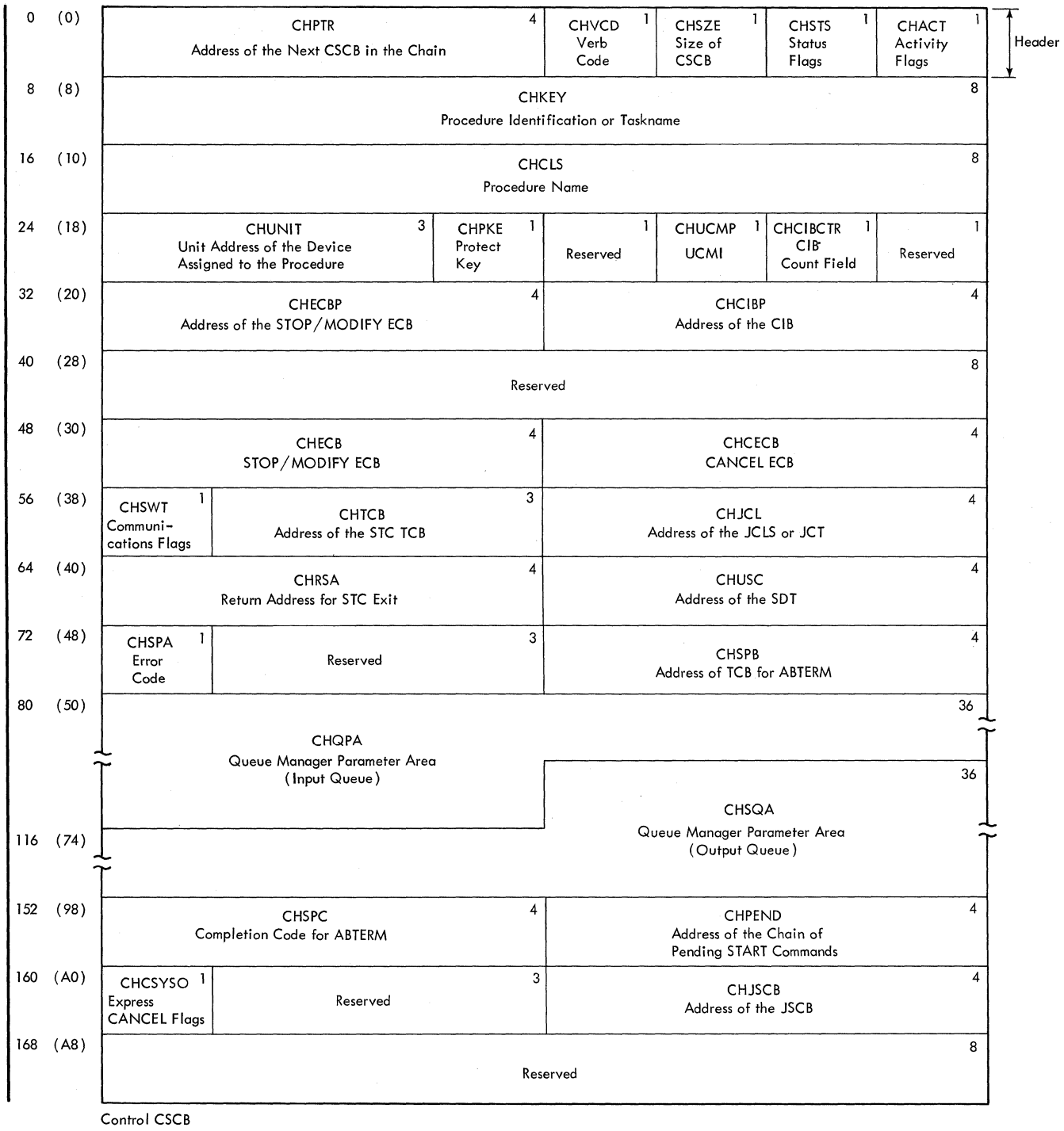
- Express CANCEL Flags: This byte indicates the parameters passed with the CANCEL command to the CANCEL processor.

| Name | Bit | Setting | Meaning |
|---|---|---|---|
| CHALL | 0 | 1 | ALL specified |
| CHINN | 1 | 1 | IN specified |
| CHOUT | 2 | 1 | OUT specified |
| CHHOLD | 3 | 1 | HOLD queue specified |
| CHQUE | 4 | 1 | Specific queue specified |
| CHDUMP | 5 | 1 | Dump specified |
| CHJB | 6 | 1 | End scan |
| CHSOUT | 7 | 1 | Cancel all SYSOUT |

Mapping Macro Instruction: IEECHAIN



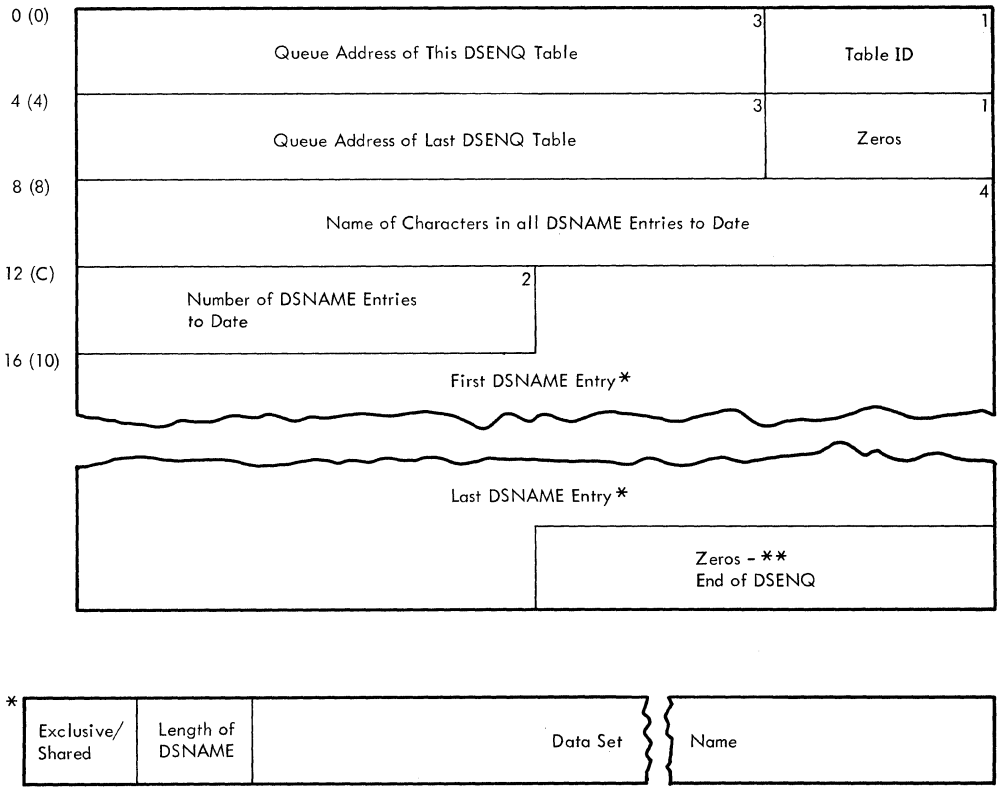

• Figure 33. Command Scheduling Control Block (CSCB) (Part 1 of 2)




• **Figure 33. Command Scheduling Control Block (CSCB) (Part 2 of 2)**

DATA SET ENQUEUE TABLE (DSENQ)

Description: The data set enqueue table (DSENQ) (Figure 34) 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 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.




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

Figure 34. Data Set Enqueue Table (DSENQ)

INTERPRETER WORK AREA (IWA)

Description: The 2048-byte interpreter work area (IWA) (Figure 35) 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-I: These fields contain internal switches used for communicating status information among the interpreter routines.

Switch A:

| Name | Bit | Setting | Meaning |
|---|---|---|---|
| JTOP | 0 | 1 | Job to process |
| JHS | 1 | 1 | Job has a step |
| JCTTQ | 2 | 1 | JCT to put cn queue |
| SCTTQ | 3 | 1 | SCT to put on queue |
| DFSH | 4 | 1 | Data flush |
| JFSH | 5 | 1 | Job flush |
| EOFR | 6 | 1 | End-of-file received |
| SAFSH | 7 | 1 | Flush to a /* |

Switch B:

| Name | Bit | Setting | Meaning |
|---|---|---|---|
| CXP | 0 | 1 | Continuation expected by Scan |
| CXPN | 1 | 1 | Continuation expected and not received |
| CXPC | 2 | 1 | Continuation expected and canceled |
| CANDD | 3 | 1 | DD * generated |
| DDAST | 4 | 1 | DD * or DD data |
| DDATA | 5 | 1 | DD data |
| FRCV | 6 | 1 | First statement received |
| SFJN | 7 | 1 | Search for job name |

Switch C:

| Name | Bit | Setting | Meaning |
|---|---|---|---|
| JCTRTN | 0 | 1 | CSCB return |
| IOERR | 1 | 1 | I/O error on input |
| NRCV | 2 | 1 | Null statement received |
| PEXP | 3 | 1 | Procedure EXEC statement expected |
| VOLTQ | 4 | 1 | Volume table to put on queue |
| DSNTQ | 5 | 1 | Data set name table to put on queue |
| PLSMB | 6 | 1 | Put last SMB for this step on queue |
| QMERR | 7 | 1 | Queue manager I/O error |

Switch D:

| Name | Bit | Setting | Meaning |
|---|---|---|---|
| JOBROLLF | 0 | 1 | Roll on job statement |
| JOBREGNS | 1 | 1 | Region on job statement |
| FEXRCV | 2 | 1 | First EXEC received this job |
| FDDRCV | 3 | 1 | First DD received this job |
| DBFST | 4 | 1 | First entry to DSENQ |
| DBLST | 5 | 1 | Last entry to DSENQ |
| DCTFST | 6 | 1 | First dictionary entry received |
| SYMPRC | 7 | 1 | First access of a procedure |

Switch E:

| Name | Bit | Setting | Meaning |
|---|---|---|---|
| PROC | 0 | 1 | Procedure library being used |
| GPI | 1 | 1 | Get procedure library input |
| PREF | 2 | 1 | Procedure library end-of-file |
| PRCV | 3 | 1 | Prime procedure buffer |
| CONCAT | 4 | 1 | Concatenation in merge |
| POVRD | 5 | 1 | Override procedure DD statement |
| POVRX | 6 | 1 | Override procedure EXEC statement |
| | 7 | | Unused |

Switch F:

| Name | Bit | Setting | Meaning |
|---|---|---|---|
| ORPARMOR | 0 | 1 | Parameter override |
| ORPARMBL | 1 | 1 | PARM parameter present |
| ORCONDOR | 2 | 1 | Condition override |
| ORTIMEOR | 3 | 1 | TIME override |
| ORTIMEO | 4 | 1 | TIME = zero |
| ORACTOR | 5 | 1 | ACCT override |
| ORREGOR | 6 | 1 | Region override |
| ORROLLOR | 7 | 1 | Roll override |

Switch G:

| Name | Bit | Setting | Meaning |
|---|---|---|---|
| ORRDOR | 0 | 1 | Reader override |
| ORSDPOR | 1 | 1 | Step dispatching priority override |
| | 2-7 | | Unused |

Switch H:

| Name | Bit | Setting | Meaning |
|---|---|---|---|
| PCPCOMM | 0 | 1 | PCP working on command |
| RDRDCBO | 1 | 1 | Reader opened |
| PROCDCBO | 2 | 1 | Procedure library opened |
| CPSYSFLG | 3 | 1 | Checkpoint restart EXEC statement |
| CPFLGXX | 4 | 1 | Reserved for checkpoint restart |
| PROCSW | 5 | 1 | Statement invokes a procedure |
| CPSTPFL | 6 | 1 | Checkpoint restart step flush |
| PCPSYSIN | 7 | 1 | SYSIN DD * encountered in PCP |

Switch I:

| Name | Bit | Setting | Meaning |
|---|---|---|---|
| BLKPRC | 0 | 1 | Block procedure library |
| IWABAS | 1 | 1 | Bypass Assign/Start |
| IWADDNM | 2 | 1 | DDNAME = Key this card |
| IWAKGSW | 3 | 1 | Blocked procedure PCP |
| BLKMLTER | 4 | 1 | Procedure library blocksize |
| DSNLIT | 5 | 1 | DSN = 'LITERAL' |
| | 6 | | Reserved |
| SPOOLDD | 7 | 1 | DD * or data indicator |

- Switch K: This field contains the Priority Change Value for the CHAP macro instruction.
- Switch L: This field contains the Default Allocation level in MSGLEVEL.
- Switch M: This field contains the Default JCL level in MSGLEVEL.
- Switch N: This field contains the length of the fixed part of the message for symbolic parameter substitution.
- Switch X1: This field is set to X'80' for a search of the DDNAME reference table or to X'40' for SYSOUT.

- Checkpoint/Restart Switches: These fields contain switches that communicate checkpoint/restart status information to the interpreter routines.

CHECKPOINT RESTART SWITCHES A:

| Name | Bit | Setting | Meaning |
|---|---|---|---|
| | 0-1 | | Unused |
| JOBRDNR | 2 | 1 | RD=NR |
| JOBRDNC | 3 | 1 | RD=NC or RD=RNC |
| JOBRDR | 4 | 1 | RD=R or RD=RNC |
| | 5-7 | | Unused |

CHECKPOINT RESTART SWITCHES B:

| Name | Bit | Setting | Meaning |
|---|---|---|---|
| CPFLG | 0 | 1 | GET/FREE SYSCHECK DD statement core |
| | 1 | | Unused |
| CPDUM | 2 | 1 | Dummy step control table required step flush |
| CRRES1 | 3 | | Reserved |
| CRRES2 | 4 | | Reserved |
| CRRES3 | 5 | | Reserved |
| CRRES4 | 6 | | Reserved |
| CRIMRS | 7 | 1 | Immediate restart (PCP) |

- Scan Switches: This field contains internal switches used by the Scan routines.

| Name | Bit | Setting | Meaning |
|---|---|---|---|
| RPRSW | 0 | 1 | Right parenthesis switch |
| PDELSW | 1 | 1 | Period delimiter switch |
| ASTSW | 2 | 1 | Asterisk switch |
| FLUSHSW | 3 | 1 | Flush switch |
| LDL | 4 | 1 | Last delimiter switch |
| DCBSW | 5 | 1 | DCB switch |
| JGC | 6 | 1 | Text sublist switch |
| FERROR | 7 | 1 | Error switch |

- Control and Scan Joint Switches: This field contains switches set by the Control routines to pass information to the Scan routine.

| Name | Bit | Setting | Meaning |
|---|---|---|---|
| CMT | 0 | 1 | Comment switch |
| DDOV | 1 | 1 | DD override switch |
| ENDS | 2 | 1 | End scan switch |
| COLST | 3 | 1 | Column 72 (continuation) switch |
| JOBSW | 4 | 1 | JOB switch |
| EXECSW | 5 | 1 | EXEC switch |
| DDSW | 6 | 1 | DD switch |
| SNPSW | 7 | 1 | Statement SYSOUT switch |

- Exit Switches: This field contains switches indicating conditions which cause exits to user routines.

| Name | Bit | Setting | Meaning |
|---|---|---|---|
| | 0-3 | | Unused |
| IWATRKS | 4 | 1 | Track stacking |
| IWAQFIOE | 5 | 1 | Job queue full |
| IWASFIND | 6 | 1 | Special procedure library FIND |
| IWAQENTR | 7 | 1 | Special queue manager entry |

- 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.

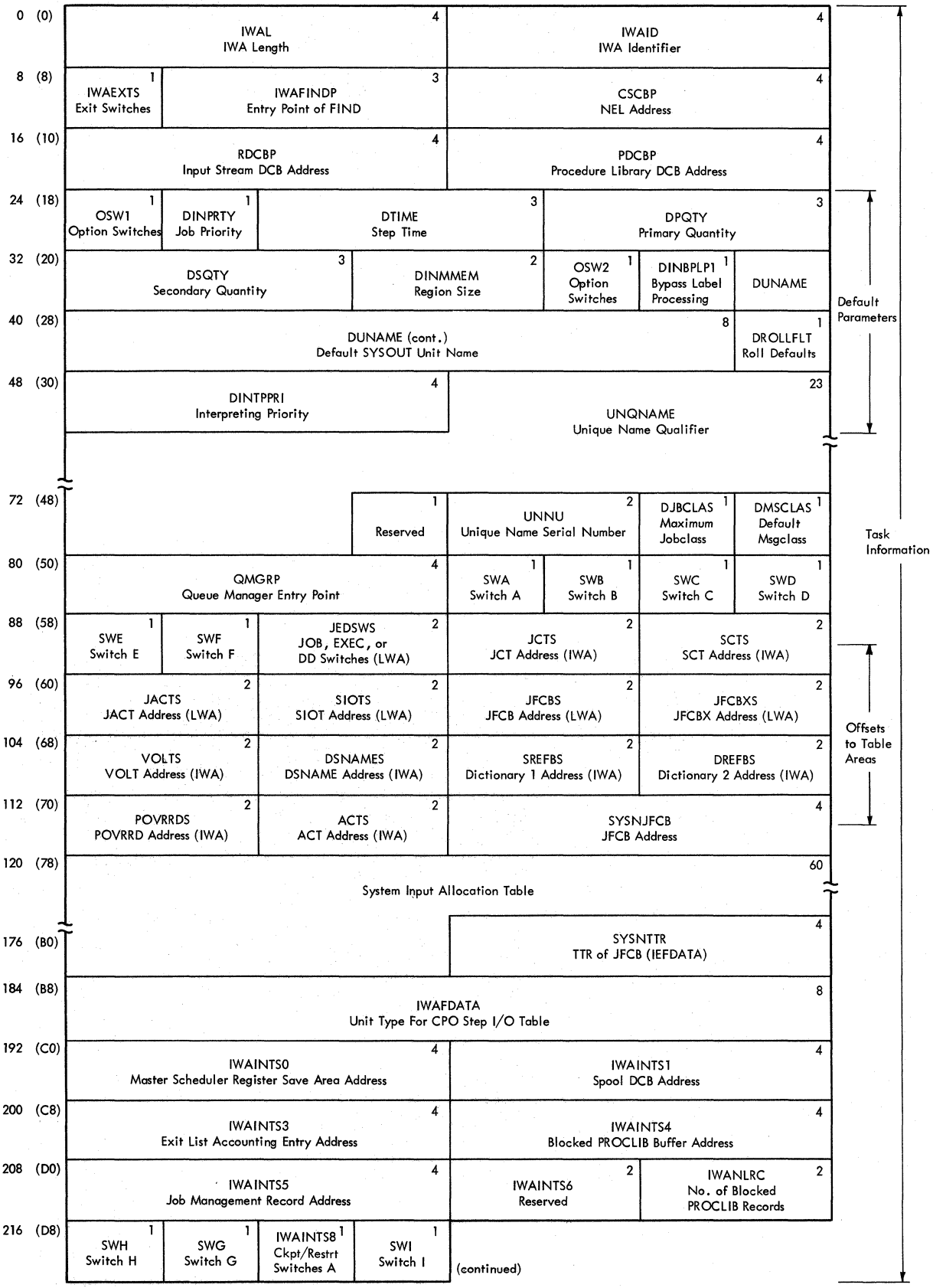
- Queue Address Table: This area contains the addresses (in TTR form) of the next two records assigned to the job's input queue entry, and the addresses (in TTR form) of the first joblib SIOT, the first scan dictionary record, and the DD override table.

- 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.

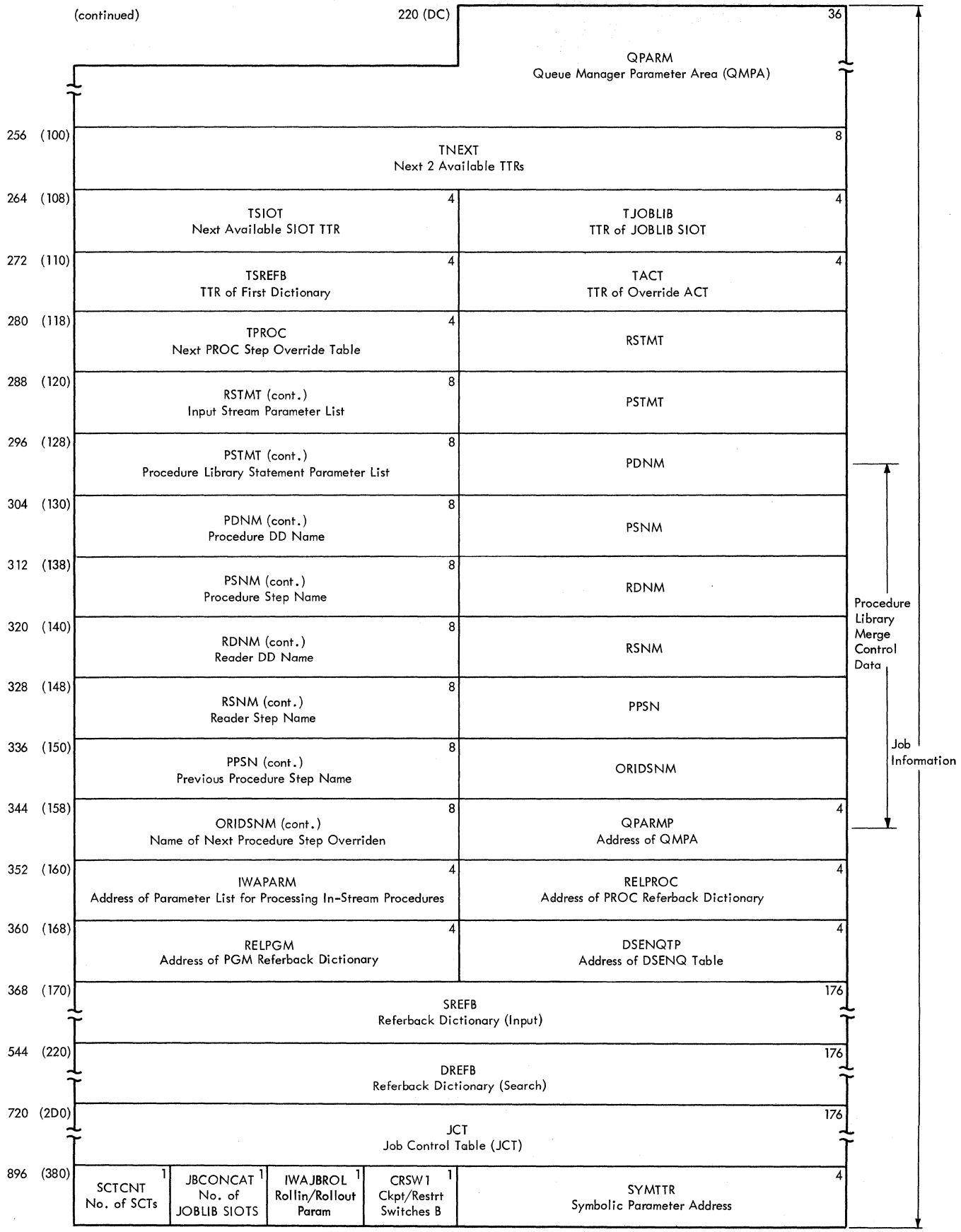
- Procedure Library Parameter List: This area describes the statement last read from the procedure library, and contains a pointer to the field currently being processed.

- 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.

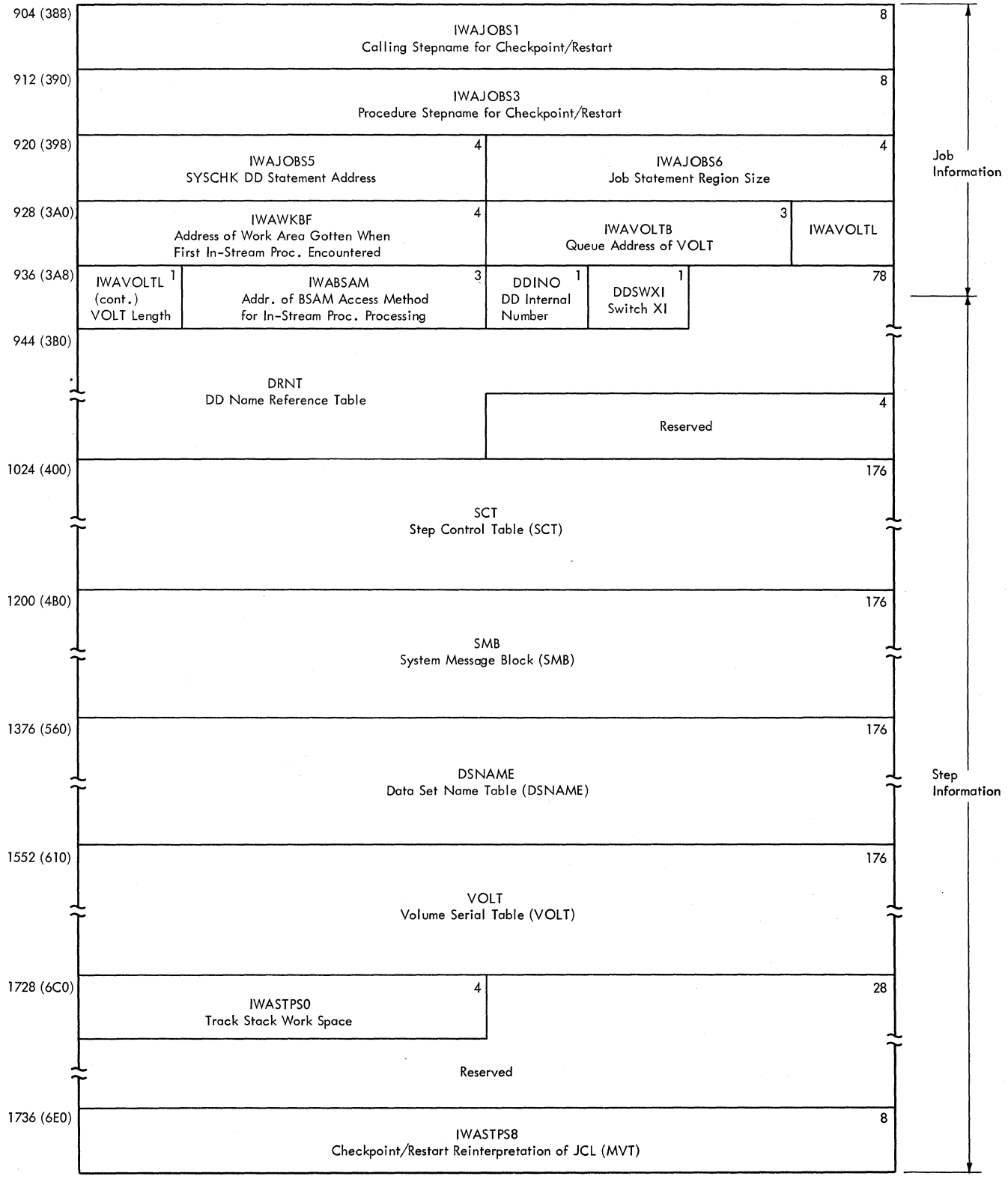
Mapping Macro Instruction: IEFVMIWA



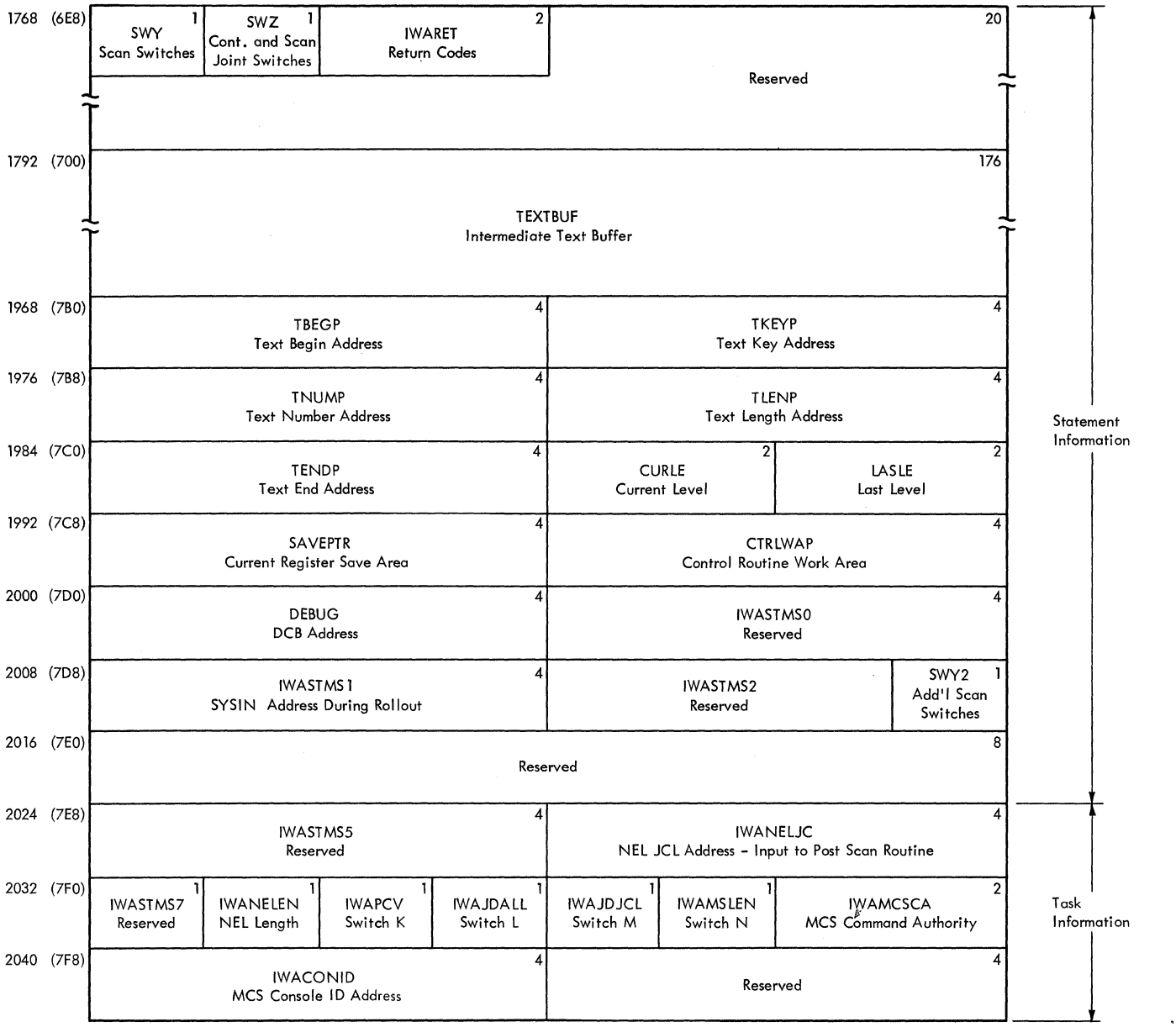

• Figure 35. Interpreter Work Area (IWA) (Part 1 of 4)




• Figure 35. Interpreter Work Area (IWA) (Part 2 of 4)




• Figure 35. Interpreter Work Area (IWA) (Part 3 of 4)




• **Figure 35. Interpreter Work Area (IWA) (Part 4 of 4)**

Description: The job control table (JCT) (Figure 36) 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 | Setting | Meaning |
|---|---|---|
| 0 | 1 | A JOBLIB DD statement is included with the job |
| 1 | 1 | Job flush |
| 2 | 1 | Job step canceled by condition codes |
| 3 | 1 | Step flush |
| 4 | 1 | JCT ABEND |
| 5 | 1 | Job failed |
| 6 | 1 | Job includes a cataloged procedure |
| 7 | 1 | Job is a "no setup" job |

- Additional Job Status Indicators: The byte indicates the status of the job as follows: Bit 0 is set to 1 to indicate spooled SYSIN data for the job. Bits 1 through 7 are reserved.

- Checkpoint/Restart Indicators: This two byte field indicates the checkpoint/restart status as shown below:

Byte 1

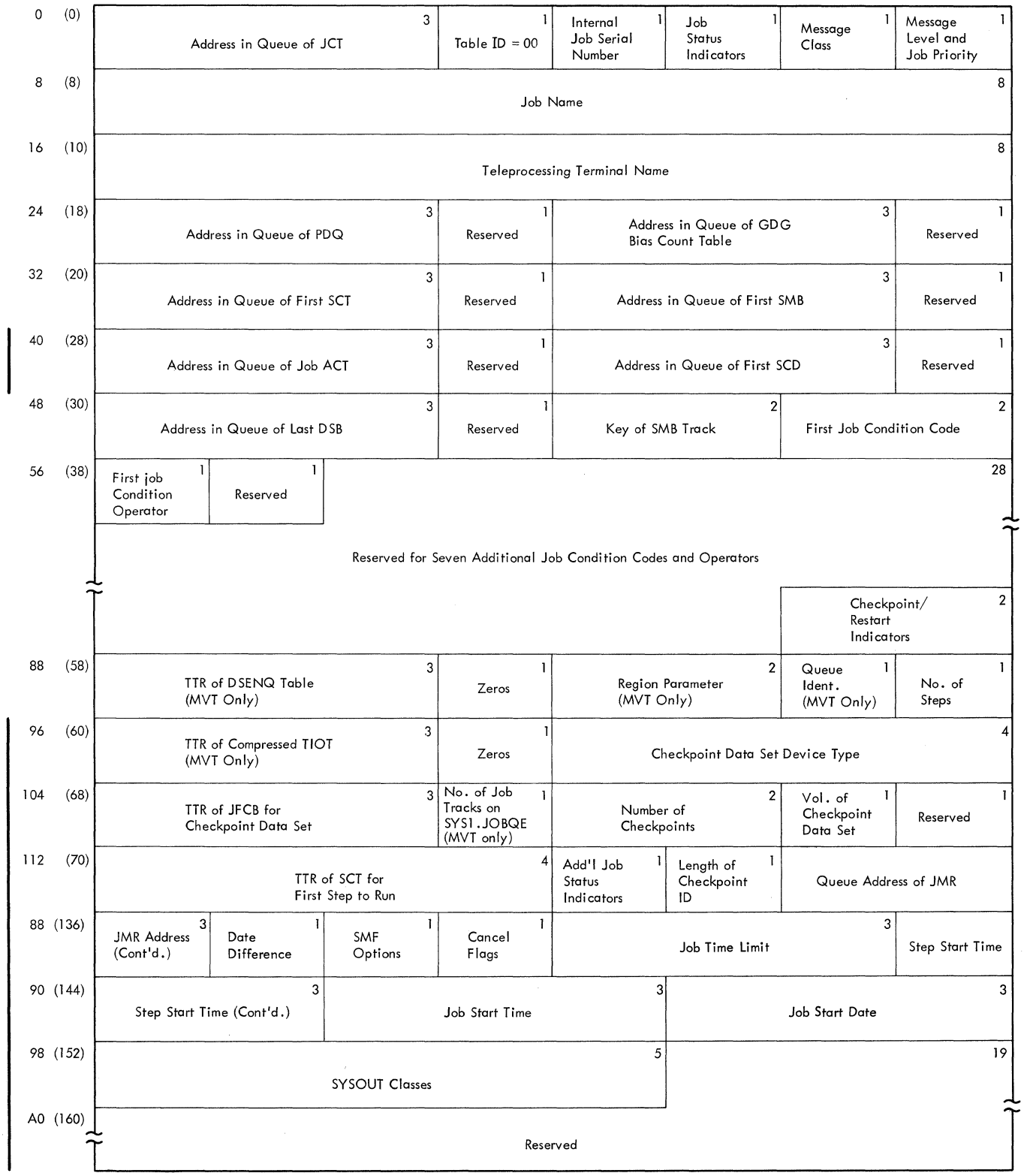
| Bit | Setting | Meaning |
|---|---|---|
| 0 | 1 | Warm start |
| 1 | | Not used by MFT |
| 2 | | Not used |
| 3 | 1 | Checkpoint taken for this step |
| 4 | 1 | Intra-step checkpoint/restart to be done |
| 5 | 1 | Step restart to be done |
| 6-7 | 0 | Must be set to zero |

Byte 2

| Bit | Setting | Meaning |
|---|---|---|
| 0 | 1 | SYSCHK DD statement is included with the job |
| 1 | 1 | RD keyword parameter is not NC |
| 2 | 1 | No restart is to be done |
| 3 | 1 | No checkpoints are to be taken |
| 4 | 1 | Do restart if necessary |
| 5 | 1 | Direct SYSOUT writer active at checkpoint |
| 6 | 1 | Job is eligible for direct SYSOUT facilities. |
| 7 | 1 | DSDR processing has not successfully ended |

- 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 36. Job Control Table (JCT)

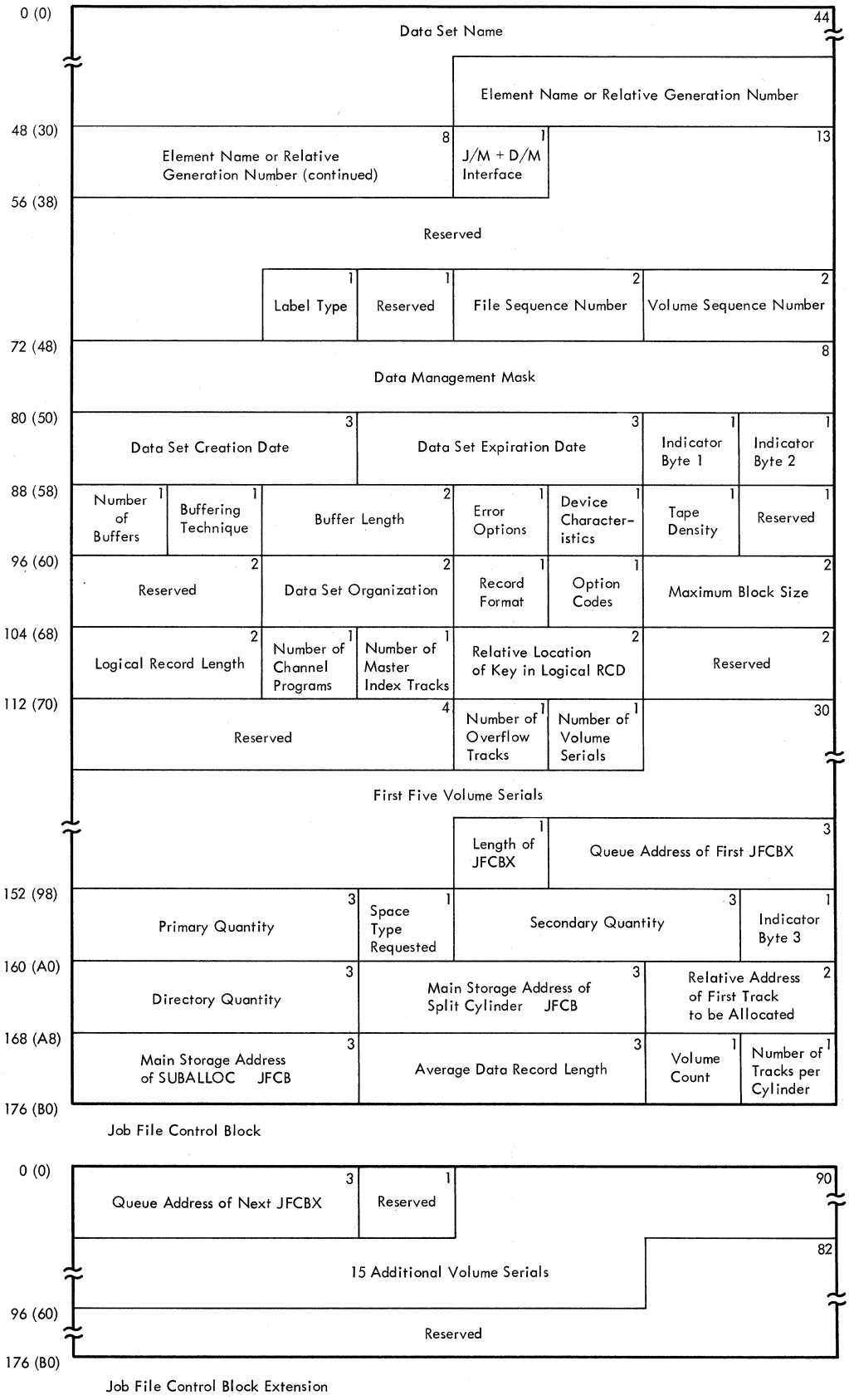



**Figure 37. Job File Control Block (JFCB) and Extension (JFCBX)**

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

Description: A job file control block (JFCB) (Figure 37) 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, GC28-6628.

Mapping Macro Instruction: IEFJFCBN

LIFE-OF-TASK (LOT) BLOCK

Description: The 384-byte life-of-task (LOT) block (Figure 38) is built in a main storage area obtained from subpool 253. It contains 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.

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, the address of the SPIL, and the ECB List.

LINKAGE CONTROL TABLE (LCT)

Description: The linkage control table (LCT) (Figure 39) is part of the LOT block constructed by the Job Select module in subpool 253. It is also built separately by System Task Control, in which case its storage is obtained from subpool zero. It is a communications area used by the routines of the Initiator, System Task Control, Allocation, and Termination.

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 40), 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.

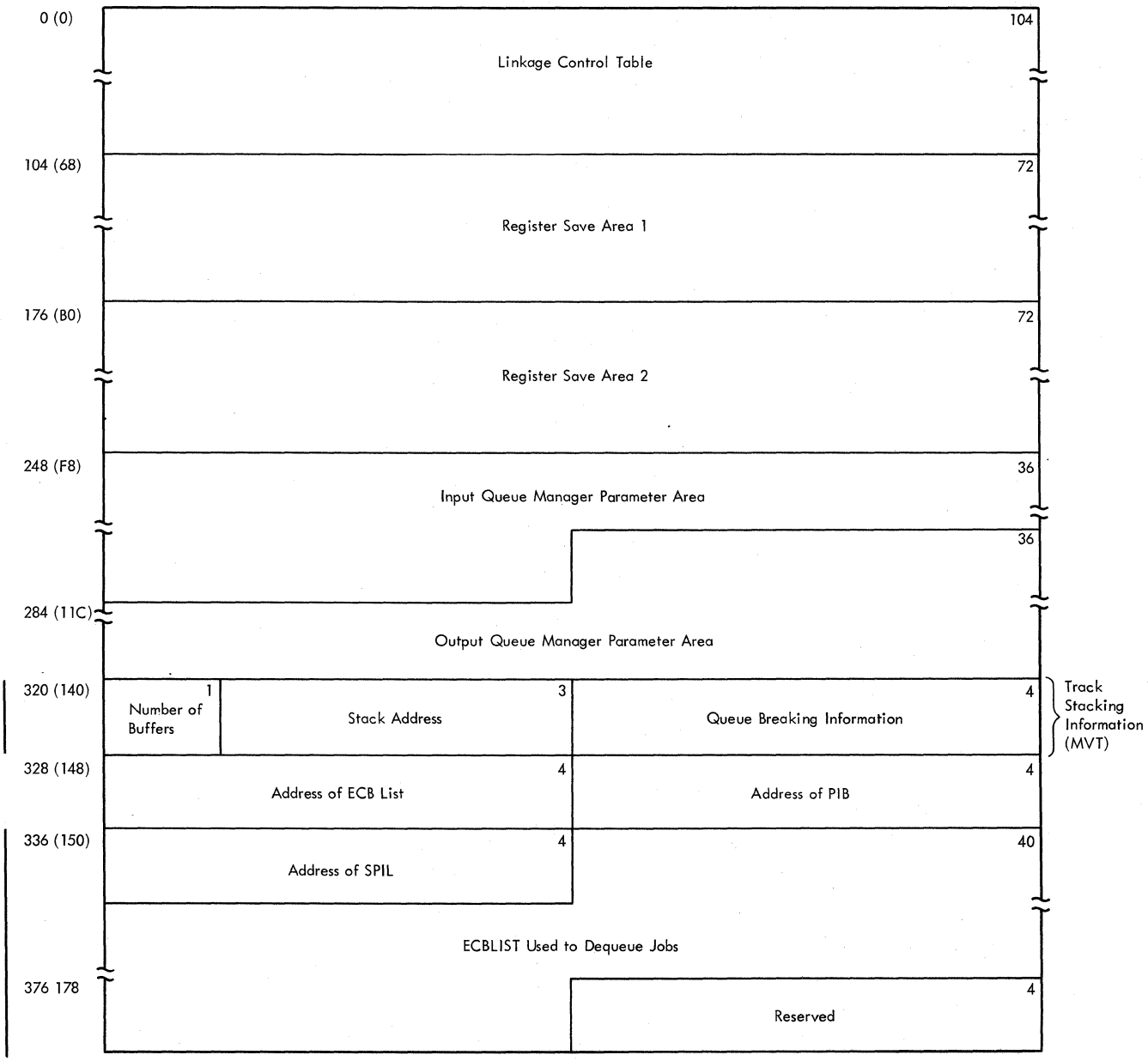
Most of the fields in the master scheduler resident data area are self-explanatory; those fields that require further explanation are described below:

- Queue Formatting Switch: If the high-order bit of this field is on, it indicates that the queue data set must be formatted.
- 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 fourth bit is set when initialization

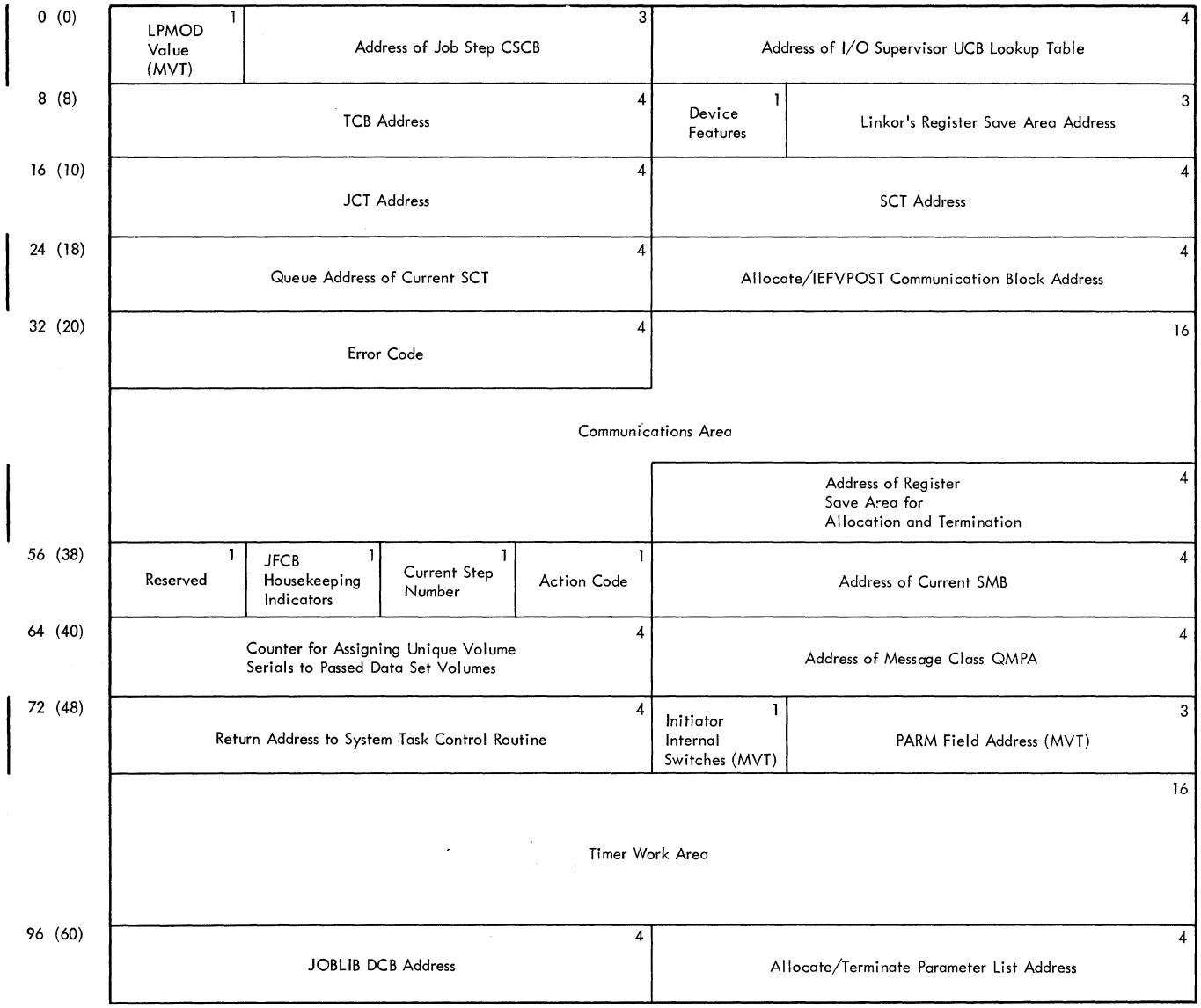
is complete to allow DEFINE commands to be accepted; the fifth bit is set on when the operator has requested partition changes at IPL; the sixth bit indicates that a small partition cannot terminate because of the DEFINE operation; the seventh bit indicates that a DEFINE command has been issued during operation; the eighth bit indicates that the system has storage protection.

- Status Flags: When set on, status flags indicate:

| Bit | Meaning |
|---|---|
| 0 | System Initialization in progress |
| 1 | DISPLAY JOBNAMES |
| 2 | Reserved |
| 3 | VARY/UNLOAD summary |
| 4 | Queue hold-release |
| 5 | DISPLAY ACTIVE processing |
| 6-7 | Reserved |




• Figure 38. Life-of-Task (LOT) Block




• Figure 39. Linkage Control Table (LCT)

- Log Status Flags:

| Bit | Meaning |
|---|---|
| 0 | Log Data Set Sysout Scheduling |
| 1 | Log Threshold Reached |

- MFT Switches: When set on, flags indicate:

| Bit | Meaning |
|---|---|
| 0 | Transient Reader Active |
| 1 | Transient Reader in Core |
| 2 | Pending START command for transient reader |
| 3 | MFT Environment switch |
| 4 | System Assigned Reader is Running |
| 5 | Core storage is in System |

- Initialization Switches: When set on, flags indicate:

| Bit | Meaning |
|---|---|
| 0 | IPL switch |
| 1 | SYSOUT IPL |
| 2 | SYSOUT job start |
| 3-4 | Reserved |
| 5 | 34 Security |
| 6 | Queue initialized |
| 7 | Procedure catalog initialized |

- System Exclusive Switches: When set on, switches indicate:

| Bit | Meaning |
|---|---|
| 0 | Console flag (PCP only) |
| 1 | CANCEL flag for ABEND (PCP only) |
| 2 | Roll-out flag (PCP only) |
| 3 | Spinoff flag (PCP only) |
| 4 | Display data set name |
| 5 | Display space |

- Pending Flags: When set on, flags indicate:

| Bit | Meaning |
|---|---|
| 0 | IPL Date |
| 1 | Region busy |
| 2 | Command move completed |
| 3 | Interpreter command return |
| 4 | System Input control purge request |
| 5 | System output control purge request |
| 6 | Blank start pending (REQ=1,START BLANK=0) |
| 7 | Console command suppressed by WTO/WTOR Exit Routine |

- ECB Flags: When set on, flags indicate:

| Bit | Meaning |
|---|---|
| 0 | External interrupt |
| 1 | WTO or WTOR |
| 2 | WTL |
| 3 | Console Attention key hit |
| 4 | System Input |
| 5 | System Output |
| 6 | Master command routine |
| 7 | Summary bit, Vary UCB scan required |

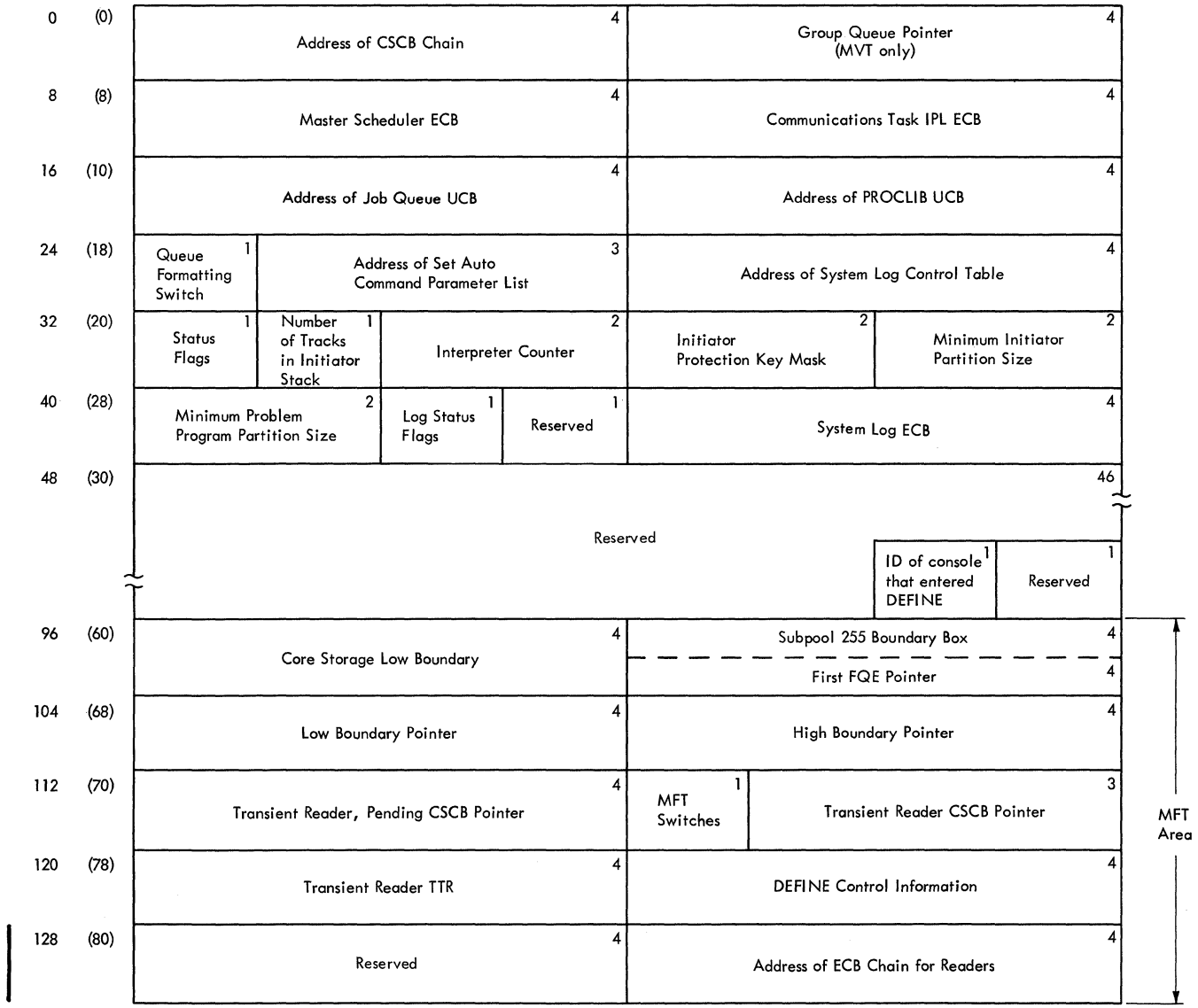
- Resident Switches: When set on, switches indicated:

| Bit | Meaning |
|---|---|
| 0 | IPL has been completed |
| 1 | WTO or WTOR pending |
| 2 | Console usage, Primary or alternate |
| 3 | Log purge request |
| 4 | Reader has reached end of file, or Start reader |
| 5 | New reader pending |
| 6 | New writer pending<br>New writer pending (Modify) |
| 7 | Job notification (1=yes) |

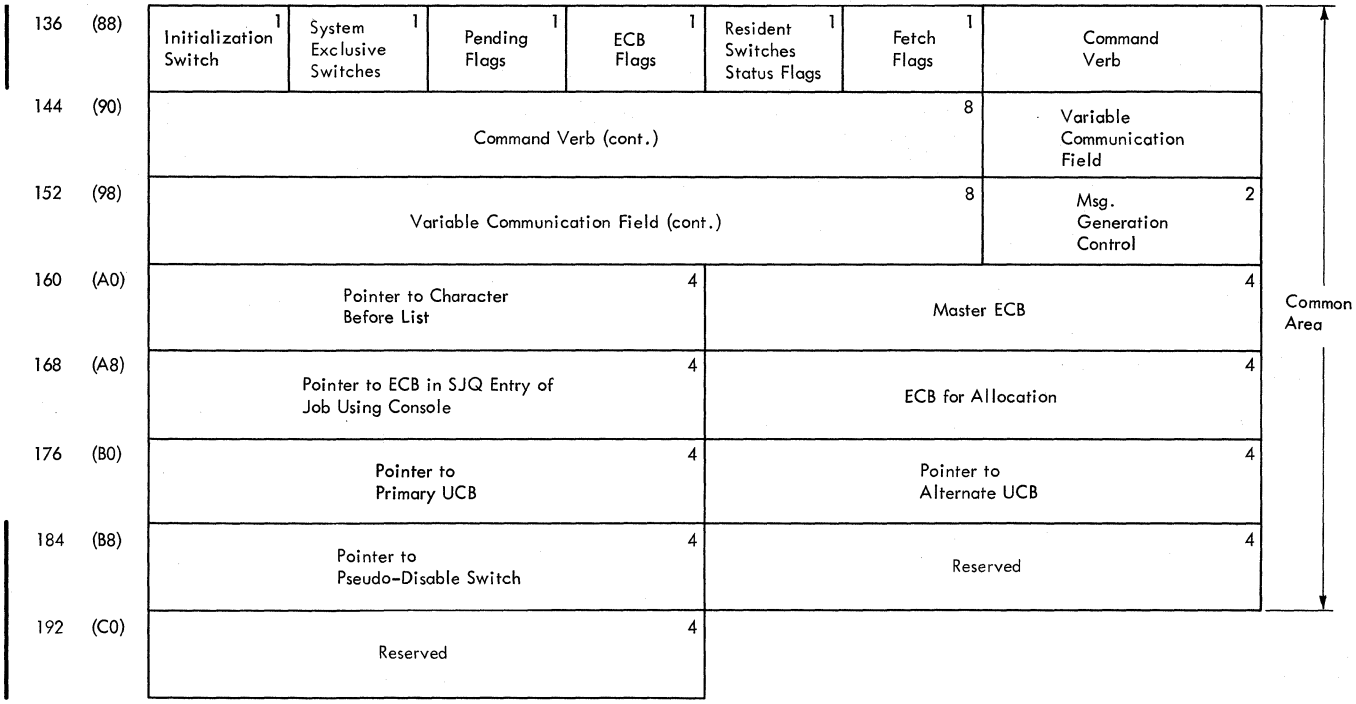
- Fetch Flags: When set on, flags indicate:

| Bit | Meaning |
|---|---|
| 0 | Named Fetch |
| 1 | Defer current command execution sequence |
| 2 | TCB Tree Trace Fetch (Locate) |
| 3 | Auxiliary FETCH given |
| 4 | Reply bit to Request attention |
| 5 | Pseudo-SYSOUT flag |
| 6 | DISPLAY STATUS |
| 7 | Queue hold-release |

- Mapping Macro Instruction: IEEBASEB.




- Figure 40. Master Scheduler Resident Data Area (Part 1 of 2)




• Figure 40. Master Scheduler Resident Data Area (Part 2 of 2)

PARTITION INFORMATION BLOCK

The 48-byte partition information block (PIB) (Figure 41) 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:

- 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 | Reserved |
| 3 | 1 | Transient reader is suspended |
| 4 | 1 | Partition is to be terminated by IEFSD599 when it next gets control |
| 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 |
| 3 | 1 | Reserved |
| 4 | 1 | Unending task present in partition |
| 5 | 1 | JOBLIB Switch |
| 6 | 1 | STEPLIB Switch |
| 7 | 1 | FETCHLIB Switch |

- 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.

- Internal Queue Status Bits:

| Bit | Setting | Meaning |
|---|---|---|
| 0 | 1 | A large partition in which the DSDR processing step for a small partition (less than 12K) is to be executed |
| 1 | 1 | Reserved |
| 2 | 1 | A DEFINE command has been received and the partition is processing jobs on its internal queue. |
| 3-7 | | Reserved |

- Job Step Timing Status Bits:

| Bit | Setting | Meaning |
|---|---|---|
| 0 | 1 | The job step TQE is being used for job step timing. |
| 1 | 1 | Indicates to the Initiator that the step being terminated was timed. |
| 2-7 | | Reserved. |

| | | | |
|---|---|---|---|
| 0 | (0) | CSCB Address of Pending Command | 4 |
| 4 | (4) | ECB Address | 4 |
| 8 | (8) | "No Work" ECB for the Initiator | 4 |
| 12 | (C) | 1 Status Bits - A | 3 Address of Current Job Step CSCB |
| 16 | (10) | 1 Status Bits - B | 3 SPIL Address |
| 20 | (14) | CSCB Address of Current Task in Partition | 4 |
| 24 | (18) | 1 Protection Key | 3 Job Class Codes |
| 28 | (1C) | CSCB Address of Suspended Reader | 4 |
| 32 | (20) | Address of the Direct SYSOUT Control Block (DSOCB) Chain | 4 |
| 36 | (24) | 1 Internal Queue Status Bits | 3 Address of Internal Queue of Job Names to be Restarted |
| 40 | (28) | 1 Job Step Timing Status Bits | 3 Address of the Job Step TQE |
| 44 | (2C) | 1 Count of Active Subtasks | 3 Address of the RB of the Most Recently Loaded Module on the JPAQ |

• Figure 41. Partition Information Block (PIB)

SMALL PARTITION INFORMATION LIST (SPIL)

Description: The 80-byte small partition information list (SPIL) (Figure 42) 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.

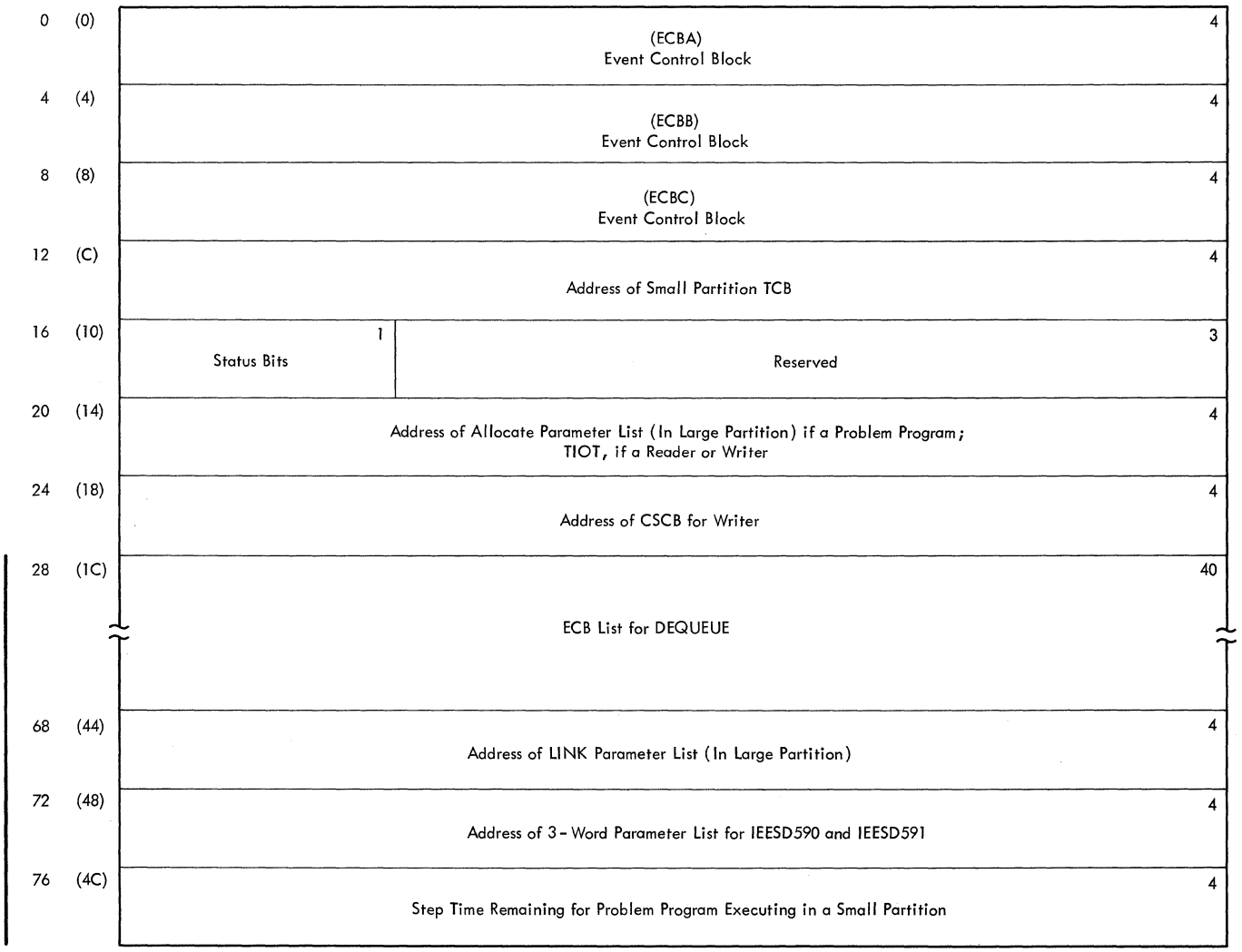
Bits 0 and 1 contain ones if a START writer or reader command has been entered.

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

Bit 3 contains a one if a problem program has requested termination.

Bit 4 contains a one if an indicative dump was requested.

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




• Figure 42. Small Partition Information List (SPIL)

Description: The step control table (SCT) (Figure 43), 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:

BYTE 4: Internal Step Status Indicators:

| Bit | Setting | Meaning |
|---|---|---|
| 0 | 1 | Step can be rolled out |
| 1 | 1 | Roll step out if necessary |
| 2 | 1 | Do not restart step |
| 3 | 1 | Do not take a checkpoint |
| 4 | 1 | Restart if necessary |
| 5 | 1 | Graphics - alter protect key |
| 6 | 1 | Graphics - ABEND exit |
| 7 | 1 | Step failed |

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.

BYTE 67: Step Type Indicators:

| Bit | Setting | Meaning |
|---|---|---|
| 0 | 1 | EXEC statement contains PGM=*.stepname.ddname |
| 1 | 1 | SYSIN is specified as DD* |
| 2 | 1 | SYSOUT is specified |
| 3 | 1 | JFCB housekeeping is complete |
| 4-6 | | Initiator Indicator |
| 7 | | Reserved |

BYTE 104: Extension of Internal Step Status Indicator

| Bit | Setting | Meaning |
|---|---|---|
| 0 | | Reserved |
| 1 | 1 | Direct system output\facilities required to output job separator or system messages. |
| 2 | 1 | Allocation for control volume |
| 3 | | Reserved |
| 4 | 1 | STEPLIB present |
| 5 | 1 | Spooled SYSIN for step |
| 6 | 1 | Job ended |
| 7 | | Reserved |

Mapping Macro Instruction: IEFASCTB

STEP INPUT/OUTPUT TABLE (SIOT)

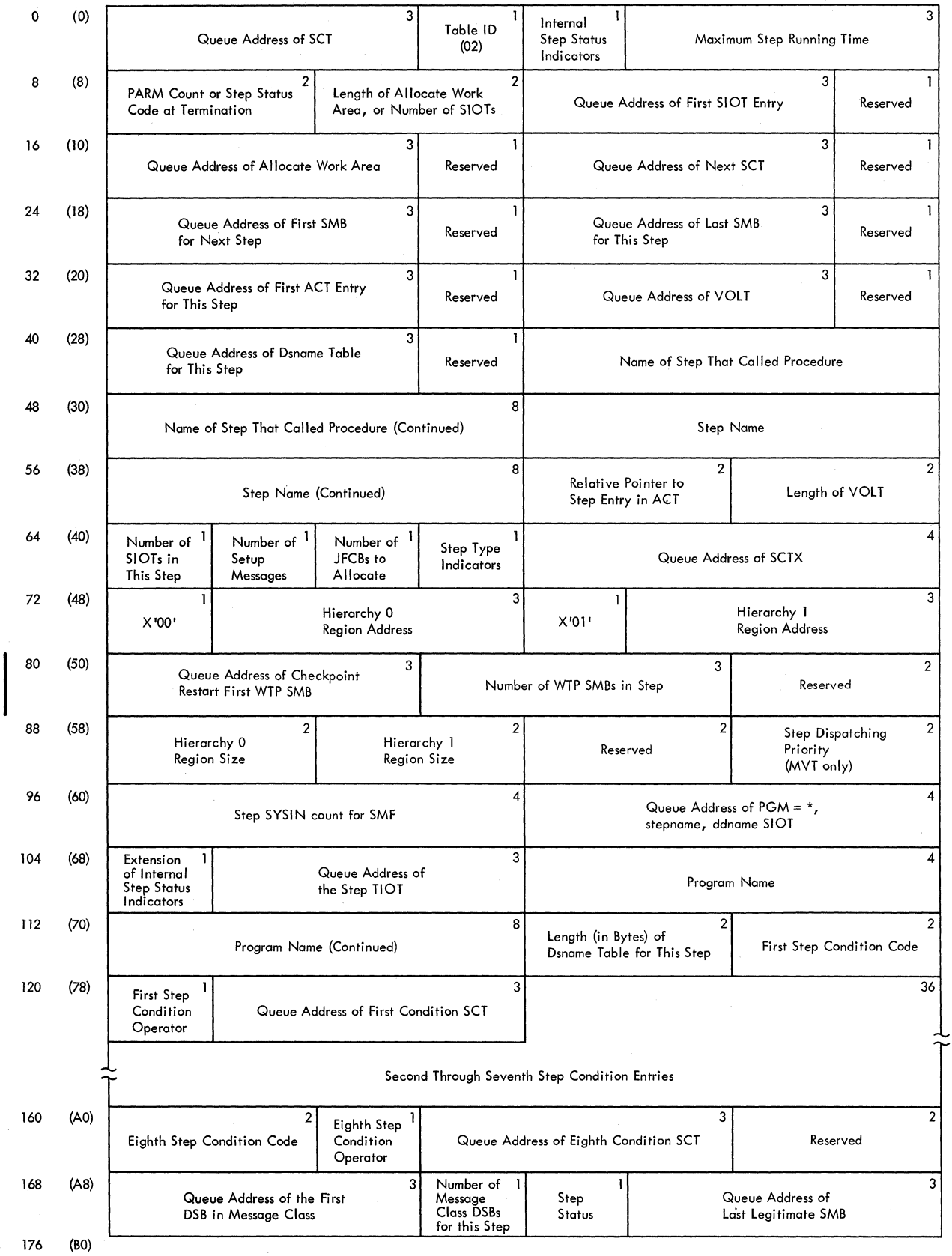
Description: The Step Input/Output Table (SIOT) (Figure 44), makes DD statement available to the initiator for use as a source of information for the TIOT and for providing DD information to allocation and disposition routines. When a DD statement is read, the interpreter creates a new SIOT and places the DD information into it. The individual bits of the disposition byte and of indicator bytes 56 through 59 in the SIOT are set to one to indicate the following conditions:

BYTE 55: Scheduler Disposition

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

BYTE 56: Indicator Byte Number 1

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




• Figure 43. Step Control Table (SCT)

BYTE 57: Indicator Byte Number 2

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

BYTE 58: Indicator Byte Number 3

| Bit | Meaning |
|---|---|
| 0 | Volume reference |
| 1 | SYSIN expected (procedures only) |
| 2 | Allocate work table volume block indicator |
| 3 | Volume reference in step |
| 4 | SYSOUT was specified |
| 5 | NEW data set |
| 6 | MOD data set |
| 7 | OLD or SHR data set |

BYTE 59: Indicator Byte Number 4

| Bit | Meaning |
|---|---|
| 0 | Set by reader to indicate GDG single |
| 4 | Step processed |
| 5 | Intra-step volume affinity |
| 6 | Data set is in passed data set queue (PDQ) |
| 7 | 1 = old or modified data set<br>0 = new data set |

BYTE 92: Conditional Disposition

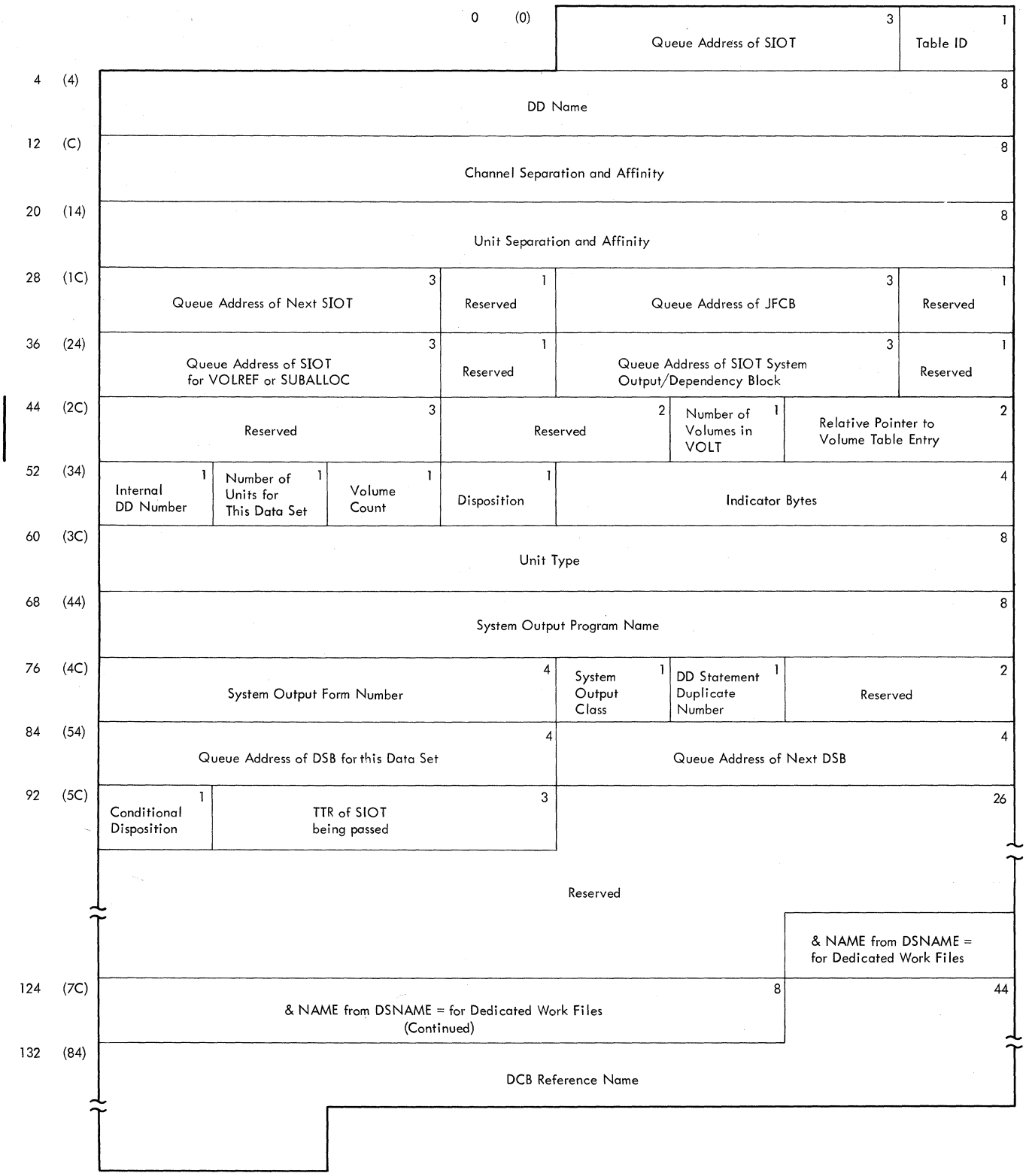
| Bit | Meaning |
|---|---|
| 0-3 | Reserved |
| 4 | Keep data set |
| 5 | Delete data set |
| 6 | Catalog data set |
| 7 | Uncatalog data set |

Mapping Macro Instruction: IEFASIOT

## TASK INPUT/OUTPUT TABLE (TIOT)

Description: The Task Input/Output Table (TIOT) (Figure 45) 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, GC28-6628.




• Figure 44. Step Input/Output Table (SIOT)

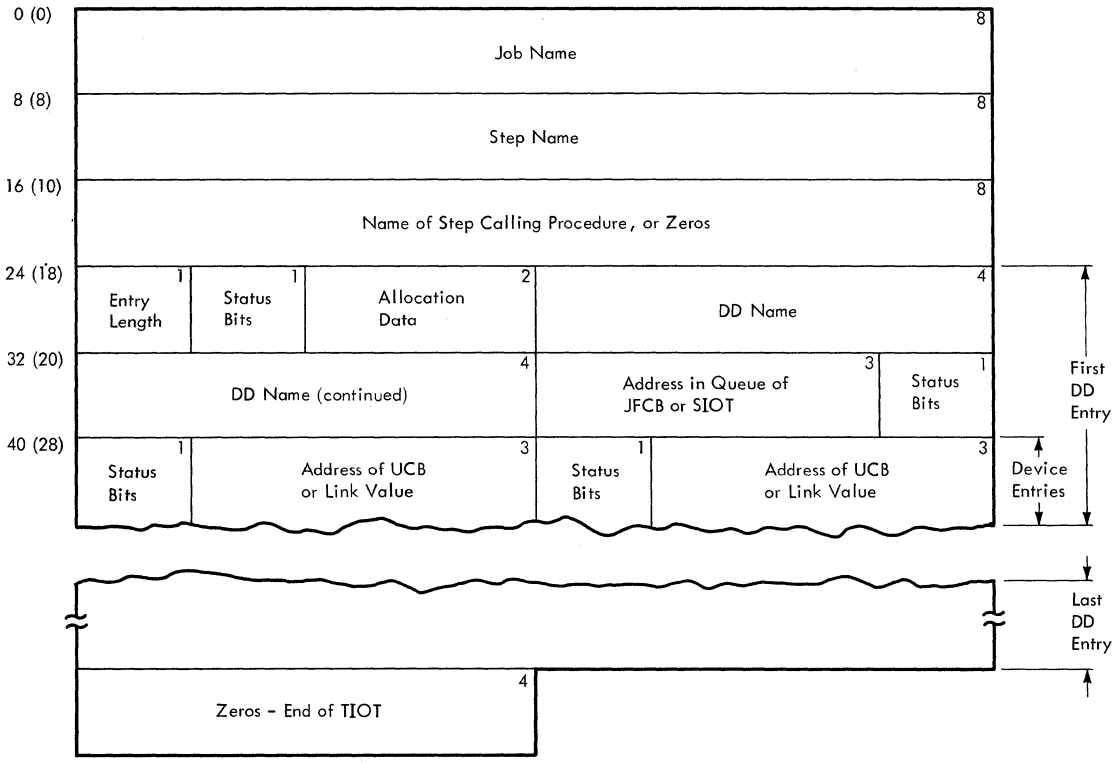
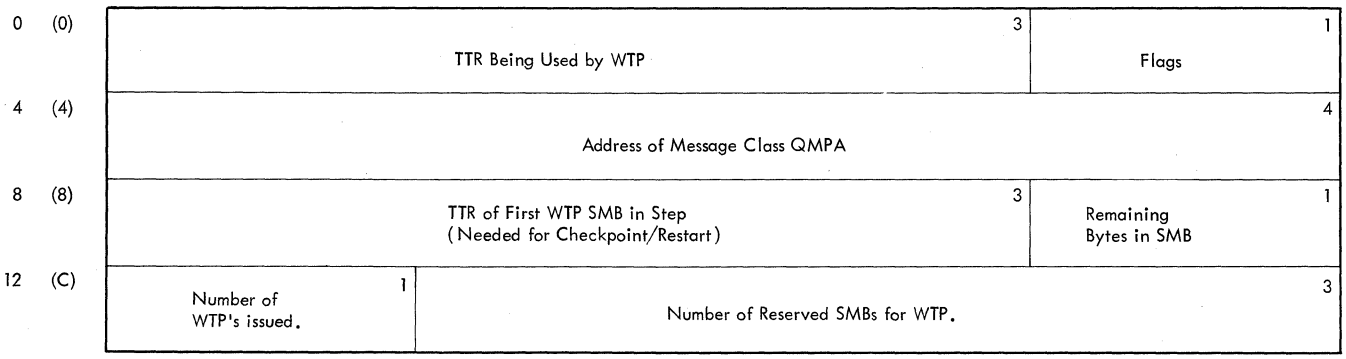



**Figure 45. Task Input/Output Table (TIOT)**

**WRITE-TO-PROGRAMMER CONTROL BLOCK (WTPCB)**

**Description:** The write-to-programmer control block (WTPCB) (Figure 46) is built by allocation interface control routine IEEVACTL in the system queue area. It is used by system tasks and problem program tasks when write-to-programmer messages are issued. The "Flags" field is defined below:

| Bit | Setting | Meaning |
|---|---|---|
| 0 | 1 | Job queue problem |
| 1 | 1 | Limit message processed |
| 2 | 1 | Step contains SYSOUT |
| 3 | 1 | Return to IEFWTP01 upon completion |
| 4 | 1 | No record message processed |
| 5 | 1 | Last SMB used for job |
| 6 | 1 | WTP invoked for this step |
| 7 | 1 | WTOR or WTO with additional routing codes being processed by WTP |




**•Figure 46. Write-to-Programmer Control Block (WTPCB)**

# Appendix B: MFT Modules

This appendix contains a table of unique MFT modules, a group of tables showing the modules of each major component, a list matching entry point and control section names with source module names, 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 3-15 which give a cross-reference to the source module. The source modules are in turn listed alphamerically for easy access. If you know the source module name, go directly to the module descriptions.

## Unique MFT Modules

Table 2 lists all modules that are unique to MFT. This table is organized by major component.

• Table 2. MFT Modules

| ABEND: | Initiator: | Master Scheduler Task: |
|---|---|---|
| IEACTM0B | IEFSD167 | IEECIR50 |
| IEADTM22 | IEFSD3BQ | IEEDFINA |
| IEADTM23 | IEFSD510 | IEEDFINB |
| IEAMTM05 | IEFSD511 | IEEDFINC |
| IEANTM0A | IEFSD512 | IEEDFIN1 |
| IEANTM0B | IEFSD513 | IEEFIN2 |
| IEANTM0C | IEFSD515 | IEEDFIN3 |
| IEANTM0D | IEFSD516 | IEEDFIN4 |
| IEANTM0E | IEFSD517 | IEEDFIN5 |
| IEANTM00 | IEFSD518 | IEEDFIN6 |
| IEANTM01 | IEFSD519 | IEEDFIN7 |
| IEANTM02 | IEFSD540 | IEEDFIN8 |
| IEANTM03 | IEFSD541 | IEEDFIN9 |
| IEANTM04 | IEFSD553 | IEESD566 |
| IEANTM05 | IEFSD554 | IEFSD569 |
| IEANTM06 | IEFSD555 | |
| IEANTM07 | IEFSD556 | |
| IEANTM08 | IEFSD558 | |
| IEANTM09 | IEFSD559 | |
| | IEFSD589 | |
| | IEFSD598 | System Task Control: |
| | IEFSD599 | IEESD590 |
| Communication Task: | | IEESD591 |
| IEEC1R45 | Reader/Interpreter: | IEESD592 |
| IEEVWTOR | IEFSD530 | IEEVACTL |
| | IEFSD531 | IEEVSMBA |
| I/O Device Allocation: | IEFSD532 | IEEVTCTL |
| IEFSD551 | IEFSD533 | IEFSD534 |
| IEFSD552 | IEFSD536 | IEFSD535 |
| IEFSD557 | IEFSD537 | IEFSD587 |
| IEF41DUM | | IEFSD588 |
| Nucleus: | System Management Facility: | |
| IEESD567 | IEESMFWT | SVC 34: |
| IEESD568 | | IEESD561 |
| | | IEESD571 |
| | System Error Task: | IEE2803D |
| | IEAMSERB | IEE3903D |

## Major Component Modules

Tables 3 through 15 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 |
|---|---|
| Control Block Validity Check | IEANTM03 |
| DAR Core Image Dump | IEADTM22 |
| DAR Task Reinstatement | IEADTM23 |
| Dump | IEANTM06 |
| Dump Test | IEANTM04 |
| Indicative Dump | IEANTM08 |
| I/O Purge | IEANTM02 |
| IQE Purge and Data Set Close | IEANTM0E*** |
| Loading Program Purge | IEANTM0C*** |
| Normal Termination and Abnormal Termination Router | IEANTM00 |
| Open Dump Data Set | IEAMTM05 |
| Open Dump Data Set | IEANTM05* |
| Recursion Processing | IEANTM09 |
| STAE and Graphics Test | IEANTM01 |
| Steal Main Storage | IEANTM0A |
| Subtask ENQ Purge | IEANTM0D*** |
| Termination | IEANTM07 |
| WTOR Purge | IEACTM0B** |
| WTOR Purge | IEANTM0B |

*Replaces IEAMTM05 for MFT with subtasking.
**Replaces IEANTM0B for MFT with MCS.
***MFT with Subtasking only.

• 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 |
| EXCP OPEN/CLOSE | IEECVOC |
| MCS Comm Task Router | IEECMAWR |
| MCS Console Switch | IEECMCSW |
| MCS Device Interface | IEECMDSV |
| MCS 1052 Device Support | IEECMPMX |
| MCS 1403/1443 Device Support | IEECMPMP |
| MCS 2540 Device Support | IEECMPMC |
| MCS 2740 Device Support | IEEC2740 |
| MCS Delete Operator Message | IEECMDOM |
| MCS WTO/WTOR Processor (SVC 35) | IEECMWSV |
| NIP Message Buffer Writer | IEECMWTL |
| User Dummy WTO/WTOR Exit | IEECVCTE |
| WTOR Purge (End of Job) | IEAGTM07 |

• Table 5. Direct System Output Modules

| Routine | Source Module |
|---|---|
| Initialization | IEFDSOCP |
| Release DSOCB Routine | IEFDSOFB |
| SIOT and JFCB Modification | IEFDSOAL |
| STOP and MODIFY Command Processor | IEFDSOSM |
| System Messages and Job Separator Writer | IEFDSOWR |
| Tape to Printer or Card Punch | IEFPRINT |

• Table 6. Initiator Modules

| Routine | Source Module |
|---|---|
| Alternate Step Deletion | IEFSD516 |
| Data Set Integrity | IEFSD541 |
| Dequeue by Jobname Interface | IEFSD519 |
| Dummy User Job Initiation Exit Routine | IEFUJI |
| Dummy User Step Initiation Exit Routine | IEFUSI |
| ENQ/DEQ Purge | IEFSD598 |
| Job Deletion | IEFSD517 |
| Job Initiation | IEFSD511 |
| Job Selection | IEFSD510 |
| Job Suspension | IEFSD168 |
| Linkage from Job Termination to Initiator for the 30K Scheduler | IEFSD33Q |
| Linkage from Job Termination to Initiator for the 44K Scheduler | IEFSD32Q |
| Linkage to IEFSD168 | IEFSD167 |
| 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 |
| Partition Recovery | IEFSD518 |
| Problem Program Interface | IEFSD513 |
| Set Problem Program State | IEFSD556 |
| Small Partition Module | IEFSD599 |
| Step Deletion | IEFSD515 |
| Step Initiation | IEFSD512 |
| TCTIOT Construction | IEFSMFAT |
| User Exit Initialization Routine | IEFSMFIE |

• Table 7. I/O Device Allocation Modules (Part 1 of 2)

| Routine | Source Module |
|---|---|
| Allocation Control | IEFXCSSS |
| Allocation Entry | IEFSD21Q |
| Allocation Exit | IEFSD41Q |
| Allocation Recovery Messages | IEFSJMSG |
| Allocation Recovery | IEFXJIMP |
| Automatic Volume Recognition | IEFXV001 |
| Automatic Volume Recognition Messages | IEFVMSG |
| Automatic Volume Recognition Nonstandard Label Routine | IEFXVNSL |
| Bit Pattern Scan Routine | IEFSCAN |
| DADSM Error Recovery | IEFXT003 |
| Decision Allocation | IEFS5000 |
| Demand Allocation | IEFWA000 |
| Device Bit Pattern | IEFDEVPT |
| 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 | IEFVMLS1 |
| 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 |

(Part 1 or 2)

Table 7. I/O Device Allocation Modules (Part 2 of 2)

| Routine | Source Module |
|---|---|
| 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 |
| Return to Initiator or System Task Control | IEF41DUM |
| Separation Strikeout | IEFXH000 |
| Space Request | IEFXT00D |
| VARY Interface and TIOT Compression | IEFXT002 |
| TIOT Construction | IEFWCIMP |
| Unsolicited Device Interrupt Handler | IEFVPOST |
| Wait for Space Decision | IEFSD097 |
| Wait for Unallocation | IEFSD195 |

Table 8. Interpreter Modules (Part 1 of 2)

| 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 |
| Dummy User JCL Validation Exit Routine | IEFUJV |
| End-of-File | IEFVHAA |
| EXEC Statement Processor | IEFVEA |
| Get Parameter | IEFVGK |
| Get | IEFVHA |
| Housekeeping | IEFVHHB |
| In-stream Procedure Compress Routine | IEZNCODE |
| In-Stream Procedure Directory Build Routine | IEFVINC |
| In-Stream Procedure Expand Routine | IEZDCODE |
| In-Stream Procedure Expand Interface Routine | IEFVIND |
| In-Stream Procedure Processor | IEFVINA |
| In-Stream Procedure Search Routine | IEFVINB |
| In-Stream Procedure Syntax Check Routine | IEFVINE |
| Initialization | IEFVH1 |
| Initialization | IEFVH2 |

(Part 1 of 2)

Table 8. Interpreter Modules (Part 2 of 2)

| Routine | Source Module |
|---|---|
| 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 | IEFVGM71 |
| 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 9. Master Scheduler Modules (Part 1 of 2)

| Routine | Source Module |
|---|---|
| Console Initialization | IEECVCTI |
| DEFINE Command Final Processor | IEEDFIN9 |
| DEFINE Command Validity Check (Core Storage) | IEEDFINC |
| DEFINE Final Processor | IEEDFIN3 |
| DEFINE Initialization | IEEDFIN1 |
| DEFINE Keyword Scan | IEEDFIN7 |
| DEFINE Listing | IEEDFIN4 |
| DEFINE Message | IEEDFIN5 |
| DEFINE Syntax Check and Router | IEEDFIN2 |
| DEFINE System Reinitialization (1) | IEEDFIN8 |
| DEFINE System Reinitialization (2) | IEEDFINB |
| DEFINE Time-Slice Syntax Check | IEEDFIN6 |
| DISPLAY A | IEESD566 |
| DISPLAY CONSOLES | IEEXEDNA |
| DISPLAY CONSOLES Get Region | IEEPDISC |
| DISPLAY U (1) | IEEUNIT1 |
| DISPLAY U (2) | IEEUNIT2 |
| DISPLAY U (3) | IEEUNIT3 |
| DISPLAY U (4) | IEEUNIT4 |
| Log Open Initialization | IEELOG02 |
| Look-Up Routine | IEEVRFRX |
| Master Scheduler Initialization | IEFSD569 |
| Master Scheduler Resident Data Area | IEESD568 |
| Message Module | IEFK1MSG |
| Queue Alter Delete | IEESD576 |
| Queue Message Class Set-Up | IEESD578 |
| Queue Restart Enqueue | IEESD577 |
| Queue Scratch | IEESD581 |
| Queue Scratch Set-Up | IEESD575 |
| Queue Search | IEESD564 |
| Queue Search Set-Up | IEESD563 |
| Queue SMB Routine | IEESD579 |
| Resident Volume Initialization | IEFPRES |
| Service | IEESD565 |

(Part 1 of 2)

Table 9. Master Scheduler Modules (Part 2 of 2)

| Routine | Source Module |
|---|---|
| SMF Initialization (1) | IEESMFIT |
| SMF Initialization (2) | IEESMFI2 |
| SMF MFT Storage Configuration Record Creation | IEEDFINA |
| SMF Open Initializer | IEESMFOI |
| SMF Parameter Processor | IEESMFI3 |
| Specific CANCEL Message Routine | IEESD575 |
| System Log Dispatcher | IEEVLDSP |
| System Log Initialization | IEEVLIN |
| System Log Open Initializer | IEEVLIN2 |
| System Log Output Writer | IEEVLOUT |
| System Log SVC (SVC 36) | IEE0303F |
| System Log SVC (SVC 36 - second load) | IEE0403F |
| System Log Wait Routine | IEELWAIT |
| Syntax Check | IEESD562 |
| User Dummy WTO/WTOR Exit | IEECVCTE |
| Wait/Router | IEECIR50 |
| Write-to-Programmer Error Processing | IEFWTP02 |
| Write-to-Programmer Initialization | IEFWTP00 |
| Write-to-Programmer Message Processor | IEFWTP01 |

• Table 10. Queue Management Modules

| Routine | Source Module |
|---|---|
| Assign | IEFQASGQ |
| Assign/Start | IEFQAGST |
| Branch | IEFQMLK1 |
| Control | IEFQBVMS |
| Delete | IEFQDELQ |
| Dequeue | IEFQMDQQ |
| Dequeue by Jobname | IEFLOCDQ |
| Dequeue by Jobname Interface | IEFSD519 |
| Dummy | IEFQMDUM |
| Enqueue | IEFQMNQQ |
| Interpreter/Queue Manager Interlock | IEFSD572 |
| Message Module | IEFSD311 |
| Queue Formatting | IEFORMAT |
| Queue Initialization | IEFSD055 |
| Queue Manager Table Breakup | IEFSD514 |
| Read/Write | IEFQMRAW |
| Resident Main Storage Reservation | IEFPRESD |
| Transient Queue Manager Initialization | IEFXQM00 |
| Transient Queue Manager Record Assignment | IEFXQM02 |
| Transient Queue Manager Track Assignment | IEFXQM01 |
| Unchain | IEFQMUNQ |

• Table 11. SVC 34 Modules

| Routine | Source Module |
|---|---|
| CANCEL Processor | IEE2803D |
| Command Translator | IEE5403D |
| CSCB and CIB Chain Manipulator | IEE0303D |
| CSCB Creation | IEE0803D |
| DEFINE, MOUNT Routine | IEESD571 |
| DISPLAY Request Processor | IEE2903D |
| DISPLAY Router | IEE3503D |
| HALT (EOD Routine) | IEE1403D |
| HARDCPY Message Routine | IEE4103D |
| LOG and WRITE LOG Routine | IEE1603D |
| Machine Status Control Routine (1) | IGF2603D |
| Machine Status Control Routine (2) | IGF2703D |
| MCS Reply Processor | IEE1A03D |
| MCS Reply Messages | IEE1B03D |
| MCS VARY Syntax Check | IEE3303D |
| Message Assembly | IEE0503D |
| Message Assembly | IEE2103D |
| Periodic STOP Handler (JOBNAMES, STATUS, DSNAME, SPACE) | IEE4503D |
| REPLY Processor | IEE1203D |
| RJE Command Processor | IEE1503D |
| Router | IEE0403D |
| SET Command Processor | IEE0603D |
| START and STOP INIT Processor (1) | IEESD561 |
| START and STOP INIT Processor (2) | IEE3903D |
| STOP and MODIFY Scheduling | IEE0703D |
| SWAP Command Processor | IGF2503D |
| System Management Facility VARY Record Handler | IEE2303D |
| Timer Maintenance | IEE0903D |
| VARY CONSOLE Keyword Scan | IEE4403D |
| VARY CONSOLE Processor | IEE4903D |
| VARY CONSOLE Information Message Routine | IEE4803D |
| VARY HARDCPY Processor | IEE4703D |
| VARY HARDCPY OFF Processor | IEE5703D |
| VARY Keyword Router | IEE3203D |
| VARY MSTCONS Processor | IEE4303D |
| VARY ONGFX/OFFGFX Handler | IEE1703D |
| VARY ON/OFFLINE of Consoles and Message Handler | IEE4603D |
| VARY PATH Command Processor | IGF2403D |
| VARY Secondary Syntax Scan | IEE4203D |
| VARY and UNLOAD Router | IEE1103D |
| VARY and UNLOAD Processor | IEE3103D |

Table 12. 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 | IEFSD084 |

Table 13. 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 |
| Reenqueue | IEFSD305 |
| Scratch Data Sets | IEFSD304 |
| Scratch Data Sets | IEFSD308 |
| TTR and NN to MBBCCHHR Conversion | IEFSD310 |

Table 14. System Task Control Modules

| Routine | Source Module |
|---|---|
| Allocation Interface | IEEVACTL |
| Internal JCL Reader | IEEVRJCL |
| Interpreter Control | IEEVRCTL |
| JCL Edit | IEEVJCL |
| Linkage to IEFSD535 | IEFSD587 |
| Linkage to IEESD591 | IEFSD584 |
| Linkage to IEFDSOSM | IEFSD585 |
| Linkage to IEFSD585 | IEFSD586 |
| 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 | IEEVSMBA |
| START Syntax Check | IEEVSTAR |
| Termination Interface | IEEVTCTL |
| Write TIOT on Disk | IEESD590 |

Table 15. Termination Modules

| Routine | Source Module |
|---|---|
| Disposition and Unallocation Messages | IEFZGMSG |
| VARY Interface and Disposition and Unallocation Messages | IEFZHMSG |
| Disposition and Unallocation | IEFZGJB1 |
| Disposition and Unallocation | IEFZGST1 |
| DSB Processing | IEFYTVMS |
| Dummy Accounting | IEFACTRT |
| 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 |
| Restart Preparation | IEFRPREP |
| SMF Writer Interface | IEFSMFWI |
| 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 Cross Reference

This section contains an alphameric list of entry point and control section names, together with the name of the module that contains them. For further information on the modules, refer to the module descriptions.

| Entry Point or Control Section Name | Module Name |
|---|---|
| GO | IEFSD515 |
| IEAGENQ1 | IEAGENQ1 |
| IEAGENQ2 | IEAGENQ2 |
| IEAMSERB | IEAMSERB |
| IEAQOT00 | IEE0903D |
| IEA0TI01 | IEA0TI01 |
| | |
| IEEBA1 | IEECVCRA |
| IEEBC1PE | IEECVCRX |
| IEECIR45 | IEECVCTW |
| IEECIR50 | IEECIR50 |
| IEECMDOM | IEECMDOM |
| | |
| IEECMDSV | IEECMDSV |
| IEECMWRT | IEECMAWR |
| IEECMWSV | IEECMWSV |
| IEECMWTL | IEECMWTL |
| IEECVCTI | IEECVCTI |
| | |
| IEECVCTR | IEECVCTR |
| IEECVCTW | IEECMAWR |
| IEECVPM | IEECVPM |
| IEECVPRG | IEECVED2 |
| IEECVXIT | IEECVCTE |
| | |
| IEEDFINA | IEEDFINA |
| IEEDFINB | IEEDFINB |
| IEEDFINC | IEEDFINC |
| IEEDFIN1 | IEEFIN1 |
| IEEDFIN2 | IEEDFIN2 |
| | |
| IEEDFIN3 | IEEDFIN3 |
| IEEDFIN4 | IEEDFIN4 |
| IEEDFIN5 | IEEDFIN5 |
| IEEDFIN6 | IEEDFIN6 |
| IEEDFIN7 | IEEDFIN7 |
| | |
| IEEDFIN8 | IEEDFIN8 |
| IEEDFIN9 | IEEDFIN9 |
| IEEDPART | IEEDFIN2 |
| IEELOG02 | IEELOG02 |
| IEELWAIT | IEELWAIT |
| | |
| IEEMSER | IEESD568 |
| IEEMSTWO | IEESD579 |
| IEEPDISC | IEEPDISC |
| IEEPSN | IEEPSN |
| IEESD562 | IEESD562 |
| IEESD563 | IEESD563 |
| IEESD564 | IEESD564 |
| IEESD565 | IEESD565 |
| IEESD566 | IEESD566 |
| IEESD567 | IEESD567 |
| | |
| IEESD575 | IEESD575 |
| IEESD576 | IEESD576 |
| IEESD577 | IEESD577 |
| IEESD578 | IEESD578 |
| IEESD579 | IEESD579 |
| | |
| IEESD580 | IEESD580 |
| IEESD581 | IEESD581 |
| IEESD590 | IEESD590 |
| IEESD591 | IEESD591 |
| IEESD592 | IEESD592 |
| | |
| IEESMFAL | IEESMFAL |
| IEESMFIO | IEESMFI3 |
| IEESMFIT | IEESMFIT |
| IEESMFI2 | IEESMFI2 |
| IEESMFI3 | IEESMFI3 |
| | |
| IEESMFI4 | IEESMFIT |
| IEESMFMS | IEESMFI3 |
| IEESMFOI | IEESMFOI |
| IEESMFOP | IEESMFOP |
| IEESMFWT | IEESMFWT |
| | |
| IEEUNIT1 | IEEUNIT1 |
| IEEUNIT2 | IEEUNIT2 |
| IEEUNIT3 | IEEUNIT3 |
| IEEUNIT4 | IEEUNIT4 |
| IEEVACTL | IEEVACTL |
| | |
| IEEVJCL | IEEVJCL |
| IEEVLIN | IEEVLIN |
| IEEVLDSP | IEEVLDSP |
| IEEVLNKT | IEEVLNKT |
| IEEVLOUT | IEEVLOUT |
| | |
| IEEVMSG1 | IEEVMSG1 |
| IEEVOMSG | IEEVOMSG |
| IEEVRCTL | IEEVRCTL |
| IEEVRFRX | IEEVRFRX |
| IEEVSMBA | IEEVSMBA |
| | |
| IEEVSMSG | IEEVSMSG |
| IEEVSTAR | IEEVSTAR |
| IEEVTCTL | IEEVTCTL |
| IEEXEDNA | IEEXEDNA |
| IEE0303D | IEE0303D |
| | |
| IEE0303F | IEE0303F |
| IEE0403D | IEE0403D |
| IEE0403F | IEE0403F |
| IEE0503D | IEE0503D |
| IEE0603D | IEE0603D |

(Continued)

| Entry Point or Control Section Name | Module Name |
|---|---|
| IEE0703D | IEE0703D |
| IEE0803D | IEE0803D |
| IEE0903D | IEE0903D |
| IEE1103D | IEE1103D |
| IEE1203D | IEE1203D |
| IEE1403D | IEE1403D |
| IEE1603D | IEE1603D |
| IEE1703D | IEE1703D |
| IEE1A03D | IEE1A03D |
| IEE1B03D | IEE1B03D |
| IEE2103D | IEE2103D |
| IEE2303D | IEE2303D |
| IEE2603D | IEE2603D |
| IEE2803D | IEE2803D |
| IEE2903D | IEE2903D |
| IEE3103D | IEE3103D |
| IEE3203D | IEE3203D |
| IEE3303D | IEE3303D |
| IEE3503D | IEE3503D |
| IEE3903D | IEE3903D |
| IEE4103D | IEE4103D |
| IEE4203D | IEE4203D |
| IEE4303D | IEE4303D |
| IEE4403D | IEE4403D |
| IEE4503D | IEE4503D |
| IEE4603D | IEE4603D |
| IEE4703D | IEE4703D |
| IEE4803D | IEE4803D |
| IEE4903D | IEE4903D |
| IEE5403D | IEE5403D |
| IEE5703D | IEE5703D |
| IEE591SD | IEESD591 |
| IEFACTLK | IEFACTFK |
| IEFACTLK | IEFACTLK |
| IEFACTRT | IEFACTRT |
| IEFALRET | IEFSD512 |
| IEFCVOL1 | IEFCVFAK |
| IEFCVOL1 | IEFMCVOL |
| IEFCVOL2 | IEFCVFAK |
| IEFCVOL2 | IEFMCVOL |
| IEFCVOL3 | IEFCVFAK |
| IEFCVOL3 | IEFMCVOL |
| IEFDPOST | IEFVPOST |
| IEFDSDRP | IEFDSDRP |
| IEFDSOAL | IEFDSOAL |
| IEFDSOCP | IEFDSOCP |
| IEFDSOFB | IEFDSOFB |
| IEFDSOSM | IEFDSOSM |
| IEFDSOWR | IEFDSOWR |
| IEFICR | IEEVRJCL |

| Entry Point or Control Section Name | Module Name |
|---|---|
| IEFIDMPM | IEFIDMPM |
| IEFIRC | IEFSD533 |
| IEFJOB | IEFQRESD |
| IEFKG | IEFSD532 |
| IEFORMAT | IEFORMAT |
| IEFPH2 | IEFSD531 |
| IEFQAGST | IEFQAGST |
| IEFQASGN | IEFQASGQ |
| IEFQASNM | IEFQASGQ |
| IEFQDELE | IEFQDELQ |
| IEFQMDQ2 | IEFQMDQQ |
| IEFQMDUM | IEFQMDUM |
| IEFQMNQ2 | IEFQMNQQ |
| IEFQMSSS | IEFQBVMS |
| IEFQMSSS | IEFQMDUM |
| IEFQMSSS | IEFQMLK1 |
| IEFQMRAW | IEFQMRAW |
| IEFQMUNC | IEFQMUNQ |
| IEFRCLN1 | IEFRCLN1 |
| IEFRCLN2 | IEFRCLN2 |
| IEFRPREP | IEFRPREP |
| IEFRSTRT | IEFRSTRT |
| IEFSD012 | IEFVSD12 |
| IEFSD017 | IEFSD017 |
| IEFSD055 | IEFSD055 |
| IEFSD068 | IEFSD167 |
| IEFSD068 | IEFSD168 |
| IEFSD070 | IEFSD070 |
| IEFSD071 | IEFSD171 |
| IEFSD078 | IEFSD078 |
| IEFSD078 | IEF078SD |
| IEFSD079 | IEFSD079 |
| IEFSD079 | IEF079SD |
| IEFSD080 | IEFSD080 |
| IEFSD081 | IEFSD081 |
| IEFSD082 | IEF082SD |
| IEFSD082 | IEFSD082 |
| IEFSD083 | IEFSD083 |
| IEFSD083 | IEF083SD |
| IEFSD084 | IEFSD084 |
| IEFSD085 | IEFSD085 |
| IEFSD086 | IEFSD086 |
| IEFSD087 | IEFSD087 |
| IEFSD088 | IEFSD088 |
| IEFSD089 | IEFSD089 |
| IEFSD090 | IEFVSD13 |
| IEFSD094 | IEFSD094 |
| IEFSD095 | IEFSD095 |
| IEFSD095 | IEFSD195 |
| IEFSD096 | IEFSD096 |

(Continued)

| Entry Point or Control Section Name | Module Name |
|---|---|
| IEFSD097 | IEFSD097 |
| IEFSD300 | IEF300SD |
| IEFSD300 | IEFSD300 |
| IEFSD301 | IEFSD301 |
| IEFSD302 | IEFSD302 |
| IEFSD303 | IEFSD303 |
| IEFSD304 | IEFSD304 |
| IEFSD304 | IEF304SD |
| IEFSD305 | IEFSD305 |
| IEFSD308 | IEFSD308 |
| IEFSD310 | IEFSD310 |
| IEFSD311 | IEFSD311 |
| IEFSD312 | IEFSD312 |
| IEFSD41R | IEF41DUM |
| IEFSD510 | IEFSD510 |
| IEFSD511 | IEFSD511 |
| IEFSD512 | IEFSD512 |
| IEFSD512 | IEFSD553 |
| IEFSD513 | IEFSD513 |
| IEFSD514 | IEFSD514 |
| IEFSD515 | IEFSD515 |
| IEFSD516 | IEFSD516 |
| IEFSD517 | IEFSD517 |
| IEFSD518 | IEFSD518 |
| IEFSD519 | IEFSD519 |
| IEFSD530 | IEFSD530 |
| IEFSD531 | IEFSD531 |
| IEFSD534 | IEFSD534 |
| IEFSD535 | IEFSD535 |
| IEFSD537 | IEFSD537 |
| IEFSD540 | IEFSD540 |
| IEFSD541 | IEFSD541 |
| IEFSD554 | IEFSD554 |
| IEFSD555 | IEFSD555 |
| IEFSD556 | IEFSD556 |
| IEFSD557 | IEFSD557 |
| IEFSD558 | IEFSD558 |
| IEFSD559 | IEFSD559 |
| IEFSD567 | IEFSD567 |
| IEFSD569 | IEFSD569 |
| IEFSD572 | IEFSD572 |
| IEFSD573 | IEFSD572 |
| IEFSD584 | IEFSD584 |
| IEFSD585 | IEFSD585 |
| IEFSD586 | IEFSD586 |
| IEFSD587 | IEFSD587 |
| IEFSD588 | IEFSD588 |
| IEFSD589 | IEFSD589 |
| IEFSD598 | IEFSD597 |
| IEFSD598 | IEFSD598 |

| Entry Point or Control Section Name | Module Name |
|---|---|
| IEFSD599 | IEFSD599 |
| IEFSD71M | IEFSD171 |
| IEFSD83M | IEFSD083 |
| IEFSD85M | IEFSD085 |
| IEFSD86M | IEFSD086 |
| IEFSD877 | IEFSD087 |
| IEFSD897 | IEFSD089 |
| IEFSMFAT | IEFSMFAT |
| IEFSMFIE | IEFSMFIE |
| IEFSMFWI | IEFSMFWI |
| IEFSMR | IEFRSTRT |
| IEFUCBL | IEFWA000 |
| IEFUJI | IEFUJI |
| IEFUJV | IEFUJV |
| IEFUSI | IEFUSI |
| IEFUSO | IEFUSO |
| IEFUTL | IEFUTL |
| IEFVAWAT | IEFSD195 |
| IEFVDA | IEFVDA |
| IEFVDBSD | IEFVDBSD |
| IEFVEA | IEFVEA |
| IEFVFA | IEFVFA |
| IEFVFB | IEFVFB |
| IEFVGI | IEFVGI |
| IEFVGK | IEFVGK |
| IEFVGM | IEFVGM |
| IEFVGM1 | IEFVGM1 |
| IEFVGM2 | IEFVGM2 |
| IEFVGM3 | IEFVGM3 |
| IEFVGM4 | IEFVGM4 |
| IEFVGM5 | IEFVGM5 |
| IEFVGM6 | IEFVGM6 |
| IEFVGM7 | IEFVGM7 |
| IEFVGM8 | IEFVGM8 |
| IEFVGM9 | IEFVGM9 |
| IEFVGM10 | IEFVGM10 |
| IEFVGM11 | IEFVGM11 |
| IEFVGM12 | IEFVGM12 |
| IEFVGM13 | IEFVGM13 |
| IEFVGM14 | IEFVGM14 |
| IEFVGM15 | IEFVGM15 |
| IEFVGM16 | IEFVGM16 |
| IEFVGM17 | IEFVGM1M |
| IEFVGM18 | IEFVGM18 |
| IEFVGM19 | IEFVGM19 |
| IEFVGM70 | IEFVGM70 |
| IEFVGM71 | IEFVGM71 |
| IEFVGM78 | IEFVGM78 |
| IEFVGS | IEFVGS |
| IEFVGT | IEFVGT |

(Continued)

| Entry Point or Control Section Name | Module Name |
|---|---|
| IEFVHA | IEFVHA |
| IEFVHAA | IEFVHAA |
| IEFVHB | IEFVHB |
| IEFVHC | IEFVHC |
| IEFVHCB | IEFVHCB |
| IEFVHE | IEFVHE |
| IEFVHEB | IEFVHEB |
| IEFVHEC | IEFVHEC |
| IEFVHF | IEFVHF |
| IEFVHG | IEFVHG |
| IEFVHH | IEFVHH |
| IEFVHHB | IEFVHHB |
| IEFVHL | IEFVHL |
| IEFVHM | IEFVHM |
| IEFVHN | IEFVHN |
| IEFVHQ | IEFVHQ |
| IEFVHR | IEFSD536 |
| IEFVH1 | IEFVH1 |
| IEFVH2 | IEFVH2 |
| IEFVINA | IEFVINA |
| IEFVINB | IEFVINB |
| IEFVINC | IEFVINC |
| IEFVIND | IEFVIND |
| IEFVINE | IEFVINE |
| IEFVJ | IEFVJIMP |
| IEFVJA | IEFVJA |
| IEFVJMSG | IEFVJMSG |
| IEFVK | IEFVKIMP |
| IEFVKMJ1 | IEFVKMSG |
| IEFVKMSG | IEFVKMSG |
| IEFVM | IEFVMLS1 |
| IEFVMCVL | IEFVMFAK |
| IEFVMCVL | IEFVMLS1 |
| IEFVMQMI | IEFVMLS1 |
| IEFVMSGR | IEFVMLS6 |
| IEFVM1 | IEFVMLS1 |
| IEFVM1 | IEFVMMS1 |
| IEFVM2 | IEFVM2LS |
| IEFVM3 | IEFVM3LS |
| IEFVM4 | IEFVM4LS |
| IEFVM5 | IEFVM5LS |
| IEFVM6 | IEFVMLS6 |
| IEFVM7 | IEFVMLS7 |
| IEFVM76 | IEFVM76 |
| IEFVRRC | IEFVRRC |
| IEFVRRCA | IEFVRRC |
| IEFVRRCB | IEFVRRC |
| IEFVRR1 | IEFVRR1 |
| IEFVRR2 | IEFVRR2 |
| IEFVRR3 | IEFVRR3 |

| Entry Point or Control Section Name | Module Name |
|---|---|
| IEFVSDRA | IEFVSDRA |
| IEFVSDRD | IEFVSDRD |
| IEFVSMBR | IEFVSMBR |
| IEFV15XL | IEFXJIMP |
| IEFV15XL | IEFSD551 |
| IEFVR2AE | IEFVRR2 |
| IEFVR3AE | IEFVRR3 |
| IEFWA000 | IEFWA000 |
| IEFWA002 | IEFWA000 |
| IEFWA7 | IEFWA000 |
| IEFWC000 | IEFWCFAK |
| IEFWC000 | IEFWCIMP |
| IEFWC002 | IEFWCIMP |
| IEFWD000 | IEFWDFAK |
| IEFWD000 | IEFWD000 |
| IEFWD001 | IEFWD001 |
| IEFWDMSG | IEFWD000 |
| IEFWLISD | IEFSD21Q |
| IEFWSTRT | IEFWSTRT |
| IEFWSWIT | IEFWSWIN |
| IEFWTERM | IEFWTERM |
| IEFW1FAK | IEFSD41Q |
| IEFW1FAK | IEF41FAK |
| IEFW2FAK | IEFSD41Q |
| IEFW2FAK | IEF41FAK |
| IEFW21SD | IEFSD21Q |
| IEFW21SD | IEFSD557 |
| IEFW22SD | IEFSD22Q |
| IEFW31SD | IEFSD31Q |
| IEFW32SD | IEFSD32Q |
| IEFW32SD | IEFSD33Q |
| IEFW41SD | IEFSD41Q |
| IEFW41SD | IEF41FAK |
| IEFW42SD | IEFSD42Q |
| IEFXA | IEFXCSSS |
| IEFXAMSG | IEFXAMSG |
| IEFXH000 | IEFXH000 |
| IEFXJMSG | IEFXJMSG |
| IEFXJX5A | IEFSD552 |
| IEFXJX5A | IEFXJIMP |
| IEFXJ000 | IEFXJFAK |
| IEFXJ000 | IEFXJIMP |
| IEFXKMSG | IEFXKMSG |
| IEFXK000 | IEFXKIMP |
| IEFXT000 | IEFXT00D |
| IEFXT002 | IEFXT002 |
| IEFXT003 | IEFXT003 |
| IEFXVMSG | IEFXVMSG |
| IEFXVNSL | IEFXVNSL |
| IEFXV001 | IEFAVFAK |

(Continued)

| Entry Point or Control Section Name | Module Name |
|---|---|
| IEFXV001 | IEFXV001 |
| IEFXV002 | IEFXV002 |
| IEFX3000 | IEFX300A |
| IEFX5000 | IEFX5000 |
| IEFYN | IEFYNIMP |
| IEFYNMSG | IEFYNMSG |
| IEFYP | IEFYPJB3 |
| IEFYPMSG | IEFYPMSG |
| IEFYS | IEFYSVMS |
| IEFYT | IEFYTVMS |
| IEFZA | IEFZAJB3 |
| IEFZG | IEFZGST1 |
| IEFZGJ | IEFZGJB1 |
| IEFZGMSG | IEFZGMSG |
| IEFZH | IEFZHMSG |
| IEF085SD | IEFSD085 |
| IEF086SD | IEFSD086 |
| IEF850SD | IEFSD085 |
| IEG056 | IEAGENQ1 |
| IEG056 | IEAGENQ2 |
| IEZDCODE | IEZDCODE |
| IEZNCODE | IEZNCODE |
| IGCXL07B | IEECMCSW |
| IGCXM07B | IEECMCSW |
| IGCXN07B | IEECMCSW |
| IGCX007B | IEECMCSW |
| IGC0001C | IEANTM00 |
| IGC0003E | IEECVWTO |
| IGC0005B | IEFVSMBR |
| IGC00060 | IEAAST00 |
| IGC00083 | IGC0008C |
| IGC00090 | IEFXQM00 |
| IGC0101C | IEANTM01 |
| IGC0103D | IGC0103D |
| IGC0103E | IEEVWTOR |
| IGC01090 | IEFXQM01 |
| IGC0111C | IEASTM11 |
| IGC0201C | IEANTM02 |
| IGC0203E | IEFWTP00 |
| IGC02090 | IEFXQM02 |
| IGC0211C | IEASTM12 |
| IGC0221C | IEADTM22 |
| IGC0301C | IEANTM03 |
| IGC0303E | IEFWTP01 |
| IGC0311C | IEANTM13 |
| IGC0321C | IEADTM23 |
| IGC0401C | IEANTM04 |
| IGC0403E | IEFWTP02 |
| IGC0411C | IEANTM14 |
| IGC048 | IEAGENQ1 |

| Entry Point or Control Section Name | Module Name |
|---|---|
| IGC048 | IEAGENQ2 |
| IGC0501C | IEAMTM05 |
| IGC0501C | IEANTM05 |
| IGC0601C | IEANTM06 |
| IGC0701C | IEANTM07 |
| IGC0801C | IEANTM08 |
| IGC0901C | IEANTM09 |
| IGC0907B | IEECMWTL |
| IGC0A01C | IEANTM0A |
| IGC0B01C | IEACTM0B |
| IGC0B01C | IEANTM0B |
| IGC0C01C | IEANTM0C |
| IGC0D01C | IEANTM0D |
| IGC0E01C | IEANTM0E |
| IGC1803D | IEESD571 |
| IGC1903D | IEESD561 |
| IGF2403D | IGF2403D |
| IGF2503D | IGF2503D |
| IGF2603D | IGF2603D |
| IGF2703D | IGF2703D |
| LCC | IEFLOCDQ |
| LCCCAN | IEFLOCDQ |
| LCCDQ | IEFLOCDQ |
| MSOFF | IEFXJMSG |
| MSRCV | IEFXJMSG |
| MSSYS | IEFXJMSG |
| SD304MG1 | IEFSD312 |
| SD304MG2 | IEFSD312 |
| SD305MG1 | IEFSD312 |
| SD55MSG1 | IEFSD311 |
| SD55MSG2 | IEFSD311 |
| SD55MSG3 | IEFSD311 |
| SMALLGO | IEFSD599 |
| SMALTERM | IEFSD515 |
| SMALTERM | IEFSD559 |
| SPRINTER | IEFPRINT |
| STRMSG01 | IEFYNMSG |
| VM7000 | IEFVMLS1 |
| VM7055 | IEFVMLS1 |
| VM7055AA | IEFVMLS1 |
| VM7060 | IEFVMLS1 |
| VM7065 | IEFVMLS1 |
| VM7070 | IEFVMLS1 |
| VM7090 | IEFVMLS1 |
| VM7100 | IEFVM2LS |

(Continued)

| Entry Point or Control Section Name | Module Name |
|---|---|
| VM7130 | IEFVMLS1 |
| VM7150 | IEFVM3LS |
| VM7200 | IEFVM4LS |
| VM7300 | IEFVM5LS |
| VM7370 | IEFVMLS1 |
| VM7600 | IEFVM76 |
| VM7700 | IEFVMLS1 |
| VM7742 | IEFJMLS1 |
| VM7750 | IEFVMLS1 |
| VM7850 | IEFVMLS1 |
| VM7900 | IEFVMLS1 |
| VM7950 | IEFVMLS1 |
| XIIB32 | IEFX5000 |
| XTTEA0 | IEFXT002 |
| XTTEA1 | IEFXT002 |

| Entry Point or Control Section Name | Module Name |
|---|---|
| XTTEB3 | IEFXT002 |
| XTTP00 | IEFXT00D |
| XTTRDJ | IEFXT002 |
| XUUB00 | IEFXT003 |
| XUUHO6 | IEFXT003 |
| X33B42 | IEFX300A |
| X55C86 | IEFX5000 |
| X55D3G | IEFX5000 |
| YPPMSG1 | IEFYPMSG |
| YPPMSG2 | IEFYPMSG |
| XPS631 | IEFZHMSG |
| ZG0E60 | IEFZHMSG |
| ZK0D1 | IEFZHMSG |
| ZK0E1 | IEFZHMSG |
| ZPOQM | IEFZGJB1 |
| ZP0QMGR1 | IEFZGST1 |

## Module Descriptions

This section contains a brief description of each of the modules used by MFT. Modules are listed alphamerically 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 module's entry points, the names of the modules to which it passes control, the major tables and work areas to which it refers, its attributes and the names of the control sections it contains.

IEAAST00: Supervisor -- STAE Service Routine

This routine is entered when the STAE macro instruction (SVC 60) is issued. It creates, cancels, or overlays a STAE control block according to the options specified. It prepares the task to intercept the scheduled abnormal termination (ABEND) processing.

- Entry: IGC00060
- Attributes: Non-resident, reentrant
- Control Section: IGC00060

IEACTM0B: Supervisor -- ABEND WTOR Purge Routine (MFT with MCS)

This routine purges outstanding WTOR requests.

- Entry: IGC0B01C from IEANTM00 or IEANTM01
- Exits: IEANTM00 for normal job step termination.
  IEANTM02 for abnormal termination.
  IEANTM0E for normal subtask termination (MFT with subtasking only).
  IEASTM13 for STAE processing.

IEADTM22: Supervisor -- DAR Core Image Dump Routine

This routine attempts to write a core image dump to a preallocated data set. It also processes primary DAR recursions.

- Entry: IGC0221C
- Exits: XCTL to IEADTM23 to continue DAR processing
- Attributes: Refreshable, disabled, privileged
- Control Section: IGC0221C

IEADTM23: Supervisor -- DAR Task Reinstatement Routine

This routine attempts reinstatement of failing system tasks. It also processes secondary DAR recursions and failing tasks which are in "Must Complete" status.

- Entry: IGC0321C
- Exits: XCTL to IEANTM07 if a jobstep task with no subtasks
  IEANTM0C if a subtask
  IEANTM0D if task has subtask that needs ENQ purge
  IEANTM0E if subtasks do not need ENQ purge
- Attributes: Refreshable, disabled, privileged
- Control Section: IGC0321C

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

IEAMSERB: System Error Task Reinitialization

This routine contains the wait RB for the system error task. It creates an FQE for the main storage occupied by the system error task when reinstatement of the failing system error task is attempted.

- Entry: IEAMSERB from IEADTM23
- Exit: IEA0DS (dispatcher) via SVC 3
- Attributes: Resident, reusable, disabled
- Control Section: IEAMSERB

IEAMTM05: Supervisor -- ABEND Open Dump Data Set Routine (MFT without subtasking)

This routine opens the dump data set for the terminating task.

- Entry: IGC0501C from IEANTM04
- Exits: IEANTM06 for normal open
  IEANTM09 if SYSABEND or SYSUDUMP entry not found in TIOT, or OPEN macro instruction failed
- Attributes: Reentrant, refreshable
- Control Section: IGC0501C

IEANTM0A: Supervisor -- ABEND Steal Main Storage Routine

This routine steals main storage to enable further successful ABEND processing.

- Entry: IGC0A01C from IEANTM04
- Exits: IEANTM04
- Attributes: Reentrant, refreshable
- Control Section: IGC0A01C

IEANTM0B: Supervisor -- ABEND WTOR Purge Routine (MFT without MCS)

This routine purges outstanding WTOR requests for the terminating task.

- Entry: IGC0B01C from IEANTM00 or IEANTM01
- Exits: IEANTM00 for normal job step termination.
  IEANTM02 for abnormal job step termination.
  IEANTM0E for normal subtask termination (MFT with subtasking only)

  IEASTM13 for STAE processing
- Attributes: Reentrant, refreshable
- Control Section: IGC0B01C

IEANTM0C: Supervisor -- ABEND Loading Program Purge Routine (MFT with subtasking only)

This routine frees partially loaded programs and FRBs for the terminating task.

- Entry: IGC0C01C from IEANTM04 or IEANTM06
- Exit: IEANTM0D
- Attributes: Reentrant, refreshable
- Control Section: IGC0C01C

IEANTM0D: Supervisor -- ABEND Subtask ENQ Purge Routine (MFT with subtasking only)

This routine performs an ENQ purge for all subtasks of the terminating task.

- Entry: IGC0D01C from IEANTM04, IEANTM06, IEANTM08, IEANTM0C
- Exit: IEANTM0E
- Attributes: Reentrant, refreshable
- Ccntrol Section: IGC0D01C

IEANTM0E: Supervisor -- ABEND IQE Purge and Data Set Close Routine (MFT with subtasking only)

This routine purges IQEs for the subtasks of the terminating task and closes data sets for all tasks and subtasks except an abnormally terminating job step task.

- Entry: IGC0E01C from IEANTM00, IEANTM04, IEANTM06, IEANTM08, IEACTM0B, IEANTM0B, IEANTM0D
- Exits: IEANTM07 for abnormally terminating job step task, SVC EXIT routine for subtask terminating abnormally, or job step task terminating normally.
- Exits:

IEANTM00: Supervisor -- ABEND Normal Termination Processing and Abnormal Terminatcn Router Routine

This routine processes normal task terminations or passes control to other ABEND modules for further processing of abnormal terminations.

- Entry: IGC0001C from SVC SLIH
  from IEACTM0B if MCS is in the system.

from IEANTM0B after WTOR requests are purged

- Exits:
  Normal
  IEFSD515 for a large partition
  IEFSD599 for a small partition
  IEACTM0B When MCS is included in the system
  IEANTM0B when there are WTOR requests to purge
  IEANTM0E when the terminating task is a subtask (MFT with subtasking only)

  Abnormal
  IEADTM22 for primary DAR recursion
  IEADTM23 for secondary DAR recursion
  IEANTM02 for OPEN, CLOSE, ABDUMP, or message recursion
  IEANTM01 for nonrecursive termination

- Attributes: Reentrant, refreshable

- Control Section: IGC0001C

IEANTM01: Supervisor -- ABEND/STAE Graphics Test Routine

This routine processes nonrecursive terminations for job step tasks, failures after a valid STAE has been issued, purge failures in STAE, or failures when GJP is included in the system.

- Entry: IGC0101C from IEANTM00

- Exits: IEASTM11 for failure after a STAE was issued or if a purge failure occurred within STAE. To caller if graphics program (GJP).
  IEANTM0B to further process nonrecursive terminations
  IEACTM0B to further process nonrecursive terminations if MCS is included in the system

- Attributes: Reentrant, refreshable

- Control Section: IGC0101C

IEANTM02: Supervisor -- ABEND I/O Purge Routine

This routine purges I/O operations in process and I/O requests for the terminating task.

- Entry: IGC0201C from IEANTM00, IEACTM0B if MCS is in the system, IEANTM0B

- Exits: IEADTM22 when the abnormally terminating task is a system task or is in "must complete" status
  IEANTM03 for nonrecursive ABENDs
  IEANTM09 for recursive ABENDs

- Attributes: serially reusable

- Control Section: IGC0201C

IEANTM03: Supervisor -- ABEND Control Block Validity Check Routine

This routine ensures the validity of FQEs, DEBs, active RBs, GQEs, load list RBs, and JPAQ RBs. If an invalid control block is discovered, it truncates the chain up to the last valid control block with the exception of an invalid FQE, which is merely dequeued.

- Entry: IGC0301C from IEANTM02

- Exit: IEANTM04

- Attributes: Reentrant, refreshable

- Control Section: IGC0301C

IEANTM04: Supervisor -- ABEND Dump Test Routine

This routine determines the type of dump, if any, that is required, and determines if enough main storage is available for providing the dump.

- Entry: IGC0401C from IEANTM03, IEANTM09, IEANTM0A

- Exits: IEANTM05 for full dump (MFT with subtasking only)
  IEAMTM05 for full dump
  IEANTM08 for indicative dump
  IEANTM0A when there is insufficient main storage available
  IEANTM0C when the ABEND is for a subtask (MFT with subtasking only)
  IEANTM0D when subtasks need ENQ purge (MFT with subtasking only)
  IEANTM0E when subtasks do not need ENQ purge (MFT with subtasking only)
  IEANTM07 for termination of a job step with no subtasks

- Attributes: Reentrant, refreshable

- Control Section: IGC0401C

IEANTM05: Supervisor -- ABEND Open Dump Data Set Routine (MFT with subtasking)

This routine replaces IEAMTM05 when MFT with subtasking is included in the system. In addition to opening the dump data set, IEANTM05 also dequeues any subtasks of the terminating task that are enqueued on the dump data set. It issues an ENQ macro instruction on the dump data set to prevent other tasks from using it until the dump is finished.

- Entry: IGC0501C from IEANTM04
- Exits: IEANTM06 for a successful OPEN
  IEANTM08 if a SYSABEND or SYSUDUMP entry is not found in TIOT, or the OPEN failed
  IEA0DS if a subtask is in the process of opening the dump data set
- Attributes: Reentrant, refreshable
- Control Section: IGC0501C

IEANTM06: Supervisor -- ABEND Dump Routine

This routine issues a WRITE macro instruction to print appropriate messages pertinent to the dump and issues a SNAP macro instruction to call ABDUMP for the terminated task.

- Entry: IGC0601C from IEANTM05
- Exits: IEANTM07 when the terminating task is a job step task with no subtasks
  IEANTM09 if ABDUMP failed
  IEANTM0C if a subtask is terminating (MFT with subtasking only)
  IEANTM0D if subtasks need ENQ purge (MFT with subtasking only)
  IEANTM0E if subtasks do not need ENQ purge (MFT with subtasking only)
- Attributes: Reentrant, refreshable
- Control Section: IGC0601C

IEANTM07: Supervisor -- ABEND Termination Routine

This routine closes all data sets associated with the terminating task, and dequeues IQEs.

- Entry: IGC0701C from IEANTM04, IEANTM06, IEANTM08, IEANTM0E
- Exits: IEFSD515 for a large partition
  IEFSD599 for a small partition
- Attributes: Reentrant, refreshable
- Control Section: IGC0701C

IEANTM08: Supervisor -- ABEND Indicative Dump Routine

This routine stores pertinent information in a dump area for printing by the job management routines.

- Entry: IGC0801C from IEANTM04.
- Exits: IEANTM07 when terminating task is a job step task with no subtasks
  IEANTM0D when subtasks need ENQ purge (MFT with subtasking only)
  IEANTM0E when subtasks do not need ENQ purge (MFT with subtasking only)
- Attributes: Reentrant, refreshable
- Control Section: IGC0801C

IEANTM09: Supervisor -- ABEND Recursion Processing Routine

This routine dequeues all request blocks created as a result of the recursion and processes OPEN, CLOSE, ABDUMP, and message recursions.

- Entry: IGC0901C from IEANTM02, IEANTM05, IEANTM06
- Exits: IEANTM04
- Attributes: Reentrant, refreshable
- Control Section: IGC0901C

IEASTM11: Supervisor -- ABEND/STAE Interface Routine

This routine purges I/O for the terminating task and schedules the user exit routine.

- Entry: IGC0111C
- Exits: IEANTM00 (IEACTM0B for MCS) if the failing task is in "must complete" status and the STAE user is a problem program, or if the STAE control block is not associated with an RB, the retry without RB purge option is selected and the user is a problem program
  IEASTM12 if the retry with purge of RBs option is selected
  IEASTM13 if the retry without purge of RBs option is selected

- Attributes: Reentrant, refreshable, disabled, privileged, non-resident
- Control Section: IGC0111C

IEASTM12: Supervisor -- ABEND/STAE Interface Routine

This module closes data sets associated with RBs on the queue between the request block for the failing problem program and the request block for the current STAE processing.

- Entry: IGC0211C from IEASTM11
- Exits: IEANTM0B for WTOR request purge (IEACTMCB for MCS)
  IEASTM14 for data sets using ISAM, BTAM, or QTAM.
- Attributes: Reentrant, refreshable, disabled, privileged, non-resident.
- Control Section: IGC0211C

IEASTM13: Supervisor -- ABEND/STAE Interface Routine

This routine creates, initializes and queues a request block for a retry routine if the RB purge option was not selected. Otherwise, the STAE requestors RB is used for the retry routine.

- Entry: IGC0311C from IEANTM0B (IEACTMOB for MCS)
  or from IEASTM11
- Exits: SVC EXIT routine
- Attributes: Reentrant, refreshable, disabled, privileged, non-resident
- Control Section: IGC0311C

IEASTM14: Supervisor -- ABEND/STAE Interface Routine

This routine closes ISAM, BTAM, and QTAM data sets associated with RBs on the queue between the request block for the failing problem program and the request block for the current STAE processing.

- Entry: IGC0411C from IEASTM12
- Exit: IEANTM0B to purge WTOR requests (IEACTM0B for MCS)
- Attributes: Reentrant, refreshable, disabled, provileged, non-resident
- Control Section: IGC0411C

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

IEECMAWR: Communications Task -- Router Module

This module waits for the posting of a communications task ECB, determines the type of interruption service required (external, attention, I/O, WTO, or DOM), and passes control to other communications task modules for further processing.

- Entry: IEECMWRT
- Exit: IEECMCSW, IEECMDSV, IEECMWSV, IEECMWTL, IEECMDOM, Dispatcher
- Tables/Work Areas: CVT, EIL, UCM, WQE
- Attributes: Reentrant, refreshable
- Control Section: IEECVCTW

IEECMCSW: Communications Task -- Console Switch Module

This routine provides console switching as a result of an unrecoverable I/O error on a console device, as a result of an external interruption, or as a result of a VARY command, and provides hard copy switching from a console device of SYSLOG.

- Entry: IGCXL07B
- Exit: IEECMAWR, IEECMDSV
- Tables/Work Areas: CVT, CXSA, RQE, UCM, WQE
- Attributes: Reentrant, refreshable
- Control Sections: IGCXL07B, IGCXM07B, IGCXN07B, IGCXO07B

IEECMDOM: Communications Task -- DOM Service Module

This module marks for deletion specified WQEs on the system output queue.

- Entry: IEECMDOM
- Exit: IEECMAWR, IEECVDT1
- Tables/Work Areas: CVT, DCM, UCM, WQE
- Attributes: Reentrant, refreshable
- Control Section: IEECMDOM

IEECMDSV: Communications Task -- Device Service Module

This module provides the interface with device support processors and provides console and system output queue management.

- Entry: IEECMDSV
- Exit: IEECMAWR, IEECMWSV, IEECMCSW, Device Support Processors
- Tables/Work Areas: IEEBASEB, CVT, EIL, UCM, WQE
- Attributes: Reentrant, refreshable
- Control Section: IEECMDSV

IEECMWSV: Communications Task -- WTO(R) Service Module

This module puts unprocessed WQEs on appropriate console output queues.

- Entry: IEECMWSV
- Exit: IEECMDSV, IEECMAWR
- Tables/Work Areas: UCM, WQE
- Attributes: Reentrant, refreshable
- Control Section: IEECMWSV

IEECMWTL: Communications Task -- NIP Message Buffer Writer Module

This module issues SVC 36 to write NIP messages to SYSLOG. If SYSLOG has not been initialized or not specified as the hard copy log, it issues SVC 35 to write the NIP messages to the operator.

- Entry: IEECMWTL
- Exit: Return to caller
- Control Section: IGC0907B

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

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

IEECVCTE: Communications Task -- User Dummy WTO/WTOR Exit Routine

This routine takes the place of the user's WTO/WTOR exit routine when an exit routine was specified at system generation, but none was supplied.

- Entry: IEECVXIT, from IEECMWSV
- Exit: Return to caller
- Control Section: IEECVXIT

IEECVCTI: Console Initialization Routine

This routine prints out the NIP message buffer in systems with the MCS option, and initializes the console configuration.

- Entry: IEECVCTI, from IEESD569
- Exit: To IEESD569
- Tables/Work Areas: CVT, EIL, UCB, and UCM
- Attributes:
- Control Section: IEECVCTI

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

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

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

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

This routine processes all WTO macro instructions.

- Entry: IGC0003E
- Exit: Return to calling program or XCTL to IEFWTP00 for write-to-programmer processing
- Tables/Work Areas: WQE, UCM, CVT, RQE
- Attributes: Reenterable
- Control Section: IGC0003E

IEEDFINA: Master Scheduler -- SMF MFT Storage Configuration Record Creation Routine

This routine creates the SMF dynamic storage configuration record for MFT. It is entered during SMF initialization and whenever a DEFINE Command is issued.

- Entry: IEEDFINA from IEESMFI2 during SMF initialization.
  From IEEDFIN9 whenever a DEFINE command is issued.
- Exit: Return to caller.
- Tables/Work Area: CVT, M/S resident data area, PIB, TCB, SMCA
- Attributes: Reentrant
- Control Sections: IEEDFINA

IEEDFINB: Master Scheduler -- System Reinitialization Routine 2

This routine waits for partitions to quiesce and then issues an ENQUEUE macro instruction specifying the boundary boxes.

- Entry: IEEDFINB
- Exits: IEEDFIN8

- Attributes: Read/only, reenterable
- Control Sections: IEEDFINB

IEEDFINC: Master Scheduler -- DEFINE Command Validity Check Routine (Core Storage)

This routine determines whether all information for the partition redefinition of core storage is correct.

- Entry: IEEDFINC
- Exits: IEEDFINB, IEEREXIT
- Tables/Work Areas: DFINDATA, CVT, M/S resident data area.
- Attributes: Read-only, reenterable
- Control Section: IEEDFINC

IEEDFIN1: Master Scheduler -- DEFINE Command Initialization Routine

This routine sets up data areas for partition definition, issues a DEFINE COMMAND BEING PROCESSED message to all active consoles, and passes control to the appropriate processing module.

- Entry: IEEDFIN1
- Exits: IEEDFIN3, IEEDFIN4, IEEDFIN5
- Attributes: Read-only, reenterable
- Control Section: IEEDFIN1

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, IEEDPART
- Exits: IEEDFIN5, IEEDFIN6, IEEDFIN7
- Attributes: Read-only, reenterable
- Control Section: IEEDFIN2

IEEDFIN3: Master Scheduler -- DEFINE Command Validity Check Routine (Processor Storage)

This routine determines whether all information for the partition redefinition of processor storage is correct.

- Entry: IEEDFIN3
- Exits: IEEDFINC, IEEREXIT
- Attributes: Read-only, reenterable
- Control Section: IEEDFIN3

IEEDFIN4: Master Scheduler -- DEFINE Command Listing Routine

This routine lists partition definitions.

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

IEEDFIN5: Master Scheduler -- DEFINE Command Message Routine

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: IEEDFIN1, IEEDFIN2 or return to calling program
- Attributes: Read-only, reenterable
- Control Section: IEEDFIN5

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

This routine checks the TMSL subparameters for proper syntax.

- Entry: IEEDFIN6
- Exits: IEEDFIN2, IEEDFIN5, IEEDPART
- Attributes: Read-only, reenterable
- Control Section: IEEDFIN6

IEEDFIN7: Master Scheduler -- Keyword Scan Routine

This routine checks keyword parameters for syntax errors. If a syntax error is discovered, the erroneous entry and all following entries are ignored, and an error message is generated. If the syntax is correct, the information is stored.

- Entry: IEEDFIN7

- Exits: IEEDFIN2, IEEDFIN3, IEEDFIN4, IEEDFIN5, IEEDPART
- Attributes: Read-only, reenterable
- Control Section: IEEDFIN7

IEEDFIN8: Master Scheduler -- System Reinitialization Routine - Part 1

This routine assigns protection keys, marks dispatchable partitions that are not of zero size, and checks that the number of problem program partitions does not exceed 15. If no error is found, IEEDFIN8 builds request blocks and boundary boxes and updates the PIB and the TCBPIB field for the defined partition.

- Entry: IEEDFIN8
- Exits: IEEDFIN9, IEEREXIT
- Attributes: Read-only, reenterable
- Control Section: IEEDFIN8

IEEDFIN9: Master Scheduler -- Command Final Processor Routine

This routine updates the time-slice control element, if time-slicing is specified, and issues a message to all active consoles that processing is complete.

- Entry: IEEDFIN9, IEEREXIT
- Exits: IEEDFIN5, IEEDFINA
- Attributes: Read-only, reenterable
- Control Section: IEEDFIN9

IEELOG02: Master Scheduler -- Log Open Initialization Module

This routine opens the system log at IPL time.

- Entry: IEELOG02
- Exit: IEESD569
- Tables/Work Areas: CVT, UCB, UCM, TIOT, M/S resident data area, JFCB, IEELCA, DCB.
- Attributes: Refreshable
- Control Section: IEELOG02

IEELWAIT: Master Scheduler -- Log Wait and Writer Module

This module writes data from the log buffer to the system log.

- Entry: IEELWAIT
- Exit: To Dispatcher
- Tables/Work Areas: CVT, LCA, MRC
- Attributes: Resident
- Control Section: IEELWAIT

IEEPDISC: Display Consoles Get Region Routine

This routine obtains a region of main storage, and sets up an environment for the execution of the DISPLAY CONSOLES command, and then frees the region when control is returned.

- Entry: IEEPDISC, from IEEVATT1
- Exit: To IEEXEDNA, Return to Master Task (SVC 3)
- Attributes: Read-only, reentrant, resident
- Control Sections: IEEPDISC

IEESD561: SVC 34 -- STOP INIT and START Command Processor (Part 1)

This routine initially processes the STOP INIT and START commands.

- Entry: IGC1903D
- Exit: IEE3903D, IEE0503D for error messages
- Tables/Work Areas: CVT, XSA
- Attributes: Read-only, transient
- Control Section: IGC1903D

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, to IEESD566 for DISPLAY active, to IEEUNIT1 for DISPLAY units, or to IEEXEDNA for DISPLAY CONSOLES
- Attributes: Read-only, reenterable
- External References: None
- Control Section: IEESD562

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, XCTL to IEESD575 for CANCEL.
- Tables/Work Areas: QCR, QMPA, CVT, CSCB
- Attributes: Read-only, reenterable
- Control Section: IEESD563

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

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

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

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: IEEMSER

IEESD571: SVC 34 -- DEFINE, MOUNT Routine

This routine schedules the execution of the DEFINE and MOUNT commands.

- Entry: IGC1803D
- Exits:
  MOUNT - XCTI to IGC0103D
  DEFINE - Return to caller
  XCTL to IEE0503D and IEE2103D due to error.
- Tables/Work Areas: CSCB, PIB, M/S resident data area, CVT
- Attributes: Reenterable
- Control Section: IGC1803D

IEESD575: Master Scheduler -- Queue Scratch Setup Routine

This routine sets up the scratch parameter list.

- Entry: IEESD575 from IEESD563
- Exit: XCTI to IEESD581 to Scratch or tc IEESD576 to delete queue entries.
- Tables/Work Areas: CSCB, CVT, JCT, JFCB, QMPA, SCT, SIOT, TIOT, UCB, UCB Lcck up Table

- Attributes: Read-only, reenterable
- Control Section: IEESD575

IEESD576: Master Scheduler -- Queue Alter Delete Routine

This routine goes to the Queue Manager delete routine IEFQDELQ to delete queue entries.

- Entry: IEESD576 from IEESD575
- Exit: XCTL to IEESD563 for queue search; XCTL to IEESD578 for message class setup; XCTL to IEESD580 for message
- Tables/Work Areas: CSCB, CVT, DSB, JCT, SCD, SMB
- Attributes: Read-only, reenterable
- Control Section: IEESD576

IEESD577: Master Scheduler -- Queue Restart Enqueue Routine

This routine links to the queue manager enqueue routine IEFQMNQQ to enqueue data sets for a canceled restarting job.

- Entry: IEESD577 from IEESD578
- Exit: XCTL to IEESD579
- Tables/Work Areas: CSCB, QMPA, SCD, SMB
- Attributes: Read-only, reenterable
- Control Section: IEESD577

IEESD578: Master Scheduler -- Queue Message Class Setup

This routine zeroes out the DSB's in the message class and sets up the QMPA for enqueuing the message class.

- Entry: IEESD578 from IEESD576
- Exit: XCTL to IEESD579 to enqueue the message class; XCTL to IEESD577 to enqueue the sysout data sets for a restarting job.
- Tables/Work Areas: CSCB, CVT, JCT, QMPA, SCD, SMB
- Attributes: Read-only, reenterable
- Control Section: IEESD578

IEESD579: Master Scheduler -- Queue SMB Routine

This routine places the appropriate CANCEL message in the SMB and goes to the queue manager enqueue routine to enqueue the message class. The operator message is also issued from this routine.

- Entry: IEESD579 from IEESD577 or IEESD578
- Exit: Return to IEECIR50
- Table/Work Areas: CSCB, CVT, QMPA, SMB
- Control Sections: IEESD579, IEEMSWTO

IEESD580: Master Scheduler -- Specific CANCEL Message Routine

This routine issues the WTO if the CANCEL command was for a specific output class other than the message class.

- Entry: IEESD580 from IEESD576
- Exit: Return to IEECIR50
- Tables/Work Areas: CSCB, QMPA
- Control Sections: IEESD580

IEESD581: Master Scheduler -- Queue Scratch Routine

This routine issues the SCRATCH macro (SVC 29) or an error message.

- Entry: IEESD581 from IEESD575
- Exit: XCTL to IEESD575
- Attributes: Read-only, reenterable
- Control Section: IEESD581

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

IEESD591: System Task Control -- Linker Routine

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

- Entry: IEESD591, IEE591SD
- Exit: XCTL to IEEVTCTL
- Tables/Work Areas: CSCB, CVT, PIB, IWA, QMPA
- Attributes: Reenterable
- Control Section: IEESD591

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

IEESMFAL: SVC 83 -- SMF Allocation Routine

This routine allocates devices for the SMF data sets.

- Entry: IEESMFAL
- Exit: IEESMFOP
- Attributes: Reentrant
- Tables/Work Areas: SMCA, CVT
- Control Sections: IEESMFAL

IEESMFIT: SMF Initialization Routine (1)

This routine obtains storage for and initializes the SMCA, and reads the SMFDEFLT parameters from SYS1.PARMLIB.

- Entry: IEESMFIT, IEESMFI4
- Exit: IEFSD569, IEESMFOI, IEESMFI2, IEESMFI3
- Attributes: Non-reentrant
- Tables/Work Areas: CVT, DCB, JFCB, M/S Resident Data Area, SMCA, TIOT, UCB
- Control Sections: IEESMFIT

IEESMFI2: SMF Initialization Routine (2)

This routine constructs the SMF IPL and initial online I/O device records, and enqueues the 10-minute TQE on the timer queue.

- Entry: IEESMFI2
- Exit: To IEEDFINA, IEESMFIT
- Attributes: Non-reentrant
- Control Sections: IEESMFI2

IEESMFI3: SMF Parameter Processing Routine

This routine processes the SMFDEFLT parameters and/or the SMF parameters entered from the operator's console.

- Entry: IEESMFI3, IEESMFIO, IEESMFMS
- Exit: IEESMFIT (at entry point IEESMFI4)
- Attributes: Serially Reusable
- Tables/Work Areas: CVT, M/S Resident Data Area, SMCA, UCB
- Control Section: IEESMFI3

IEESMFOI: SMF Open Initializer

This routine stores the DCBs and JFCBs for the SMF data sets.

- Entry: IEESMFOI
- Exit: IEESMFIT
- Attributes: Serially reusable
- Tables/Work Areas: DCB, JFCB, SMCA, TIOT
- Control Sections: IEESMFOI

IEESMFOP: SVC 83 -- Open Routine

This routine opens the SMF data sets, and switches between the primary and alternate data sets.

- Entry: IEESMFOP
- Exit: IEESMFAL, return to caller
- Tables/Work Areas: DCB, JFCB, SMCA, TIOT
- Attributes: Reentrant
- Control Sections: IEESMFOP

IEESMFWT: SMF Writer Routine

This routine writes the contents of the SMF buffer in the SMF data set.

- Entry: IEESMFWT
- Exit: return to caller
- Tables/Work Areas: CVT, DCB, SMCA
- Attributes: Reentrant
- Control Section: IEESMFWT

IEEUNIT1: Master Scheduler -- DISPLAY U Routine (1)

This routine syntax checks the DISPLAY U command and, for valid commands, defines the type of output requested. For invalid commands it passes control to IEEUNIT3 for issuance of the appropriate error message.

- Entry: IEEUNIT1
- Exits:
  Normal -- IEEUNIT4 to continue DISPLAY U processing
  Error -- IEEUNIT3 to issue error messages and return control to the Master Scheduler
- Tables/Work Areas: CVT, CSCB, UCB, Device Name Table, UCM, DCM, M/S resident XSA data area, XSA
- Attributes: Reenterable
- Control Section: IEEUNIT1

IEEUNIT2: Master Scheduler -- DISPLAY U Routine (2)

This routine constructs the lines of the tabular display of the unit status based on information in the UCBs and the Device Name Table.

- Entry: IEEUNIT2
- Exits:
  Normal -- IEEUNIT3 to scan a data cell UCB
  -- IEEUNIT4 to write a line of text
  Error -- IEEUNIT3 to issue error messages and return control to the Master Scheduler
- Tables/Work Areas: CSCB, CVT, DEVICE Name Table, UCB, XSA
- Attributes: Reenterable
- Control Section: IEEUNIT2

IEEUNIT3: Master Scheduler -- DISPLAY U Routine (3)

This rcutine issues all of the error messages for the DISPLAY U command processing. It also constructs the display lines that contain the data cell information.

- Entry: IEEUNIT3
- Exits:
  Normal -- IEEUNIT2 when the data cell scan is complete but the line of text is not full
  -- IEEUNIT4 to write a full line of text
  -- IEECIR50 to return control to the Master Scheduler

  Error -- None
- Tables/Work Areas: CSCB, CVT, DCM, Device Name Table, M/S Resident Data Area, UCB, UCM, XSA

IEEUNIT4: Master Scheduler -- DISPLAY U Routine (4)

This routine displays, via the WTO macro instruction, the lines of text prepared by IEEUNIT2 or IEEUNIT3. It also builds a list of valid UCB addresses for processing by IEEUNIT2 or IEEUNIT3.

- Entry: IEEUNIT4
- Exits:
  Normal -- IEEUNIT2 to complete the display
  -- IEEUNIT3 to complete the data cell scan or to return to the Master Scheduler
  Error -- IEEUNIT3 to issue error messages and to reutrn to the Master Scheduler
- Tables/Work Areas: CSCB, CVT, UCB, XSA
- Attributes: Reeenterable
- Control Section: IEEUNIT4

IEEVACTL: System Task Control -- Allocation Interface Routine

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

- Entry: IEEVACTI
- Exits: To IEFSD21Q, IEEVMSG1, IEEVSMSG, IEEVTCTL, or IEEVSMBA
- Attributes: Reenterable
- Control Section: IEEVACTL

IEEVICLR: Internal JCL Reader

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

- Entry: IEEVICLR, IEFICR
- 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 IEEVSTAR
- Exit: XCTL to IEEVRCTL
- Tables/Work Areas: SDT, CSCB
- Attributes: Reenterable
- Control Section: IEEVJCL

IEEVLDSP: Master Scheduler -- Log Dispatcher Routine

This routine puts the log data set on the system output queue.

- Entry: IEEVLDSP
- Exit: Master Scheduler
- Tables/Work Areas: IEEBASEA, CT, IEELCA, UCB, JFCB.
- Attributes: Reentrant
- Control Section: IEEVLDSP

IEEVLIN: Master Scheduler -- Log Initialization Routine

This routine initializes the system log.

- Entry: IEEVLIN
- Exit: IEFSD569, IEEVLIN2
- Tables/Work Areas: UCM, CVT, UCB, TIOT, M/S resident data area, IEELCA.
- Attributes: Refreshable
- Control Section: IEEVLIN

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

IEEVLOUT: Log Data Set Reinitialization Routine

This routine opens and closes the log data set to reinitialize the DS1LSTAR and DS1TREAL fields of the DSCB associated with the log data set.

- Entry: IEEVIOUT, from IEFSD171
- Exit: IEFSD171
- Tables/Work Areas: CVT, DSCB, LCA, M/S Resident Data Area
- Attributes: Reenterable
- Control Section: IEEVLOUT

IEEVMSG1: System Task Control -- Message Interface Routine

This routine sets up the parameter list for the message writing routine.

- Entry: IEEVMSG1 from IEEVRCTL, IEEVACTL, or IEEVTCTL
- Exit: IEEVSMSG, 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

IEEVRFRX: Master Scheduler -- Table Lookup Routine

This routine can be used to obtain the following information; the CVT address, the contents of a CVT entry, or the contents at the CVT pointer address, a pointer to the TCB or the RB, the TIOT pointer, the TIOT entry, the TIOT TTR, or the TIOT UCB pointer. The routine can also be used to insert a TIOT pointer, a TIOT TTR, or a TIOT UCB pointer in the CVT.

- Entry: IEEVRFRX
- Exit: Return to calling program
- Tables/Work Areas: CVT, TCB, RB, TIOT, UCB
- Attributes: Reenterable
- Control Section: IEEVRFRX

IEEVRJCL: System Task Control -- Internal JCL Reader

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

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

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 IEEVMSG1, IEFSD533, IEFVH1, or IEEVACTL
- 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: IEEVSTAR
- 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: 74

IEEVTCTL: System Task Control -- Termination Interface Routine

This routine initializes the necessary tables for terminating a task that was established via a START command, and releases storage obtained by IEEVSTAR for the task's JSCB and WTPCB.

- Entry: IEEVTCTL, from IEESD590, IEEVACTL or IEFW31SD
- Exit: IEFQMSS, IEEVMSG1, IEFW42SD, IEFQDELE, IEEPRTN2, IEEVOMSG
- 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

IEEXEDNA: DISPLAY CONSOLES Processor

This routine processes the DISPLAY command with the CONSOLES operand and displays the system console configuration on the requesting console.

- Entry: IEEXEDNA to IEESD562
- Exit: To IEECIR50
- Attributes: Reentrant
- Control Sections: IEEXEDNA

IEE0303D: SVC 34 -- CSCB and CIB Chain Manipulator

This routine manipulates the CSCB and CIB chain as requested by the caller of SVC 34.

- Entry: IEE0303D
- Exit: To IEE5403D, or return to caller
- Tables/Work Areas: CVT, M/S resident data area, CSCB, XSA
- Control Section: IEE0303D

IEE0303F: SVC 36 -- WRITE-TO-LOG

This module copies text records from an input area to the log buffer and posts the log ECB when the buffer is full.

- Entry: IEE0303F
- Exit: Returns to Master Scheduler, IEE0403F.
- Tables/Work Areas: IEEBASEA, IEELCA, CVT
- Attributes:
- Control Section: IEE0303F

IEE0403D: SVC 34 -- Router Routine

This routine identifies the command verb, ensures that the console has authority to enter the command, and passes control to the appropriate routine.

- 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

IEE0403F: SVC 36 (Load 2) -- Log Buffer Management Module

This module opens, closes, and switches system log buffers.

- Entry: IEE0403F
- Exit: IEE0303F
- Tables/Work Areas: IEEBASEA, IEELCA, UCB, JFCB, DCB, CVT, TIOT.
- Attributes: Reentrant
- Control Section: IEE0403F

IEE0503D: SVC 34 -- 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
- Attributes: Reenterable, read-only
- Control Section: IEE0503D

IEE0603D: SVC 34 -- SET Command Routine

This routine processes the SET command.

- Entry: IEE0603D
- Exits: To IEE0903D, IEE0503D, or return to caller
- Tables/Work Areas: XSA, CVT, M/S resident data area
- Attributes: Reenterable, self-relocating, read only transient
- Control Section: IEE0603D

IEE0703D: SVC 34 -- STOP and MODIFY Scheduling

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 IEE0503D or IEE2103D.
- Tables/Work Areas: M/S Resident Data Area, XSA, CVT, CSCB
- Attributes: Reenterable, self-relocating, read-only, transient
- Control Section: IEE0703D

IEE0803D: SVC 34 -- CSCB Creation Routine

This routine schedules the execution of commands that cannot be completely processed by the command scheduling routines. It performs this function by adding a CSCB to the CSCB chain and issuing a POST macro instruction to the master scheduling task.

- Entry: IEE0803D
- Exit: IEE0503D, IEE2103D, or return to caller
- Tables/Work Areas: XSA, M/S resident data area, CVT, CSCB, and UCM
- Attributes: Reenterable, transient, partially disabled.
- Control Section: IEE0803D

IEE0903D: SVC 34 -- Timer Maintenance Routine

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 Scan Routine for Non-MCS Systems

This routine examines the command and its operand.

- Entry: IEE1103D
- Exit: IEE2203D for multiprocessing VARY commands, IEE2303D for VARY ONLINE and CONSOLES operands when SMF is present, to IEE3103D for all other VARY operands and UNLOAD, and to IEE0503D for errors.
- Tables/Work Areas: XSA, CVT, M/S resident data area, and UCM.
- Attributes: Reenterable, self-relocating, read-only, and transient.
- Control Sections: 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

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

IEE1603D: SVC 34 -- Log and Writelog Processor Routine

This routine issues a WTL macro instruction when a LOG command is issued, and stores the WRITELOG command and posts the Log ECB, for WRITELOG processing.

- Entry: IEE1603D, from IEE0403D
- Exit: IEE0503D for errors, and return to caller of SVC 34.
- Tables/Work Areas: XSA, CVT, LCA, and M/S resident data area.
- Attributes: Reentrant, self-relocating, read-only, and transient.
- Control Sections: IEE1603D

IEE1703D: SVC 34 -- VARY ONGFX/OFFGFX

This routine processes the GVARY command. It checks the parameters for validity and if an error is found, it passes control to IEE0503D via an XCTL macro instruction. If the parameters are valid, the routine sets appropriate bits in the Overall Control Table (OCT) of the GFX reader task. It then issues a POST macro instruction on the ECB in the OCT for each graphics device (2250) placed in the online status.

- Entry: IEE1703D
- Exit: IEE0503D, return to issuer of SVC 34
- Tables/Work Areas: CVT, OCT, XSA
- Attributes: Reenterable, read-only, self-relocating
- Control Section: IEE1703D

IEE1A03D: SVC 34 -- MCS Reply Processor Routine

The purpose of this routine is to process valid operator replies to WTOR macro instructions.

- Entry: IEE1A03D
- Exit: To IEE1B03D to issue error messages or return to the caller of SVC 34.
- Control Sections: IEE1A03D

IEE1B03D: SVC 34 -- MCS Reply Message Routine

This routine assembles, edits, and broadcasts the accepted reply to a WTOR macro instruction for the MCS Reply Processor routine (module IEE1A03D) of the Command Scheduling routine, and to write error messages to the operator whose command is in error.

- Entry: IEE1B03D, from IEE1A03D
- Exit: Return to the caller of SVC 34
- Control Sections: IEE1B03D

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
- Attributes: Reenterable, self-relocatory, read-only, transient
- Control Section: IEE2103D

IEE2303D: SVC 34 -- SMF VARY Processor

This routine initially processes the ONLINE and CONSOLES operand of the VARY command when the system has the SMF option. It builds and issues an SMF record for each device placed in online status.

- Entry: IEE2303D from IEE1103D or IEE3303D
- Exit: IEE3103D, IEE4203D, or IEE4403D
- Tables/Work Areas: CVT, SMCA, XSA
- Attributes: Reentrant, read-only, self-relocating
- Control Sections: IEE2303D

IEE2803D: SVC 34 -- CANCEL Command

This routine checks the syntax and processes the CANCEL command.

- Entry: IEE2803D
- Exit: BALR to ABTERM
  XCTL to IEE0803D
  XCTL to IEE0503D and IEE2103D for messages
- Tables/Work Areas: CSCB, TCB, M/S Resident data area
- Attributes: Read-only, reenterable
- Control Section: IEE2803D

IEE2903D: SVC 34 -- Display Requests Routine

This routine displays to the requesting operator the ID of all outstanding WTORs, the unit name of each device for outstand-

ing MOUNT messages, and an indication as to whether any AVR mount messages are pending.

- Entry: IEE2903D, from IEE0803D
- Exit: Return to caller of SVC 34
- Tables/Work Areas: Message work area
- Attributes: Reentrant, Refreshable, transient
- Control Sections: IEE2903D

IEE3103D: SVC 34 -- VARY and UNLOAD Processor Routine

This routine processes the UNLOAD command, all VARY command operands in a system without the MCS option, and VARY ONLINE and OFFLINE operands for non-console devices in a system with the MCS option.

- Entry: IEE3103D, initially from IEE1103D or IEE2303D and returns from IEE4603D.
- Exit: Return to Caller
- Tables/Work Areas: XSA, UCM, CVT, M/S resident data area, and UCB.
- Attributes: Reentrant, self-relocating, read-only, transient.
- Control Sections: IEE3103D

IEE3203D: SVC 34 - Vary Keyword Router Routine

This routine identifies the first keyword in a VARY command, checks its validity, and passes control to the appropriate command keyword processing routine.

- Entry: IEE3203D from IEE0403D
- Exit: To IEE1103D, IEE3303D, IEE4303D, IEE4703D, IEE2403D, IEE1703D
- Tables/Work Areas: XSA
- Attributes: Reentrant, read-only, self-relocating
- Control Sections: IEE3203D

IEE3303D: SVC 34 -- MCS VARY Syntax Check Routine

This routine scans the command syntax in an MCS environment.

- Entry: IEE3303D from IEE3203D
- Exit: IEE2303D, IEE2203D, IEE4203D, or IEE4403D
- Tables/Work Areas: XSA, UCB, UCM, CVT
- Attributes: Reenterable, self-relocating, read only
- Control Sections: IEE3303D

IEE3503D: SVC 34 -- Display Commands Router Routine

This routine examines the operand of DISPLAY commands and routes to the task which processes the command.

- Entry: IEE3503D from IEE0403D
- Exits: IEE0503D, IEE0803D, IEE3103D, IEE2903D, or return to caller
- Table/Work Areas: M/S resident data area, XSA, CVT, and UCM
- Attributes: Reenterable, transient
- Control Section: IEE3503D

IEE3903D: SVC 34 -- STOP INIT and START Command Processing Routine (Part 2)

This routine completes the processing of the STOP INIT and START commands.

- Entry: IGC3903D
- Exit: Return to caller
- Tables/Work Areas: CVT, XSA, CSCB, PIB, M/S resident data area
- Attributes: Read-only, transient
- Control Section: IEE3903D

IEE4103D: SVC 34 -- Hardcopy Message Issuing Routine

This routine issues messages concerning the status of the hard copy log.

- Entry: IEE4103D, from IEE4703D
- Exit: Return to caller
- Tables/Work Areas: XSA, message area, UCB, CVT, XSA, and UCM
- Attributes: Reentrant, transient
- Control Sections: IEE4103D

IEE4203D: SVC 34 -- VARY ONLINE/OFFLINE and CONSOLE Secondary Scanner Routine

This module performs authority and operand validity checking, and passes control to the routine that will process the command.

- Entry: IEE4203D, from IEE3303D or IEE2303D
- Exit: To IEE3103D, IEE4603D, or IEE4903D
- Tables/Work Areas: XSA, CVT, UCM, and UCB.
- Attributes: Reenterable, self-relocating, read-only, transient
- Control Sections: IEE4203D

IEE4303D: SVC 34 -- VARY MSTCONS Routine

This routine processes the VARY MSTCONS command.

- Entry: IEE4303D from IEE3203D VARY CONSOLE
- Exit: To IEE0503D or IEE2103D on errors, SVC 72 to Console Switch Routine (module IEECMCSW) and upon return to caller of SVC 34
- Tables/Work Areas: UCB, CVT, XSA, and UCM
- Attributes: Reentrant, self-relocating, read-only, transient
- Control Sections: IEE4303D

IEE4403D: SVC 34 -- VARY CONSOLE Keyword Scan Routine

This routine determines the validity of VARY CONSOLE and sets appropriate bits in the XSA.

- Entry: IEE4403D, from IEE3303D or IEE2303D
- Exit: To IEE4203D for VARY CONSOLES or to IEE0503D for errors.
- Tables/Work Areas: XSA, UCM, CVT, and UCB
- Attributes: Reentrant, transient
- Control Sections: IEE4403D

IEE4503D: SVC 34 -- Periodic STOP Command Handler Routine

This routine processes the commands STOP JOBNAMES/STATUS/SPACE/DSNAME.

- Entry: IEE4503D, from IEE0403D
- Exit: IEE0503D for errors, and return to caller of SVC 34
- Tables/Work Areas: XSA, M/S resident data area, CVT, and UCM
- Attributes: Reentrant, self-relocating, read-only, transient
- Control Section: IEE4503D

IEE4603D: SVC 34 -- VARY ONLINE/OFFLINE Routing Routine for Console Devices

This routine processes VARY ONLINE/OFFLINE for all MCS consoles.

- Entry: IEE4603D, from IEE4203D to process VARY ONLINE-OFFLINE
- Exit: IEE3103D or return to caller
- Tables/Work Areas: XSA, CVT, UCB, UCM, and M/S resident data area
- Attributes: Reentrant, self-relocating, read-only, transient
- Control Sections: IEE4603D

IEE4703D: SVC 34 -- VARY HARDCPY Processor Routine

This routine processes VARY HARDCPY commands.

- Entry: IEE4703D from IEE3203D
- Exit: To IEE4103: or IEE5703D
- Tables/Work Areas: XSA, UCM, M/S resident data area, CVT and UCB
- Attributes: Reentrant, transient
- Control Sections: IEE4703D

IEE4803D: SVC 34 -- VARY CONSOLE Information Message Routine

This routine constructs a message which shows the current status of the varied console.

- Entry: IEE4803D, from IEE4903D
- Exit: Return to caller
- Tables/Work Areas: XSA, message area, UCB, CVT, and UCM
- Attributes: Reentrant, transient
- Control Sections: IEE4803D

IEE4903D: SVC 34 -- VARY CONSOLE Processor Routine

This module processes the VARY CONSOLE command.

- Entry: IEE4903D, from IEE4203D
- Exit: To IEE4803D to construct console message
- Table/Work Areas: XSA, CVT, UCB, and UCM
- Attributes: Reentrant, self-relocating, read-only, transient
- Control Sections: IEE4903D

IEE5403D: SVC 34 -- Command Translator Routine

This routine translates lower case letters (except those within apostrophes) into upper case letters.

- Entry: IEE5403D
- Exit: IEE0403D
- Tables/Work Areas: CVT, Internal Translation Table, UCM, UCME, XSA
- Control Section: IEE5403D

IEE5703D: SVC 34 -- VARY Hardcopy OFF and Message Routing Routine

This routine removes the hardcopy log and routes any error messages to the appropriate error message routines.

- Entry: IEE5703D from IEE4703D
- Exit: IEE0503D or IEE2103D
- Tables/Work Areas: XSA, UCM, UCB, CVT, MRC
- Attributes: Reenterable, transient
- Control Sections: IEE5703D

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

IEFDSDRP: Data Set Descriptor Record Processing Routine

This routine processes the job queue information in the DSDR record to make a restarting job's queue entry reflect the environment when the checkpoint was taken.

- Entry Point: IEFDSDRP
- Exit: Return to caller
- Table/Work Areas: JCT, SCT, SIOT, JFCB, TIOT, UCB, CVT, VOLT, TCB, QMPA, CSCB, DCBD, DCB, JFCBX, SCTX, LCT
- Attributes: Reenterable
- Control Section: IEFDSDRP

IEFDSOAL: Direct System Output -- SIOT and JFCB Modification

This routine modifies the SYSOUT SIOTs and JFCBs of steps that will use DSO.

- Entry: IEFDSOAL from IEFVMLS1
- Exit: IEFVMLS1 or IEFQBVMS
- Tables/Work Areas: CVT, LCT, QMPA, DSOCB, JCT, SCT, SIOT, JFCB, PIB, UCB
- Attributes: Reenterable
- Control Sections: IEFDSOAL

IEFDSOCP: Direct System Output -- Initialization Routine

This routine initializes DSO by constructing the DSOCB and performing DSO housekeeping.

- Entry: IEFDSOCP
- Exit: Return to caller
- Tables/Work Areas: TCB, DSOCB, TIOT, JFCB, UCB, CVT, ECB/IOB, QMPA, PIB
- Attributes: Reenterable
- Control Sections: IEFDSOCP

IEFDSOFB: Direct System Output -- Release DSOCB Routine

This routine frees DSOCBs allocated to a job.

- Entry: IEFDSOFB from IEFSD168, or IEFSD518
- Exit: Return to caller
- Tables/Work Areas: LCT, DSOCB, CVT, PIB, M/S resident data area, JCT
- Attributes: Reenterable
- Control Sections: IEFDSOFB

IEFDSOSM: Direct System Output -- Stop and Modify Command Processing Routine

This routine processes Stop and Modify commands for DSO.

- Entry: IEFDSOSM
- Exit: SVC 3
- Tables/Work Areas: CVT, JCB, DSOCB, UCB, PIB, CSCB, JCT, ECB/IOB, QMPA
- Attributes: Reenterable
- Control Sections: IEFDSOSM

IEFDSOWR: Direct System Output -- System Messages and Job Separator Writer

This routine writes job separators and system messages to the assigned DSO device.

- Entry: IEFDSOWR from IEFSD512 or IEFSD31Q
- Exit: IEFSD512, IEFSD31Q, IEFQMRAW, IEFSD094 or the user's separator program
- Tables/Work Areas: TCB, LCT, QMPA, DSOCB, JCT, SCT, PIB, UCB, CVT, TIOT, DCB, SMB
- Attributes: Reenterable
- Control Sections: IEFDSOWR

IEFIDMPM: Termination -- Message Module

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

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

IEFLOCDQ: Queue Management -- Dequeue by Jobname Routine

This routine searches a queue for a named job or list of named jobs, and can return information, or dequeue or cancel the job.

- Entry: LOCDQ, LOCCAN, LOC
- Exit: Return to caller
- Tables/Work Areas: QCR, LTH
- Attributes: Reenterable
- External References: IEFCNVRT, IEFRDWRT

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

IEFPRINT: Direct System Output -- Tape to Printer or Card Punch Routine

This routine writes a DSO - written tape to a printer or card punch.

- Entry: SPRINTER
- Exit: Return to caller
- Control Sections: SPRINTER

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

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

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 resident data area, CVT
- Attributes: Reenterable
- Control Section: IEFQDELE

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

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

IEFRCLN1: Restart Reader Linkage

This routine receives control from IEFVRRC and LINKS to interpreter initialization routine IEFVH1.

- Entry: IEFRCLN1
- Exit: XCTL to IEFVRRC at entry IEFVRRCA
- Attributes: Reenterable
- Control Section: IEFRCLN1

IEFRCLN2: Restart Reader Linkage

This routine receives control from IEFVRRC and LINKS to interpreter initialization routine IEFVH1.

- Entry: IEFRCLN2
- Exit: XCTL to IEFVRRC at entry IEFVRRCB

- Attributes: Reenterable
- Control Section: IEFRCLN2

IEFRPREP: Termination -- Restart Preparation Routing

This routine determines whether a job step that has been abnormally terminated can be restarted.

- Entry: IEFRPREP from IEFYNIMP
- Exit: Return to caller
- Attributes: Reenterable
- Tables/Work Areas: LCT, JCT, SCT, PDQ, QMPA
- Control Section: IEFRPREP

IEFRSTRT: Restart SVC Issuing Routine

This routine issues the Restart SVC. When called by its alias, IEFSMR, it issues the Restart SVC and then returns to the caller.

- Entry: IEFRSTRT, IEFSMR
- Exit: SVC 52 (RESTART), return to caller
- Attributes: Reenterable
- Control Sections: IEFRSTRT

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

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

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

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

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

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

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

IEFSD167: Initiator -- Linkage to IEFSD168

This routine passes control via an XCTL macro instruction to IEFSD168 in the 30K scheduler.

- Entry: IEFSD068
- Exit: IEFSD168
- Tables/Work Areas: None
- Attributes: Reenterable
- Control Section: IEFSD068

IEFSD168: Initiator -- Job Suspension

This routine causes a terminated job to be reenqueued so that the job can be reactivated.

- Entry: IEFSD068
- Exit: Branch to IEFSD598 to purge resources, branch to IEFSD510 to reinitiate job, link to IEFDSOFB
- Tables/Work Areas: QMPA, LCT, JCT, SCD, SCT
- Attributes: Reenterable
- Control Section: IEFSD068

- External Reference: IEFQMRAW, IEFQMNQ2, IEFVSDRA

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 IEEVLOUT, 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

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 LTH 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. If DSO was used, the DSOCBs are released; if the message class is assigned to DSO, the routine links to IEFDSOWR.

- Entry: IEFW31SD
- Exit: IEFSD32Q (44K Scheduler), IEFSD33Q (30K Scheduler), or IEFDSOWR
- Tables/Work Areas: JCT, SCT, SMB, QMPA, ECB, CVT, M/S resident data area, DSOCB, PIB
- 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

IEFSD32Q: Initiator -- Linkage From Job Termination to the Initiator for the 44K Scheduler

This routine receives control from job termination exit routine IEFSD31Q in MFT systems with the 44K scheduler. It returns control to step deletion routine IEFSD515 via the RETURN macro instruction.

- Entry: IEFW32SD
- Exit: Return to IEFSD515
- Tables/Work Areas: None
- Attributes: Reenterable
- Control Section: IEFW32SD

IEFSD33Q: Initiator -- Linkage From Job Termination to the Initiator for the 30K Scheduler

This routine receives control from job termination routine IEFSD31Q in MFT systems with the 30K scheduler. It passes control to job deletion routine IEFSD517 with the address of the Life-of-Task block in register 1.

- Entry: IEFW32SD
- Exit: Branch to IEFSD517
- Tables/Work Areas: None
- Attributes: Reenterable
- Control Section: IEFW32SD

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, LCT, SCT, SMB, CMPA
- 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. it also provides an interface for entry to the LOG function if a LOG data set is scheduled to be added to the SYSOUT queue.

- 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, LINK to IEFSD519, XCTL to IEFSD586, IEFSD589, SMALTERM
- Tables/Work Areas: LOT block, CSCB, SPIL, CVT, TCB, PIB, DSOCB
- Attributes: Read-only, reenterable
- Ccntrol Section: IEFSD510
- External References: IEFQMDQQ, IEFQMUNC

IEFSD511: Initiator -- Job Initiation Routine

This routine initializes information pertaining to a job. If DSO is available for the job's system output classes, the routine selects the DSOCBs to be used by the job.

- Entry: IEFSD511, IEFDSOSL
- Exit: Branch to IEFSD541
- Tables/Work Areas: Life-of-Task Block, CSCB, JCT, SCT, SCD, PIB, IOB2, DSOCB
- Attributes: Read-only, Reenterable
- Control Section: IEFSD511, IEFDSOSL
- External References: IEFQMRAW

IEFSD512: Initiator -- Step Initiation Routine

This routine passes control to allocation as a closed subroutine via an XCTL macro instruction and receives control back from Allocation at entry point IEFALRET. 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, IEFALRET
- Exits: Branch to IEFSD513, IEFSD516, or IEFSD518, XCTL to IEFSD510, IEFSD556
- Tables/Work Areas: LOT Block, JCT, SCT, APL, TIOT, CSCB, IOB1, IOB2, QMPA, SMB, DSOCB
- Attributes: Read-only, reenterable
- Control Section: IEFSD512
- External References: IEFQMRAW, IEFSD556, IEFSD514, IEFDSOWR

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 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

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

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, IEFSD558, IEFSD167 (30K Scheduler), IEFSD168 (44K Scheduler), or BALR to IEFSD42Q
- Tables/Work Areas: Life-of-Task Block, Terminate Parameter List, CVT, TCB, PIB, IOB, CSCB, DCB, JCT, SCT, SPIL, DSOCB
- Attributes: Read-only, reenterable
- Control Section: IEFSD515
- External References: IEFQMRAW, IEFSD514, IEFSD42Q, IEFSD598

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

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

IEFSD518: Initiator -- Partition Recovery Routine

This routine determines the status of main storage required for a checkpoint/restart.

- Entry: IEFSD518
- Exits: Return to IEFSD512
- Tables/Work Areas: SPIL, CVT, TCB, JCT, PIB, LOT, QMPA, CSCB, DSOCB
- Attributes: Reenterable
- Control Section: IEFSD518
- External Reference: IEFQMRAW, IEFQMNQ2, IEFSD598, IEFDSOFB

IEFSD519: Queue Management -- Dequeue by Jobname Interface Routine

This routine builds a seven-word parameter list used by IEFLOCDQ to locate a job by jobname on the checkpoint/restart internal queue.

- Entry: IEFSD519
- Exit: Return to IEFSD510
- Tables/Work Areas: LOT, PIB
- Attributes: Reenterable
- Control Section: IEFSD519
- External Reference: IEFLOCDQ, IEFQMRAW

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

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

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: IEFKG
- Exits: XCTL to IEFVHN or IEFSD530, or branch to IEFVHHB
- Tables/Work Areas: IWA, LWA, QMPA, PIB, CVT

- Attributes: Read-only, reenterable
- Control Section: IEFKG

IEFSD533: Interpreter -- Interface Routine

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

- Entry: IEFIRC
- Exits: XCTL to IEFSD537. RETURN to STC if error.
- Tables/Work Areas: CSCB, CVT, QMPA
- Attributes: Reenterable, read-only
- Control Section: IEFIRC

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

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

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 initializes the communications task and the system log. It 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: IEE0503D, Branch to dispatcher
- Attributes: Read-only, non-reenterable
- Control Section: IEFSD569

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

IEFSD584: System Task Control -- Linkage to IEESD591

This routine places system task control in the problem program mode.

- Entry: IEFSD584
- Exit: XCTL to IEESD591
- Control Section: IEFSD584

IEFSD585: System Task Control -- Linkage to IEFDSOSM

This routine places the DSO processor in the problem program mode.

- Entry: IEFSD585
- Exit: XCTL to IEFDSOSM
- Control Section: IEFSD585

IEFSD586: System Task Control -- Linkage to IEFSD585

This routine links to IEFSD585 so that upon return the initiator will be in supervisor state.

- Entry: IEFSD586
- Exit: Link to IEFSD585
- Control Section: IEFSD586

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: 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

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

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 IEF589SP or IEFSD586
- Tables/Work Areas: SPIL, allocate parameter list (APL), DSOCB, PIB
- Attributes: Read-only, reenterable
- Control Section: IEFSD599
- External Reference: IEFQMUNC

IEFSMFAT: Initiator -- TCTIOT Construction Routine

This routine constructs the TCTIOT, appends it to the TCT, initializes the TCT storage map, and stores the user routine address in the TCT.

- Entry: IEFSMFAT
- Exit: Return to caller
- Tables/Work Areas: PQE, SMCA, TCB, TCT, TCTIOT, TIOT
- Attributes: Reentrant
- Control Sections: IEFSMFAT

IEFSMFIE: Initiator -- User Exit Initialization Routine

This routine initializes the parameter lists for the Job Initiation and Step Initiation user exits.

- Entry: IEFSMFIE
- Exit: Return to caller
- Tables/Work Areas: JCT, JMR, LCT, SCT, TCT
- Attributes: Reentrant
- Control Sections: IEFSMFIE

IEFSMFLK: Termination Routine -- User Exit Initialization Routine

This routine initializes the parameter lists for the Job Termination and Step Termination user exits.

- Entry: IEFACTLK
- Exit: Return to caller
- Tables/Work Areas: JCT, JMR, LCT, SCT, SMCA, TCB, TCT
- Attributes: Reentrant
- Control Sections: IEFACTLK

IEFSMFWI: Termination Routine -- SMF Writer Interface Routine

This routine constructs the SMF job termination and step termination records.

- Entry: IEFSMFWI
- Exit: Return to caller
- Tables/Work Areas: JCT, JMR, LCT, SCT
- Attributes: Reentrant
- Control Sections: IEFSMFWI

IEFUJI: Initiator -- Dummy User Job Initiation Exit Routine

This routine simulates the presence of a user-supplied job initiation exit routine.

- Entry: IEFUJI
- Exit: Return to caller
- Attributes: Reentrant
- Control Sections: IEFUJI

IEFUJV: Interpreter -- Dummy User JCL Validiation Exit Routine

This routine simulates the presence of a user-supplied JCL validiation routine.

- Entry: IEFUJV
- Exit: Return to caller
- Attributes: Reentrant
- Control Sections: IEFUJV

IEFUSI: Initiator -- Dummy User Step Initiation Exit Routine

This routine simulates the presence of a user-supplied step initiation exit routine.

- Entry: IEFUSI
- Exit: Return to caller
- Attributes: Reentrant
- Control Sections: IEFUSI

IEFUSO: Dummy User SYSOUT Limit Exit Routine

This routine simulates the presence of a user-supplied SYSOUT limit exit routine.

- Entry: IEFUSO
- Exit: Return to caller
- Attributes: Reentrant
- Control Sections: IEFUSO

IEFUTL: Dummy User Time Limit Exit Routine

This routine simulates the presence of a user-supplied time limit exit routine.

- Entry: IEFUTL
- Exit: Return to caller
- Attributes: Reentrant
- Control Sections: IEFUTL

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, IEFVINB
- Tables/Work Areas: IWA, EXEC work area, interpreter key table, JCT, SCT, SIOT, QMPA, procedure override table, DCBD, PARMLIST, WORKAREA
- 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

IEFVGM71: Interpreter -- Message Module

This routine contains interpreter messages 40-47.

- Attributes: Non-executable
- Control Section: IEFVGM71

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, DCBD
- 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, DCBD, PARMLIST, WORKAREA
- 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 complete writing a data set, control passes to IEFVHR. If the input reaches end-of-file, contrcl 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

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

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

IEFVINA: Interpreter -- In-stream Procedure Processor

This routine processes the in-stream procedure. It uses the other in-stream procedure routines as subroutines to perform additional processing.

- Entry: IEFVINA
- Exit: IEFVHA, IEFVHCB
- Tables/Work Areas: EWA, IWA, JCT, PARMLIST, QMPA, WORKAREA
- Attributes: Reenterable
- Control Section: IEFVINA

IEFVINB: In-stream Procedure Search Routine

This routine searches the in-stream directory for a procedure.

- Entry: IEFVINB
- Exit: Return to caller
- Tables/Work Areas: IWA, PARMLIST, QMPA, WORKAREA
- Attributes: Reenterable
- Control Section: IEFVINB

IEFVINC: In-stream Procedure Directory Build Routine

This routine builds a directory entry, if one is needed, for an in-stream procedure.

- Entry: IEFVINC
- Exit: Return to caller
- Tables/Work Areas: IWA, PARMLIST, QMPA, WORKAREA
- Attributes: Reenterable
- Control Section: IEFVINC

IEFVIND: In-stream Procedure Expand Interface Routine

This routine reads a record from the job queue and issues a LOAD macro instruction specifying the expand routine IEZDCODE to expand the record.

- Entry: IEFVIND
- Exit: Return to caller
- Tables/Work Areas: DCBD, IWA, PARMLIST, QMPA
- Attributes: Reenterable
- Control Section: IEFVIND

IEFVINE: In-stream Procedure Syntax Check Routine

This routine syntax checks the PROC and END statements for invalid or non-existant labels and/or null operands with comments.

- Entry: IEFVINE
- Exit: Return to caller
- Tables/Work Areas: 256 byte translate and test table
- Attributes: Reenterable
- Control Section: IEFVINE

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, IEFVMCVL, IEFVMQMI, VM7000, VM7055, VM7055AA, VM7060, VM7070, VM7090, VM7130, VM7370, VM7700, VM7742, VM7750, VM7850, VM7900, VM7950
- Exits: To IEFVM2LS, IEFVM3LS, IEFVM4LS, IEFVM5LS, IEFVM6LS, and IEFXCSSS, IEFDSOAL
- 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: XCTL to IEFVMLS1
- 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 generation data group (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

IEFVRRC: Reinterpretation Control Routine

This routine passes control among the routines that modify the queue entry of a restart step so that they appear as they were prior to the initiation of the step.

- Entry: IEFVRRC, IEFVRRCA, IEFVRRCB
- Exit: Return to caller
- Attributes: Read-only reenterable
- Tables/Work Areas: NEL, JCT, SCT, SIOT, JFCB, JFCBX, VOLT, SMB, DSENQ, SCD, DSB, QMPA
- Control Section: IEFVRRC

IEFVRR1: Dequeue Interface Routine

This routine interfaces with queue management to cause a specific job to be dequeued and the JCT for that job to be read into main storage.

- Entry: IEFVRR1
- Exit: Return to caller
- Tables/Work Areas: QMPA, JCT
- Attributes: Read-only, reenterable
- Control Section: IEFVRR1

IEFVRR2: Table Merge Routine

This routine merges the reinterpreted queue entry tables of a restart step with the original queue tables for that step.

- Entry: IEFVRR2, IEFVR2AE
- Exit: Return to caller
- Tables/Work Areas: QMPA, JCT, ACT, SMB, SCT, SIOT, JFCB, DSENQ, VOLT, JFCBX, NEL
- Attributes: Reenterable
- Control Section: IEFVRR2

IEFVRR3: Reinterpretation Delete/Enqueue Routine

This routine deletes the reinterpreted input and output queue entries of a restart step, constructs the internal JCL necessary for processing a checkpoint restart, and reenqueues the job's queue entry.

- Entry: IEFVRR3, IEFVR3AE
- Exit: Return to caller
- Tables/Work Areas: QMPA, JCT, SCT, SIOT, JFCB
- Attributes: Reenterable
- Control Section: IEFVRR3

IEFVSDRA: Restart Activation Routine

This routine issues a START Restart Reader command for one or more jobnames. This routine is entered from IEFSD168 during a problem program restart or IEFSD305 during a warm start.

- Entry: IEFVSDRA
- Exit: Return to caller
- Tables/Work Areas: CSCB, CVT, TCB
- Attributes: Reenterable
- Control Section: IEFVSDRA

IEFVSDRD: Restart Determination Routine

This routine initiates automatic restarts.

- Entry: IEFVSDRD
- Exit: To IEFSD305
- Tables/Work Areas: JCT, SCT, QMPA, CVT, TIOT, LCT
- Attributes: Reenterable
- Control Section: IEFVSDRD

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

IEFVSMBR: SMB Reader Routine

This routine reads the SMBs associated with a restarting job and converts the JCL statements to their original format.

- Entry: IGC0005B
- Exits: If called during restart reader processing, return to caller; if called during restart, XCTL to the first load of restart housekeeping.
- Tables/Work Areas: QMPA, DCB, JCT, SMB, RRCWKAR, SCT
- Attributes: Reenterable
- Control Section: IEFVSMBR

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, IEFWC002
- Exit: To IEFWCIMP
- Control Section: IEFWC000, IEFWC002

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, IEFWDMSG
- Exits: To IEFXT000, IEFW41SD, IEFXK000
- Tables/Work Areas: SCT, LCT, TIOT, UCB
- Control Section: IEFWD000, IEFWDMSG

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

IEFWTP00: Write-to-programmer Initialization Routine

This routine initializes storage and registers to process write-to-programmer messages if the WTP call is valid.

- Entry: IGC0203E from IEECVWTO
- Exits: Normal to IEFWTP01, to calling program if only a WTP is requested and the WTP limit has been exceeded, to IEECVWTO if WTP request cannot be processed or a WTO message was also requested.
- Tables/Work Areas: WTPCB, JSCB, UCM, CVT, WPL, IEFQMNGR, IEFQMRES
- Attributes: Reenterable

- Control Section: IGC0203E

IEFWTP01: Write-to-programmer Message Processing Routine

This routine processes the WTP messages and writes them on the job queue using the transient queue manager (SVC-90).

- Entry: IGC0303E from IEFWTP00 or IEFWTP02
- Exits: Normal to IEECVWTO or to calling program if only a WTP is requested; to IEFWTP02 for processing I/O errors which occur while writing a WTP message or for job queue problems.
- Tables/Work Areas: WTPCB, JSCB, UCM, CVT, WPL, IEFQMNGR, IEFQMRES
- Attributes: Reenterable
- Control Section: IGC0303E

IEFWTP02: Write-to-programmer Error Processing Routine

This routine handles WTP processing using the reserved WTP SMBs for messages when there are I/O errors in the job queue or when WTP is unable to get a record assigned for a WTP message using the transient queue manager.

- Entry: IGC0403E from IEFWTP01
- Exits: Return to IEECVWTO, to calling program if only a WTP was requested, or IEFWTP01 if a system WTP message is to be processed following a NO RECORD message.
- Tables/Work Areas: WTPCB, JSCB, UCM, CVT, WPL, IEFQUNGR, IEFQMRES
- Attributes: Reenterable
- Control Section: IGG0403E

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, IEFV15XI, 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, MSCFF
- 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

IEFXQM00: Transient Queue Manager Initialization and Read/Write Routine

This routine initializes tables and read or writes job queue records.

- Entry: IGC00090
- Exits: XCTL to IGC01090 or return to caller
- Tables/Work Areas: Q/M resident data area, QMPA, CVT, ECB/IOB
- Attributes: Reenterable
- Control Section: IGC00090

IEFXQM01: Transient Queue Manager Track Assignment Routine

This routine assigns logical tracks as required.

- Entry: IGC01090
- Exits: XCTL to IGC02090 or return to caller
- Tables/Work Areas: QM resident data area, QMPA, CVT, ICB/IOB
- Attributes: Reenterable
- Control Section: IGC01090

IEFXQM02: Transient Queue Manager Record Assignment Routine

This routine assigns records to a queue entry.

- Entry: IGC02090
- Exits: Return to caller
- Tables/Work Areas: QM resident data area, QMPA, CVT, ECB/IOB
- Attributes: Reenterable
- Control Section: IGC02090

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 -- VARY Command Interface TICT Compression Routine

This routine reduces the TIOT to its final size and provides an interface with the VARY command.

- Entry: IEFXT002, XTTRDJ, XTTEE3, XTTEA1, XTTEA0
- Exits: to IEFXKIMP, IEFXT003, IEF41FAK
- 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

IEFXV002: I/O Device Allocation -- Automatic Volume Recognition, Label Processing

This routine reads the label of a newly mounted volume, extracts the serial number, and places it into the UCB for the corresponding device.

- Entry: IEFXV002
- Exits: To IEFXVNSL via CALL, return to caller.
- Tables/Work Areas: LUT, UCB, CVT, DEB, IOB
- Attributes: Reusable
- Control Section: IEFXV002

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, IEFRPREP
- 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, IEFYNIMP

- 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. If a job used DSO, the data set blocks are marked inactive.

- Entry: IEFYT
- Tables/Work Areas: JCT, SCT, TIOT, SIOT, QMPA, DSCB, LCT, CVT, JFCB, DSOCB, PIB
- 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 IEFZGJ, 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: IEFZGJ, 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

IEFZGST2: Termination -- Unallocation Routine

This routine makes available to other data sets the units used by the terminating job step.

- Entry: IEFZG2, ZGOK09, ZOOA1, ZOOE10, ZPOC10, ZPOQMGR2

- Exit: Return to caller
- Tables/Work Areas: LCT, PDQ, SIOT, TIOT, UCB, JFCB, QMPA
- Control Section: IEFZG2

IEFZHMSG: Termination -- VARY Command Interface and Disposition and Deallocation Message Routine

This routine prepares messages to the programmer and to the operator for the disposition and allocation routines. It also provides an interface with the VARY command.

- 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 Cutput 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

IEF41DUM: Allocation -- Return to Initiator or System Task Control

This routine returns control to the Initiator or to System Task Control after device allocation has completed. If allocation was called by the Initiator, the routine returns control to step initiation routine IEFSD512 at entry point IEFAIRET via an XCTL macro instruction. If allocation was called by System Task Control, the routine returns control to the caller.

- Entry: IEFSD41R
- Exits: IEFAIRET or return to caller
- Tables/Work Areas: CSCB, LCT
- Attributes: Reenterable
- Control Section: IEFSD41R

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

IEF589SP: Initiator -- Linkage to IEFSD584

This routine links to system task control so that upon return, the initiator will be in supervisor state.

- Entry: IEF589SP from IEFSD599
- Exit: Link to IEFSD584
- Control Section: IEF589SP

IEZDCODE: Interpreter -- In-stream Procedure Expand Routine

This routine expands a given statement to its original form by inserting a given character.

- Entry: IEZDCODE
- Exit: Return to caller
- Tables/Work Areas: IEZPARM
- Attributes: Reenterable
- Control Section: IEZDCODE

IEZNCODE: Interpreter -- In-stream Procedure Compress Routine

This routine compresses a given statement by removing a given character. The new statement that is formed contains the number of the removed character.

- Entry: IEZNCODE
- Exit: Return to caller
- Tables/Work Areas: IEZPARM
- Attributes: Reenterable
- Control Section: IEZNCODE

IGC0008C: SVC 83 -- SMF Buffer Manager and Writer Routine

The purpose of this routine is to move SMF records into the SMF buffer, and to cause the records to be written when the buffer is full.

- Entry: IGC00083
- Exit: IEESMFOP, return to caller
- Tables/Work Areas: CVT, SMCA
- Attributes: Reentrant
- Control Sections: IGC00083

IGC0103D: SVC -- Master Command EXCP Routine

This routine processes the MOUNT Command.

- Entry: IGC0103D
- Attributes: Reenterable, transient
- Control Section: IGC0103D.

IGF2403D: SVC 34 - VARY PATH Command Processor

This routine is used only if the Alternate Path Retry option is supported. It processes an operator's request for varying a PATH to a device and causing that path to be either logically brought online or logically removed from the system.

- Entry: IGF2403D from IEE3202D
- Exit: To IEE0503D, IEE2103D
- Attributes: Reenterable
- Tables/Work Areas: Test Channel Table, SVRB Extended Save Area
- Control Sections: IGF2403D

IGF2503D: SVC 34 - SWAP Command Processor

This routine is used only if Dynamic Device Reconfiguration is in the system. It processes the operator's command to SWAP volumes for Dynamic Device Reconfiguration. The routine checks the command for proper format, sets a switch if the status of DDR is to be changed, validates CUAs, and fills in certain fields of the I/O RMS Communications Area.

- Entry: IGF2503D from IEE0403D or from IGF0408E (DDR Tape Reposition)
- Exit: To IEE0503D, IEEZ103D, IGF0408E (DDR Tape Reposition)
- Attributes: Reenterable
- Tables/Work Areas: I/O RMS Communications Area, SVRB Extended Save Area
- Control Sections: IGF2503D

IGF2603D: SVC 34 -- Machine Status Control Routine

This routine is available only for the model 85. It processes the status parameter of the MODE command.

- Entry: IGF2603D
- Exit: IGF2703D
- Tables/Work Areas: CVT, XSA
- Attributes: Reenterable, read-cnly, self-relocating
- Control Section: IGF2603D

IGF2703D: SVC34 - Machine Status Control Routine

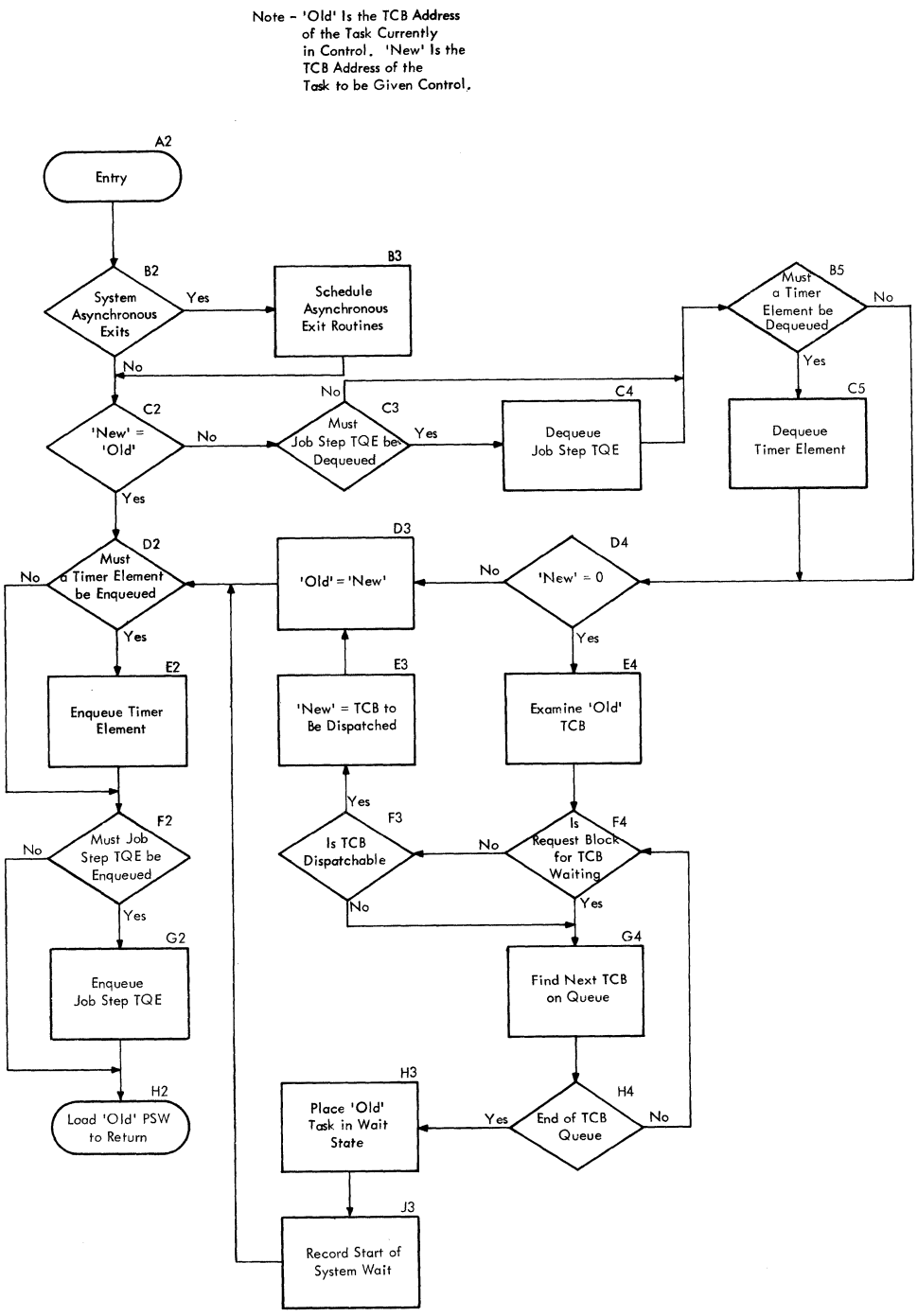
This rcutine is available only for the model 85. It processes all parameters of the MCDE command but the status parameter.

- Entry: IGF2703D
- Exit: Return to issuer of SVC 34
- Tables/Work Areas: CVT, XSA
- Attributes: Reenterable, read-only, self-relocating
- Control Sections: IGF2703D

# Appendix C: Flowcharts

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, GY28-6660.

Chart 01. Task Dispatcher (Without Time Slicing)



## Chart 02. Task Dispatcher (With Time Slicing (Part 1 of 2)

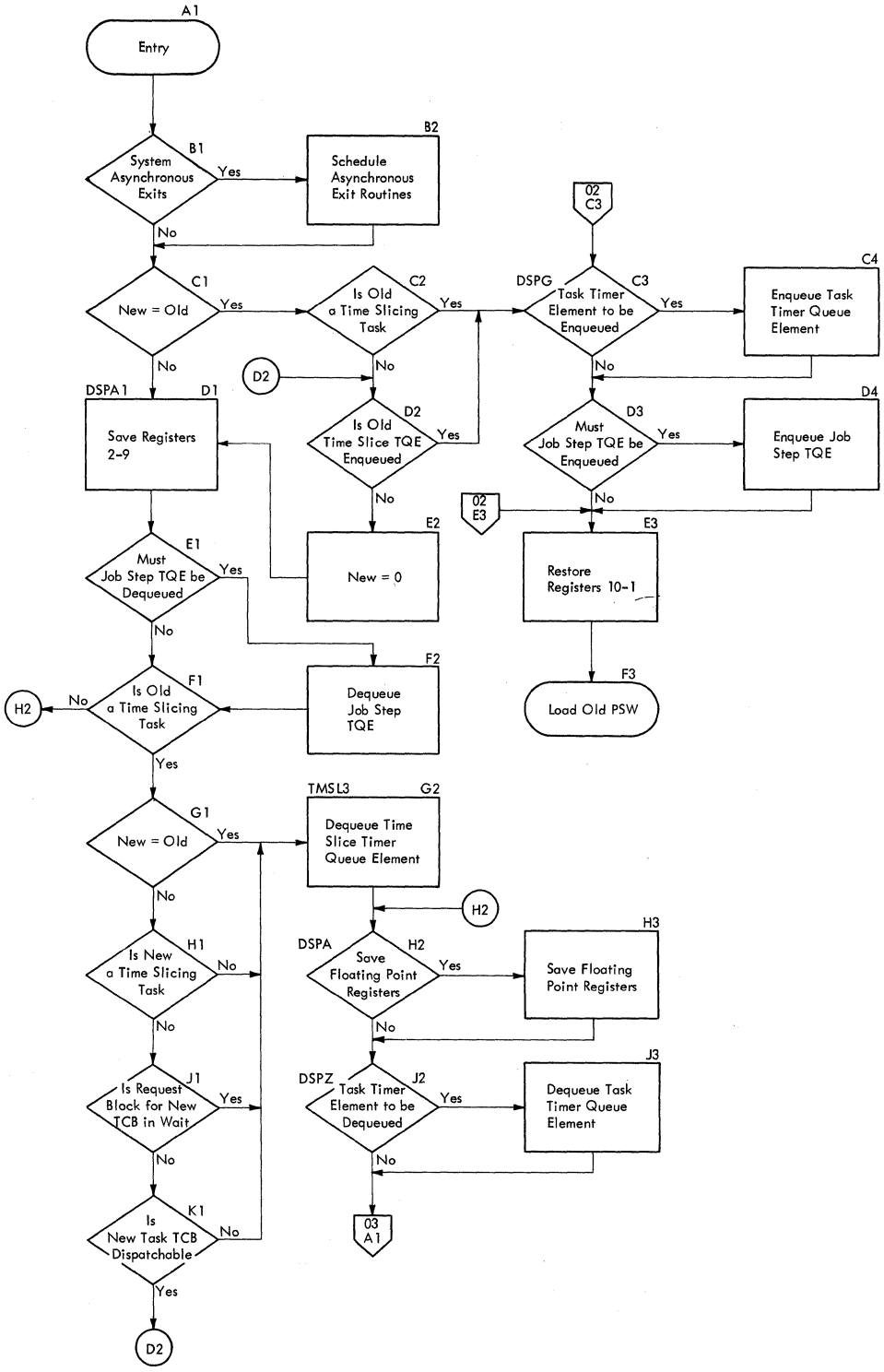
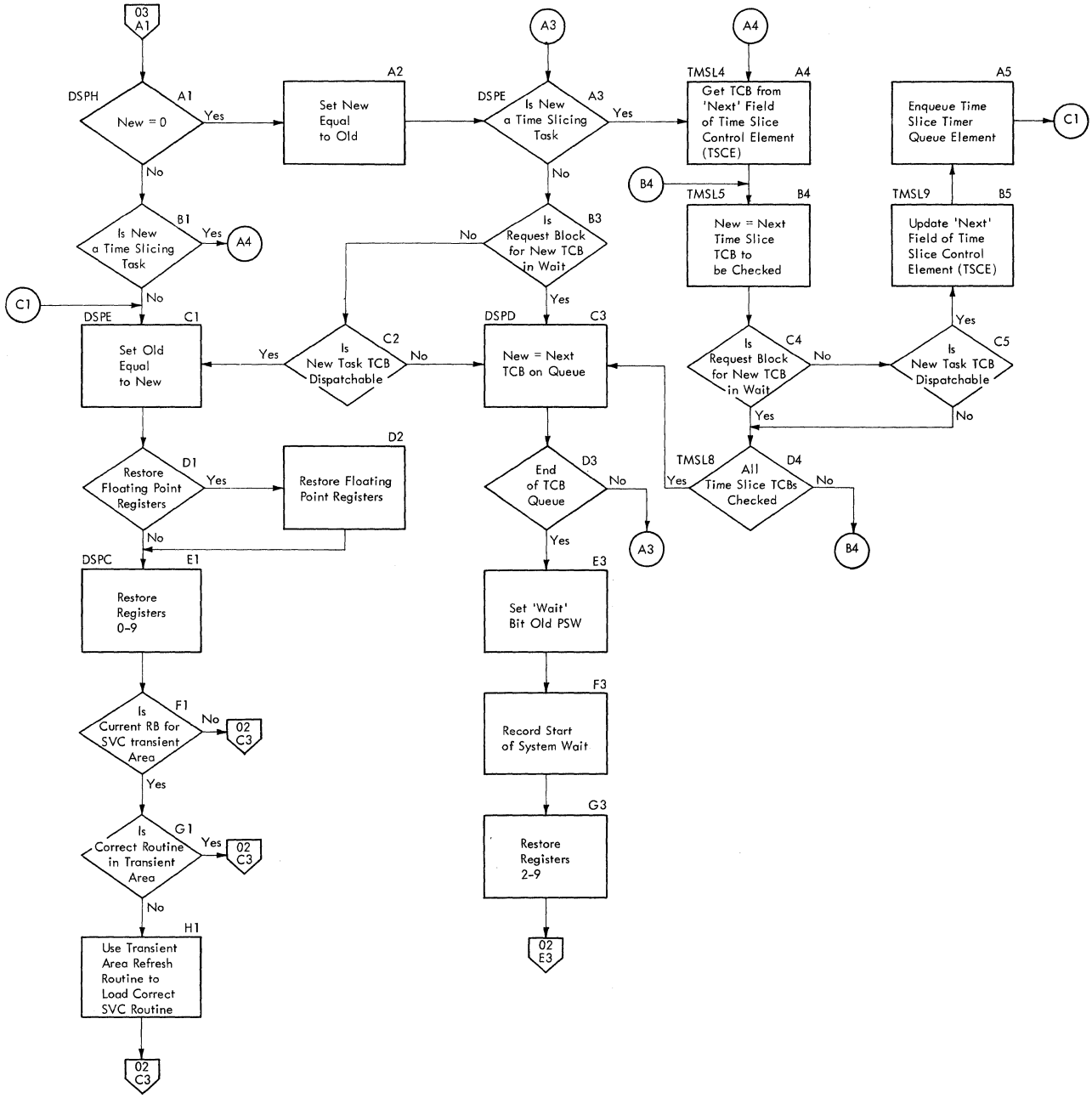
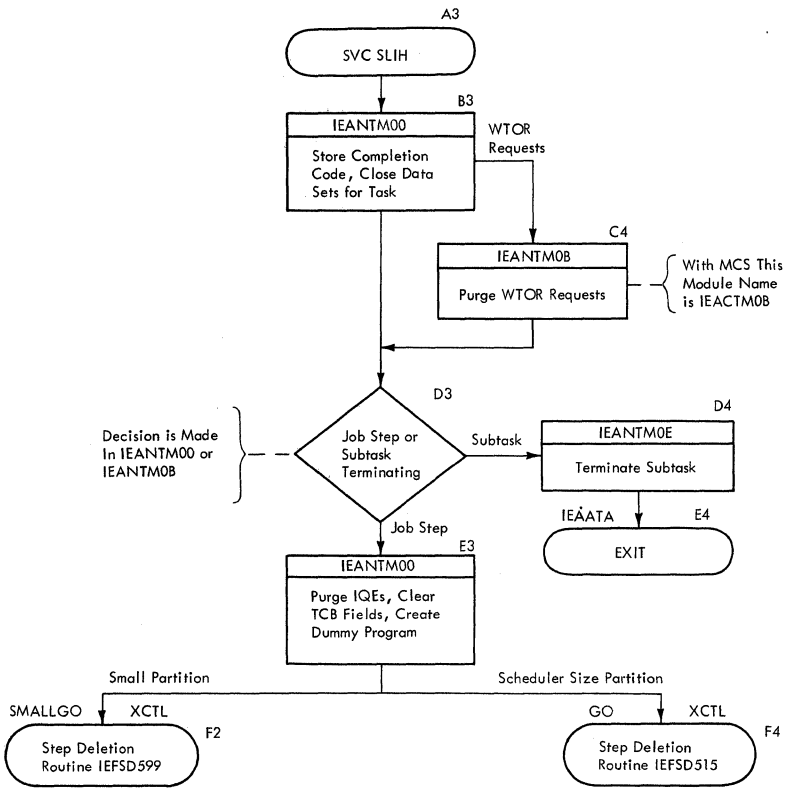


Chart 03. Task Dispatcher (With Time-Slicing) (Part 2 of 2)



## Chart 04. Normal Termination



## Chart 05. Small Partition Routine (Part 1 of 4)

Note - At Entry, Small Partition Has Zero Protection in TCB, PSW and Hardware. Also, PSW is Supervisor State.

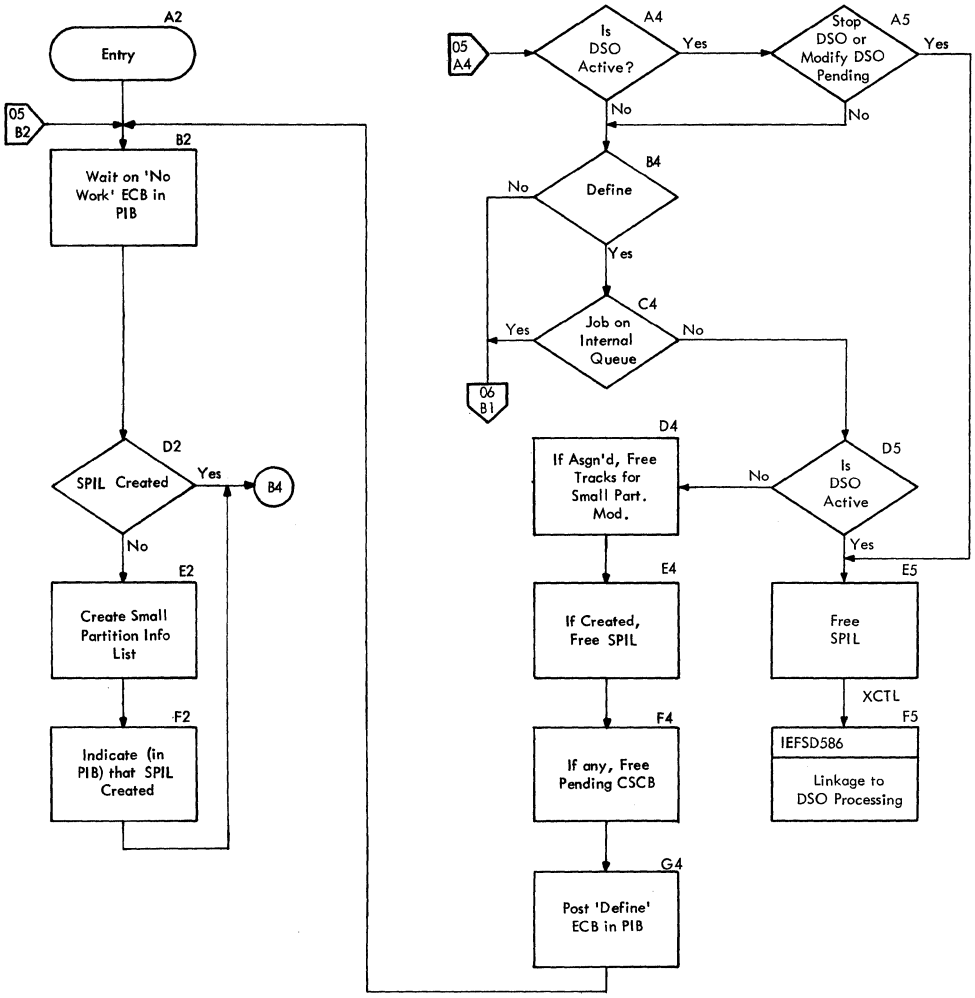


Chart 06. Small Partition Routine (Part 2 of 4)

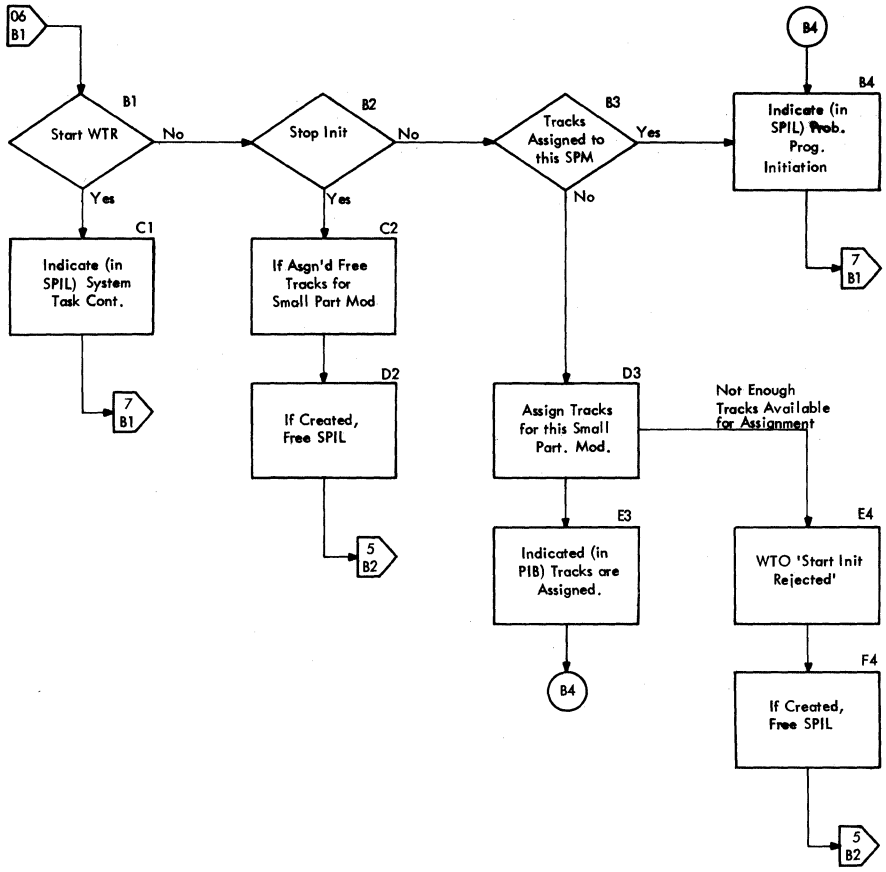


Chart 07. Small Partition Routine (Part 3 of 4)

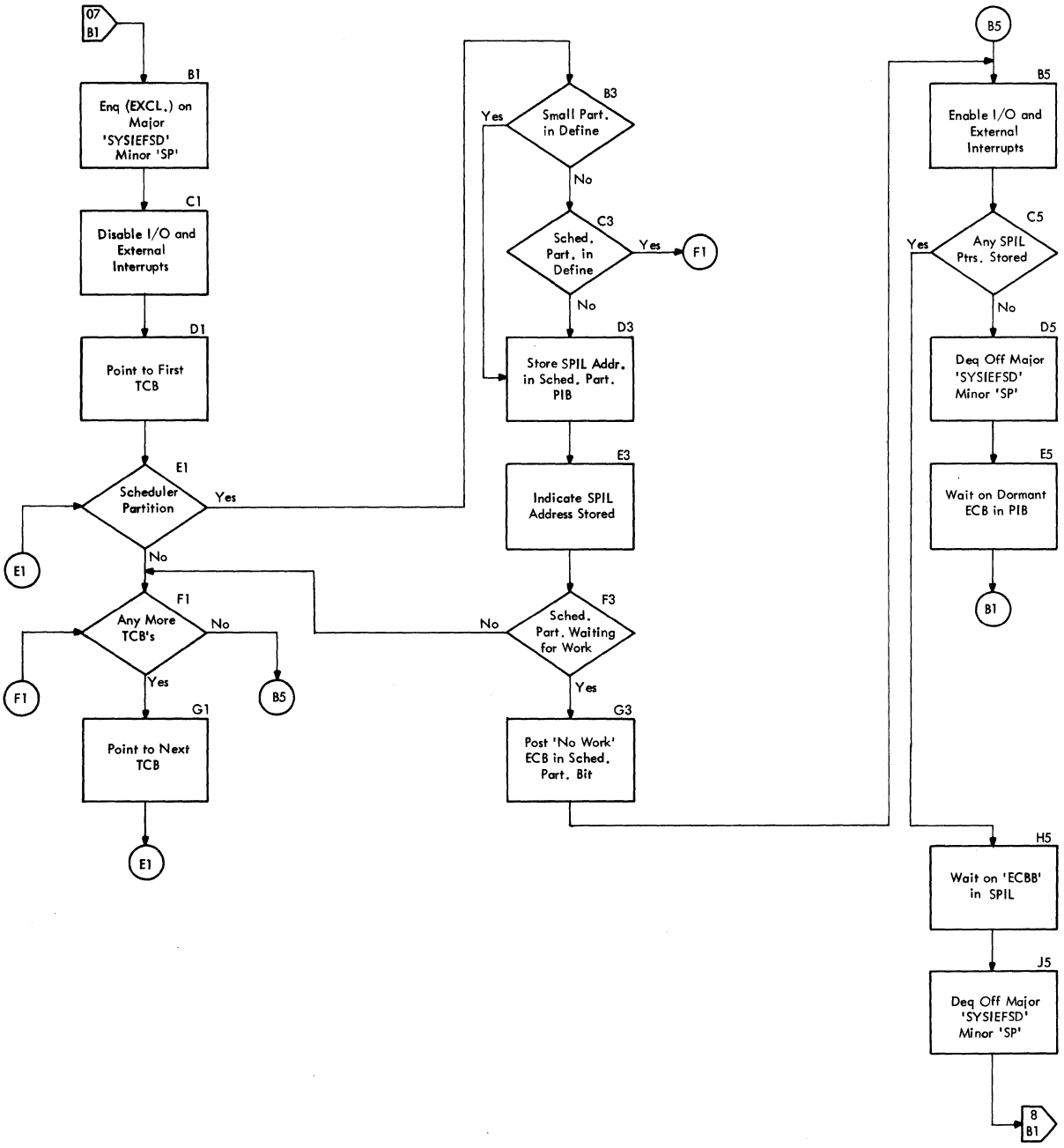


Chart 08. Small Partition Routine (Part 4 of 4)

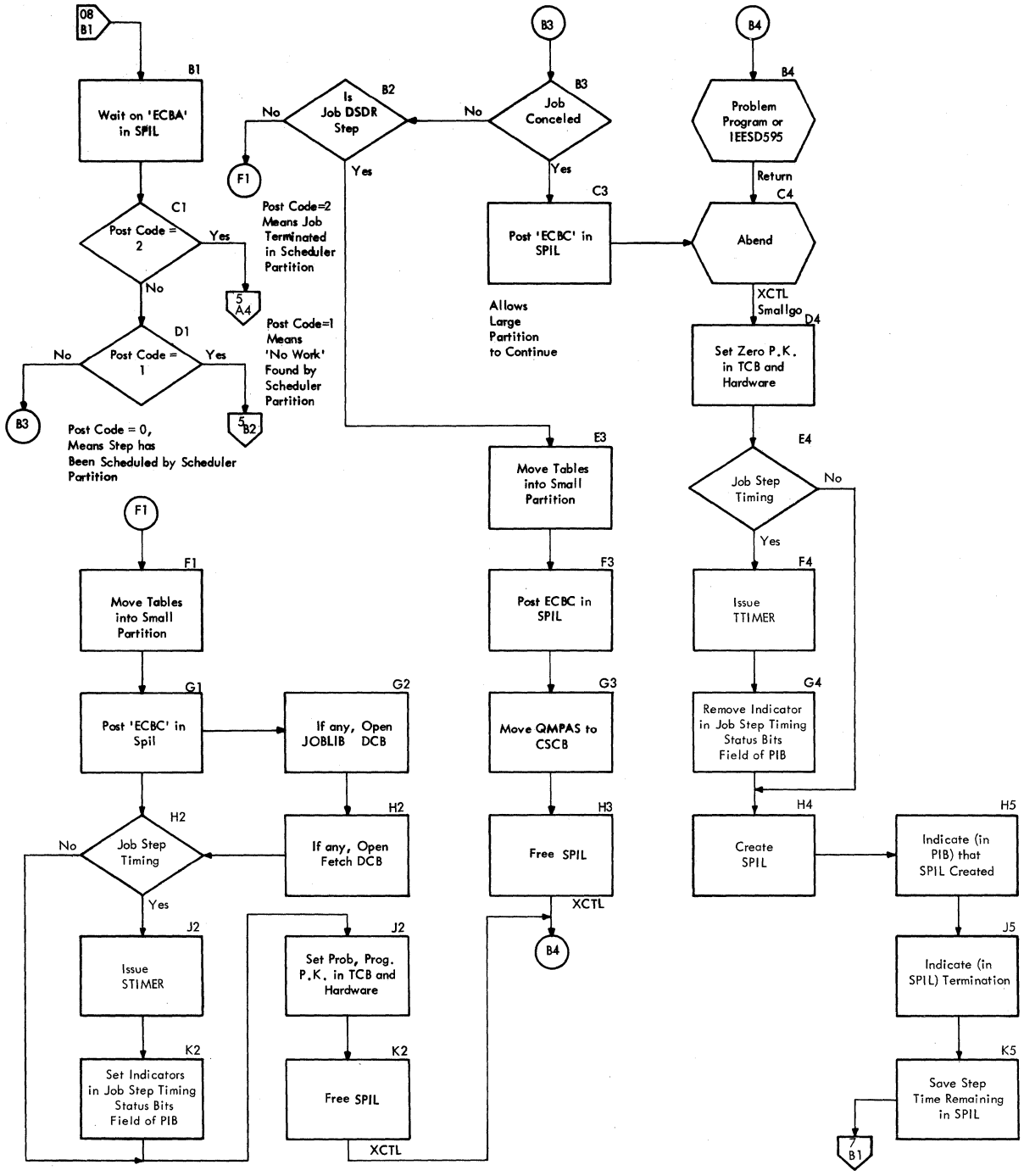


Chart 09. Master Scheduler Task

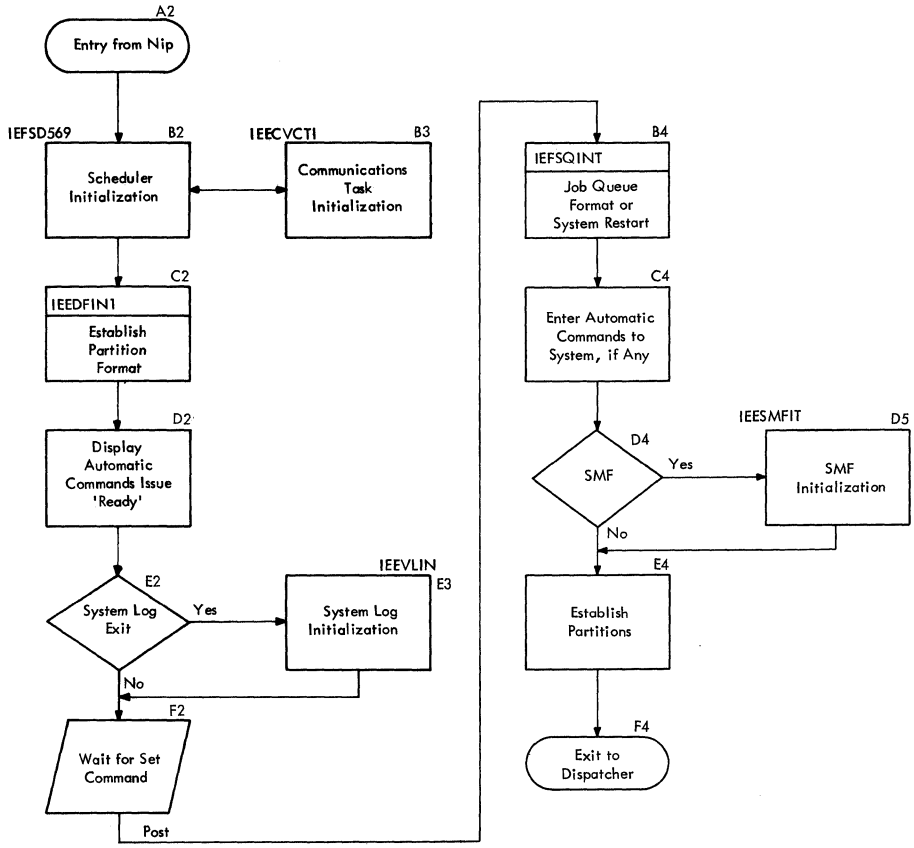


Chart 10. Queue Alter

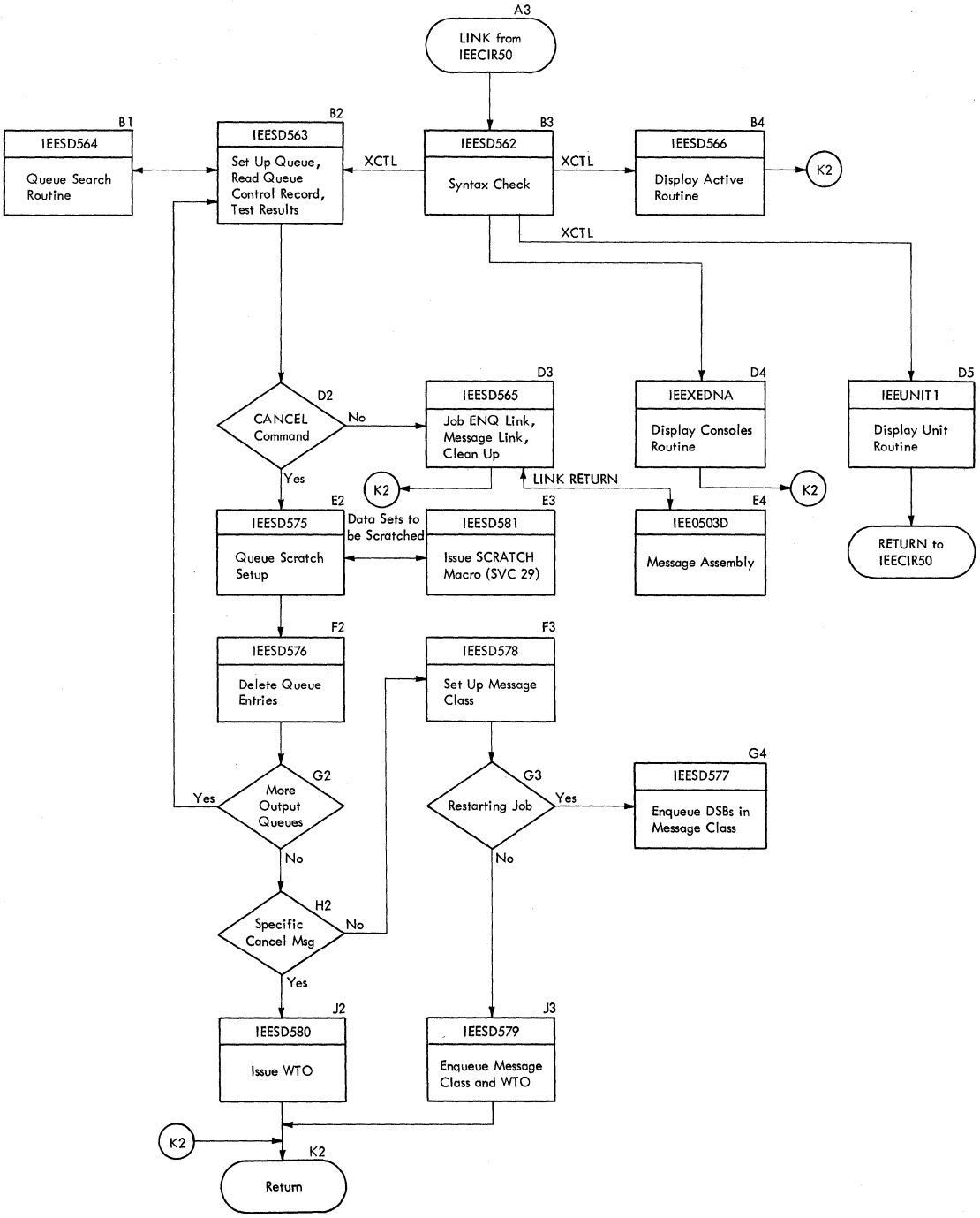


Chart 11. Queue Manager Table Breakup Routine

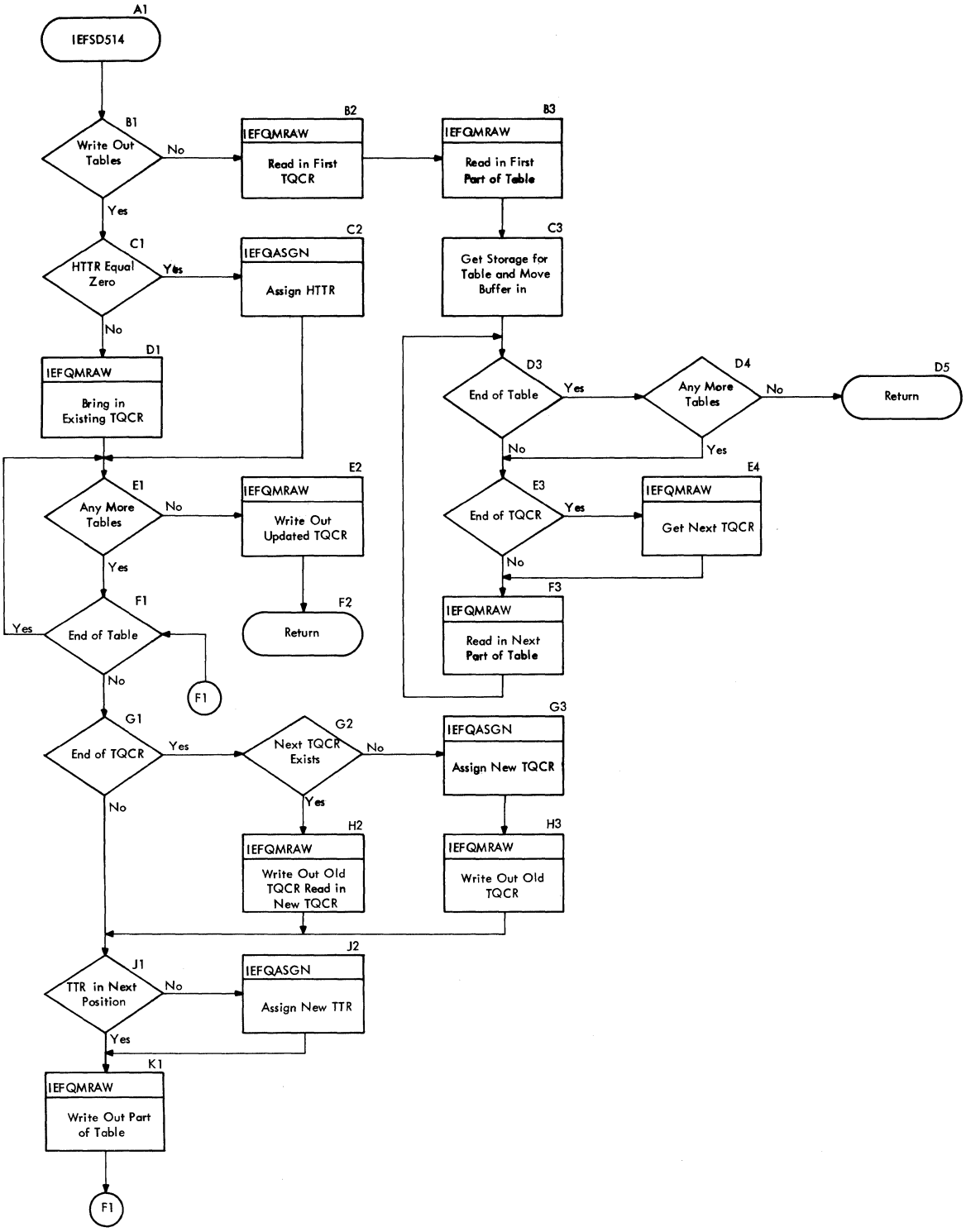


Chart 12. Master Scheduler Resident Command Processcr

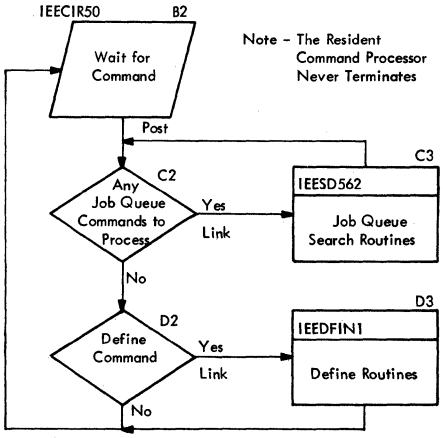
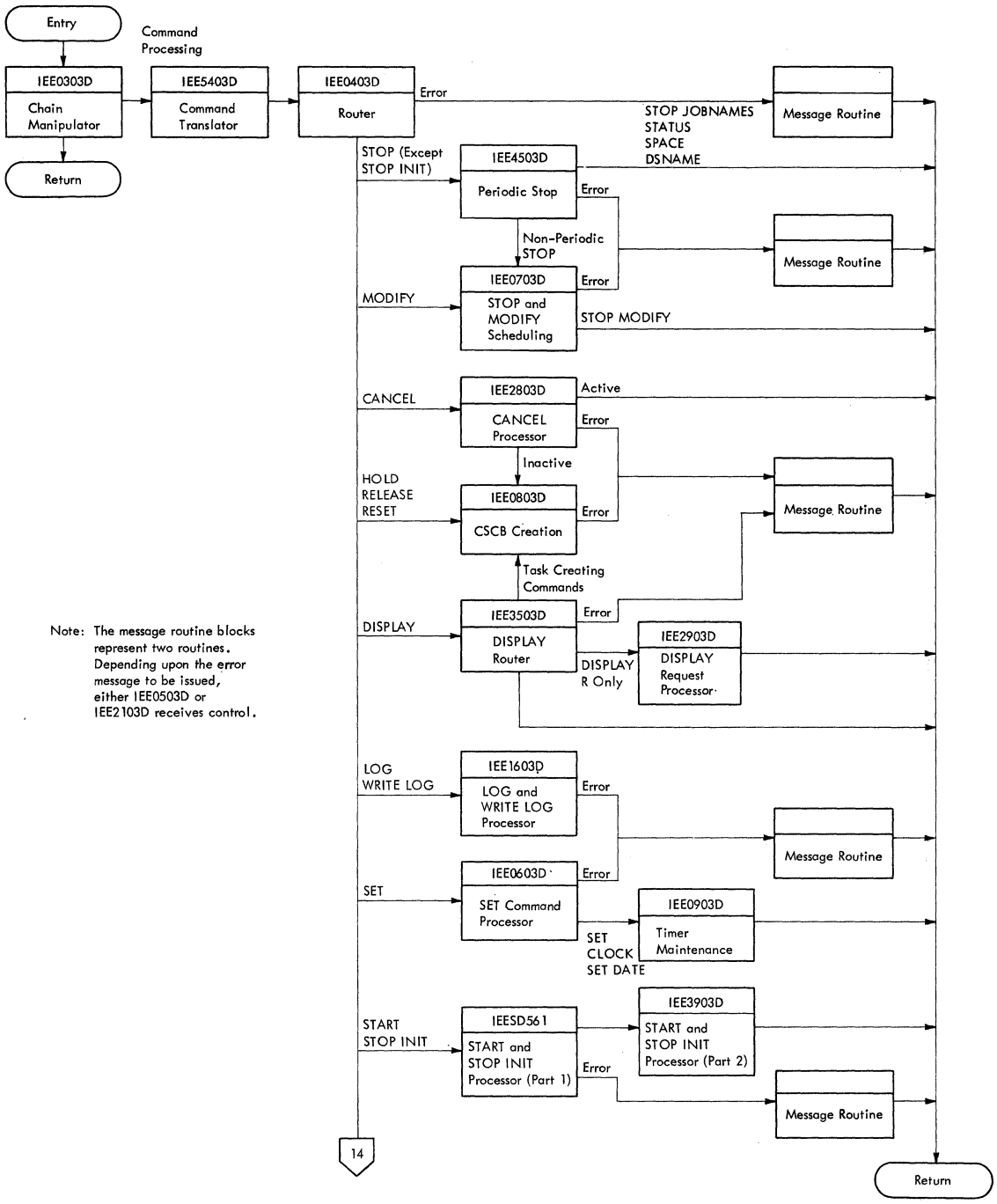


Chart 13. SVC 34 Command Processing (Part 1 of 3)



## Chart 14. SVC 34 Command Processing (Part 2 of 3)

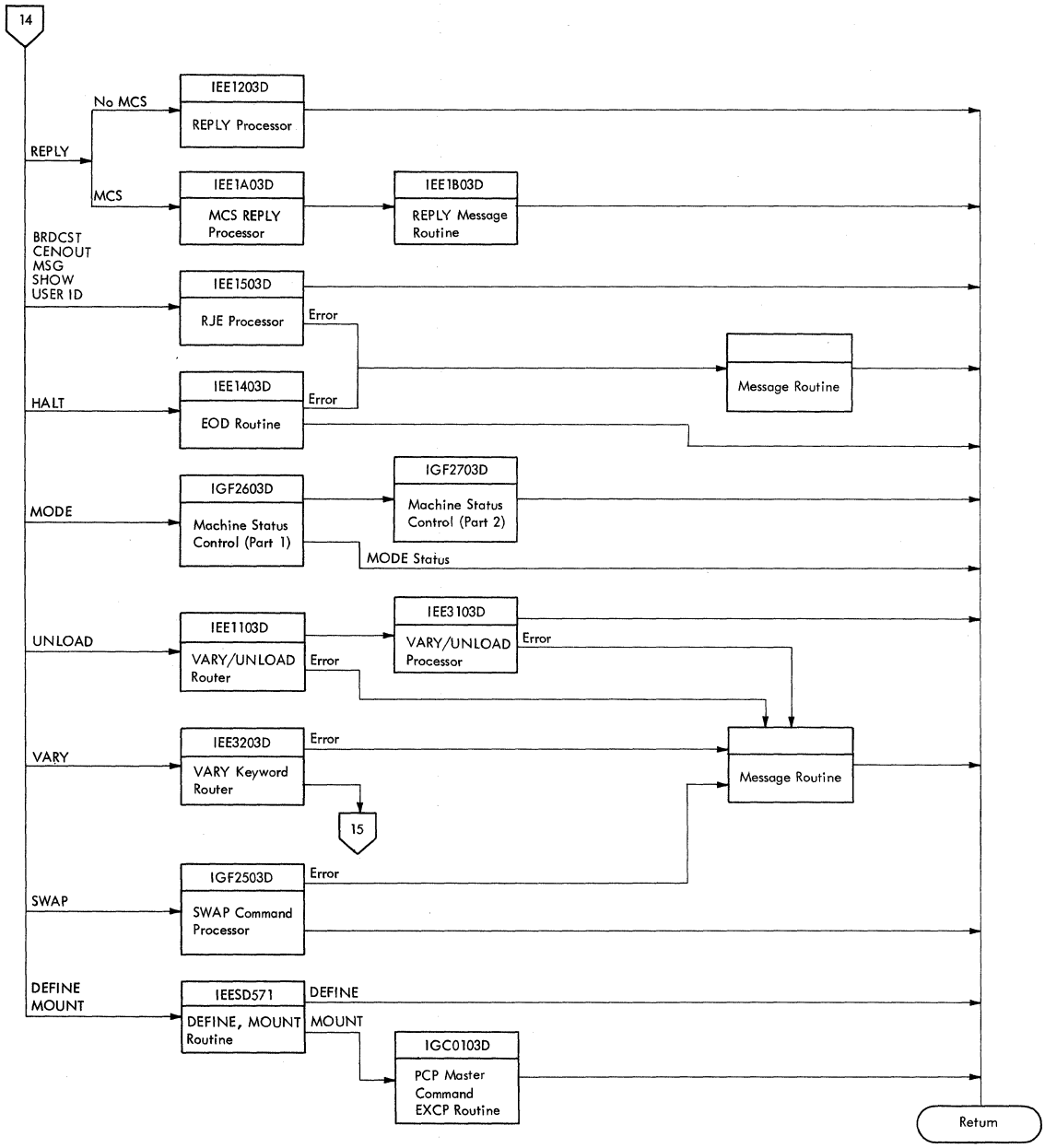
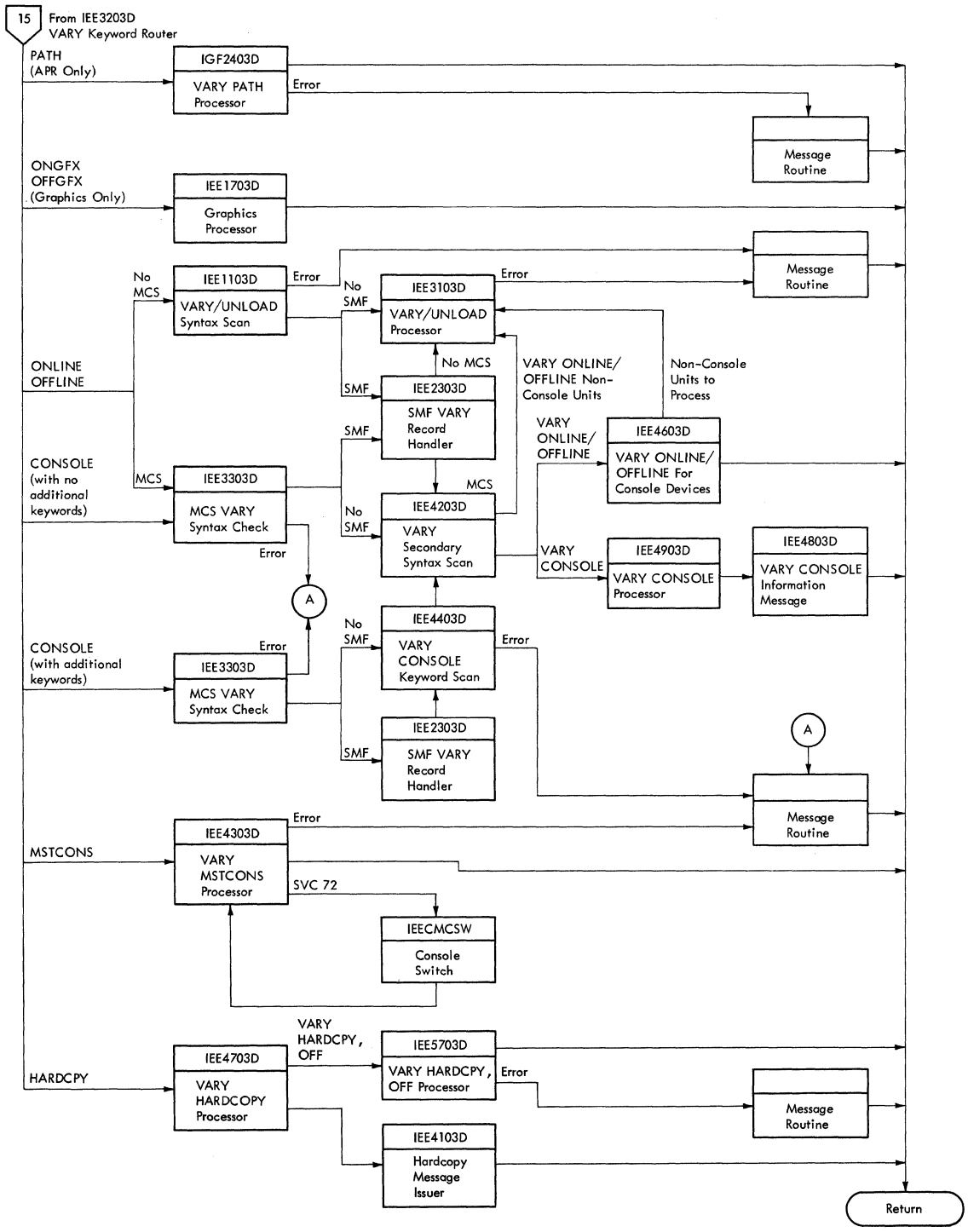


Chart 15. SVC 34 Command Processing (Part 3 of 3)



# Chart 16. Communications Task

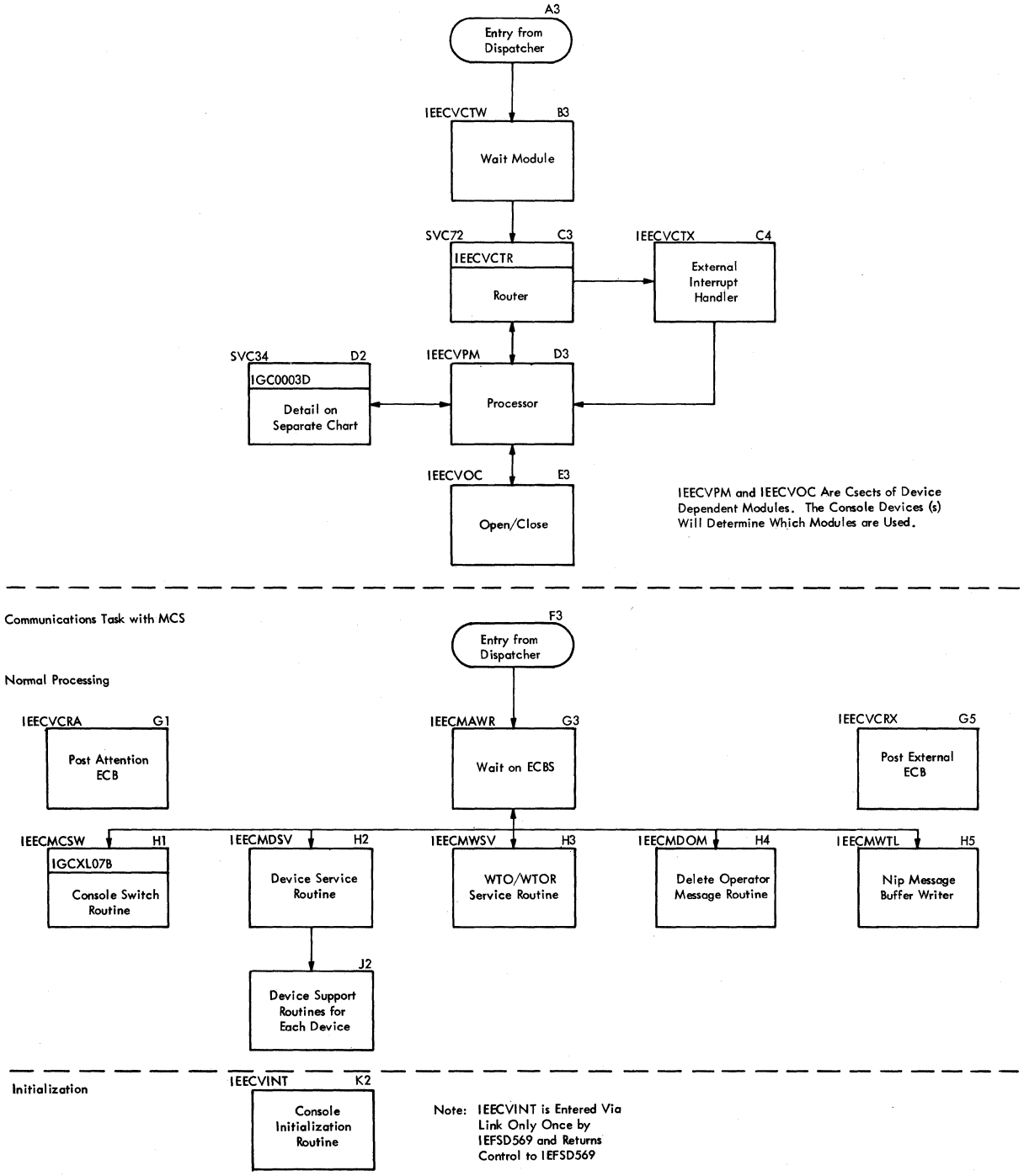


Chart 17. IEFSD518 -- Partition Recovery Routine

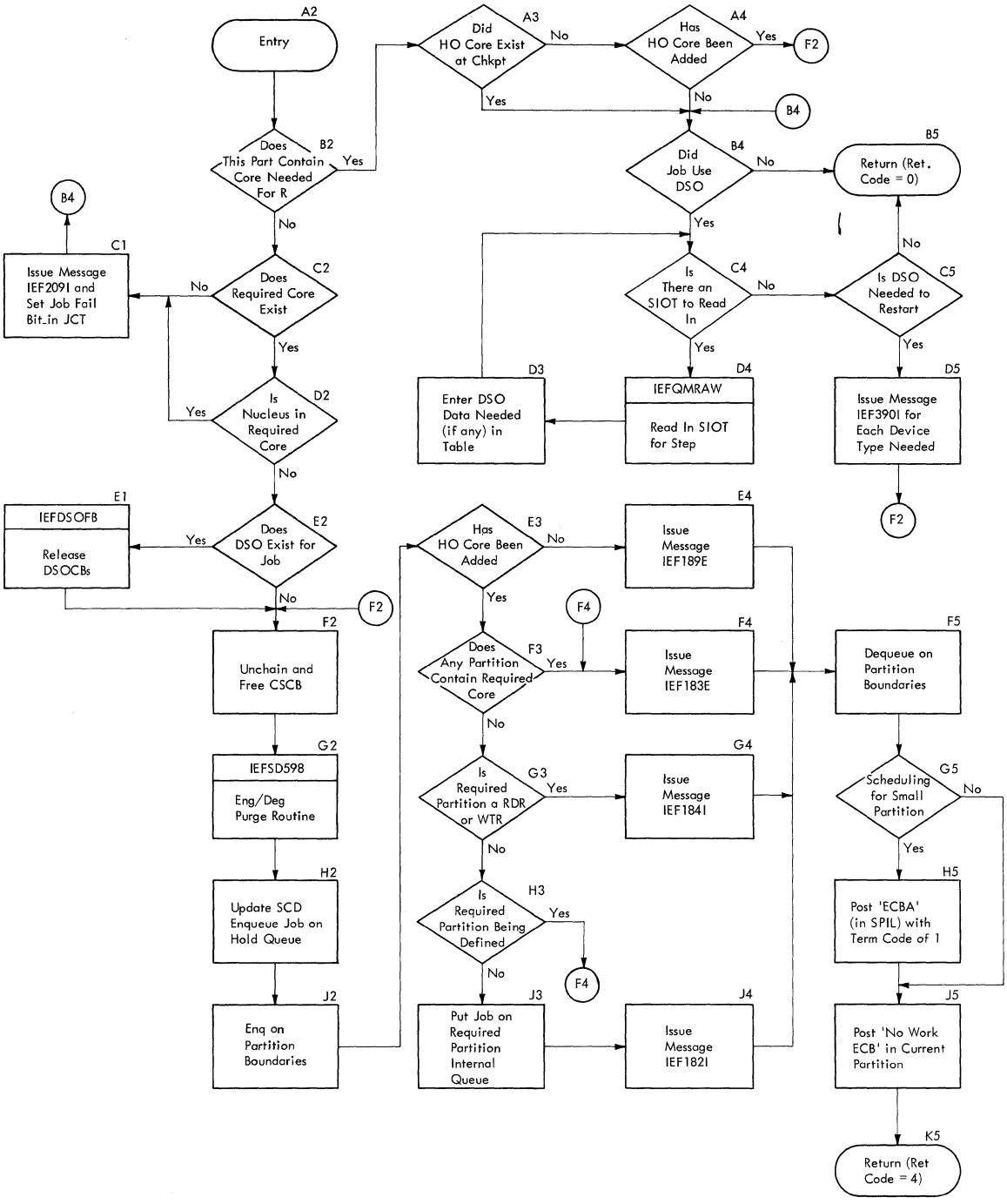


Chart 18. Initiator Control Flow

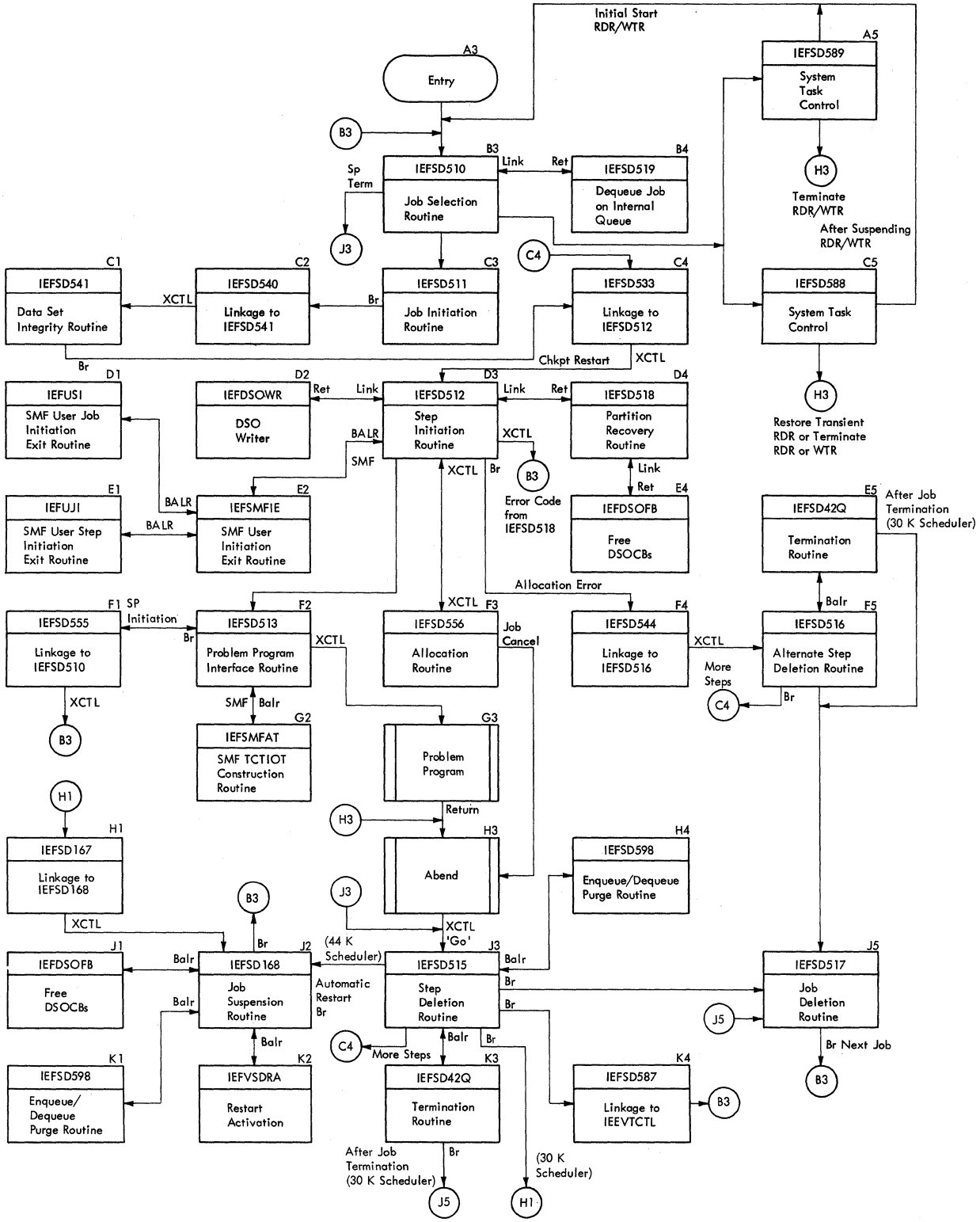


Chart 19. Job Selection Routine (Part 1 of 5)

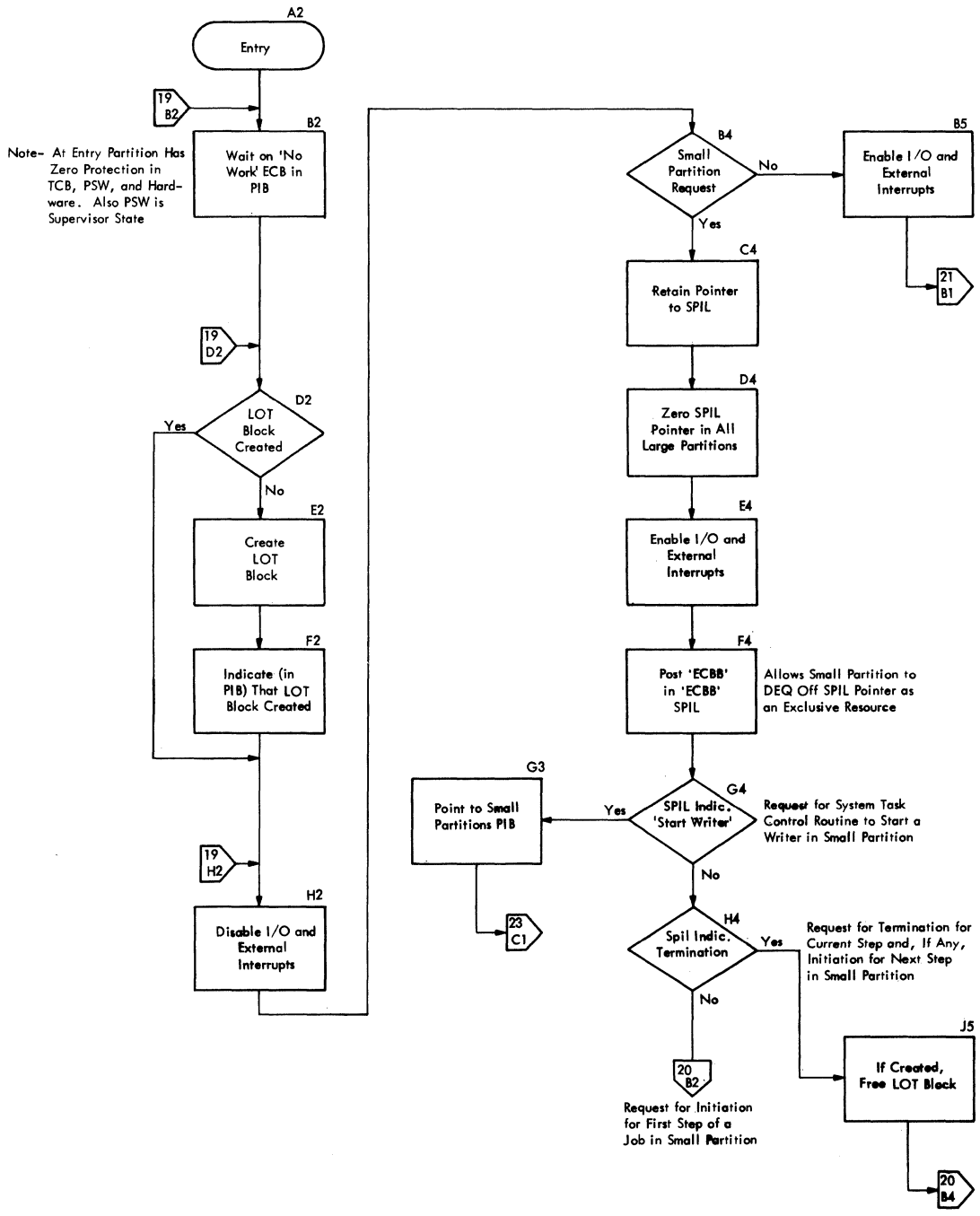


Chart 20. Job Selection Routine (Part 2 of 5)

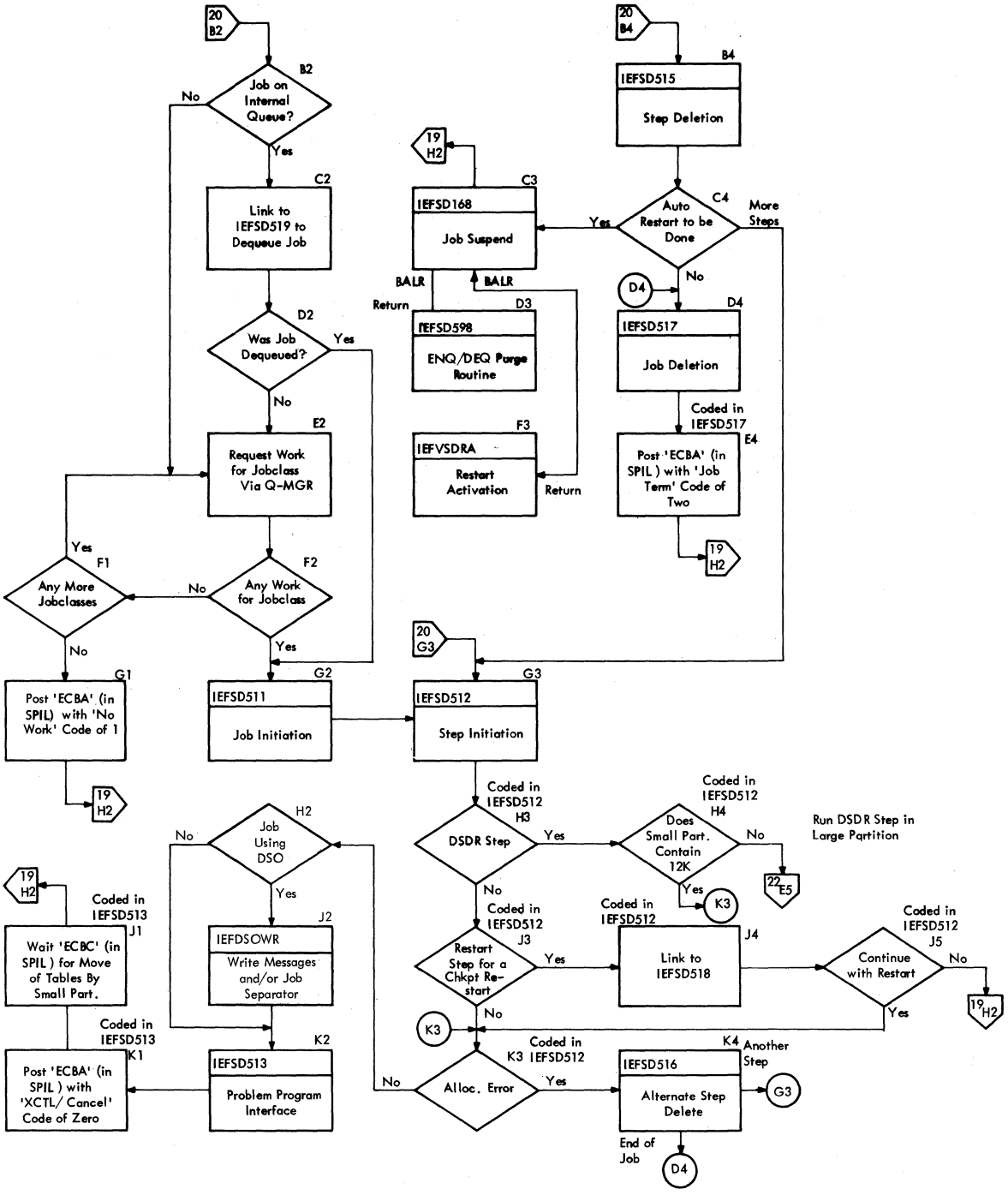


Chart 21. Job Selection Routine (Part 3 of 5)

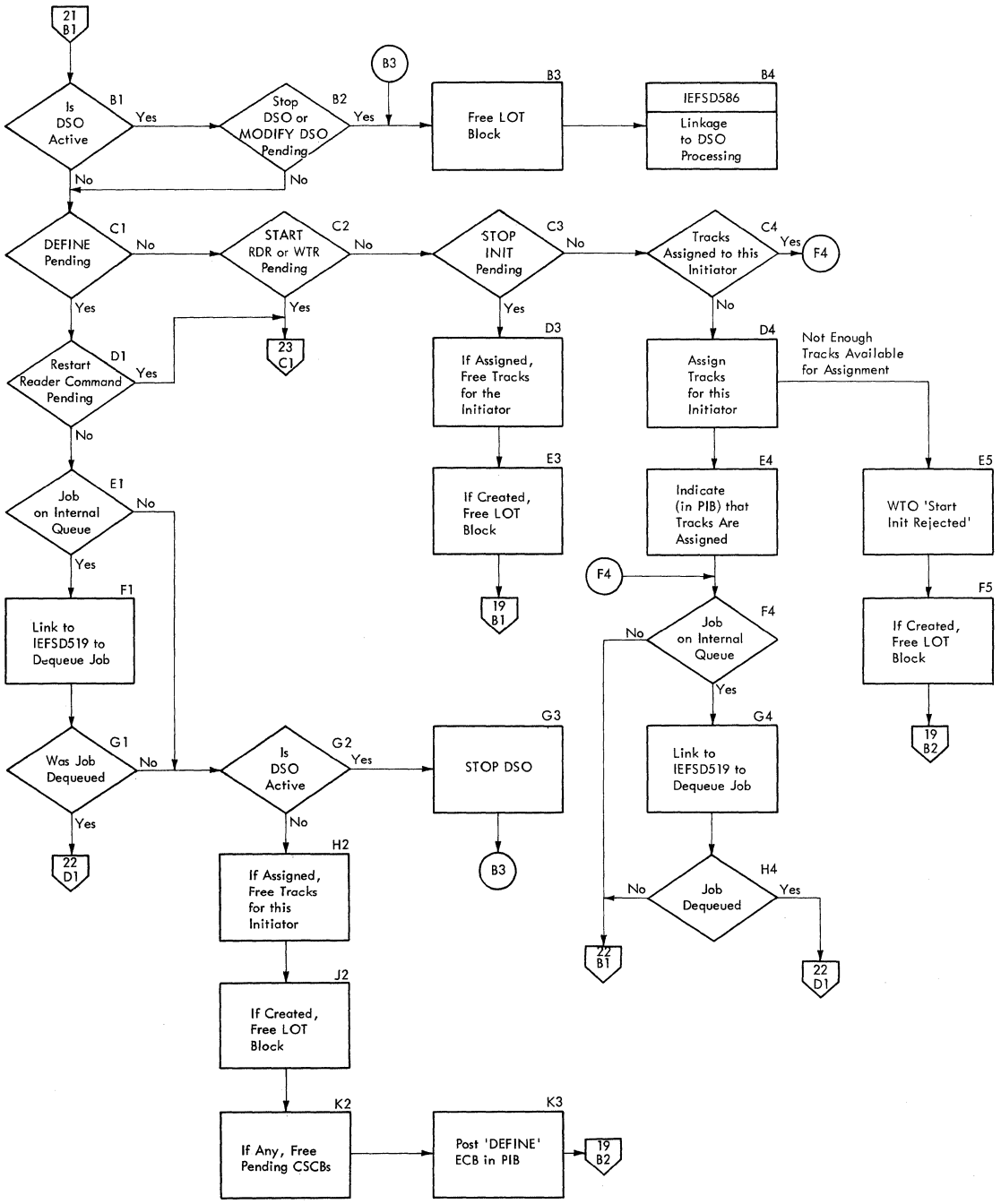


Chart 22. Job Selection Routine (Part 4 of 5)

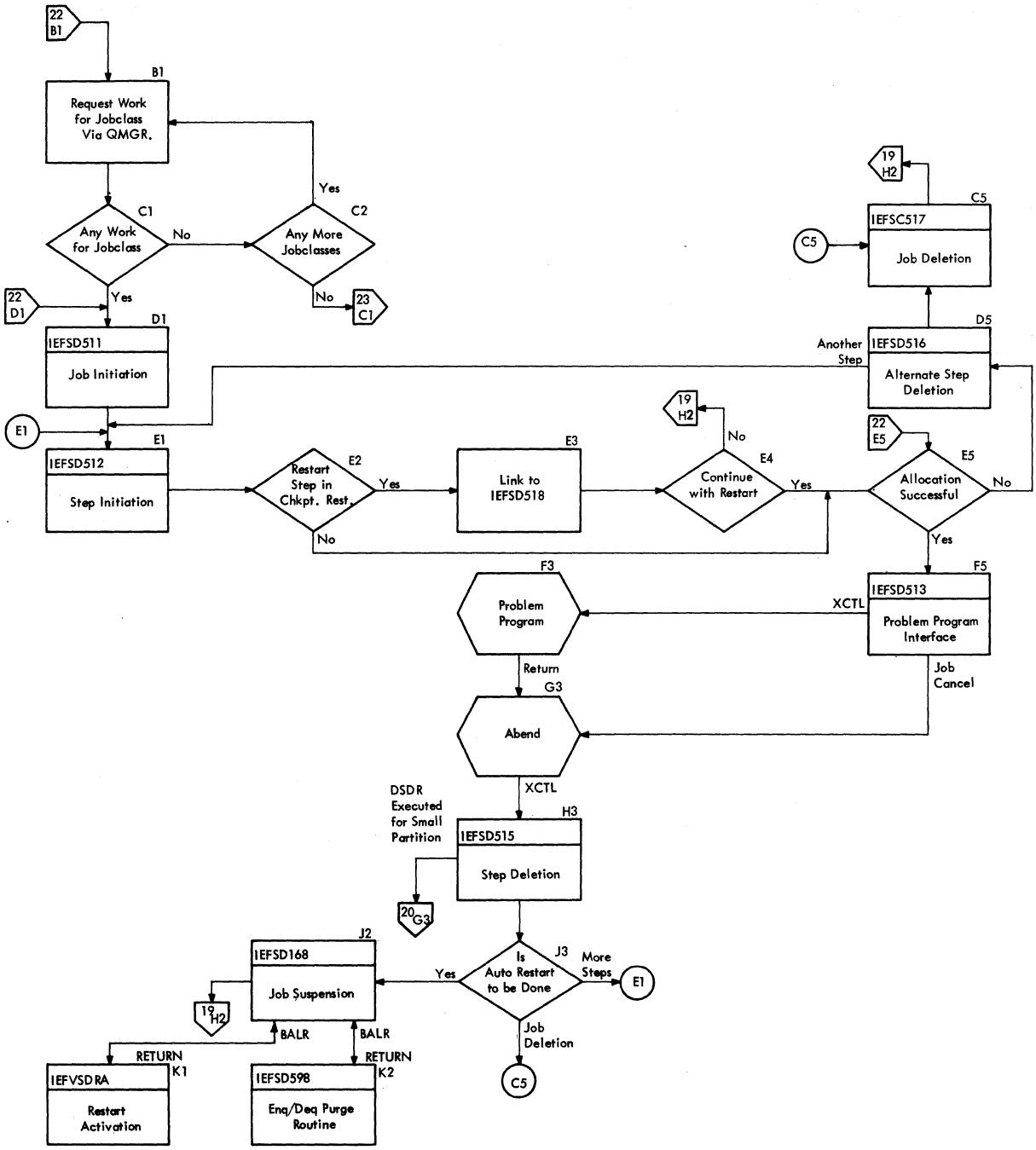


Chart 23. Job Selection Routine (Part 5 of 5)

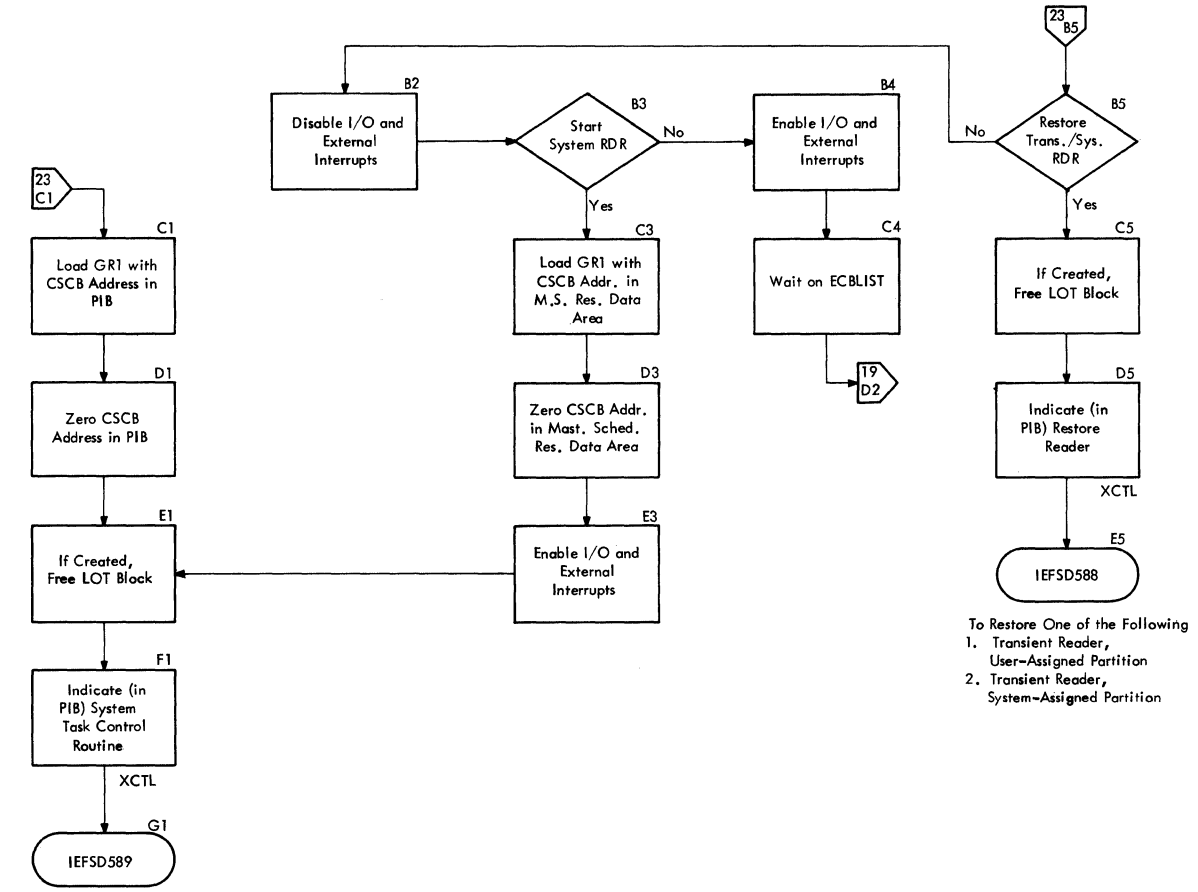


Chart 24. Reader/Interpreter (Part 1 of 3)

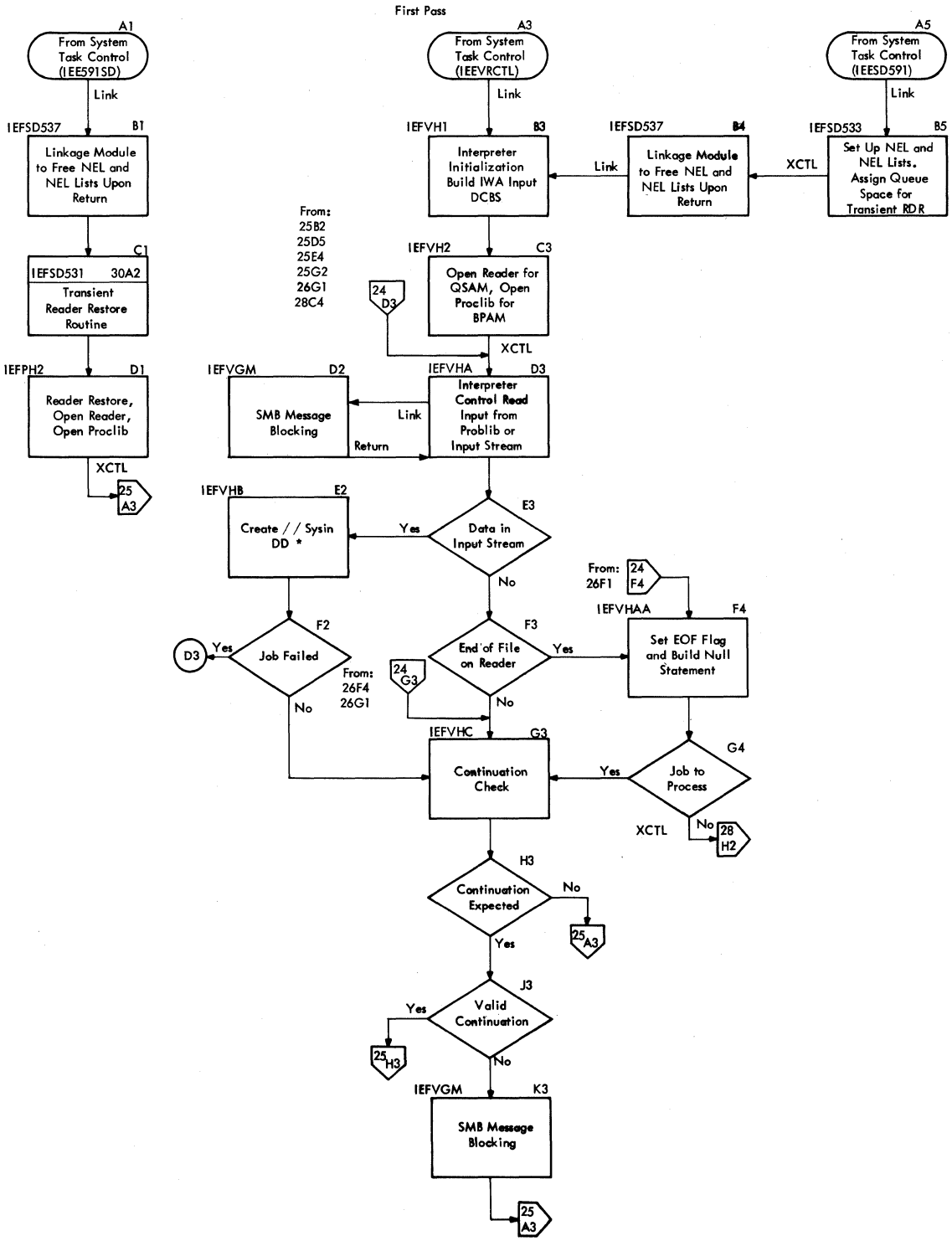
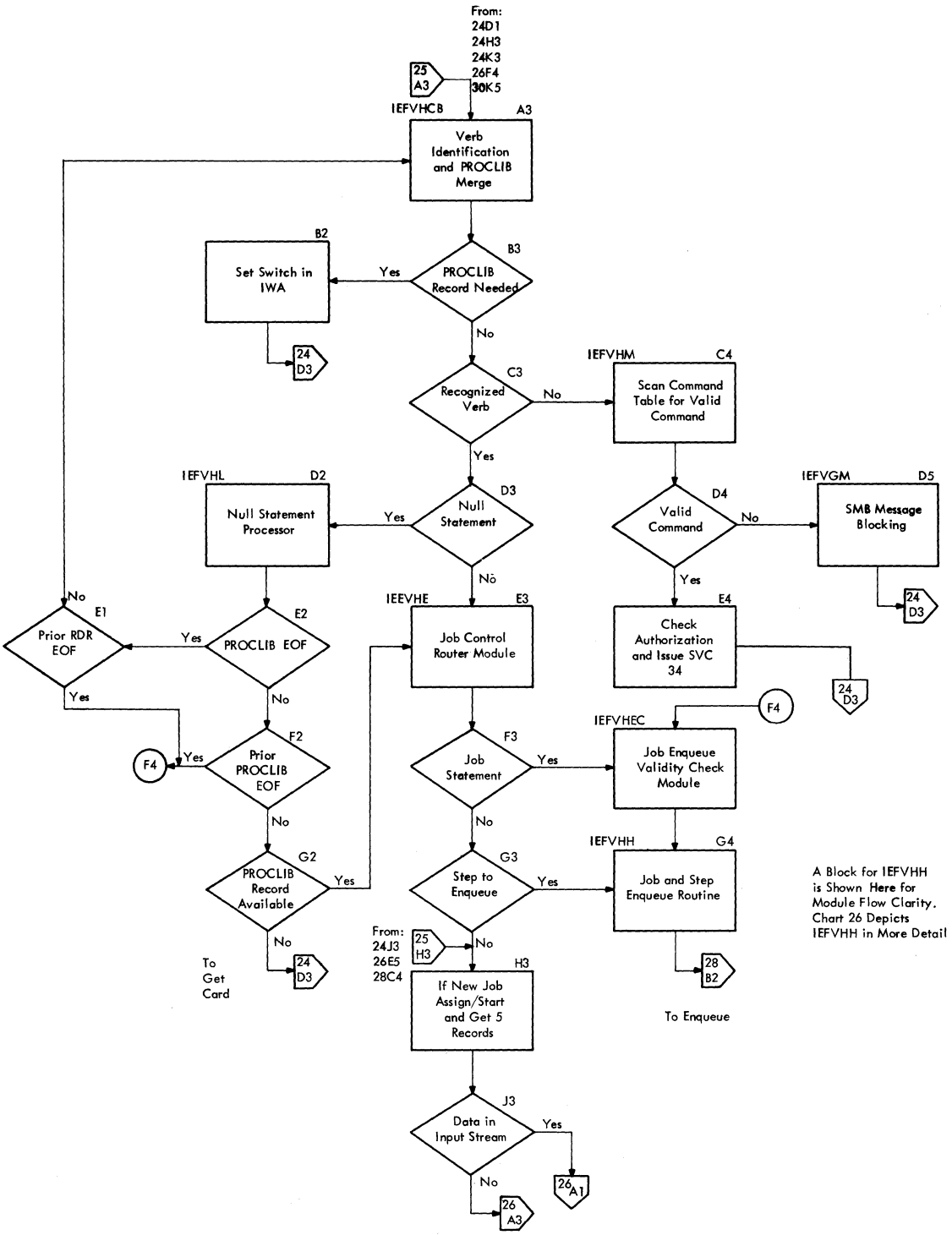


Chart 25. Reader/Interpreter (Part 2 of 3)



## Chart 26. Reader Interpreter (Part 3 of 3)

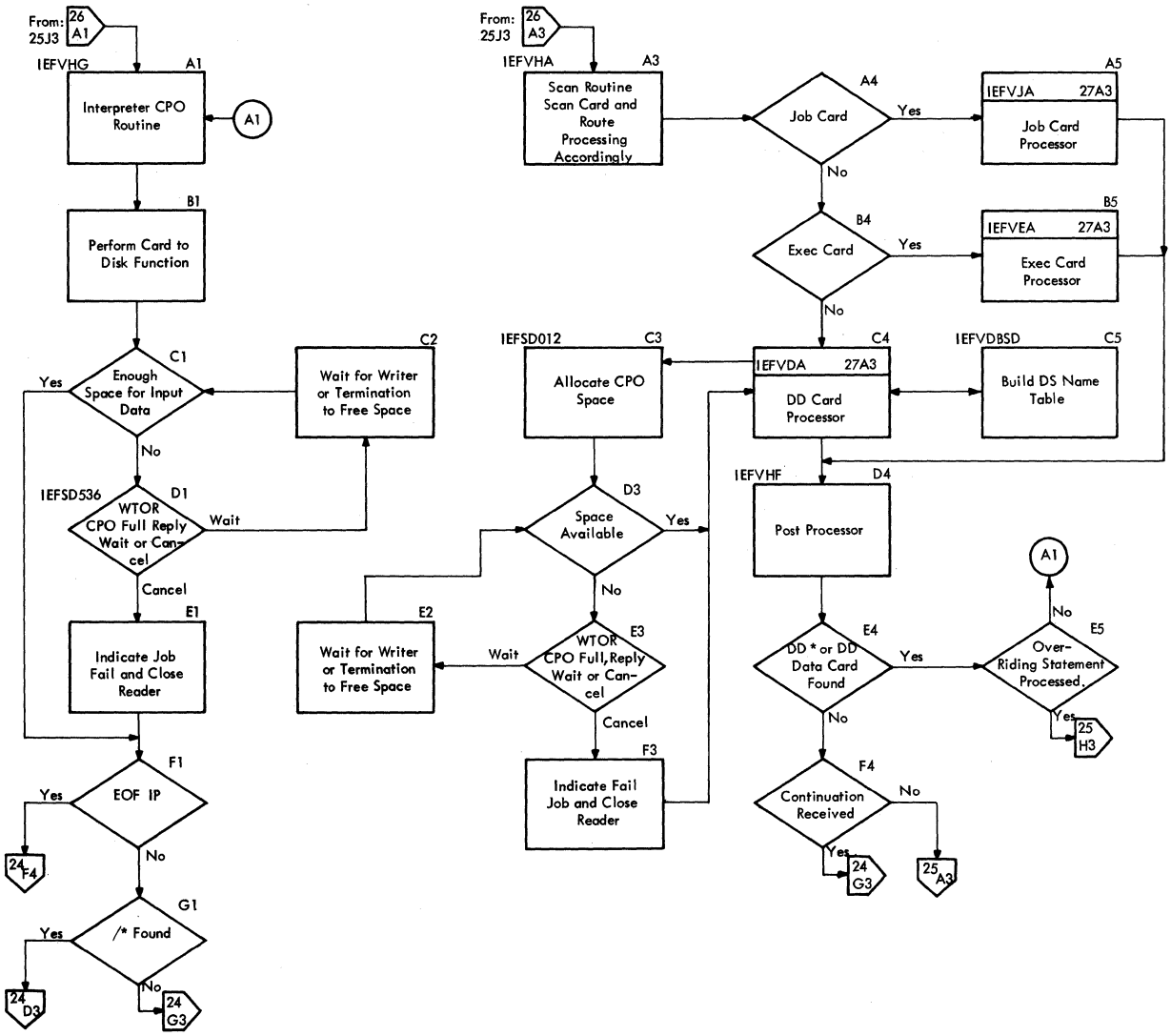


Chart 27. JCL Statement Processor

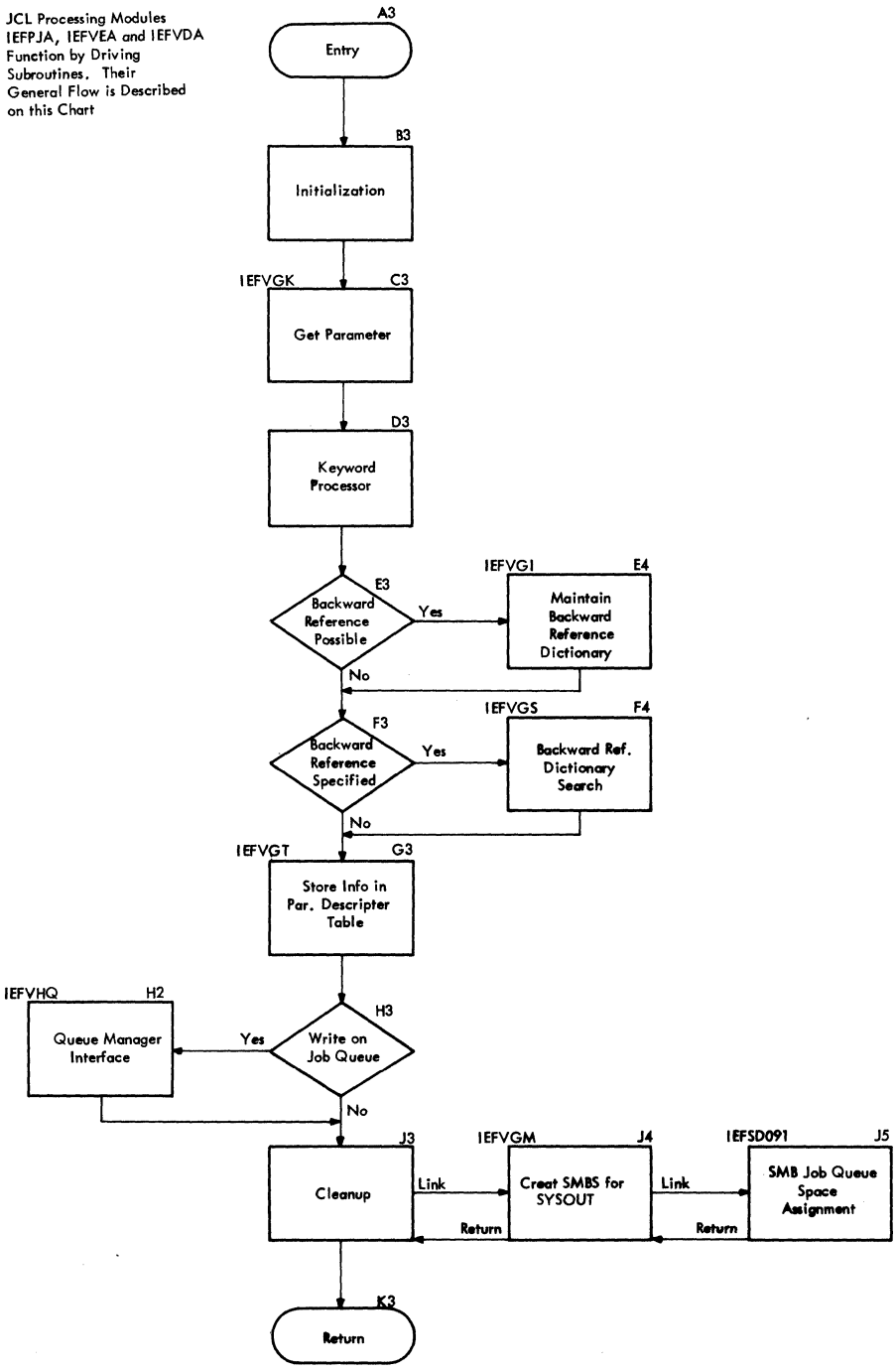
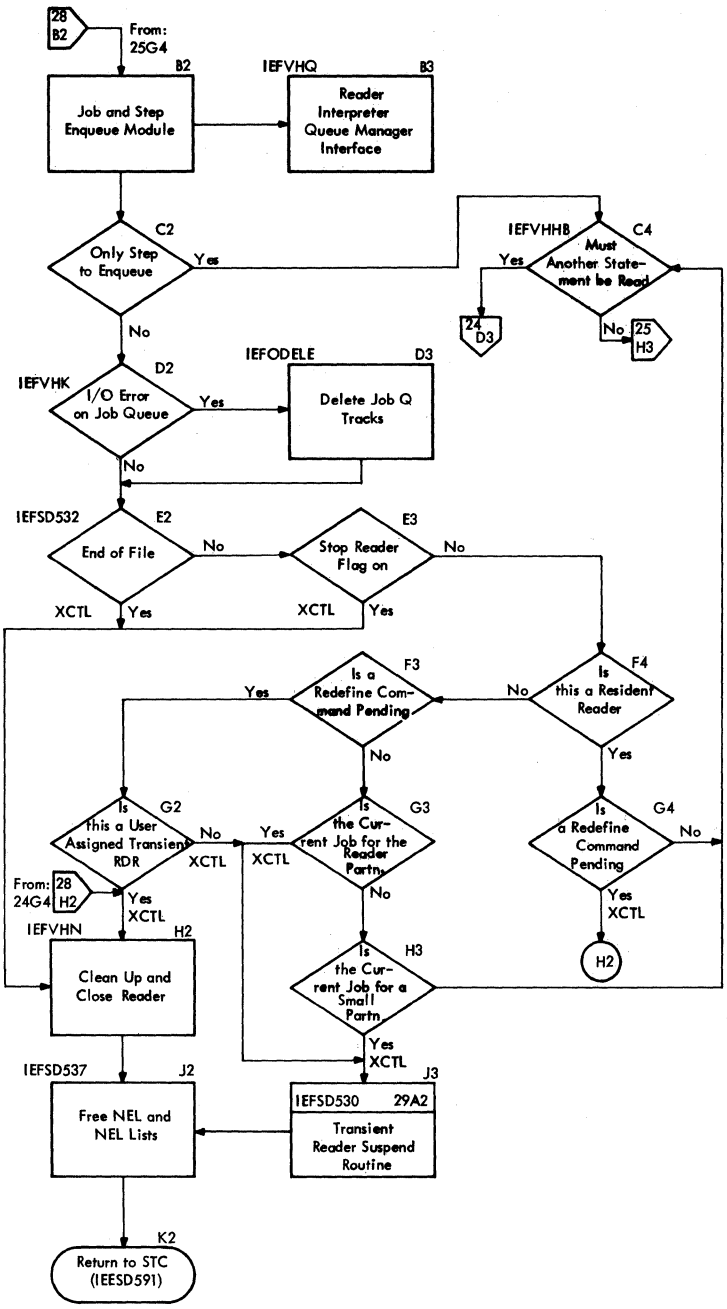


Chart 28. Job and Step Enqueue Routine



## Chart 29. Transient Reader Suspend Routine

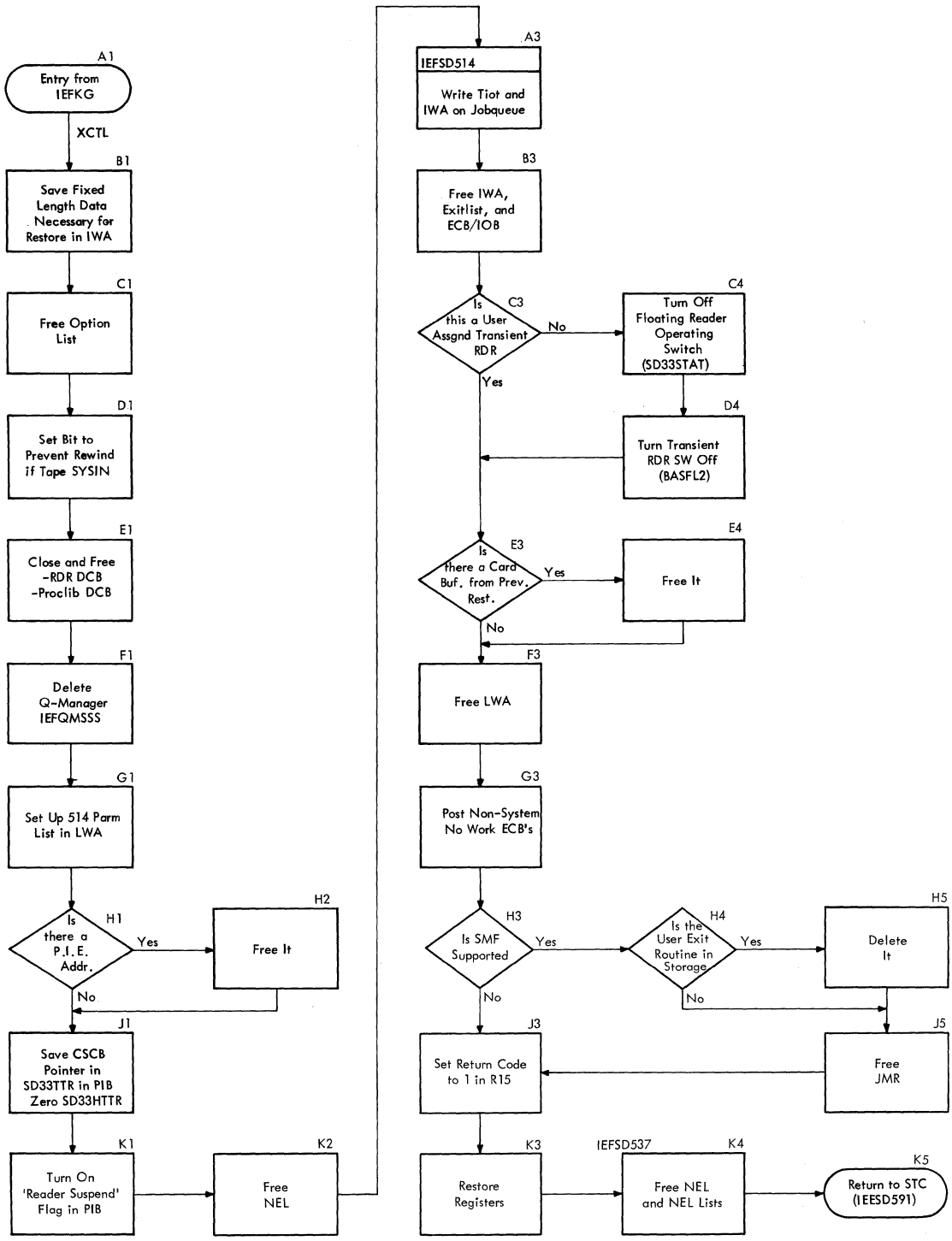


Chart 30. Transient Reader Restore Routine

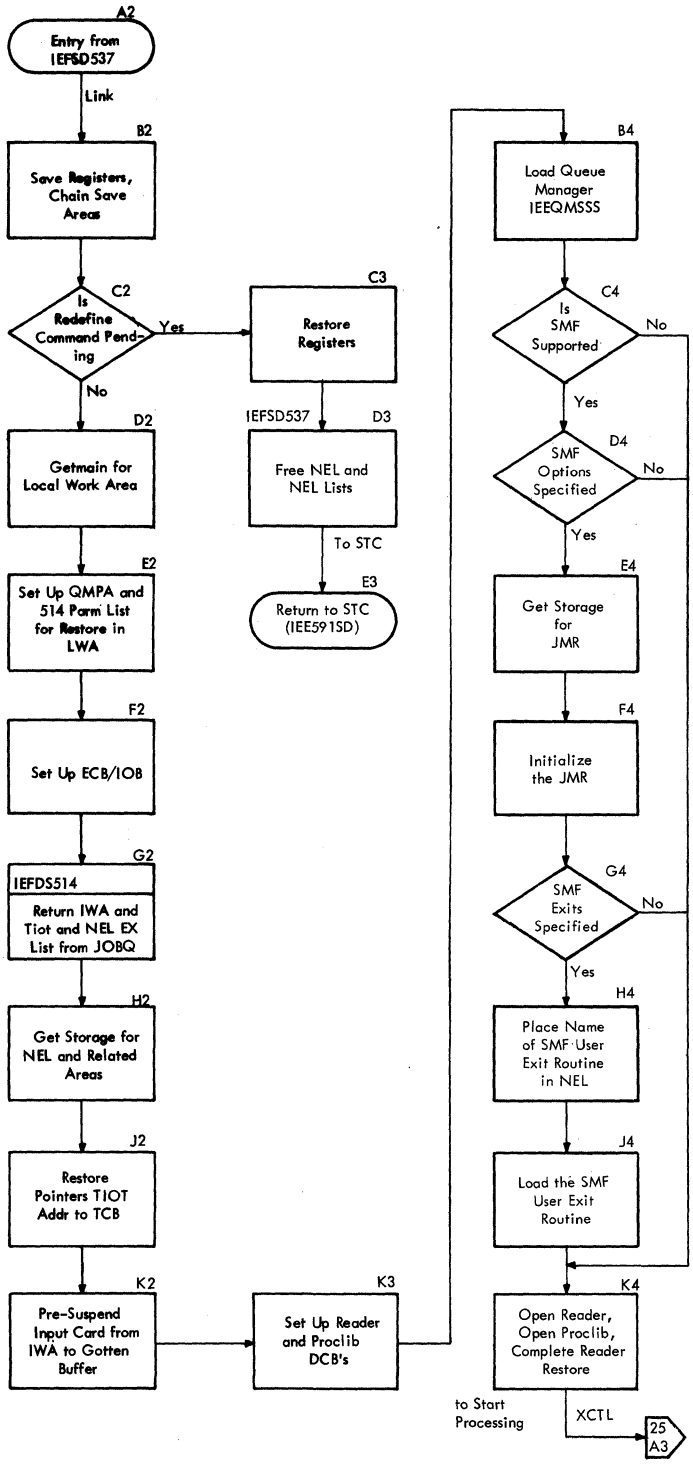
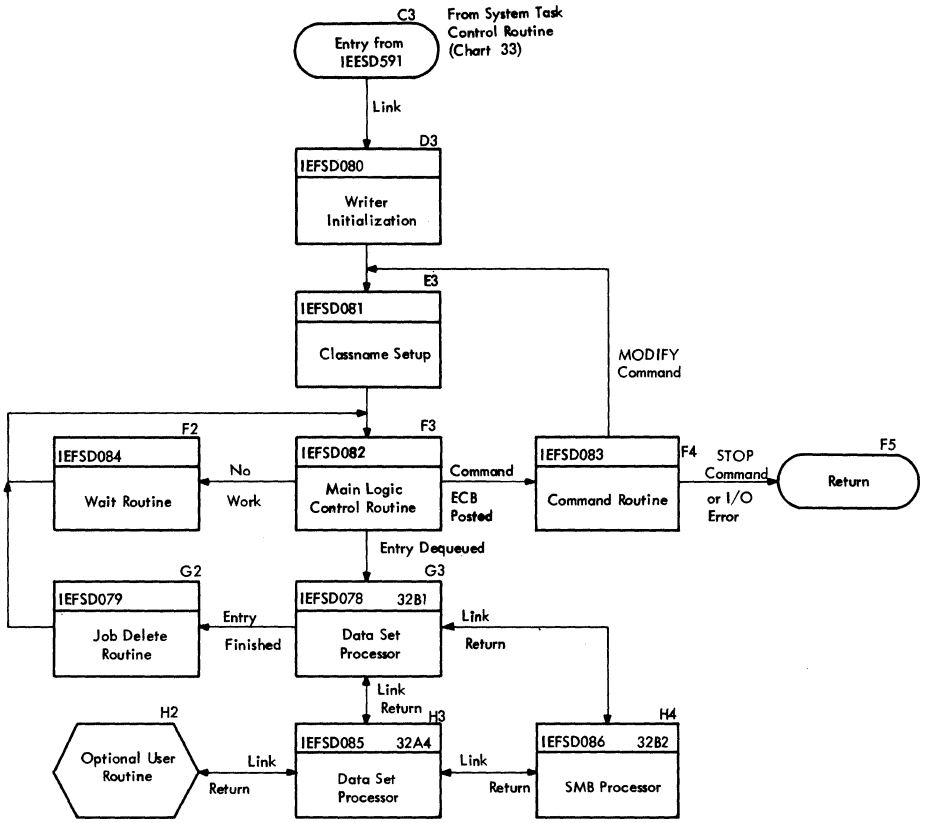


Chart 31. System Output Writer Control Flow



## Chart 32. System Output Writer

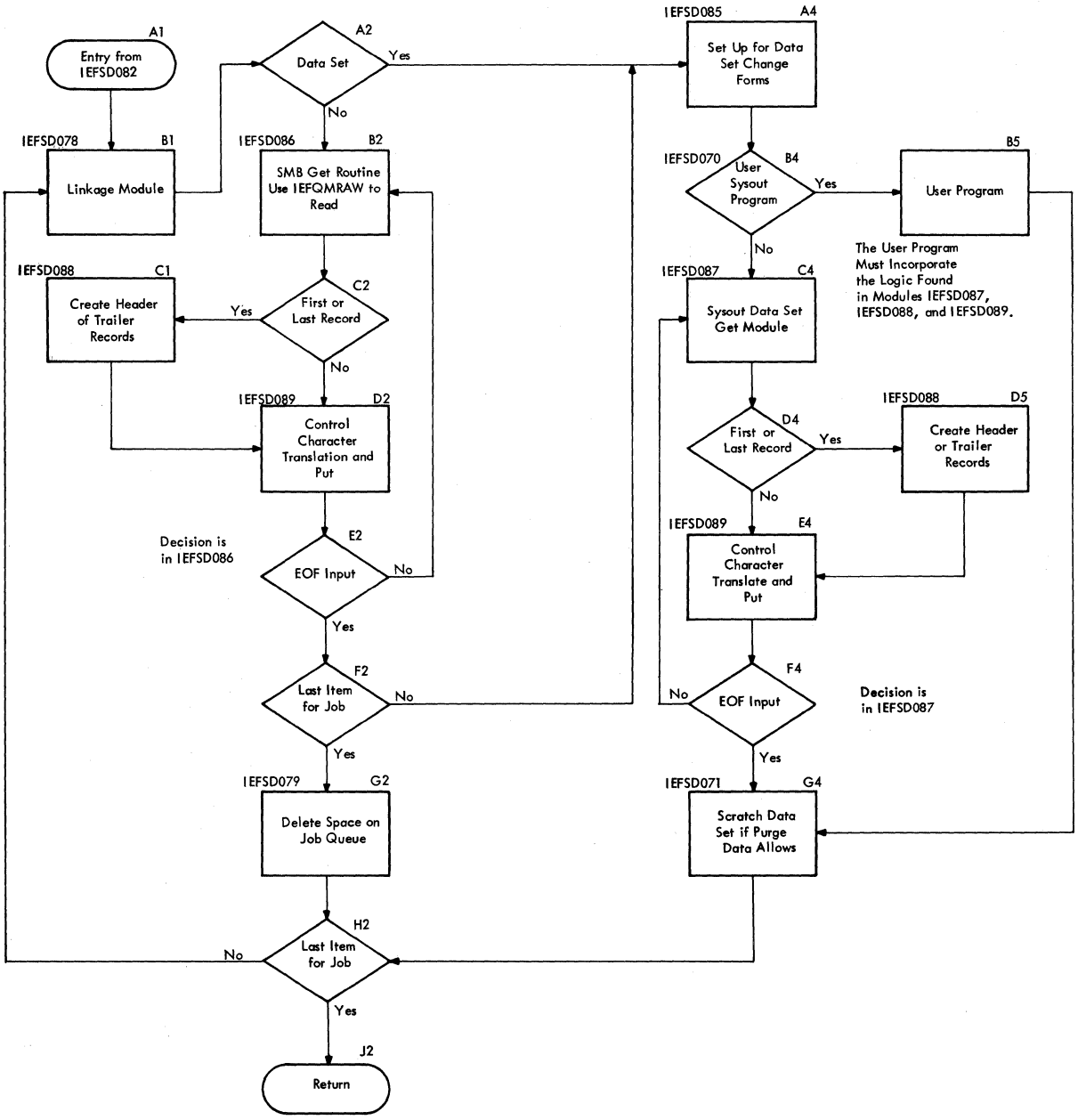
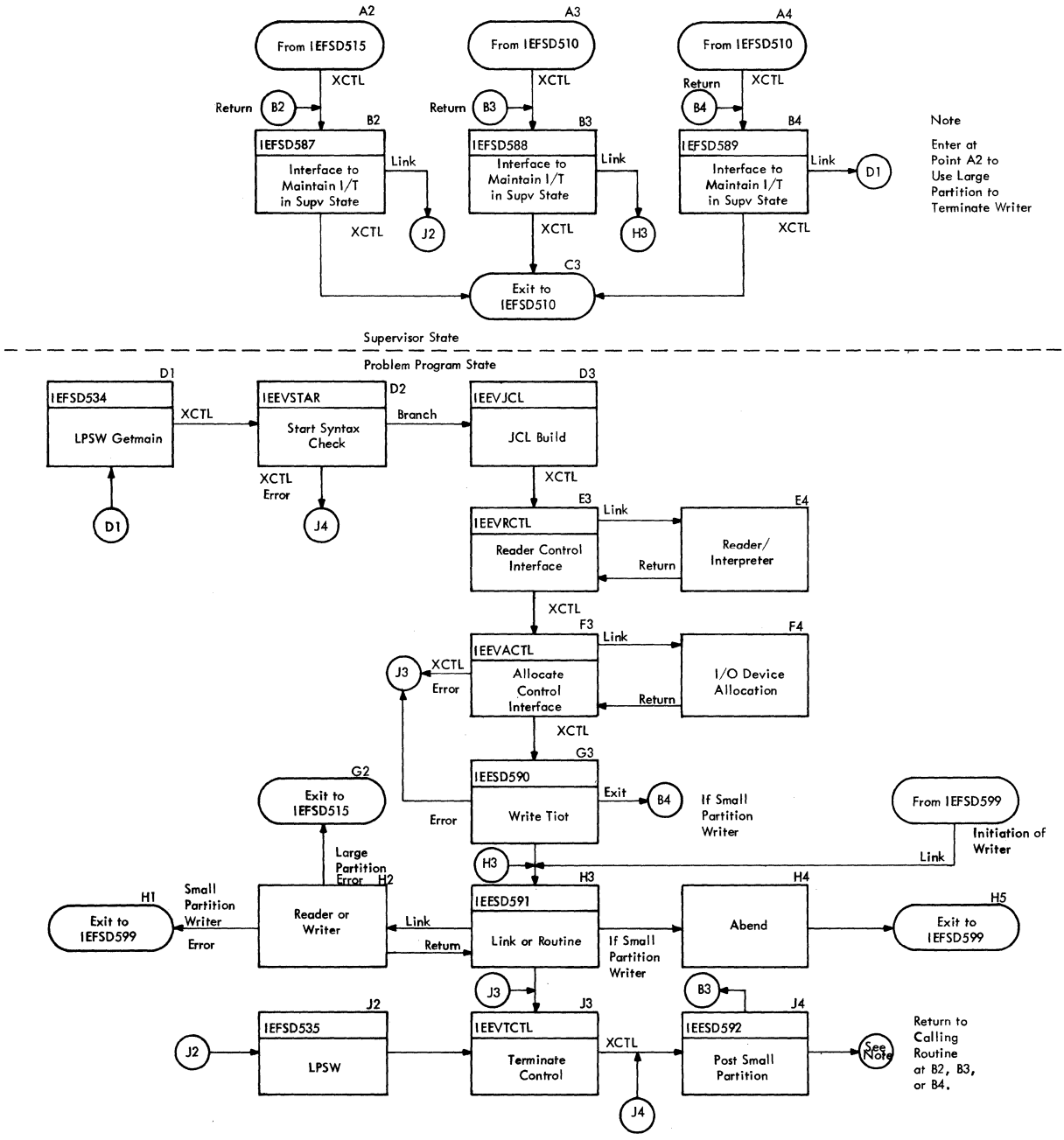
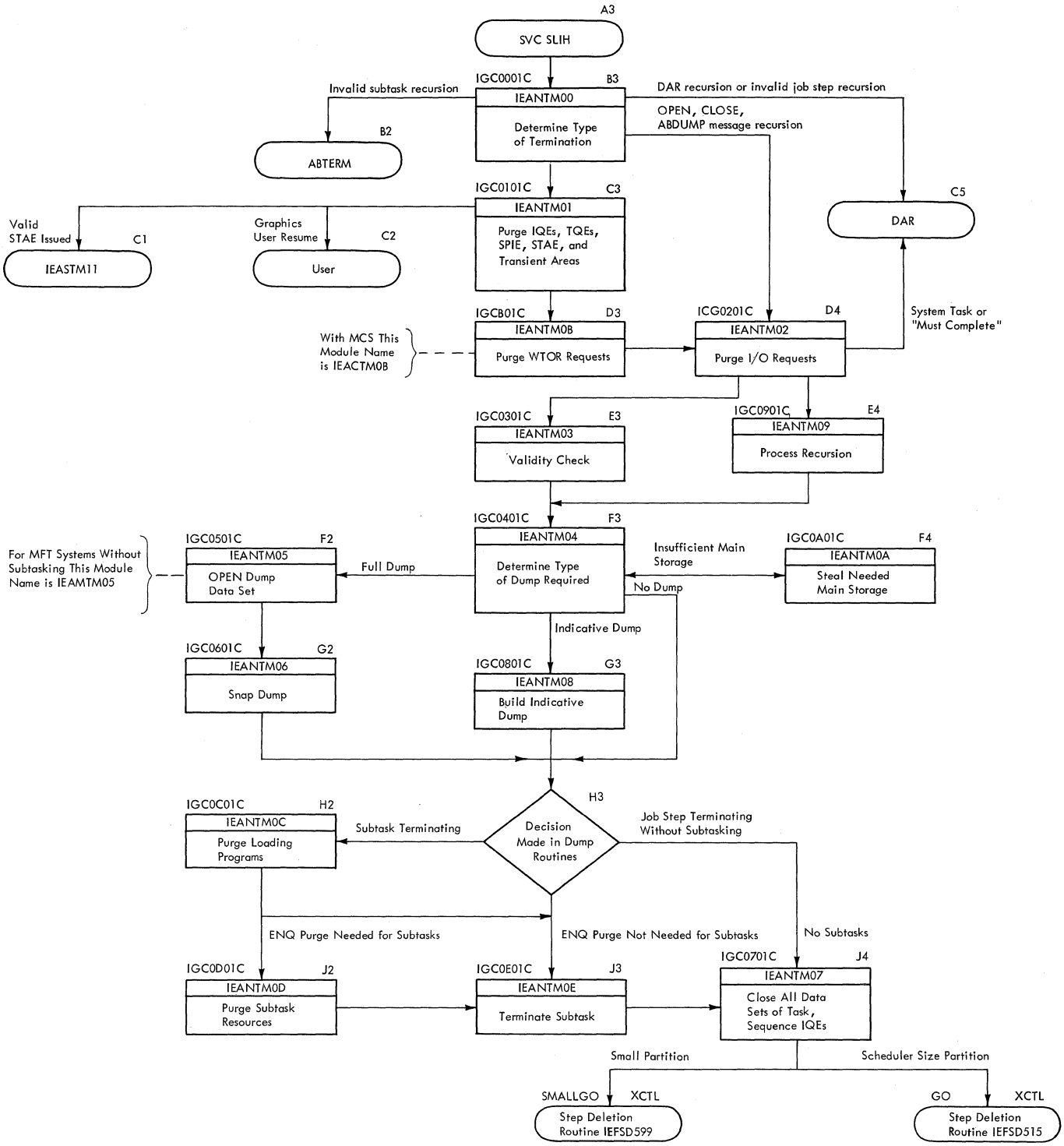


Chart 33. System Task Control



# Chart 34. Abnormal Termination



# Index

Indexes to program logic manuals are consolidated in the publication IBM System/360 Operating System: Program Logic Manual Master Index, GY28-6717. For additional information about any subject listed below, refer to other publications listed for the same subject in the Master Index.

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

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

Order No. GY27-7128-5

- 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 manual is part of a library that serves as a reference source for systems analysts, programmers and operators of IBM systems. Your answers to the questions on the back of this form, together with your comments, 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.

Note: Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.

Cut Along Line

Fold Fold

FIRST CLASS
PERMIT NO. 81
POUGHKEEPSIE, N.Y.

**BUSINESS REPLY MAIL**

NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY ...

IBM Corporation
P.O. Box 390
Poughkeepsie, N.Y. 12602

Attention: Programming Systems Publications
Department D58

Fold Fold

IBM®

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

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

## READER'S COMMENT FORM

IBM System/360 Operating System
Control Program With MFT
Program Logic Manual

Order No. GY27-7128-5

- 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 manual is part of a library that serves as a reference source for systems analysts, programmers and operators of IBM systems. Your answers to the questions on the back of this form, together with your comments, 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.

Note: Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.

Fold Fold

FIRST CLASS
PERMIT NO. 81
POUGHKEEPSIE, N.Y.

**BUSINESS REPLY MAIL**

NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY ...

IBM Corporation
P.O. Box 390
Poughkeepsie, N.Y. 12602

Attention: Programming Systems Publications
Department D58

Fold Fold

Cut Along Line

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]