SY28-0763-0
File No. S370-36

Systems

OS/VS2
System Logic Library
Volume 3

VS2.03.804
VS2.03.807
VS2.03.810

IBM

Pages numbered as duplicates in this publication must be retained because
each of these documents information specific to individual Selectable Units.

This minor revision incorporates the following Selectable Units:

Scheduler Improvements	VS2.03.804
Supervisor Performance #2	VS2.03.807
IBM 3800 Printing Subsystem	VS2.03.810

The selectable unit to which the information applies, is noted in the upper corner of the page.

First Edition (July, 1976)

This is a reprint of SY28-0715-0 incorporating changes released in the following
Selectable Unit Newsletters:

SN28-2683 (dated May 28, 1976)
SN28-2692 (dated May 28, 1976)
SN28-2698 (dated May 28, 1976)

This edition applies to Release 3.7 of OS/VS2 and to all subsequent releases of OS/VS2 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/370 Bibliography,* GC20-0001, 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, Publications Development, Department D58, Building 706-2, PO Box 390, Poughkeepsie, N.Y. 12602. Comments become the property of IBM.

System Logic Library comprises seven volumes. Following is the content and order number for each volume.

OS/VS2 System Logic Library,

Volume 1 contents: SY28-0713
  MVS logic introduction
  Abbreviation list
  Index for all volumes
Volume 2 contents: SY28-0714
  Method of Operation diagrams for
  Communications Task
  Command Processing
  Region Control Task (RCT)
  Started Task Control (STC)
  LOGON Scheduling
Volume 3 contents: SY28-0715
  Method of Operation diagrams for
  System Resources Manager (SRM)
  System Activity Measurement Activity (MF/1)
  JOB Scheduling
    —Subsystem Interface
    —Master Subsystem
    —Initiator/Terminator
    —SWA Create Interface
    —Converter/Interpreter
    —SWA Manager
    —Allocation/Unallocation
    —System Management Facilities (SMF)
    —System Log
    —Checkpoint/Restart
Volume 4 contents: SY28-0716
  Method of Operation diagrams for
  Timer Supervision
  Supervisor Control
  Task Management
  Program Management
  Recovery/Termination Management (R/TM)
Volume 5 contents: SY28-0717
  Method of Operation diagrams for
  Real Storage Management (RSM)
  Virtual Storage Management (VSM)
  Auxiliary Storage Management (ASM)
Volume 6 contents: SY28-0718
  Program Organization
Volume 7 contents: SY28-0719
  Directory
  Data Areas
  Diagnostic Aids

Please note that if you use only one order number, you will only receive that volume. To receive all seven volumes, you must either use all seven form numbers or, simply the following number: SBOF-8210. If you use SBOF-8210, you will receive all seven volumes.

The publication is intended for persons who are debugging or modifying the system. For general information about the use of the MVS system, refer to the publication *Introduction to OS/VS Release 2,* GC28-0661.

How This Publication is Organized

This publication contains six chapters. Following, is a synopsis of the information in each section:

- *Introduction and Master Index* — an overview of each of the functions this publication documents, an abbreviation list of all acronyms used in the publication, and a complete index for all seven volumes.
- *Method of Operation* — a functional approach to each of the subcomponents, using both diagrams and text. Each subcomponent begins with an introduction; all the diagrams and text applying to that subcomponent follow.
- *Program Organization* — a description of module-to-module flow for each subcomponent; a description of each module's function, including entry and exit. The module-to-module flow is ordered by subcomponent. The module descriptions are in alphabetic order without regard to subcomponent.
- *Directory* — a cross-reference from names in the various subcomponents to their place in the source code and in the publication.
- *Data Areas* — a description of the major data areas used by the subcomponents (only those, however, that are not described in *OS/VS Data Areas,* SYB8-0606, which is on microfiche); a data area usage table, showing whether a module reads or updates a data area; a control block overview diagram for each subcomponent, showing the various pointer schemes for the control blocks applicable to each subcomponent; a table detailing data area acronyms, mapping macro instructions, common names, and symbol usage table.

- *Diagnostic Aids* — the messages issued, including the modules that issue, detect, and contain the message; register usage; return codes; wait state codes; and miscellaneous aids.

Corequisite Reading

The following publications are corequisites:

- *OS/VS2 JES2 Logic,* SY28-0622
- *OS/VS Data Areas,* SYB8-0606 (This document is on microfiche.)
- *OS/VS2 System Initialization Logic,* SY28-0623

Contents

Section 2: Method of Operation . . . 3-1
System Resources Manager (SRM) . . . 3-3
  SRM Interface . . . 3-5
  Locking Considerations . . . 3-5
 Method-of-Operation Diagrams . . . 3-6
  6-1. SRM Interface (IRARMINT) . . . 3-6
  6-1. SRM Interface (IRARMINT) **(VS2.03.807)** . . . 3-6
  6-1A. SRM Service Routine (IRARMSRV) **(VS2.03.807)** . . . 3-9.2
  6-1B. Obtain/Free SQA Storage (IRARMI04) **(VS2.03.807)** . . . 3-9.6
  6-1C. Requeue SRM TQE (IRARMI05) **(VS2.03.807)** . . . 3-9.8
  SYSEVENT Processor . . . 3-11
  List of SYSEVENTs **(VS2.03.807)** . . . 3-11
  6-2. SYSEVENT Processor . . . 3-12
  SRM Control . . . 3-23
  6-3. SRM Control (IRARMCTL) . . . 3-24
  6-4. Timer Action Analysis (IRARMCAT) . . . 3-26
  6-5. Deferred Action Processor (IRARMCEN) . . . 3-28
  6-6. Algorithm Processor (IRARMCEL) . . . 3-30
  6-7. Periodic Entry Point Scheduling (IRARMCET) . . . 3-32
  6-8. Full Analysis (IRARMCAS) . . . 3-34
  6-9. Partial Analysis (IRARMCAP) . . . 3-36
  6-9. Swap Analysis (IRARMCAP) . . . 3-36
  6-10. Control Swap-In (IRARMCSI) . . . 3-40
  6-11. Control Swap-Out (IRARMCSO) . . . 3-42
  6-11A. Select User for Swap-In (IRARMCPI) **(VS2.03.807)** . . . 3-43.0
  6-11B. Select User for Swap-Out (IRARMCPO) **(VS2.03.807)** . . . 3-43.2
  6-11C. User Evaluation (IRARMCVL) **(VS2.03.807)** . . . 3-43.4
  Resource Use Algorithms . . . 3-45
   Storage Management . . . 3-45
   I/O Management . . . 3-45
   CPU Management . . . 3-45
   Resource Monitor **(VS2.03.807)** . . . 3-45
  6-12. Storage Management (IRARMSTM) . . . 3-46
  6-12. Main Storage Occupancy Analysis (IRARMMS2) . . . 3-52
  6-14. I/O Management (IRARMIOM) . . . 3-54
  6-15. I/O Load Balancing Swap Analysis (IRARMIL2) . . . 3-56
  6-16. I/O Load Balancing User I/O Monitoring (IRARMIL0) . . . 3-58
  6-17. CPU Management (IRARMCPM) . . . 3-62
  6-18. CPU Load Balancing Swap Analysis (IRARMCL2) . . . 3-66
  6-18. Resource Monitor Periodic Monitoring (IRARMRM1) **(VS2.03.807)** . . 3-66
  6-18A. Resource Monitor MPL Adjustment Processing (IRARMRM2) **(VS2.03.807)** . . . 3-68
  Workload Management . . . 3-69
  Workload Management(IRARMWLM) **(VS2.03.807)** . . . 3-69
  6-19. Swappable User Evaluation (IRARMWM2) **(VS2.03.807)** . . . 3-70
  6-20. Individual User Evaluation (IRARMWM3) **(VS2.03.807)** . . . 3-73.0
  6-21. User Ready Processing (IRARMHIT) **(VS2.03.807)** . . . 3-73.2
  6-22. Initialize for MF/1 (IRARMWR1) **(VS2.03.807)** . . . 3-73.6
  6-23. Collect Data for MF/1 (IRARMWR2) **(VS2.03.807)** . . . 3-73.8
System Activity Measurement Facility (MF/1) . . . 3-75
 Method-of-Operation Diagrams . . . 3-80
  7-1. Measurement Facility Control (MFC) Mainline (IRBMFMFC) . . . 3-80
  7-2. Input Merge Control (IRBMFINP) . . . 3-82
  7-3. Syntax Analyzer (IRBMFANL) . . . 3-84
  7-4. List Option Subroutine (MFLISTOP) . . . 3-86
  7-5. MFSTART Mainline (IGX00013) . . . 3-88
  7-6. Initialization Mainline (MFIMAINL) . . . 3-90
  7-7. CPU Activity Initialization (IRBMFICP) or Paging Activity Initialization (IRBMFIPP) . . . 3-96
  7-8. Workload Initialization (IRBMFIWK) . . . 3-98
  7-9. Channel Initialization (IRBMFIHA) . . . 3-100
  7-10. Device Initialization (IRBMFIDV) . . . 3-104
  7-11. Data Control (IRBMFDTA) . . . 3-106
  7-12. Termination Processor (IRBMFTMA) . . . 3-110
  7-13. MF/1 Message Processor (IRBMFMPR) . . . 3-112
  7-14. MFDATA SVC Mainline (IGX00014) . . . 3-114
  7-15. Interval MG Routine for CPU (IRBMFDCP) . . . 3-118

7-16. Interval MG Routine for Paging (IRBMFDPP) 3-122
7-17. Interval Routine for Workload (IRBMFDWP) 3-126
7-18. Interval MG Routine for Channels (IRBMFDHP) 3-130
7-19. Interval MG Routine for Devices (IRBMFDDP) 3-134
7-20. MFROUTER SVC Processor (IRBMFEVT) 3-138
7-21. Channel Sampling Module (IRBMFECH) 3-140
7-22. Second CPU Test Channel Sampling Module (IRBMFTCH) 3-142
7-23. Device Sampling Module (IRBMFEDV) 3-144
7-24. Report Generator Control (IRBMFRGM) 3-146
7-25. Report Generators for CPU, Paging, Workload, Channels, and Devices (IRBMFRCR, IRBMFRPR, IRBMFRWR, IRBMFRHR, and IRBMFRDR) 3-150
Job Scheduling Overview 3-153
Subsystem Interface 3-159
Method of Operation Diagram 3-164
8-1. Subsystem Interface 3-164
Master Subsystem 3-169
Method-of-Operation Diagrams 3-172
9-1. Common Request Router (IEFJRASP) 3-172
9-2. Subsystem Determination (IEFJSDTN) 3-174
9-3. Subsystem Initiation (IEFJJOBS) 3-176
9-4. Converter/Interpreter Interface (IEFJCNTL) 3-178
9-5. Pseudo Access Method (IEFJACTL) 3-182
9-6. Subsystem Initiation Message Writer (IEFJWTOM) 3-186
9-7. Data Set Name Assignment (IEFDSNA) 3-188
9-8. Subsystem Job Termination (IEFJJTRM) 3-190
Initiator/Terminator 3-193
Method-of-Operation Diagrams 3-196
10-1. Initiator: Job Initiation 3-196
10-2. Initiator: Step Initiation 3-200
10-3. Initiator: Step and Job Deletion 3-208
10-4. Initiator: Recovery Processing 3-212
SWA Create Interface 3-215
Method-of-Operation Diagram 3-216
11-1. SWA Create Interface (IEFIB600) 3-216
Converter/Interpreter 3-223
Method-of-Operation Diagrams 3-223
12-1. Converter: Initialization (IEFVH1) 3-224
12-2. Converter: Identifying Verbs on JCL Statements 3-226
12-3. Converter: Processing Commands in the Input Stream (IEFVHM) . 3-230
12-4. Converter: Processing In-Stream and Cataloged Procedures (IEFVINA) 3-232
12-5. Converter: Processing Symbolic Parameters (IEFVFA, IEFVFB) . . 3-234
12-6. Converter: Converting Statements to Internal Text (IEFVFA) . . . 3-236
12-7. Converter: Entering Defaults into Internal Text (IEFVFA) 3-240
12-8. Converter: Termination (IEFVHF) 3-242
12-9. Interpreter: Initialization (IEFNB903) 3-246
12-10. Interpreter: Analyzing Parameter Values 3-248
12-11. Interpreter: Creating and Chaining Tables (IEFVGT) 3-252
12-11. Interpreter: Writing Tables into SWA (IEFVHH) 3-256
12-13. Interpreter: Termination (IEFVHN) 3-258
SWA Manager 3-261
Method-of-Operation Diagrams 3-264
13-1. SWA Manager: Move Mode (IEFQB550) 3-264
13-2. SWA Manager: Locate Mode (IEFQB555) 3-266
Allocation/Unallocation 3-269
Introduction to Allocation/Unallocation 3-271
Batch Initialization and Control 3-271
Dynamic Initialization and Control 3-271
JFCB Housekeeping 3-271
Common Allocation Control 3-271
Data Set Requests and Unit Requests 3-271
Order of Processing Requests 3-271
Generic Allocation Control 3-272
Recovery Allocation 3-273
The Retry Situation 3-273
Processing Tape Requests 3-273
Common Unallocation Control 3-275
Volume Mount & Verify (VM&V) Control 3-275
Allocation/Unallocation Module Name Conventions 3-275
Organization of Allocation/Unallocation Method-of-Operation Diagrams . 3-275

Selected Terms Used in Allocation/Unallocation 3-276
Method-of-Operation Diagrams 3-280
14-1. Common Allocation Control (IEFAB421) 3-280
14-2. Fixed Device Control (IEFAB430) 3-294
14-3. Specific Volume Allocation Control (IEFAB433) 3-298
14-4. Allocate Request to Unit (IEFAB434) 3-302
14-5. Nonspecific Volume Allocation Control (IEFAB436) 3-308
14-6. JFCB Housekeeping Control (IEFAB451) 3-314
14-7. DD Function Control (IEFAB454) 3-322
14-8. JLOCATE (IEFAB469) 3-334
14-9. Generic Allocation Control (IEFAB471) 3-338
14-10. Allocation Via Algorithm (IEFAB476) 3-348
14-11. Demand Allocation (IEFAB479) 3-354
14-12. Recovery Allocation (IEFAB485) 3-358
14-13. Offline/Allocated Device Allocation (IEFAB486) 3-366
14-14. Common Allocation Cleanup (IEFAB490) 3-378
14-15. Allocation/Volume Mount & Verify (VM&V) Interface (IEFAB492) 3-386
14-16. Volume Mount & Verify (VM&V) Control (IEFAB493) 3-390
14-17. Initiator/Allocation Interface (IEFBB401) 3-396
14-18. Initiator/Unallocation Interface (IEFBB410) 3-402
14-19. Job Unallocation (IEFBB416) 3-410
14-20. SVC 99 Control (IEFDB400) 3-412
14-21. Dynamic Allocation Control (IEFDB410) 3-414
14-22. Dynamic Unallocation Control (IEFDB4A0) 3-416
14-23. Dynamic Concatenation (IEFDB450) 3-418
14-24. Dynamic Deconcatenation (IEFDB460) 3-420
14-25. Dynamic Information Retrieval (IEFDB470) 3-422
14-26. Remove In-Use Attribute (IEFDB480) 3-424
14-27. Ddname Allocation (IEFDB490) 3-428
14-28. Common Unallocation Control (IEFAB4A0) 3-430
14-29. Disposition Processing (IEFAB4A2) 3-440
14-30. Unit Unallocation (IEFAB4A4) 3-444
System Mangement Facilities (SMF) 3-447
Method-of-Operation Diagrams 3-450
15-1. Writing SMF Records (IEEMB829, IEEMB830) 3-450
15-2. Switching SMF Data Sets (IEEMB829) 3-454
15-3. STAE Exit Processing for SMF (IEEMB825) 3-458
15-4. SMF Cross-Memory POST Error Exit (IEEMB827) 3-460
System Log 3-463
Method-of-Operation Diagrams 3-466
16-1. System Log Initialization (IEEMB803) 3-466
16-2. Terminating the System Log (IEEMB803) 3-470
16-3. Switching Log Data Sets (IEEMB803) 3-472
16-4. Log Writer Processing (IEEMB803) 3-474
16-5. Processing Log Task Abnormal Termination (IEEMB806) 3-476
16-6. Writing Data on the System Log (IEEMB804) 3-480
Checkpoint/Restart 3-483
DSDR Processing 3-483
The Job Journal 3-483
Journal Routines 3-483
Method-of-Operation Diagrams 3-486
17-1. Processing Data Set Descriptor Records (IEFXB609) 3-486
17-2. Job Journal to SWA Merging (IEFXB601) 3-492
17-3. Step Continue Processing (IEFXB601) 3-494
17-4. System Restart Processing (IEFXB601) 3-496
17-5. Automatic Checkpoint Restart (IEFXB601) 3-498
17-6. Automatic Step Restart (IEFXB601) 3-500
17-7. Merge Cleanup (IEFXB601) 3-502
17-8. Updating the Virtual Addresses in SWA (IEFXB601) 3-504
17-9. Journal Merge Reading (IEFXB601) 3-506
17-10. Journal Merge Error Processing (IEFXB601) 3-508
17-11. Restart Interface Processing (IEFXB602) 3-510
17-12. Building a Step Header Record for Job Journal (IEFXB604) 3-512
17-13. Preparing Abended Job Step for Restart (IEFRPREP) 3-516
17-14. Writing Blocks to the Job Journal (IEFXB500) 3-520
17-15. Journal for Restarted Jobs (IEFXB500) 3-525

Index **I-1**

Figure 2-9 System Resources Manager (SRM) Visual Contents 3-4
Figure 2-9A SRM Module/Entry Point Cross Reference (**VS2.03.807**) 3-3.2
Figure 2-9B Processing Algorithms and Actions in IRARMCTL (**VS2.03.807**) 3-23.2
Figure 2-9C RMEP Algorithm and Action Invocation Flags (**VS2.03.807**) . . 3-23.3
Figure 2-10 System Activity Measurement Facility (MF/1) Visual Contents . . 3-79
Figure 2-11 Job Scheduling: Initiation of the Master Scheduler 3-164
Figure 2-12 Job Scheduling: Initiation of the Job Entry Subsystem 3-165
Figure 2-13 Job Scheduling: START/LOGON/MOUNT Initiation 3-166
Figure 2-14 Job Scheduling: Normal Job Entry and Initiation 3-167
Figure 2-15 Subsystem Interface Summary 3-171
Figure 2-16 Master Subsystem Visual Contents 3-179
Figure 2-17 Initiator/Terminator Visual Contents 3-203
Figure 2-17A Converter Visual Contents 3-223
Figure 2-17B Interpreter Visual Contents 3-245
Figure 2-18 General Format of a SWA Control Block and an Example of the JFCB as it Appears in SWA 3-262
Figure 2-19 SWA Manager Visual Contents 3-263
Figure 2-20 Relationship of the Six Major Functions of Allocation/Unallocation 3-267
Figure 2-21 Allocation/Unallocation Functions and Related Method-of-Operation Diagrams 3-270
Figure 2-22 The Division of Generic Device Types into Device Groups . . . 3-272
Figure 2-23 Tape Device Types and Supported Densities 3-274
Figure 2-24 Tape Device Eligibility 3-274
Figure 2-25 Batch and Dynamic Allocation/Unallocation Visual Contents . . 3-277
Figure 2-26 Common Allocation Visual Contents 3-279
Figure 2-27 Function Map of Common Allocation Parameter List 3-293
Figure 2-28 Function Map of JFCB Housekeeping Parameter List 3-321
Figure 2-29 System Management Facilities (SMF) Recording: Visual Contents 3-449
Figure 2-30 System Log Visual Contents 3-465
Figure 2-31 Job Scheduler Checkpoint/Restart: Visual Contents 3-485

Section 2: Method of Operation

This section uses diagrams and text to describe the functions performed by the scheduler, supervisor, MF/1, SRM, and ASM functions of the OS/VS2 operating system. The diagrams emphasize functions performed rather than the program logic and organization. Logic and organization is described in "Section 3: Program Organization."

The method-of-operation diagrams are arranged by subcomponent as follows:

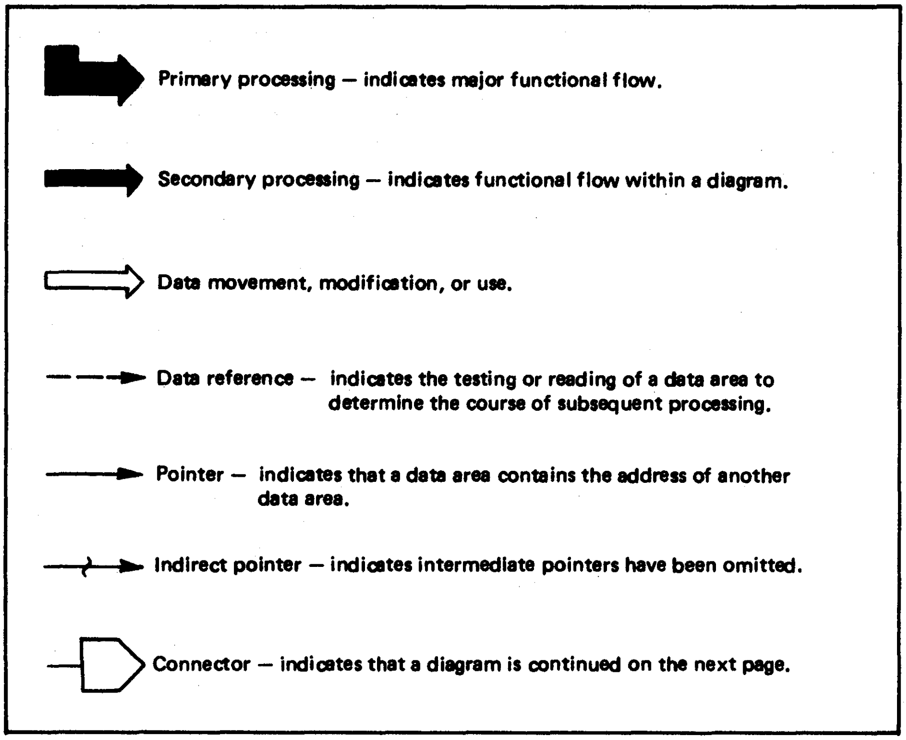
- Communications Task.
- Command Processing (includes Reconfiguration Commands).
- Region Control Task (RCT).
- Started Task Control (STC) (includes START/LOGON/MOUNT).
- LOGON Scheduling
- System Resources Manager
- System Activity Measurement Facility (MF/1)
- Job Scheduling:
  - Subsystem Interface.
  - Master Subsystem.
  - Initiator/Terminator.
  - SWA Create Interface.
  - Converter/Interpreter.
  - SWA Manager.
  - Allocation/Unallocation.
  - System Management Facilities (SMF).
  - System Log.
  - Checkpoint/Restart.
- Timer Supervision.
- Supervisor Control.
- Task Management.
- Program Management.
- Recovery/Termination Management (R/TM).
- Real Storage Management (RSM).
- Virtual Storage Management (VSM).
- Auxiliary Storage Management (ASM).

The diagrams for each subcomponent are preceded by an introduction that summarizes the subcomponent's function. Following each introduction is a visual table of contents that displays the organization and hierarchy of the diagrams for that subcomponent.

The diagrams cross-reference each other using diagram numbers and module names. As an aid in locating the diagrams that are cross-referenced, an alphabetic list of all diagram names and their corresponding page numbers follows this introduction.

Method-of-operation diagrams are arranged in an input-processing-output format: the left side of the diagram contains data that serves as input to the processing steps in the center of the diagram, and the right side contains the data that is output from the processing steps. Each processing step is numbered; the number corresponds to an amplified explanation of the step in the "Extended Description" area. The object module name and labels in the extended description point to the code that performs the function.

Note: The relative size and the order of fields within input and output data areas do not always represent the actual size and format of the data area.



Primary processing — indicates major functional flow.

Secondary processing — indicates functional flow within a diagram.

Data movement, modification, or use.

Data reference — indicates the testing or reading of a data area to determine the course of subsequent processing.

Pointer — indicates that a data area contains the address of another data area.

Indirect pointer — indicates intermediate pointers have been omitted.

Connector — indicates that a diagram is continued on the next page.

Figure 2-1. Key to Symbols Used in Method-of-Operation Diagrams

System Resources Manager (SRM)

In MVS, address spaces may be swapped into or out of real storage. When an address space is swapped out, its entire working storage is moved to auxiliary storage, and the real page frames it formerly occupied may be used for paging activity or to swap in a previously swapped-out address space. The system resources manager (SRM) is the system's swap decision maker. By swapping, the SRM attempts to manage the system to predetermined multiprogramming levels (MPLs) within domains of work as indicated in the IPS.

Domains provides a mechanism of controlling how many of a group of users are swapped in at one time. That is, a domain associates a multi-programming level (MPL with aggregate or group users. The total MPL is the number of swappable memories in real storage at a given time. When the SRM's resource monitor determines that the total MPL may increase, the MPL of one domain will be incremented by one. Similarly, the MPL of one domain is decremented when the total MPL should be lowered. The domain descriptions in the IPS indicate ranges for the MPL of each domain and a weighting factor for each domain which indicates to the SRM which domain to increment or decrement should a change in the system MPL be required.

Also, SRM monitors the system resources of CPU, I/O, and storage. It keeps statistics and uses them to make swap decisions that can prevent either a depletion or an under-utilization of these resources.

Specifically:

- SRM maintains data concerning real and auxiliary storage. It uses the real storage manager (RSM) and the auxiliary storage manager (ASM) to keep track of frame (RSM) and slot (ASM) usage. Using this data SRM is able to detect shortages and use swapping to correct them.
- SRM monitors the I/O resource and makes decisions concerning the allocation of devices based on I/O load balancing considerations.
- SRM monitors and controls CPU utilization through its ability to balance the CPU load through swapping and by its ability to maintain the automatic priority group (APG).

The SRM's structure consists of five functional groupings:

- The interface function is the means through which other system components communicate with the SRM, and through which the SRM requests the services of other system components.
- The SYSEVENT processor analyzes communications to the SRM and translates them into requests for specific SRM services. It also formulates responses as required by the SYSEVENTs.
- The control function performs swapping analyses, obtains swap recommendations from other SRM components, and translates these recommendations into specific swapping decisions. It also requests that previously deferred SRM functions be performed when it is possible to do so.
- The resource use algorithms consist of CPU, I/O, and storage management functions, which monitor the utilization of these resources, and make swapping recommendations that affect their future use. Also, as a result of this monitoring, recommendations to raise or lower individual domain multiprogramming levels (MPLs) are made and adjustments to the MPLs occur within the constraints of the IPS.
- The workload manager function attempts to maintain each address space's usage of system resources (their service) as specified for different user classes in the IPS (Installation Performance Specification). It exercises this control by influencing the Control function's swapping decisions. Additionally, the workload manager interfaces with MF/1 so that reports concerning the rates of system resources usage can be easily obtained.

The primary way in which the installation may affect the functioning of the SRM is by changing the tuning parameters and the IPS parameters. These are explained in more detail in the *OS/VS2 MVS Initialization and Tuning Guide.* The SRM's principal control block is the resources manager control table (RMCT). All SRM routines and subroutines have access to this table and can access most other SRM blocks via pointers in the RMCT or by displacements from the origin of the RMCT. The origin of the RMCT is the entry point of

IRARMCNS, the SRM constants module. This module contains all the control tables, constants and parameters of execution of the SRM as well as pointers to all the key SRM routines.

The SRM maintains a control block (OUCB) associated with each active address space. OUCBs are maintained on one of three queues, depending on the status of the associated address space:

"IN" queue	- consists of address spaces currently in real storage.
"OUT" queue	- consists of address spaces currently swapped out of real storage and awaiting SRM analysis for swap-in.
"WAIT" queue	- consists of address spaces currently swapped out of real storage and in "long wait" status.

SRM is packaged as several modules, but each module does not directly correspond to a unique SRM function. Specifically, each function in SRM is identified by an entry point in one of the modules that comprise the SRM component. Figure 2-9A summarizes all SRM entry points and shows in what module the entry point exists. A description of each entry point is included at the end of the section in which its containing module is diagrammed.

SRM MODULES / SRM ENTRY POINTS	IRARMCPM	IRARMCTL	IRARMERR	IRARMEVT	IRARMINT	IRARMIOM	IRARMRMR	IRARMSRV	IRARMSTM	IRARMWAR	IRARMWLM
CHAP	X										
CPLRVSWF	X										
CPUTLCK	X										
CPUWAIT	X										
IGC095					X						
IRAPRCSR				X							
IRARMAP1	X										
IRARMASM	X										
IRARMCAP		X									
IRARMCED		X									
IRARMCEL		X									
IRARMCEN		X									
IRARMCET		X									
IRARMCL0	X										
IRARMCL1	X										
IRARMCL3	X										
IRARMCPI		X									
IRARMCPO		X									
IRARMCQT		X									
IRARMCRD		X									
IRARMCRL		X									
IRARMCRN		X									
IRARMCRT		X									
IRARMCRY		X									
IRARMCSI		X									
IRARMCSO		X									
IRARMCVL		X									
IRARMDEL				X							
IRARMEQ1	X										
IRARMHIT											X
IRARMI00					X						
IRARMI01					X						
IRARMI02								X			
IRARMI03								X			
IRARMI04								X			
IRARMI05								X			
IRARMI06								X			
IRARMI07								X			
IRARMI09								X			
IRARMI10						X					
IRARMI48						X					
IRARMIL0						X					
IRARMIL1						X					
IRARMIL3						X					
IRARMIL4						X					
IRARMIPS				X							
IRARMMS2									X		
IRARMMS6									X		
IRARMNOP								X			
IRARMPR1									X		

Figure 2-9A. SRM Module/Entry Point Cross Reference (Part 1 of 2)

SRM MODULES SRM ENTRY POINTS	IRARMCPM	IRARMCTL	IRARMERR	IRARMEVT	IRARMINT	IRARMIOM	IRARMRMR	IRARMSRV	IRARMSTM	IRARMWAR	IRARMWLM
IRARMPR5									X		
IRARMRM1							X				
IRARMRM2							X				
IRARMRPS		X									
IRARMRR1			X								
IRARMRR2			X								
IRARMSQA									X		
IRARMTRC					X						
IRARMUXB				X							
IRARMWM1											X
IRARMWM2											X
IRARMWM3											X
IRARMWM4											X
IRARMWM5											X
IRARMWM7											X
IRARMWMI											X
IRARMWMJ											X
IRARMWMK											X
IRARMWMN											X
IRARMWMO											X
IRARMWMQ											X
IRARMWMR											X
IRARMWMY		X									
IRARMWR1										X	
IRARMWR2										X	
IRARMWR3										X	
IRARMWR4										X	
IRARMWR5										X	
IRARMWR6										X	
IRARMWR7										X	
IRARMWR8										X	
IRARMXPS				X							
IRARMXTL		X									
LCHUSE						X					
NEWDP	X										
RMRR1CKQ			X								
RMRR2GST			X								
RMRR2INT			X								
RMRR2PER			X								
RMRR2REQ			X								
RMRR2RTY			X								
RMRR2SPR			X								
RMRR2VFB			X								
RMRR2VLD			X								
STEAL									X		

Figure 2-9A. SRM Module/Entry Point Cross Reference (Part 2 of 2)

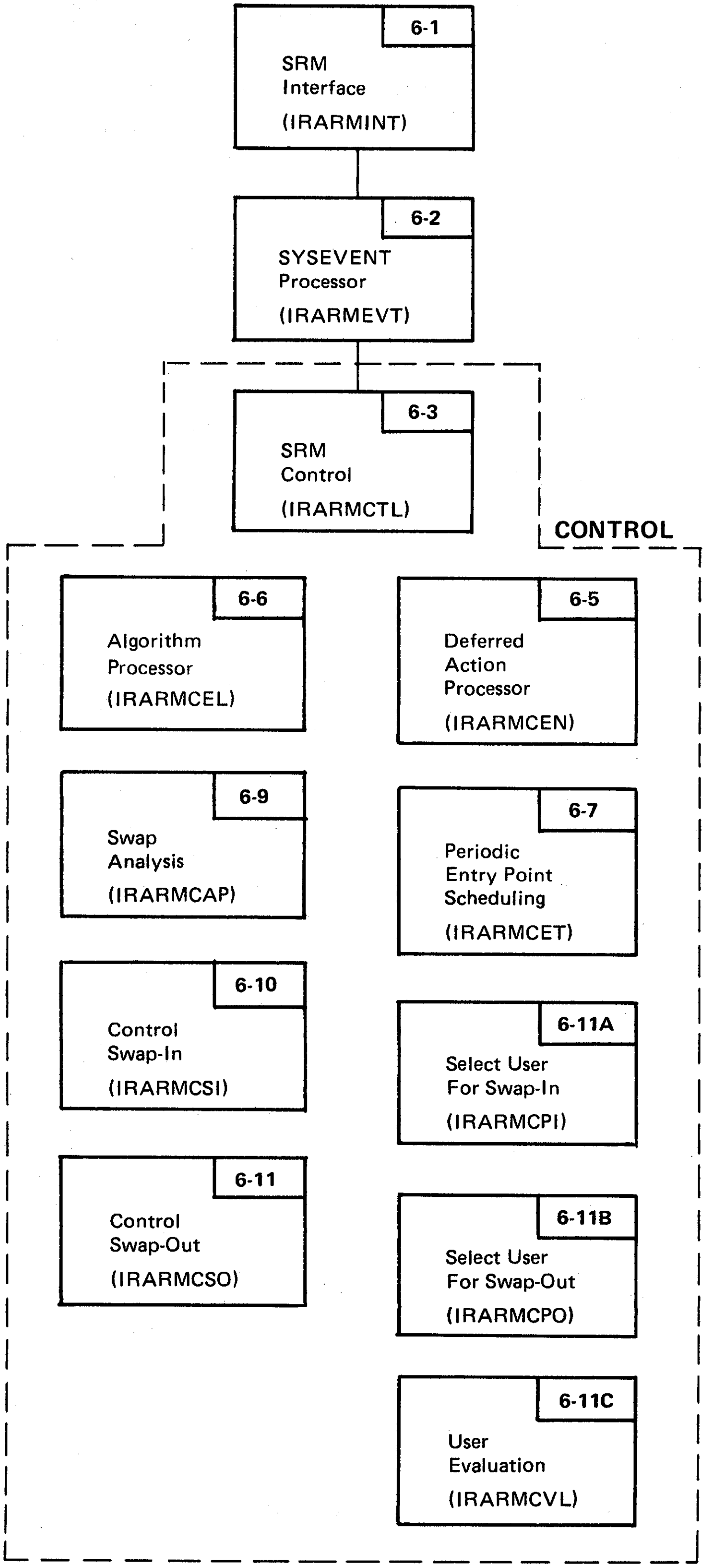


Figure 2-9. System Resources Manager (SRM) Visual Contents (Part 1 of 3)

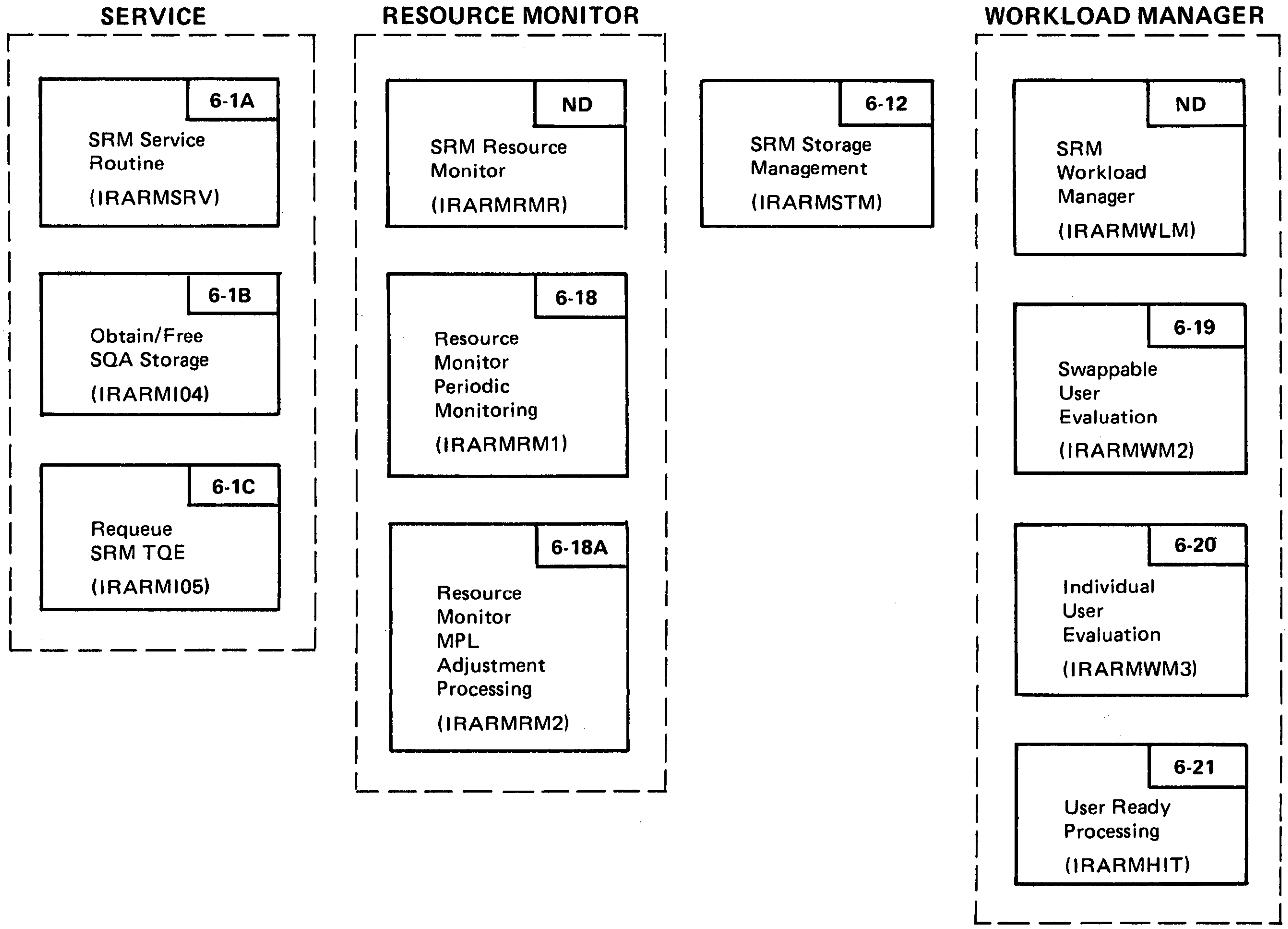


Figure 2-9. System Resources Manager (SRM) Visual Contents (Part 2 of 3)

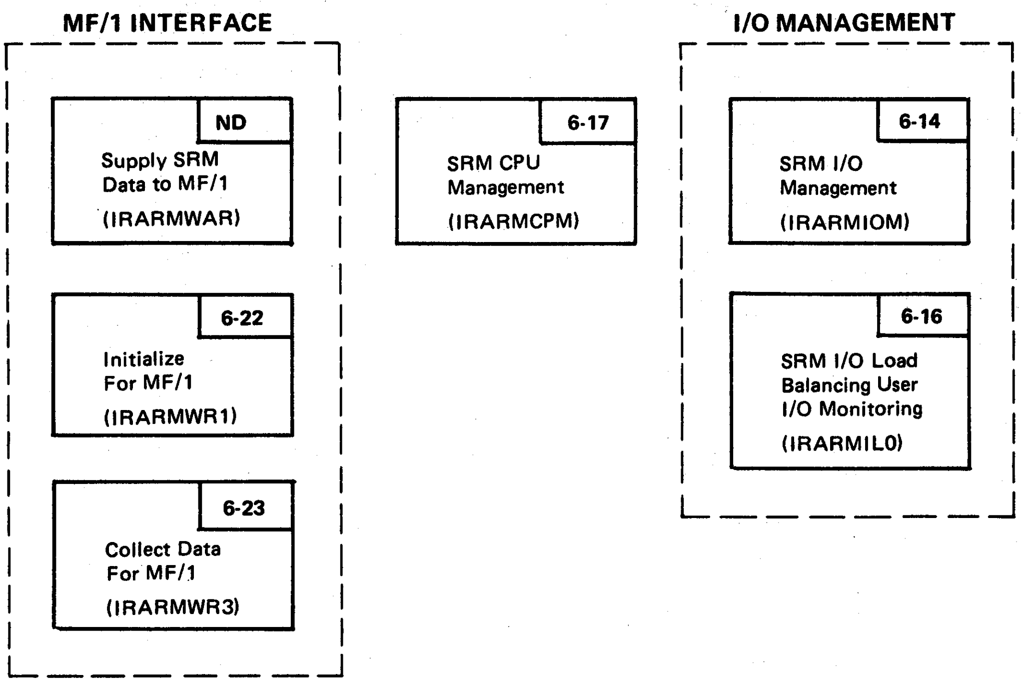


Figure 2-9. System Resources Manager (SRM) Visual Contents (Part 3 of 3)

SRM Interface

Other system components communicate with the SRM by means of the SYSEVENT macro instruction. SYSEVENTs fall into three classes:

- Address space SYSEVENTs are issued to notify the SRM of a change in status for a particular address space.
- System status SYSEVENTs are issued to notify the SRM of a change in status applicable to the system as a whole.
- SRM services SYSEVENTs are issued to request particular SRM support functions.

The SRM interface receives control as a result of the execution of a SYSEVENT macro instruction. The SYSEVENT macro serves as an extended routing function based on the SYSEVENT code generated by the SYSEVENT macro from the specified mnemonic name operand. Each individual SYSEVENT code represents a logically distinct interface to the system resources manager, with its own circumstances, its own input and output conventions, and its own resultant system resources manager actions. The use of the SYSEVENT macro is restricted to those components/modules which have reached prior agreement with system resources manager module owners. The SYSEVENT macro instruction generates either a branch or SVC entry (SVC 95) into the SRM. Branch entry callers must be in supervisor state, key 0-7, and associated data areas must be fixed. Disabled page faults that occur when user data areas are referenced will cause the SYSEVENT issuer to be abnormally terminated (ABEND code '15F'). Branch entry callers must also pass, in register 13, the address of a 72-byte save area, which can be stored into by using the caller's key. The SYSEVENT issuer is responsible for serializing the use of this save area (via disablement, global or local lock).

SYSEVENT 38 requires no authorization. SYSEVENTs 41 and 42 either require APF authorization or must be issued from a program that the initiator recognizes as "DONTSWAP" authorized (ASCBNSWP='1' at initiator attach time). All other SYSEVENT issuers using the SVC entry facility must be APF authorized, and associated data areas must be fixed. Unauthorized use of the SVC entry, or page faults occurring while referencing user data areas, will cause the SYSEVENT issuer to be abnormally terminated (ABEND code '15F').

The SRM interface passes control to the SYSEVENT processor for processing related to the particular SYSEVENT; depending upon the SYSEVENT, the SRM may then perform further processing not necessarily related to the invoking SYSEVENT. Thus many SYSEVENTs serve not only as status change notifiers or service requestors, but also as occasions for performing a wide-range of SRM processing.

The SRM interface also processes requests from internal SRM routines servicing system components. These include such services as cross memory post, obtaining SQA storage and issuing a Write-to-Operator (WTO) message. The interface function is used to provide a common point of invocation and simplified access for internal SRM routines. The service interface routines are packaged together in the IRARMSRV module, each routine having its own entry and exit point. See the M.O. Diagrams for more detail.

SRM Error Recovery

One functional recovery routine (FRR) provides recovery for all of the SRM routines. The address of this routine is identified to the recovery termination manager (RTM) at the beginning of SRM processing, when obtaining the SRM lock for non-globally locked entries and upon entry for globally locked entries. The FRR (address) is cancelled upon exit from SRM processing. The only section of the SRM component not covered by the functional recovery routine is the (non-globally locked) code preceding the obtaining of the SRM lock; such code is restricted from performing any updating of system data.

The functional recovery routine recovers SRM from a percolated error, from machine checks, from the restart key, and from program checks. The routine requests that error recording/storage dumping be performed, supplying additional information about the error.

The processing performed by SRM's FRR depends upon the nature of the error. The actions taken for different errors are described below.

1. If the ABEND macro was issued by SRM, or if the restart key was depressed recursively, the error is percolated.
2. If the error occurred in the SRM workload activity recording routine, the MF1 task is abended. If SRM was running in the same address space as the MF1 task, the error is percolated.
3. If a translation or protection exception occurred in SYSEVENT processing, the abend code is changed to X'15F'. The FRR

validates queues and status data maintained by SRM and percolates the error.

4. For other errors occurring within SRM, the FRR validates queues and status data maintained by SRM and performs a retry of the SRM routine that failed. If the error is repeated, and if the error is associated with an action or algorithm, another retry is attempted bypassing the routine in error. Otherwise, the error is percolated.

The SRM interface also processes requests from internal SRM routines, servicing other system components. The SRM interface M.O. diagram illustrates the functioning of this subcomponent.

The issuing of most SYSEVENTs prior to SRM NIP processing (performed by IEAVNP10) will result in a direct return to the issuer without any SRM processing. An exception is SYSEVENT RSMCNSTS (22), for which normal processing will be performed.

Locking Considerations

All issuers of enabled, branch entry SYSEVENTs must hold the local lock when the SYSEVENT is issued.

The SRM lock will be obtained by the SRM on all SYSEVENT entries to the SRM except the following SYSEVENTs:

USERRDY (4)
SWOUTCMP (15)
SWPINST (16)
RSMCNSTS (22)
AVQLOW (23)
AVQOK (24)
SQALOW (25)
SQAOK (26)
SYQSCST (35)
SYQSCCMP (36)

It is required that issuers of the above SYSEVENTs be disabled on issuing the SYSEVENT, because the SRM uses CPU-related save areas while processing these SYSEVENTs. Issuers of other SYSEVENTs (those not listed above) must not hold any global locks higher in the system locking hierarchy than the SRM lock when they issue the SYSEVENT. These issuers must not hold the SRM lock. SRM must be able to obtain the SRM lock when entered via any of these SYSEVENTs.

The method-of-operation diagrams that follow describe the specific functions performed by the SRM interface. The functions are:

- SRM Interface (IRARMINT).
- SRM Service Routine (IRARMSRV).

Diagram 6-1. SRM Interface (IRARMINT) (Part 1 of 4)

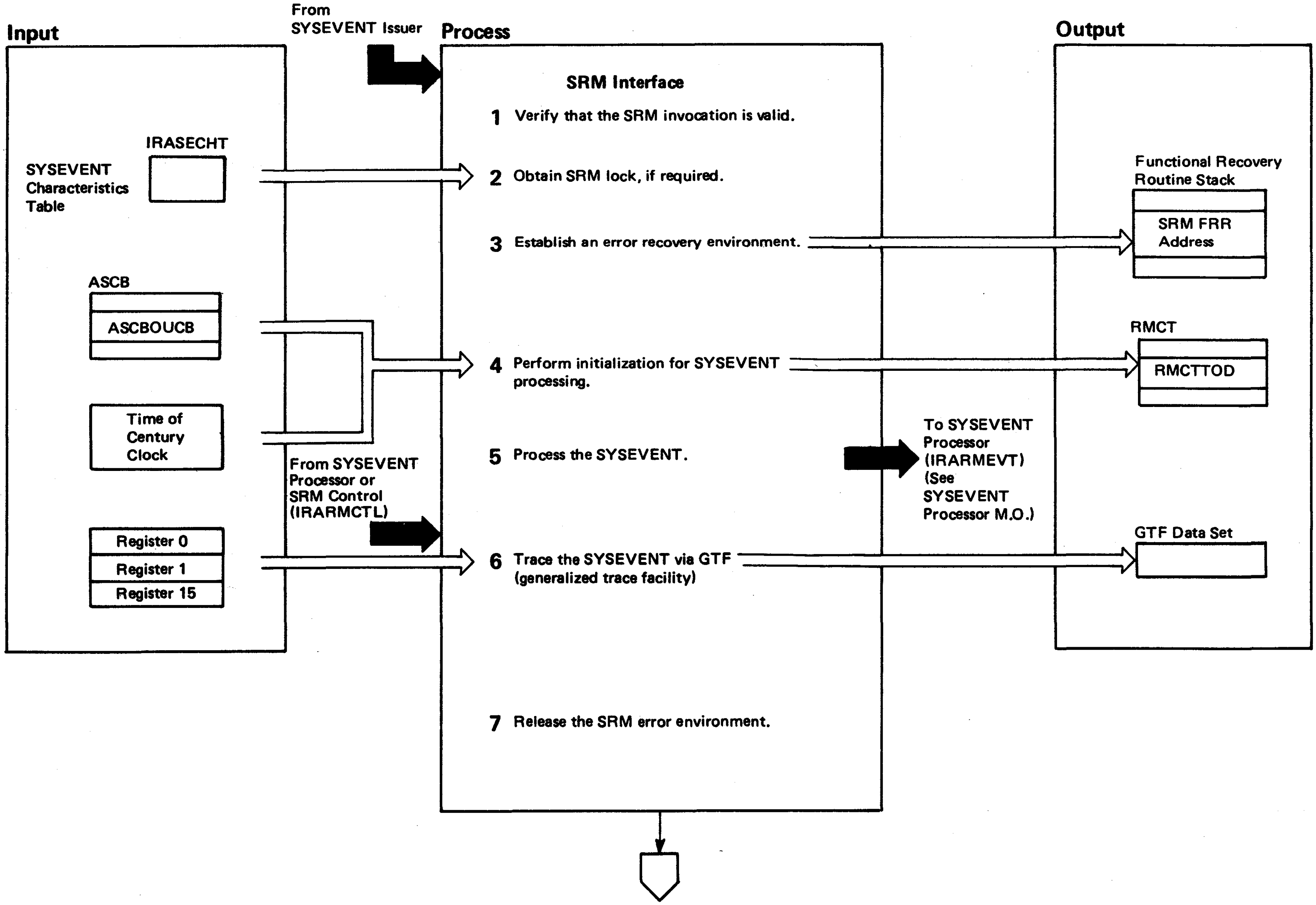
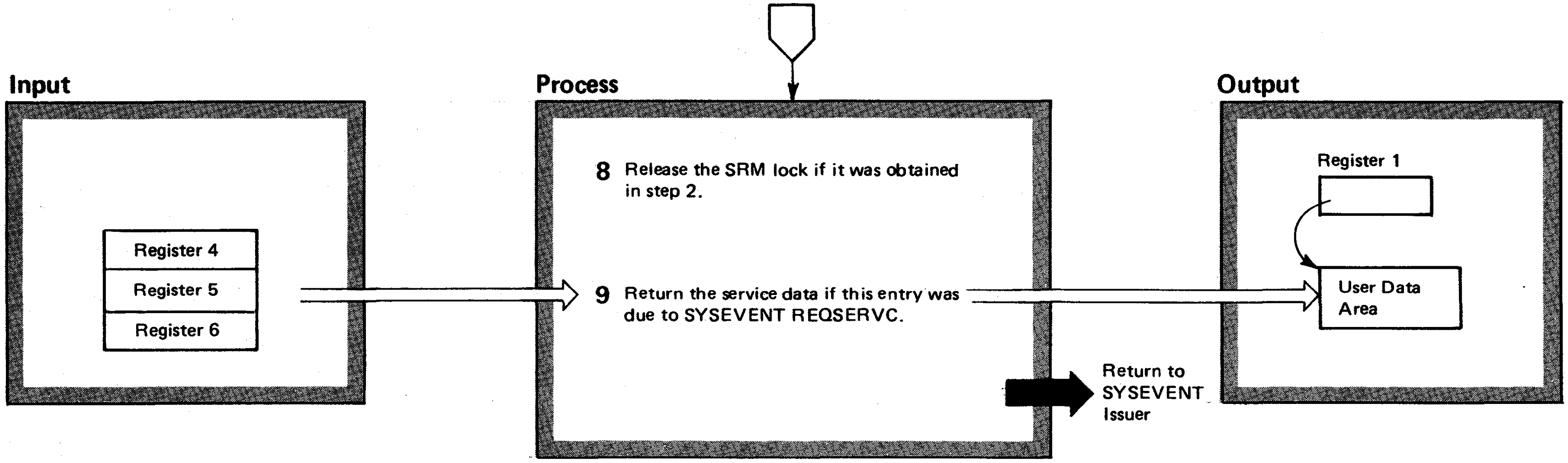


Diagram 6-1. SRM Interface (IRARMINT) (Part 2 of 4)

Extended Description	Module	Label
The SRM Interface receives control when a SYSEVENT macro instruction is issued, or when the SRM requests the services of another system component. When the interface receives a SYSEVENT, it performs the locking necessary to ensure that SRM functions which must be serialized are not performed simultaneously on more than one CPU. SRM requests the SRM lock unconditionally before passing control to the SYSEVENT processor. If the lock is held by another CPU, the lock manager will spin waiting for the lock to be released. Otherwise, SRM will acquire the lock and continue processing. In either case the SRM lock serializes SRM processing in a multi-CPU environment.	IRARMINT IRARMSRV	
1 For all SYSEVENTs that generate supervisor call entries to the SRM (SVC 95), except for SYSEVENT REQSERVC (38), the issuer must be authorized. For SYSEVENTS 41 and 42, "DON'T SWAP" authorization is valid. For all other SYSEVENTs, the user is considered authorized only if he is executing in supervisor state or protection keys 0-7, or is authorized by APF (authorized program facility).	IRARMINT	IGC095
For SYSEVENTs that generate a branch entry to the SRM, the issuer must be executing in protection key 0-7, and must be in supervisor state.		IRARMI00
2 The SYSEVENT characteristics table indicates, for each SYSEVENT entry, whether or not the SRM lock must be obtained for SRM serialization purposes.	IRARMINT	IRARM000
3 The SRM is protected from unexpected errors via a functional recovery routine (FRR). The processing will be performed for an error situation depends upon whether or not the SRM lock was held (see ERROR PROCESSING, below).	IRARMINT	RMINT005

Extended Description	Module	Label
4 Before passing control to the SYSEVENT processor, a pointer is obtained to the SRM user control block (OUCB) corresponding to the input ASID (address space identifier); for SYSEVENT MEMCREAT, there will not yet be an OUCB (an OUCB is obtained by IRARMEVT if no Resource shortages exist). The current time-of-day is obtained and formatted for SRM use. The time-of-day clock value is stored and shifted 22 bits to the right, and the rightmost 32 bits of the resulting value are used by the SRM. Therefore, SRM constants representing time are in units of 1024 microseconds (approximately 1 millisecond).	IRARMINT	IRARM001
5 The interface invokes the SYSEVENT processor to initiate the appropriate processing (see SYSEVENT PROCESSOR table).	IRARMEVT	
6 A GTF trace record is produced (via the HOOK macro) if GTF is active. This record includes:		
• Register 0 (as input, except that the ASID is placed here even when it was not included as input). • Register 1 (as input, with the addition of possible return indicators which may overlay input data). • Register 15 (containing any necessary return code in byte 3).	IRARMINT	
7 The address of the SRM FRR is removed from the system FRR stack.	IRARMINT	IRARMI01

Diagram 6-1. SRM Interface (IRARMINT) (Part 3 of 4)



8 Release the SRM lock if it was obtained in step 2.

9 Return the service data if this entry was due to SYSEVENT REQSERVC.

Diagram 6-1. SRM Interface (IRARMINT) (Part 4 of 4)

Extended Description	Module	Label
8 The SRM lock will have been obtained if the invoking routine did not already hold a lock higher in the locking hierarchy than the SRM lock (except for SYSEVENTS SYQSCST and SYQSCCMP).		RMINT010
9 To prevent disabled page faults and an invalid SRM invocation, and to insure system integrity, the service data is stored while not holding the SRM lock, and in the user's protection key.		
Error Processing		
The issuer of a SYSEVENT will be abnormally terminated (ABEND code '15F'X) if:	IRARMINT	
• an invalid ASID or SYSEVENT code was supplied (reason code 4).		
• the program was not authorized to issue the SYSEVENT (reason code 8).		
• a page fault occurred in referencing a data area assumed to be fixed (reason code 12).		
• the program did not have the correct storage key for storing into a parameter data area (reason code 16).		
• the SRM lock was held on entry to the SRM (reason code 20).		
A SYSEVENT issuer will be terminated (ABEND code '25F') if the SRM determines that a system failure has resulted in the loss of data used by the SRM in controlling an address space. Similarly, the System Activity Measurement Facility (MF/1) task, and the Set IPS task will be terminated (ABEND code '25F') when the SRM receives an error occurring during SRM processing relating to a Set to New IPS command or to the collection of workload activity data for MF/1.		

Extended Description	Module	Label
A functional recovery routine (FRR) provides the error recovery environment for SRM processing. When an error occurs during SRM processing (or when an error occurring in a routine invoked by the SRM has been passed back (percolated) to the SRM), the recovery/termination manager gives control to the SRM FRR. If the SRM was operating without holding the SRM lock when the error occurred, error processing will consist of making one attempt at retrying the failing routine; a second failure will result in the error being passed to the previous routine in the FRR stack. If the SRM was operating under the SRM lock when the error occurred, the FRR will perform queue validation before making an attempt at retrying the failing routine; queue validation consists of verifying that the three OUCB queues are properly chained (re-chaining where necessary), and that OUCBs, OUXBs (user control block extensions), and OUSBs (user swappable blocks) exist and are valid, where they are required. Likewise, the pointers between the ASCBs and OUCBs is checked. Where it is necessary to create a new OUCB or OUXB, a bit is set in the OUCB to indicate that the data reflected in these newly created blocks may not be valid.	IRARMERR	IRARMRR1 IRARMRR2 RMRR2VLD
On errors occurring during SRM locked processing, retry action depends upon whether the error occurred during SYSEVENT related or non-SYSEVENT related processing. For SYSEVENT-related processing, 1 retry will be attempted. Subsequent failure will result in the error being passed to the previous routine in the FRR stack. For non-SYSEVENT-related processing (i.e., processing which SRM control was driving), 1 retry of the failing routine will be attempted. A second error will case an attempt to bypass the twice failing routine. Subsequent errors will result in the error being passed to the previous routine in the FRR stack.		

IRARMINT Module Entry Point Summary

IGC095 - SVC entry point to SRM.
IRARMI00 - Branch entry point to SRM.
Handle all external SYSEVENTs.
IRARMI48 - Branch entry point to SRM.
Handle the internal SYSEVENT. (48).
IRARMI01 - Entry point from RARMEVT or RARMCTL.
Return to the SYSEVENT issuer.
IRARMI10 - Entry point to SRM.
Abend a user of SRM.

Diagram 6-1A. SRM Service Routine (IRARMSRV) (Part 1 of 6)

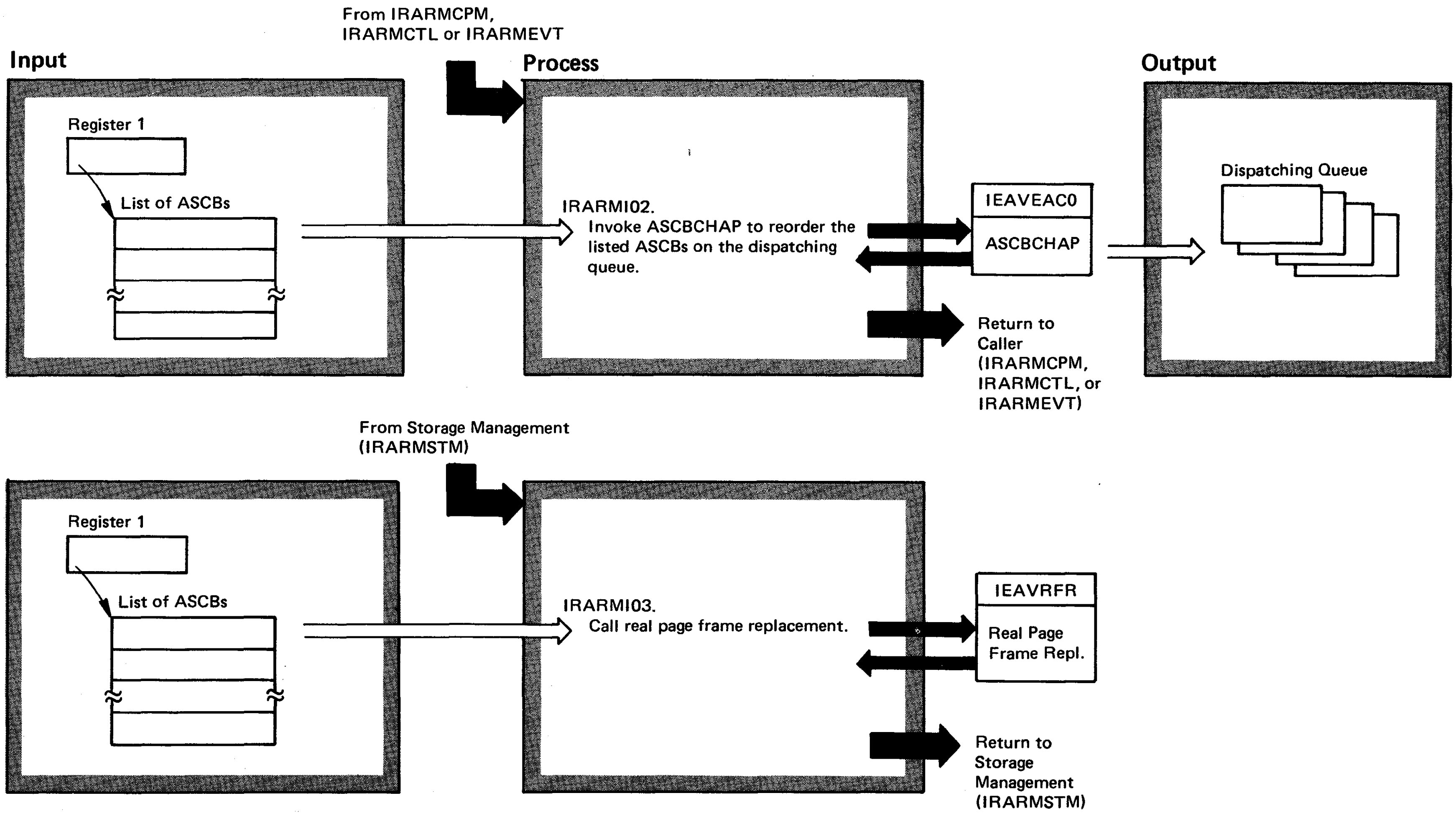


Diagram 6-1A. SRM Service Routine (IRARMSRV) (Part 2 of 6)

Extended Description	Module	Label (or Segment)
This module is a collection of several independent routines which act as interfaces between SRM and various system services.	IRARMSRV	IRARMSRV
IRARMI02 Reposition the listed ASCBs in the ASCB dispatching queue to reflect their new dispatching priorities.	IRARMSRV IEAVEAC0	IRARMI02
IRARMI03 Update UICs in pages belonging to all users listed. Steal pages from users which then have UICs that meet the steal criterion.	IRARMSRV IEAVRFR	IRARMI03

Diagram 6-1A. SRM Service Routine (IRARMSRV) (Part 3 of 6)

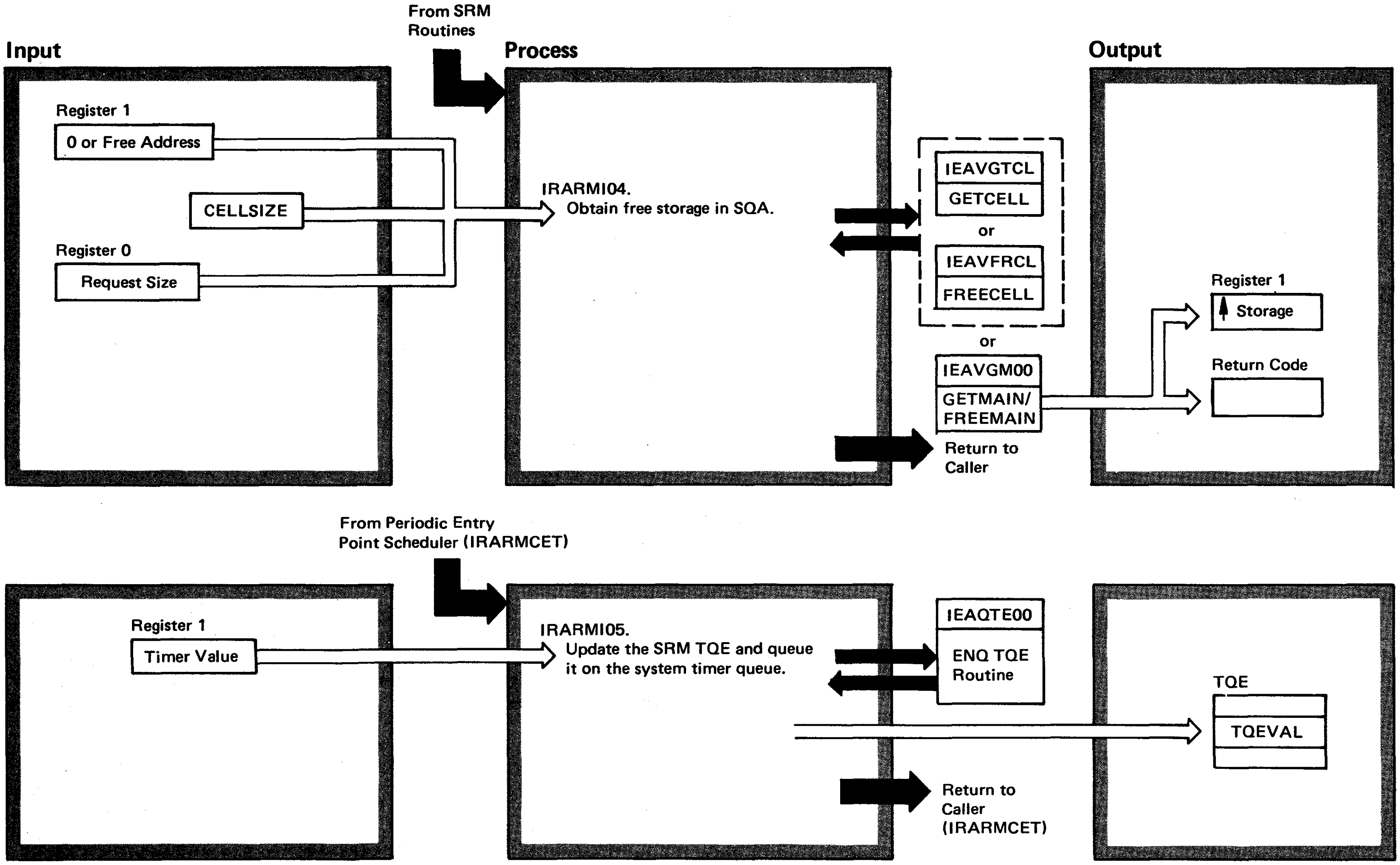


Diagram 6-1A. SRM Service Routine (IRARMSRV) (Part 4 of 6)

Extended Description	Module	Label (or Segment)
IRARMI04	IRARMSRV	IRARMI04
Obtain free SQA storage either from a cell in 'RM1' cellpool or from other available SQA. (See Obtain/Free Storage (IRARMI04) M.O.)	IEAVGTCL or IEAVFRCL or IEAVGM00 IEAVBLDP	
IRARMI05	IRARMSRV	IRARMI05
Store a new timer value in the SRM TQE and queue the TQE on the system timer queue. (See Requeue SRM TQE (IRARMI05) M.O.)	IEAVRTI0 IEAVRTI0	IEAQTD00 IEAQTE00

Diagram 6-1A. SRM Service Routine (IRARMSRV) (Part 5 of 6)

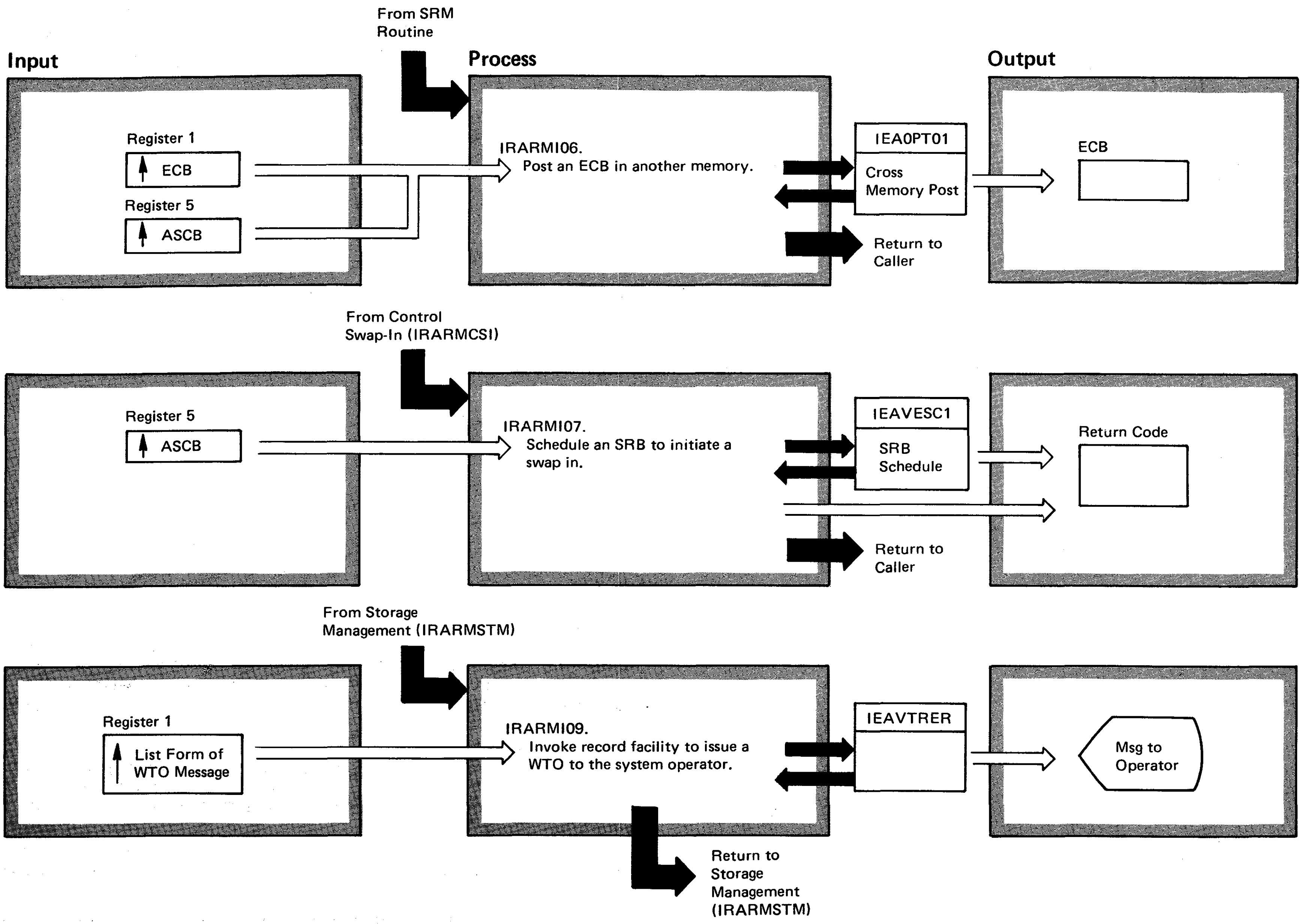
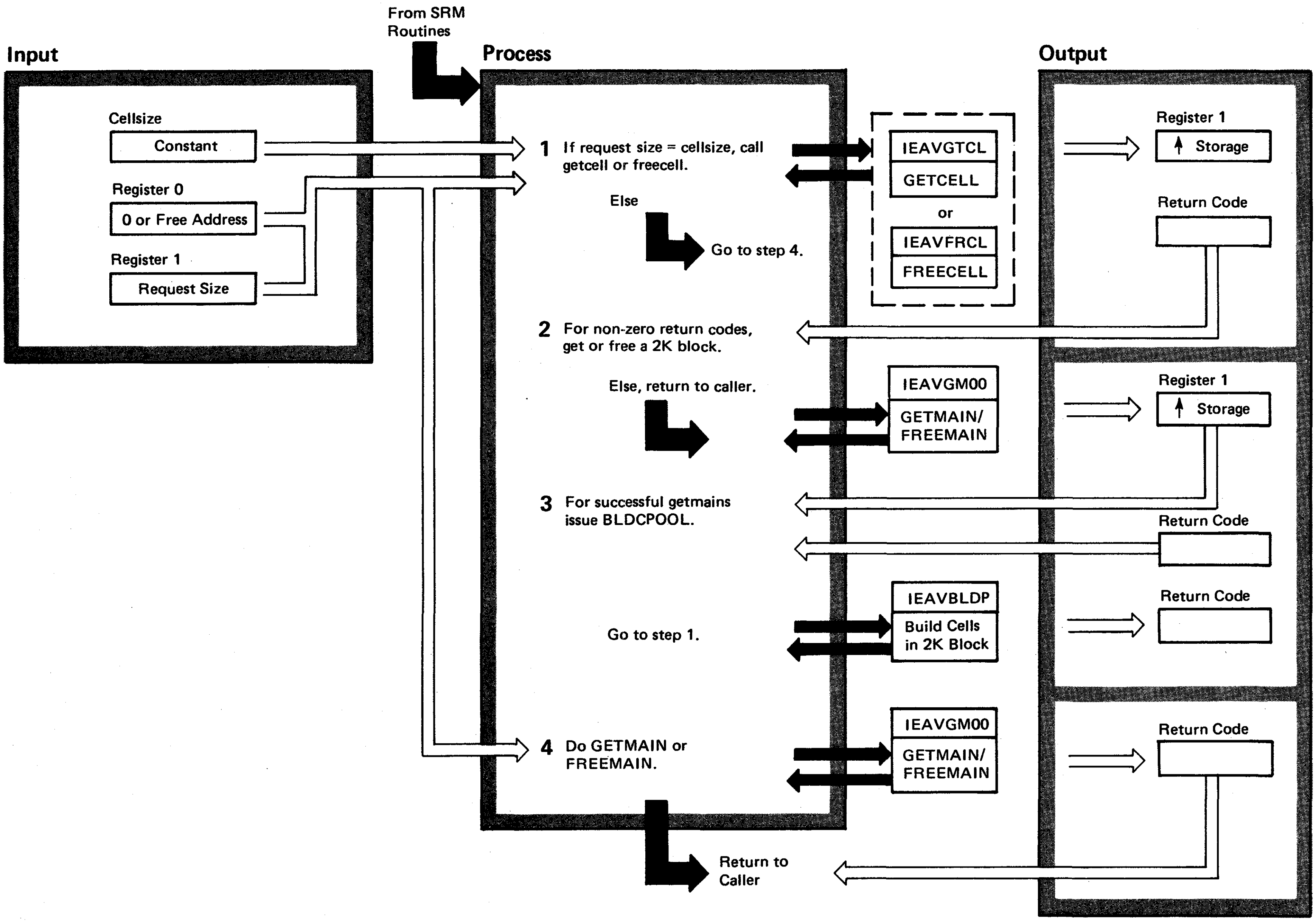


Diagram 6-1A. SRM Service Routine (IRARMSRV) (Part 6 of 6)

Extended Description	Module	Label (or Segment)
IRARMI06	IRARMSRV	IRARMI06
This entry point is used by the swap-out routine to post the region control task (for example).		
If an error is encountered during the cross memory post, the error routine (IRARMXPE) gets control and attempts cleanup while running under an FRR.	IEAOPT01 IRARMSRV	IEAOPT01 IRARMXPE
IRARMI07	IRARMSRV	IRARMI07
Initiates a swap-in, gets an SRB and schedules it to run the RSM swap in routine (IEAVSWIN) in the master memory.	IEAVGTCL IEAVESC0	IEAVGTCL IEAVESC1
IRARMI09	IRARMSRV	IRARMI09
The record facility is invoked to issue a WTO to the system operator console because the requesting SRM routines hold the lock and cannot therefore issue a WTO.	IEAVTRER	IEAVTRER

Diagram 6-1B. Obtain/Free SQA Storage (IRARMI04) (Part 1 of 2)

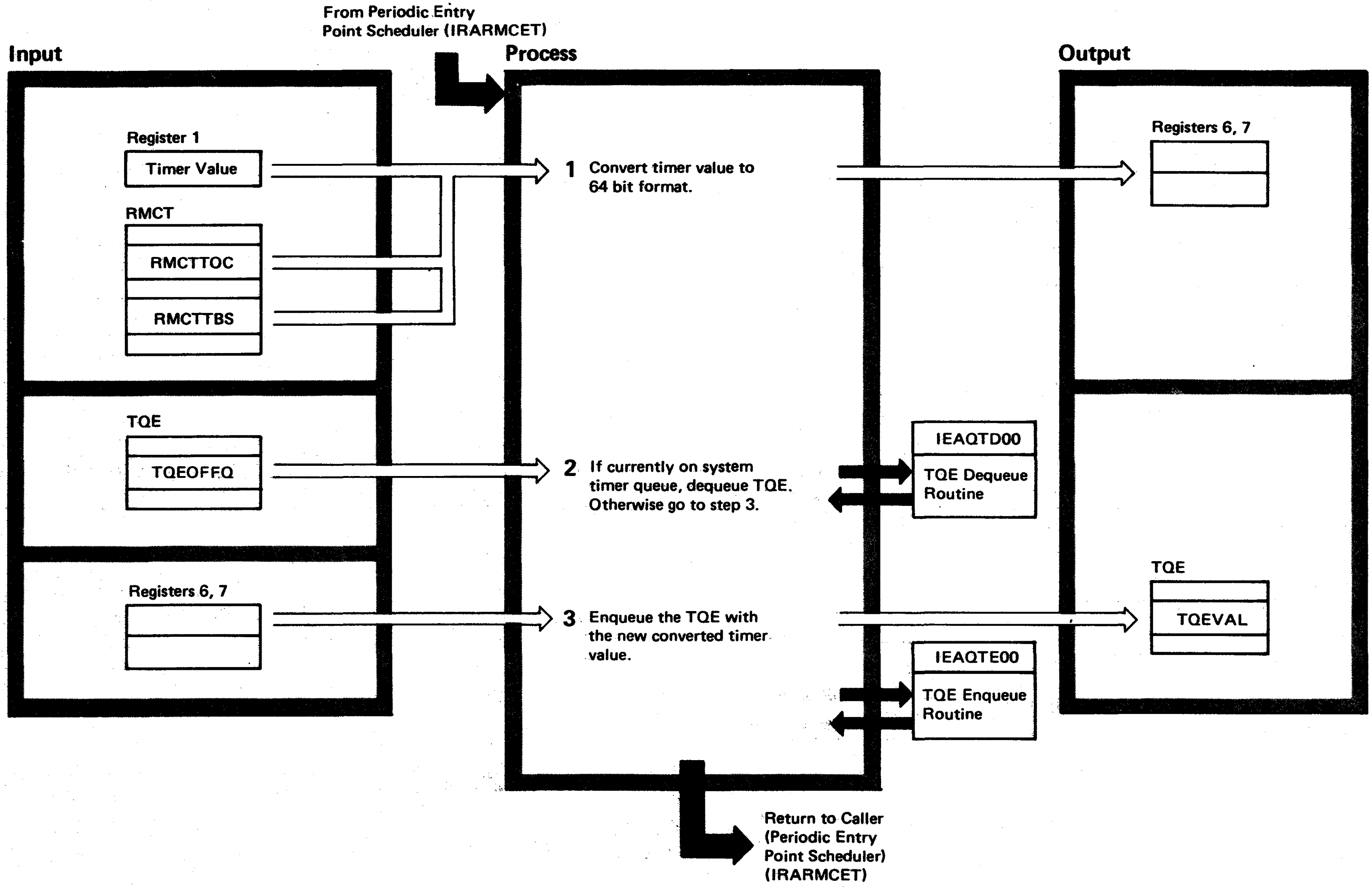


VS2.03.807

Diagram 6-1B. Obtain/Free SQA Storage (IRARMI04) (Part 2 of 2)

Extended Description	Module	Label (or Segment)
This routine is used by the SRM for obtaining and freeing control blocks in key 0, subpool 245 storage (SQA).	IRARMSRV	IRARMI04
Request processing follows the same procedure for both obtaining and freeing storage. If register 0 contains zero, the request is a get.		
1 If the request length matches the cellsize for the IRARMRM1 cellpool, call GETCELL or FREECELL. Otherwise, go to step 4.	IEAVGTCL IEAVFRCL	IEAVGTCL
2 If the GETCELL or FREECELL fails, call GETMAIN or FREEMAIN, for a 2048-byte block. Otherwise, return to caller of IRARMI04 and the GETMAIN/FREEMAIN return code becomes the IRARMI04 return code.	IEAVGM00	
3 If a GETMAIN was done and was successful, issue BLDCPOOL to segment the returned storage into cells. If BLDCPOOL succeeds, go to step 1. Otherwise, return to the caller.	IEAVBLDP	IEAVBLDP
4 Get the SALLOC lock and perform a GETMAIN or FREEMAIN for the original request size. Then release the SALLOC lock. The GETMAIN/FREEMAIN return code becomes the IRARMI04 return code.	IRARMSRV IEAVELK IEAVGM00	GETSTOR or FREESTOP

Diagram 6-1C. Requeue SRM TQE (IRARMI05) (Part 1 of 2)



1 Convert timer value to 64 bit format.

2 If currently on system timer queue, dequeue TQE. Otherwise go to step 3.

3 Enqueue the TQE with the new converted timer value.

Diagram 6-1C. Requeue SRM TQE (IRARMI05) (Part 2 of 2)

Extended Description		**Module**	**Label (or Segment)**
This routine updates the SRM timer queue element with a new timer value and enqueues the element on the system timer queue.		IRARMSRV	IRARMI05
1	Convert timer value to hexidecimal format.	IRARMSRV	IRARMI05
2	Dequeue timer queue element (TQE) if currently queued. This is done under the dispatcher lock.	IEAVRTI0	IEAQTD00
3	Enqueue the TQE. This is done while holding the dispatcher lock.	IEAVRTI0	IEAQTE00

IRARMSRV Module Entry Point Summary

IRARMI02 - ASCB CHAP entry point.
IRARMI03 - Real Page Frame Replacement entry point.
IRARMI04 - Obtain or Free SQA Storage.
IRARMI05 - Requeue SRM TQE Routine.
IRARMI06 - Cross-Memory Post entry point.
IRARMI07 - Swap SRB SCHEDULE Routine.
IRARMI09 - RECORD entry point.

IRARMERR Module Entry Point Summary

IRARMRR1 - Functional Recovery for Globally Locked Entries (entries to SRM in which the SRM lock could not be obtained).
Retry the failing SRM routine when possible. Otherwise percolate the error.
RARMRR2 - Functional Recovery for Non-Globally Locked Entries (entries to SRM in which the SRM lock was obtained). Validate queues and cleanup. Retry the failing routine if possible; otherwise, percolate the error.
RMRR2RTY - Return to RTM indicating retry.
RMRR2PER - Return to RTM indicating percolation.
RMRR2INT - FRR initialization.
RMRR2VLD - Validate control blocks.
RMRR2GST - Release the dispatcher lock in order to call IRARMI04.
RMRR2CKQ - Verify the location of an OUCB.
RMRR1VFB - Verity addresses.
RMRR2REQ - OUCB enqueue routine entry point.
RMRR2SPR - Return with the return code in register 15.

SYSEVENT Processor

The SYSEVENT processor function (IRARMEVT) receives control from the interface function to perform processing related to the SYSEVENT, and, in most cases, to request the services of other internal SRM routines. In a multiprocessing environment the system may not be able to perform some of these routines immediately because of concurrent SRM processing on another CPU. In these cases, execution of the requested routines is deferred until a later SRM invocation. Listed are all SYSEVENTs in alphabetical order along with their associated codes.

The next diagram lists the SYSEVENTs (numerically by code), the situation occasioning their issuance, information passed to and returned from them, internal SRM routines that they may explicitly invoke, the functions of these routines, and any exceptional notes about the SRM action taken as a result of a SYSEVENT. Also, this diagram indicates for each SYSEVENT whether the SRM lock is obtained by the SRM interface routine and where control passes after the SYSEVENT is processed. All SYSEVENTs receive the associated SYSEVENT code (listed below the SYSEVENT name) as input information (in byte 3 of register 0), although this information is not explicitly mentioned in the figure. Where an ASID is listed as input, it is passed in register 0, bytes 0 and 1.

Note that some SYSEVENTs do not hold the SRM lock. These SYSEVENTs return directly to the SRM interface for return to the issuer. Most other SYSEVENTs exit to the SRM control function, which may then invoke algorithm processing. Thus, for a given SYSEVENT entry to the SRM, processing may be performed that is unrelated to the purpose of the original SRM entry.

ALTCPREC (33)
AVQOK (24)
AVQLOW (23)
BRINGIN (44)
CONFIGCH (29)
COPYDMDT (40)
DEVALLOC (28)
DONTSWAP (41)
ENQHOLD (20)
ENQRLSE (21)
INITATT (10)
INITDET (11)
JOBSELCT (8)
JOBTERM (9)
MEMCREAT (6)
MEMDEL (7)
NEWIPS (32)
NIOWAIT (3)
OKSWAP (42)
QSCEFL (18)
QSCEMCP (13)
QSCEST (12)
RSMCNSTS (22)
REQPGDAT (39)
REQSERVC (38)
REQSVDAT (49)
REQSWAP (43)
RESETPG (31)
RSTORCMP (19)
SETDMN (37)
SQALOW (25)
SQAOK (26)
SWINFL (17)
SWOUTCMP (15)
SWPINST (16)
SYQSCCMP (36)
SYQSCST (35)
TERMWAIT (2)
TGETTPUT (34)
TIMEREXP (1)
USERRDY (4)
VERIFYPG (30)
WKLDCOLL (46)
WKLDINIT (45)
WKLDTERM (47)
(48)

Diagram 6-2. SYSEVENT Processor (Part 1 of 16)

SYSEVENT	When Issued	Information		Routines Invoked	Function of Invoked Routine	SRM Action	SRM Lock Held	Exit To
		Passed	Returned					
TIMEREXP (1)	SRM timer interval has expired.	Indication whether TOD clock initialization (01) or not (00). (register 1, byte 3).	None	Periodic Entry Point Initialization (IRARMWMY).	Resets the "time due" fields of the time driven queue according to the current time, for TOD initialization.	This SYSEVENT provides the exclusive means for invoking the majority of the SRM algorithms.	Yes	SRM Control (IRARMCTL)
				Periodic Entry Point Scheduling (IRARMCET).	See Periodic Entry Point Scheduling M.O.			
TERMWAIT (2)	Issued by TGET or TPUT when a user enters terminal wait.	● ASID. ● Input (00) or output (80) indication. (register 1, byte 0).	None	Control Swapout (IRARMCSO).	See Control Swapout M.O. called for swappable users.		Yes	SRM Control (IRARMCTL)
NIOWAIT (3)	Issued by WAIT macro processing when some task in an address space enters long wait.	● ASID.	None	Control Swapout (IRARMCSO).	See Control Swapout M.O. called for swappable users.		Yes	SRM Control (IRARMCTL)
USERRDY (4)	An SRB has been scheduled for an address space for which QUIESCE is running, or for a swapped out address space.	● ASID.	None	User Ready Processing (IRARMHIT).	The ready user is placed on the "OUT" queue.	User-ready processing is performed through the action request routine.	No	Invoker Via IRARMI01
MEMCREAT (6)	An ASID has been associated with a new address space and space has been obtained for an ASCB and OUSB.	● ASID. ● START(01)/ LOGON(02)/ MOUNT(03) indication. (register 1, byte 0).	Indication whether or not memory creation should not proceed because of a resource shortage. (00—proceed 80—do not proceed). (register 1, byte 0).	Storage Request (IRARMI04).	Obtain storage for an OUCB and OUXB if no resource shortages exist.		Yes	SRM Control (IRARMCTL)
				User Control Block Repositioning (IRARMRPS).	Place user on "in" queue.	User Control Block Repositioning is performed through the action request routine.		
MEMDEL (7)	Storage associated with an ASCB is about to be freed, and an ASID disassociated with an address space.	● ASID.	Indication that memory delete may not proceed and must wait. —04 (register 1, byte 3).	OUCB and OUXB delete (IRARMDEL).	Free storage associated with an OUCB and an OUXB, and indicate that memory delete processing may proceed by issuing XMPOST to the Master Memory.	OUCB and OUXB delete is performed indirectly, through action request routine.	Yes	SRM Control (IRARMCTL)

Diagram 6-2. SYSEVENT Processor (Part 2 of 16)

SYSEVENT	When Issued	Information		Routines Invoked	Function of Invoked Routine	SRM Action	SRM Lock Held	Exit To
		Passed	Returned					
JOBSELCT (8)	An address space has begun using system services, on behalf of a new job, START or MOUNT command, or a TSO session.	• ASID. • Address of job-name or user-id. • Performance Group number (register 0, byte 2).	None	Control Swapout (IRARMCSO).	Called to swapout an address space if a second level auxiliary page shortage exists or an excess of fixed frames exists.	This SYSEVENT authorizes the accumulation of service for the job. SRM validates the performance group number indicated for the address space. If it is not valid, a default value is assigned.	Yes	SRM Control (IRARMCTL)
				Transaction Stop Routine (IRARMWMO).	Updates the accumulated time and service for a job. Also indicates that the current transaction has ended or been suspended. If workload activity reporting is active, invokes IRARMWR4 to accumulate report information.			
JOBTERM (9)	An address space has completed using system resources on behalf of a job, START or MOUNT command, or a TSO session.	• ASID. • Address of job-name or user-id.	None	Transaction Stop Routine (IRARMWMO).	Updates the accumulated time and service for a transaction. Also indicates that the current transaction has ended. If workload activity reporting is active, invokes IRARMWR4 to accumulate report information.	This SYSEVENT revokes authorization for starting new transactions.	Yes	SRM Control (IRARMCTL)
INITATT (10)	Whenever an initiator attaches a task.	• ASID. • Performance Group number. (register 1, byte 2).	None	Transaction Resume Processing (IRARMWMR).	Resumes a suspended transaction, if the performance group number for a new non-TSO job step is the same as for the previous step; otherwise starts a new transaction.	SRM validates the performance group number indicated for the address space. If it is not valid, a default value is assigned. If the input dispatching priority is in the APG, the SRM will follow the IPS specification for this user.	Yes	SRM Control (IRARMCTL)
		• Dispatching Priority. (register 1, byte 3).		Change Dispatching Priority (IRARMI02).	Move ASCB to correct position on dispatcher queue.			

Diagram 6-2. SYSEVENT Processor (Part 3 of 16)

SYSEVENT	When Issued	Information		Routines Invoked	Function of Invoked Routine	SRM Action	SRM Lock Held	Exit To
		Passed	Returned					
INITATT (10) (continued)		• Nonswap Authorization (ASCBNSWP bit of ASCB).		Start New Transaction (IRARMWMN).	Indicate the start of a new transaction. If workload activity reporting is active, calls IRARMWR6 to indicate that a transaction has ended.			
INITDET (11)	Whenever an initiator detaches a task.	• ASID. • Dispatching Priority. (register 1, byte 3).	None	Transaction Stop Routine (IRARMWMO).	Updates the accumulated time and service for a transaction. Also indicates that the current transaction has ended or been suspended. If workload activity reporting is active, invokes IRARMWR4 to accumulate report information.		Yes	SRM Control (IRARMCTL)
				I/O Load Balancing IMCB Deletion (IRARMIL4).	Frees I/O measurements control block (which has been created if the user is a heavy I/O user).	IMCB deletion is performed through action request.		
				Change Dispatching Priority (IRARMI02).	Move ASCB to correct position on dispatcher queue.			
QSCEST (12)	Issued during quiesce processing when the status of all associated tasks has been deterdetermined.	• ASID. • Long wait indication. (00—not in long wait 80—in long wait). (register 1, byte 0).	• Continue with (00) or terminate (08) quiesce processing. (register 1, byte 3).	I/O Load Balancing User I/O Monitoring (IRARMILO).	An I/O measurement control block is created for heavy I/O users. The IMCB is updated with channel useage data from the Timing Control Table I/O Table (TCTIOT). (See I/O Management M.O. (IRARMIOM) and I/O Load Balancing User I/O Monitoring M.O. (IRARMILO.)	**Note:** After this SYSEVENT, no further quiesce processing is performed for: • non-swappable users, and • users being swapped because of a long wait, and who are no longer in a long wait status.	Yes	SRM Control (IRARMCTL)

Diagram 6-2. SYSEVENT Processor (Part 4 of 16)

SYSEVENT	When Issued	Information		Routines Invoked	Function of Invoked Routine	SRM Action	SRM Lock Held	Exit To
		Passed	Returned					
QSCECMP (13)	Issued when the RCT has completed quiesce processing for an address space.	• ASID. • Long wait indicator. (00—not in long wait 80—in long wait). (register 1, byte 0).	• Indication whether USERRDY SYSEVENT (4) has been received for this task since quiesce start (00—received 80—not received). (register 1, byte 0). • Indication whether to initiate swapout (00) or begin restore (08). (register 1, byte 3). • If Reg 1 byte 3 is 00, Reg 1 byte 2 contains the swap out reason code.	User Control Block Repositioning (IRARMRPS).	Changes the status of the memory to out-of-real-storage and positions it on the correct queue (normally the "out" queue; however, will be the "wait" queue for users entering long wait, or for users swapped because a resources shortage exists).	User Control Block Repositioning is performed indirectly through action request routine.	Yes	SRM Control (IRARMCTL)
				CPU Load Balancing Profile Adjustment (IRARMCLO).	Updates mean time to wait indication for use by CPU load balancing (see CPU Load Balancing Swap Analysis M.O.) and users in the APG.			
				Transaction Quiesce Processing (IRARMWMQ).	Increments the cumulative service received by a transaction by the amount received during a real storage residence period. Also updates the performance group period indication if a transaction has completed a performance group period. Determines whether to continue the transaction, or to stop or suspend it at this point for the reason that caused the swapout. If workload activity reporting is active, invokes IRARMWR4 to accumulate report information.	**Note**: After this SYSEVENT, no further quiesce processing is performed for: • non-swappable users, and • users being swapped because of a long wait, and who are no longer in a long wait status.		

Diagram 6-2. SYSEVENT Processor (Part 5 of 16)

SYSEVENT	When Issued	Information		Routines Invoked	Function of Invoked Routine	SRM Action	SRM Lock Held	Exit To
		Passed	Returned					
SWOUTCMP (15)	All I/O required to swapout a memory has completed.	● ASID. ● Pointer to parameter list (register 1) containing: – number of pages swapped out. (word 1, bytes 0 & 1). – working set size, in number of pages to be swapped in. (word 1, bytes 2 & 3). – indication whether address space is waiting for an unfinished RSM service. (word 2, byte 3, bit 7 on means the address space is waiting for service).	None	Free OUXB Storage (IRARMUXB).	Free storage associated with an OUXB, when the address space is swapped out.	IRARMUXB is performed indirectly, through action request.	No	Invoker Via IRARMI01
				Swap Analysis (IRARMCAP).	Swap analysis is requested when a user is voluntarily swapped out.	Swap Analysis is invoked through algorithm request rotuine.		
				User Ready Processing (IRARME04).	See SYSEVENT USERRDY (4).	User Ready processing is invoked if user ready indicator is off, but an indication of an unfinished RSM service is received.		
SWPINST (16)	By RSM to notify of Swap Status	● ASID ● Code in Reg 1: 01–Swap-in Starting 02–Stage one of Swap-In complete	None	None		None	No	Invoker Via IRARMI01
SWINFL (17)	Swapin processing failed to obtain or initialize the LSQA storage for an address space.	● ASID	None	User Control Block Repositioning (IRARMRPS).	Changes the status of the address space to "out of real storage" and positions the OUCB on the correct queue (normally the "out" queue).	User Control Block Repositioning is performed indirectly, through an action routine.	Yes	SRM Control (IRARMCTL)
				Free Storage (IRARMI04).		Free OUXB		

Diagram 6-2. SYSEVENT Processor (Part 6 of 16)

SYSEVENT	When Issued	Information		Routines Invoked	Function of Invoked Routine	SRM Action	SRM Lock Held	Exit To
		Passed	Returned					
QSCEFL (18)	The RCT failed to complete quiesce processing because of an abnormal situation.	• ASID.	None	User Control Block Repositioning (IRARMRPS).	Changes the current status of the user from "out of real storage" to "in real storage".	User Control Block Repositioning is performed indirectly, through an action routine.	Yes	SRM Control (IRARMCTL)
RSTORCMP (19)	The RCT has completed restore processing for an address space.	• ASID. • Long wait indicator. (00—not in long wait 80—in long wait) (register 1, byte 0).	None	Restore Completed Processing (IRARMWMR).	Invoked so the workload manager can initialize the fields used for monitoring service during a period of real storage residence.		Yes	SRM Control (IRARMCTL)
				User Control Block Repositioning (IRARMRPS).	Changes the current status of the user from "out of real storage" to "in real storage".	User Control Block Repositioning is performed indirectly, through an action routine.		
				Control Swapout (IRARMCSO).	Requests that a swappable user still in long wait status be swapped out.			
ENQHOLD (20)	A user's execution is delayed because of a request for a resource being held by another user.	• ASID of memory holding resource. • Address of QCB for resource. (register 1).	None	None		Users in real storage, holding resources in demand by other users, are given a "spurt" of non-swappable service equal to the Enqueue Residence Value (ERV) (see CPU Management M.O.). Users out of storage are marked as holding a resource so that CAP will initiate a swap in.	Yes	SRM Control (IRARMCTL)
ENQRLSE (21)	A contention situation has disappeared because of the release of a resource by a user for whom an ENQHOLD SYSEVENT had previously been received.	• ASID. • Address of QCB for resource. (register 1).	None	None		If user has freed all resources in contention, eliminate special treatment.	Yes	SRM Control (IRARMCTL)

Diagram 6-2. SYSEVENT Processor (Part 7 of 16)

SYSEVENT	When Issued	Information		Routines Invoked	Function of Invoked Routine	SRM Action	SRM Lock Held	Exit To
		Passed	Returned					
RSMCNSTS (22)	Real storage has been configured into or out of the system (because of a VARY storage command, or a storage error).	● Number of pages of functioning real storage. (register 1, bytes 0 & 1). ● New Available Page Queue low limit. (register 1, bytes 2 & 3).	None	None		None	No	Invoker Via IRARMI01
AVQLOW (23)	The number of available real storage page frames has fallen below the Available Page Queue low limit.	● Indication of cause. (register 1, byte 3). 1– Available queue (AVQ) is below limit. 2– AVQ is 1 when a page fault occurred. 3– AVQ is 0 when a page fault occurred. 4– Ratio of fixed frames to total real frames is above a limit.	None	Main Storage Occupancy Analysis (IRARMMS2).	For level 1, 2, or 3, initiate page stealing (see Main Storage Occupancy Analysis M.O.). For level 4, swap out user acquiring fixed frames at the fastest rate. Notify system operator and inhibit creating memories. Repeat swap outs until shortage is relieved.	Because it is important that the Main Storage Occupancy Analysis algorithm be run as soon as possible, an SRB is scheduled after requesting the algorithm; the SRB will issue SYSEVENT 48 when it is dispatched, which will result in the CONTROL function being invoked. This algorithm will then be executed.	No	Invoker Via IRARMI01
AVQOK (24)	Enough real storage pages have been freed to alleviate a shortage condition.		None	None		SRM ceases its special efforts to free up real storage.	No	Invoker Via IRARMI01
SQALOW (25)	There exists a critical shortage of SQA pages.	● Indication whether shortage is of severity 1 (01) or 2 (02). (register 1, byte 3).	None	SQA Shortage Message Writer (IRARMSQA).	Inform System operator of the SQA shortage (see Storage Management M.O.).	The Message Writer algorithm is scheduled for execution the next pass thru the CONTROL function. SRM does not permit the creation of new address spaces when an SQA shortage exists.	No	Invoker via IRARMI01

Diagram 6-2. SYSEVENT Processor (Part 8 of 16)

SYSEVENT	When Issued	Information		Routines Invoked	Function of Invoked Routine	SRM Action	SRM Lock Held	Exit To
		Passed	Returned					
SQAOK (26)	An SQA page shortage has been relieved.	Code indicating the level of the relieved shortage (register 1, byte 3): above level 1 (01) above level 2 (02)	None	SQA Shortage Message Writer (IRARMSQA).	Inform system operator of the fact that an SQA shortage has been relieved (see Storage Management M.O.).	Issue a message if all SQA shortages are relieved (i.e. level 1).	No	Invoke Via IRARMI01
DEVALLOC (28)	A device allocation choice must be made from two or more candidates.	• ASID. • Pointer to a three word list (register 1) containing: – address of a list of candidate UCB addresses. (word 1). – address of a list of UCB addresses already allocated to requestor. (word 2). – address of a two word return area. (word 3).	• Pointer to same three word list as on entry (register 1), with return area containing: – address of the candidate list entry that was selected (word 1). • Successful (00) or unsuccessful (08) indication. (Register 15, byte 3).	None		The UCB is selected by applying the following selection principles in the order indicated: • Avoid contention (reallocating same UCB to same user) for Direct Access. • Avoid allocation on units with premounted volumes. • Give preference to less heavily utilized logical channels, assuming that each previous allocation for this user has know projected constant impact on utilization. • For direct access devices, pick the one with the lowest allocated user count. • Choose randomly, if more than one candidate remains.	Yes	SRM Control (IRARMCTL)
CONFIGCH (29)	A VARY command has been issued for a channel or CPU.	• ASID. • SMF record describing the change. (pointed to by register 1).	None	None		Update SRM control information for logical channel utilization monitoring.	Yes	SRM Control (IRARMCTL).
VERIFYPG (30)	An interpreter has received a performance group number which needs verification.	• Performance group number. (register 1, byte 3).	• Valid (00)/ Invalid (01–non-TSO user; 02–TSO user ASID) indication. (register 1, byte 2).	None		The IPS is checked for performance group number validity. If the number is invalid, a default is provided.	Yes	SRM Control (IRARMCTL)

Diagram 6-2. SYSEVENT Processor (Part 9 of 16)

SYSEVENT	When Issued	Information		Routines Invoked	Function of Invoked Routine	SRM Action	SRM Lock Held	Exit To
		Passed	Returned					
RESETPG (31)	The system operator has entered a RESET command for a particular address space.	• ASID. • New performance group number. (register 1, byte 3).	Return code indicating – request honored (00) or –performance group number invalid (04) or –ASID not currently assigned (08). (register 1, byte 2).	Start New Transaction (IRARMWMN).	For users in real storage, a new transaction is started. For swapped out users a new transaction will be started upon swapin. If workload reporting is active, IRARMWR6 is called to indicate that a transaction has ended.	Starting a new transaction results in the user being associated with the performance objective and domain corresponding to the first period of the performance group definition.	Yes	SRM Control (IRARMCTL)
NEWIPS (32)	The system operator has entered a SET command with the IPS keyword.	• ASID. • Pointer to WMST describing new IPS. (register 1).	• Old IPS description. • Indication whether SET command may proceed (indicated by posting an ECB).	Set to New IPS (IRARMSET).	If Workload Activity reporting is active for MF/1, the reporting is terminated (it will later be re-established by MF/1). The performance group number of all active transactions are examined. If the corresponding performance group has changed in the new IPS, a new transaction is begun; if it is the same, the old transaction continues; if the performance group number is not defined in the new IPS, a default performance group number is substituted.	The IRARMSET routine is called by IRARMIPS, which is performed indirectly, through the action request routine.	Yes	SRM Control (IRARMCTL)
ALTCPREC (33)	As a result of an error some CPU has had to be configured out of the system.	• CPU address. (register 1).	None	None		Updates SRM control information for logical channel utilization imbalances.	Yes	SRM Control (IRARMCTL)

Diagram 6-2. SYSEVENT Processor (Part 10 of 16)

SYSEVENT	When Issued	Information		Routines Invoked	Function of Invoked Routine	SRM Action	SRM Lock Held	Exit To
		Passed	Returned					
TGETTPUT (34)	A TGET or TPUT instruction has completed some I/O to a terminal.	● ASID. ● TGET (0) or TPUT (1) indication. (register 1, byte 0, bit 0). ● (for TGET) entire message transferred indicator. (0—all transferred; 1—at least one more TGET required). (register 1, byte 0, bit 0).	None	Start New Transaction (IRARMWMN).	For TGET, indicates the start of a new TSO transaction. If workload reporting is active, IRARMWR6 is called to indicate that a transaction has ended. If the TGETTPUT SYSEVENT was preceeded by a TERM-WAIT condition the IRARMWMN routine was instead called at the time the address space was swapped in.	Starting a transaction results in the user being associated with the first period of his performance group.	Yes	SRM Control (IRARMCTL)
SYSQSCT (35)	The system start/stop routine has been entered to stop the system.	None	None	None		The SRM saves the time at which the system was stopped.	No	Invoker Via IRARMI01
SYQSCCMP (36)	The system start/stop routine is about to restart the system.		None			Steps forward transaction starting times by the duration of the system stoppage.	No	Invoker Via IRARMI01
SETDMN (37)	The operator entered a SETDMN command to change constraint values for a domain.	Data area address (register 1). Byte 0—Domain number Byte 1—Flags Bit 0—New minimum passed Bit 1—New maximum passed Bit 2—New weight passed. Byte 2— New minimum Byte 3—New maximum Byte 4—New weight.	Return code (register 15) 0: Successful 4: Invalid domain 8: Minimum exceeds maximum	None		Update the domain control table with the new ranges or weights.	Yes	Invoker Via IRARMI01

Diagram 6-2. SYSEVENT Processor (Part 11 of 16)

SYSEVENT	When Issued	Information		Routines Invoked	Function of Invoked Routine	SRM Action	SRM Lock Held	Exit To
		Passed	Returned					
REQSERVC (38)	Issued by the TSO TIME command, to obtain user related service data.	• ASID. • Address of 4-word return area. (register 1).	• Return area for a TSO user: – Total service (word 1). – Total transaction active time for all transactions (word 2). – Last performance group number (word 3, bytes 0 & 1). – Total number of transactions (word 3, bytes 2 & 3). • Return area for a non-TSO user: – Total service (word 1). – Total transaction active time (word 2). – Last performance group number (word 3, bytes 0 & 1). • Indication whether data was successfully returned (00) or not (04). (register 15, byte 3).	Service Calculation Routine (IRARMWM1).	Calculates the service accumulated during the current "in real storage" interval. This is added to previous accumulated service to obtain total service.	Accumulated service information is stored in the user's area (while not holding the SRM lock) and under the user's protect key.	Yes	Invoker Via IRARMI01

Diagram 6-2. SYSEVENT Processor (Part 12 of 16)

SYSEVENT	When Issued	Information		Routines Invoked	Function of Invoked Routine	SRM Action	SRM Lock Held	Exit To
		Passed	Returned					
REQPGDAT (39)	Issued by SMF during step termination, to obtain user paging data. **Note:** This SYSEVENT is intended for use only by SMF. It should not be issued by callers others than SMF, because the related data fields in the OUSB and the OUXB are reset to zero on readout. If requested by another caller, the data would be lost to SMF.	• ASID. • Address of 14-word return area. (register 1).	• Return area: – Non VIO page-ins (word 1). – Non VIO page-outs (word 2). – Non VIO reclaims (word 3). – VIO page-ins (word 4). – VIO page-outs (word 5). – VIO reclaims (word 6). – Pages swapped in (word 7). – Pages swapped out (word 8). – Swapouts (word 9). – Common area page-ins (word 10). – Common area reclaims (word 11). – Pages stolen (word 12). – CPU page-seconds (words 13,14). • Indication whether data was successfully returned (00) or not (04). (register 15, byte 3).	None		The SRM obtains paging data from SRM control blocks and resets related fields in these blocks to zero.	Yes	SRM Control (IRARMCTL)

Diagram 6-2. SYSEVENT Processor (Part 13 of 16)

SYSEVENT	When Issued	Information		Routines Invoked	Function of Invoked Routine	SRM Action	SRM Lock Held	Exit To
		Passed	Returned					
COPYDMDT (40)	Issued when a "DISPLAY" command with the keyword "DMN" has been entered.	Pointer to a fixed area of 2584 bytes. (register 1).	Pointer to same area. (register 1) Byte 0 and 1 — Count of Domains. Byte 2 and 3 — Reserved. Byte 4 — 2583 contain copy of Domain Table.	None		Duplicate Domain Information	Yes	Invoker Via IRARMI01
DONTSWAP (41)	Issued to notify SRM that the issuing address space must not be swapped out until either a OKSWAP (42) or INITDET (11) SYSEVENT.	• ASID.	• Indication whether request was honored (00), was dishonored because it was not for the current address space (04), or was dishonored because it was not authorized (08). (register 1, byte 3).	Swap Status Change Request (IRARMWMK).	Determine SRM algorithms applicable to user, and reposition user on SRM swap queue.		Yes	SRM Control (IRARMCTL)
OKSWAP (42)	Issued to notify SRM that issuing address space, which had previously issued a DONTSWAP SYSEVENT, may again be considered for swapping.	• ASID.	• Indication whether request was honored (00) was not for the current address space (04), or was not authorized (08). (register 1, byte 3).	Swap Status Change Request (IRARMWMK).	Same as for DONTSWAP (41).		Yes	SRM Control (IRARMCTL)

Diagram 6-2. SYSEVENT Processor (Part 14 of 16)

SYSEVENT	When Issued	Information		Routines Invoked	Function of Invoked Routine	SRM Action	SRM Lock Held	Exit To
		Passed	Returned					
REQSWAP (43)	Issued when a VARY storage command has been issued, to swapout the address space that occupies the storage to be taken offline. Issued also at job step initiation of a non-swappable user, so that, when swapped back in, the user may be allocated particular page frames to enhance recovery from real storage errors.	• ASID. • Address of ECB to be posted (if dependency exists on requested swap). (register 1).	• Indication whether request was honored (00), was ignored because the address space is non-swappable (04), or was ignored because the address space is in the process of swapout (08). (register 1, byte 3).	Control Swapout (IRARMCSO).	Initiates the swapout of the address space (see CONTROL SWAPOUT M.O.).	Quiesce is posted to begin the swapout. If swap completion notification is requested (by providing an ECB), the ECB will be posted when the address space is next swapped in.	Yes	SRM Control (IRARMCTL)
BRINGIN (44)	Issued when the system operator has issued a CANCEL command for a particular job.	• ASID.	• Indication whether request was honored (00), or was not honored because the address space was in the process of being swapped (08). (register 1, byte 3).	Simulate User Ready Notification (IRARMHIT).	Invokes IRARMWMU to make the memory eligible for swap-in.	Expedite the swap-in of a memory that is swapped-out.	Yes	SRM Control (IRARMCTL)
WKLDINIT (45)	Issued by MF/1 to request that SRM begin collecting workload activity data.	• ASID. • Data collection buffer address. (register 1).	• Indication whether request was honored (00), was not honored because of incorrect buffer size (08), or data collection is already active (20). (register 15, byte 3).	Workload Activity Recording Initialization (IRARMWR1).	Constructs and initializes the workload activity measurement table (WAMT).		Yes	SRM Control (IRARMCTL)

Diagram 6-2. SYSEVENT Processor (Part 15 of 16)

SYSEVENT	When Issued	Information		Routines Invoked	Function of Invoked Routine	SRM Action	SRM Lock Held	Exit To
		Passed	Returned					
WKLDCOLL (46)	Issued by MF/1 at the end of a reporting interval, to collect workload activity data.	● ASID. ● Data Buffer address. (register 1).	● Indication whether request was honored (00), whether an IPS change has occurred (04), or data buffer had not yet been established (40). (register 15, byte 3).	Workload Activity Recording Data Collection (IRARMWR3).	Moves the contents of the WAMT into a collection buffer.		Yes	SRM Control (IRARMCTL)
WKLDTERM (47)	Issued by MF/1 to terminate workload activity data recording, at MF/1 termination or when an IPS change has occurred.	● ASID.	● Address of the buffer no longer used by SRM. (register 1). ● Indication whether the request was honored (00) or the data collection buffer had not yet been established (40). (register 15, byte 3).			The SRM indicates that workload activity data collection no longer be performed.	Yes	SRM Control (IRARMCTL)
(48)	Issued by the SRM when the control function must be invoked immediately (i.e., without waiting for the next SYSEVENT issued by another component).	● ASID. ● Address of issuing SRB. (register 1).	None	SRM Control (IRARMCTL).	Performs control mainline processing, in the course of which a scheduled critical function will be performed (see SRM Control M.O.).	Frees up SRM SRB for reuse.	Yes	SRM Control (IRARMCTL)

Diagram 6-2. SYSEVENT Processor (Part 16 of 16)

SYSEVENT	When Issued	Information		Routines Invoked	Function of Invoked Routine	SRM Action	SRM Lock Held	Exit To
		Passed	Returned					
REQSVDAT (49)	Issued by SMF during job session termination to obtain user related service data.	• ASID. • Address of 4-word return area. (register 1).	• Return area for a TSO user: – Total service (word 1). – Total transaction active time for all transactions (word 3). – Last performance group number (word 3, bytes 0& 1). – Total number of transactions (word 3, bytes 2 & 3). – Session Residency time (word 4). • Return area for a non-TSO user: – Total service (word 1). – Total transaction active time (word 2). – Last performance group number (word 3, bytes 0& 1). – Session Residency time (word 4). • Indication whether data was successfully returned (00) or not (04). (register 15, byte 3).	Service Calculation Routine (IRARMWM1).	Calculates the service accumulated during the current "in real storage" interval. This is added to previous accumulated service to obtain total service.	Accumulated service information is stored in the caller's area under the caller's protect key.	Yes	Invoker Via IRARMI01

IRARMEVT Module Entry Point Summary

IRARMEVT	SYSEVENT processor. Begin to process the indicated SYSEVENT.
IRARMXVT	SYSEVENT retry. Prepare a retry of a sysevent that had incurred a system error.
IRARMDEL	Synchronize memory delete processing.
IRARMIPS	Set new IPS. Invoke IRARMSET to establish a new IPS.
IRARMUXB	Synchronize OUXB deletion at swapout completion time.

SRM Control

SRM Control is the dispatcher of SRM. It is packaged in the module IRARMCTL along with the swap analysis algorithm and various other SRM routines (see volume table of contents, figure 2-9). Most SYSEVENTs which execute holding the SRM lock exit to SRM Control to perform the following functions.

- SRM Control executes deferred actions (routines which execute on behalf of a single user memory). Examples of actions are:
  - moving a user control block from one SRM queue to another.
  - memory delete processing.
- SRM Control executes deferred algorithms (routines which execute on behalf of the entire operating system). Examples of algorithms are:
  - Real Page Shortage Prevention.
  - Real Page Shortage Page Replacement.
- Following the TIMEREXP SYSEVENT, SRM Control schedules timed algorithms. Examples of timed algorithms are:
  - assigning swappable users their current workload level (Swappable User Evaluation Algorithm).
  - Keeping the multiprogramming level (MPL) at its target level in each domain by performing user swaps (Swap Analysis Algorithm).

Action/Algorithm Scheduling

Actions and algorithms can be requested/scheduled by any of the components of SRM. These requests are processed by request handling subroutines within IRARMCTL. Requests for actions are processed in one of the following ways:

- The action is called inline if the SRM lock is held and if the action was not requested by another action.
- Otherwise, the action is deferred. A flag is set in the OUCB to indicate that the action was requested.

Requests for algorithms are always deferred. A flag is set in the RMCT to indicate that the algorithm was requested. If an action or algorithm which has been deferred is critical, the request handling subroutine schedules an SRB to another entry point, IRARMCED, within IRARMCTL. IRARMCED executes SYSEVENT 48. SYSEVENT 48 exits to SRM Control where the deferred action or algorithm is executed.

Non-critical actions and algorithms which have been requested but deferred are executed the next pass through SRM Control. This execution will normally occur after processing the next SYSEVENT while holding the SRM lock.

SRM Control identifies which actions and algorithms to execute by bit strings in the OUCB (for actions) and the resource manager control table (RMCT) (for algorithms). "On" bits in the OUCB (OUCBACN field) and in the RMCT (RMCTALA and RMCTALR fields) identify deferred action and algorithm requests, respectively. The actual addresses of the individual routines that process actions and algorithms are located in resource manager entry point elements (RMEPs) which are chained together. One RMEP chain exists for actions and another for algorithms. SRM Control compares the "on" bits in the bit string (the OUCB or RMCT) against each RMEP in the action/algorithm RMEP chain. When a match is found, the entry point address in the isolated RMEP identifies the action or algorithm routine that will get control. As a part of routing to the identified routine, SRM Control turns off the bit in the OUCB or RMCT used in finding the proper RMEP. When all bits in the OUCB and RMCT bit strings are "off" SRM Control has processed all deferred actions and algorithms and exits to a return point in the SRM interface module IRARMINT. Figures 2-9B and 2-9C show in more detail the routines and bit settings used in processing algorithms and actions.

Swap Analysis

The swap analysis algorithm is concerned with maintaining the multiprogramming level at the target value in each domain defined to the system. A domain is a group of user memories defined in the installation performance specification (IPS) which have common execution characteristics (for example, all TSO users might be assigned to one domain). The multiprogramming level (MPL) in a domain is the number of users in that domain which are in real storage. The target multiprogramming level is the number of users in real storage which the SRM resource monitor has determined is optimal for this domain.

SRM recognizes user memories, (i.e. address spaces) as being in one of three states. Each state corresponds in concept to a queue on which OUCBs that describe address spaces are placed. The three possible states of an address space are:

IN - The working set of an address space in this state occupies real storage.

WAIT- The working set of an address space in this state does not occupy real storage (that is, has been swapped out), and the address space is incapable of being placed into execution.

OUT - The working set of an address space in this state does not occupy real storage, however, the address space is capable of executing, and may be considered for swap-in.

The decision to swap address spaces is made based on a number of input factors supplied by other SRM functions. The workload manager provides workload levels for each user. The resource-use algorithms tell which users are significant users of system resources (via individual recommendation values). Swap analysis combines the individual recommendations of the workload manager and resource managers into a composite recommendation value. The steps of the swap analysis algorithm are defined below in the order of execution. In steps one and three all domains are considered in numerical order. The algorithm is terminated at the end of any step which has resulted in at least one swap.

1. Unilateral Swap-Out. In each domain the required number of user memories are swapped out to lower the MPL to its target value. Users which have the smallest recommendation values (RVs) in each domain are selected for swap out.
2. Express Swap-In. If there is a user out of real storage which is enqueued on a resource requested by another user, the enqueued user is swapped in if this can be done without exceeding the target MPL in that domain. If the MPL would be exceeded, the user with the smallest RV in that domain is swapped out to lower the MPL. The enqueued user will be swapped in on the next invocation of swap analysis. If no user is swappable, the enqueued user is swapped in. This raises the MPL in that domain above its target temporarily.
3. Unilateral Swap-In. In each domain, the required number of user memories are swapped in to raise the MPL to its target value. Users which have the largest RVs in each domain are selected for swap in.
4. Exchange Swap. For a domain having its MPL at the target, an in-real-storage user memory and an out-of-real-storage memory are selected for an exchange. The user in real storage with the smallest recommendation value are selected. The difference in their recommendation values must exceed a limit (RMPTXCHT) to proceed with the exchange. If an exchange is justified, the swap out of the in-real-storage user is initiated, and the swap in of the out-of-real-storage user memory is deferred until a subsequent invocation of swap analysis.

The following M.O.s describe SRM Control processing and other important routines within IRARMCTL:

- Swap analysis (IRARMCAP), which analyzes users and, if it determines a swap desirable, requests it.
- Control swap-out (IRARMCSO), which initiates requested user swap-outs.
- Control swap-in (IRARMCSI), which initiates requested user swap-ins.
- Select user for swap-in (IRARMCPI), which finds the user with the highest recommendation value in its domain.
- Select user for swap-out (IRARMCPO), which finds the user with the lowest recommendation value in its domain.
- User evaluation (IRARMCVL), which calculates a recommendation value for a specific user.

	SRM Lock Held	Scheduling Routine	RMEP Chain	Bit String	SRM Lock Held	Executing Routines	RMEP Chain
ACTIONS	NO YES*	IRARMCRN	EPDT	OUCBACN**	YES	(IRARMCEN, IRARMCRT)	EPDT
ALGORITHMS	NO YES	IRARMCRL IRARMCRL	EPAT EPAT	RMCTALA RMCTALR	YES	(IRARMCEL, IRARMCRT)	EPAT
TIMED ALGORITHMS	YES	IRARMCET	IRACTMQE	RMCTALR	YES	(IRARMCEL, IRARMCRT)	EPAT

*If SRM lock is held when an action is requested, it is not deferred (except where an action invokes another action). Control passes to IRARMCRY (if the action is IRARMCSI or IRARMCSO) or to IRARMCRN (for all other actions) and then directly to the action.

**During execution this field is inspected only in OUCBs which have been queued on the action queue by the action-scheduling routine (IRARMCRN).

Figure 2-9B. Processing of Algorithms and Actions in IRARMCTL

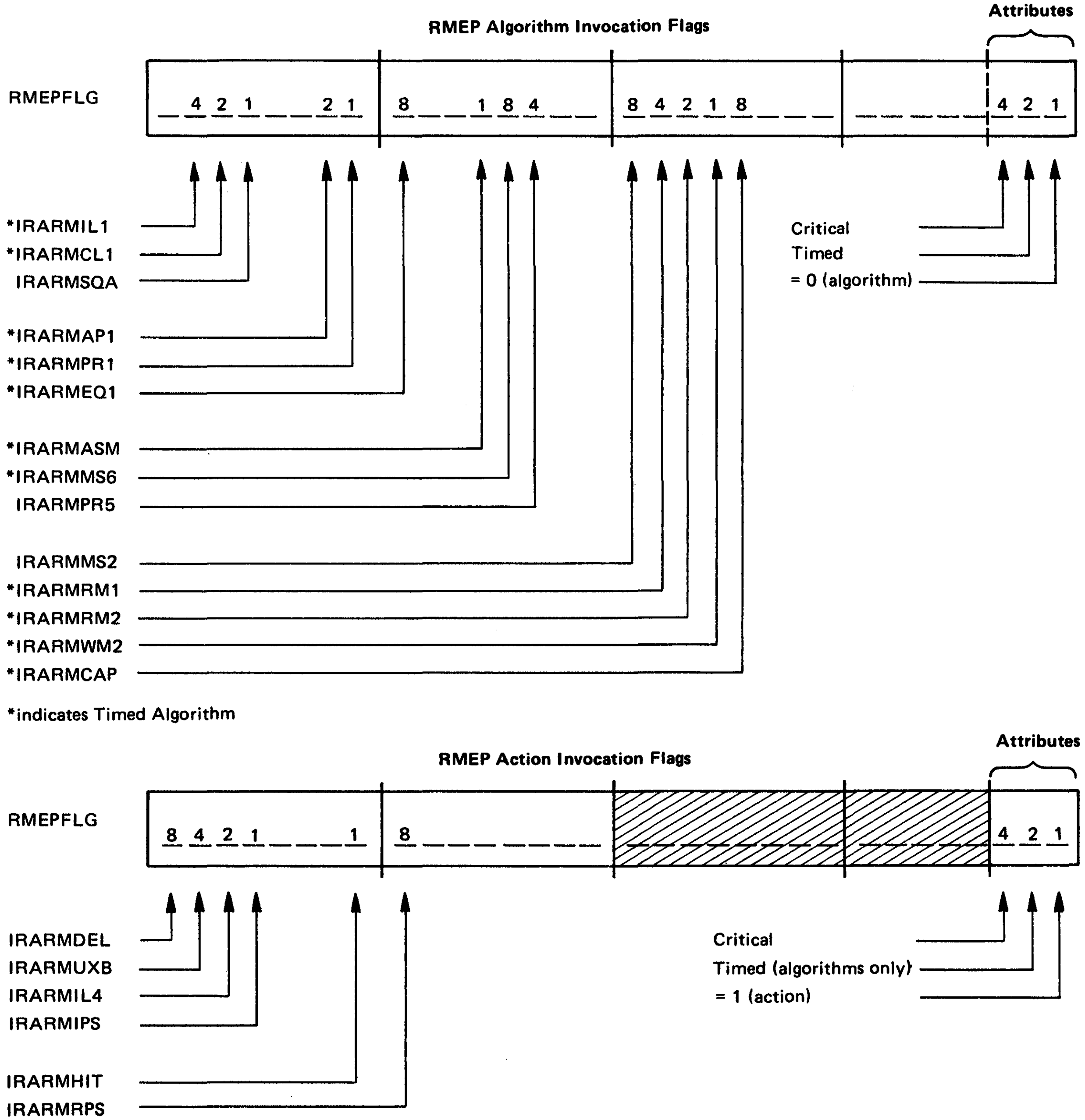
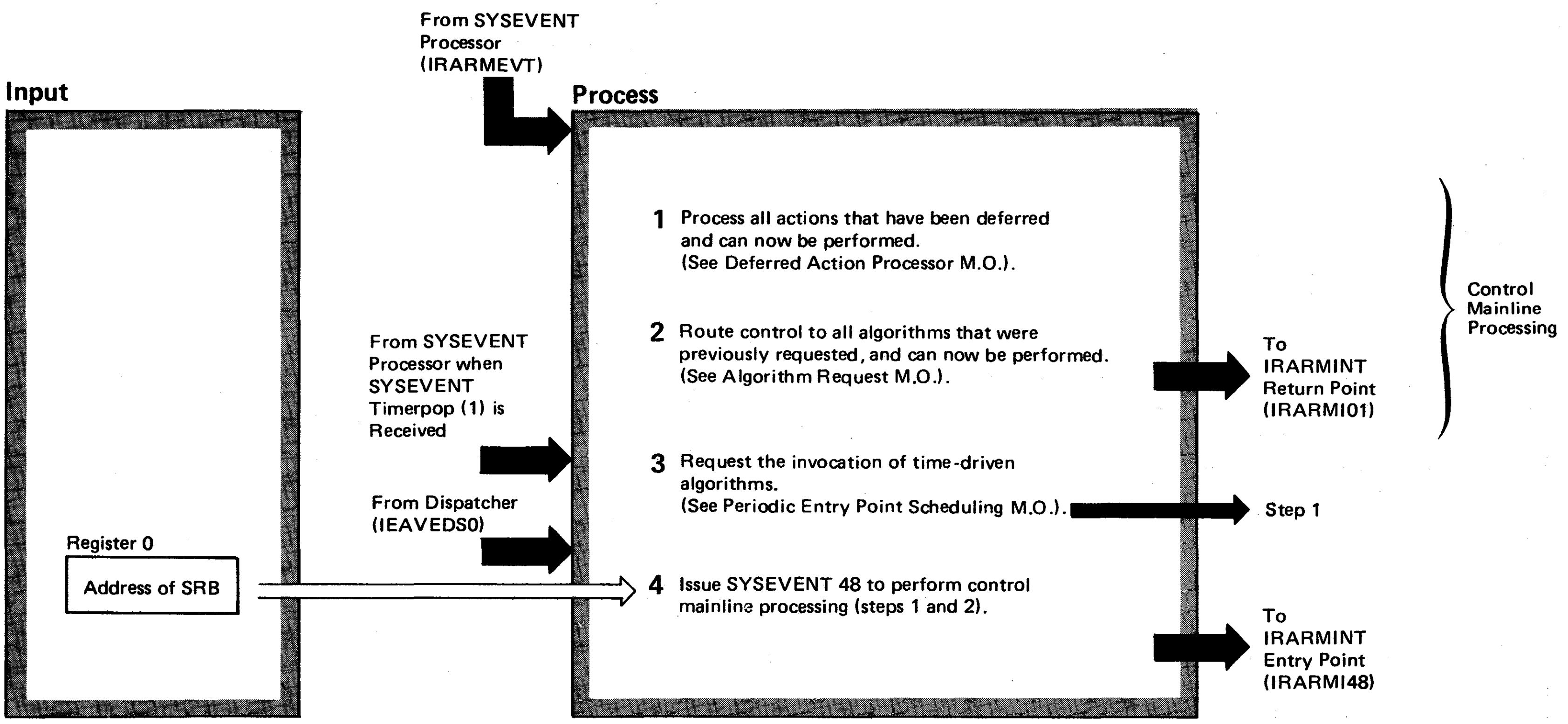


Figure 2-9C. RMEP Algorithm and Action Invocation Flags

Diagram 6-3. SRM Control (IRARMCTL) (Part 1 of 2)



1 Process all actions that have been deferred and can now be performed. (See Deferred Action Processor M.O.).

2 Route control to all algorithms that were previously requested, and can now be performed. (See Algorithm Request M.O.).

3 Request the invocation of time-driven algorithms. (See Periodic Entry Point Scheduling M.O.).

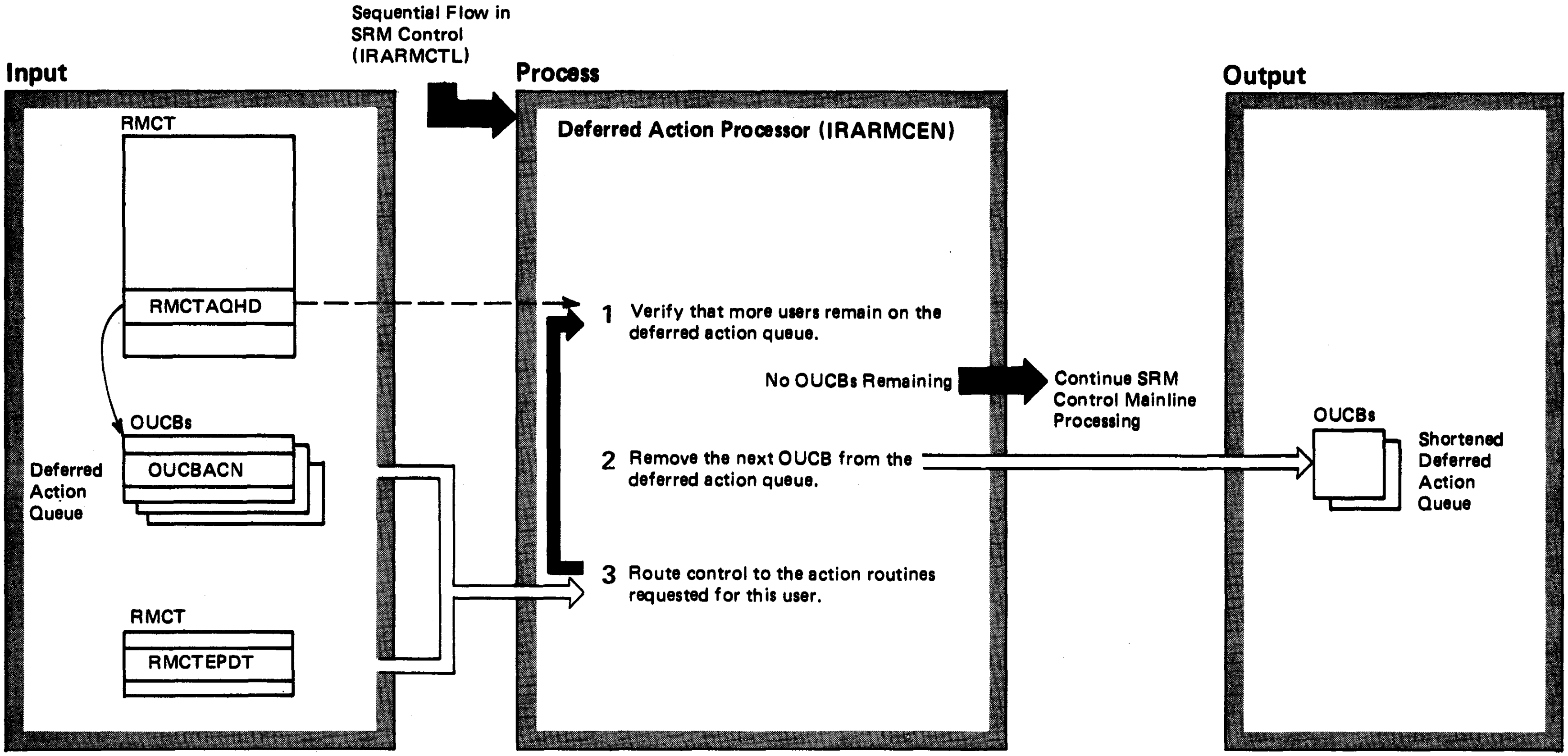
4 Issue SYSEVENT 48 to perform control mainline processing (steps 1 and 2).

Diagram 6-3. SRM Control (IRARMCTL) (Part 2 of 2)

Extended Description	Module	Label
SRM Control routes control to actions and algorithms which have been requested and also to timed algorithms which have come due.	IRARMCTL	
1 Route control to actions which have been requested but deferred. Actions are SRM functions performed on behalf of a single user.	IRARMCTL	IRARMCEN
2 Route control to algorithms which have been requested. Algorithms are SRM functions performed on behalf of the system.	IRARMCTL	IRARMCEL
3 Request the invocation of time-driven algorithms which are now due. The queue of time-driven algorithms is scanned, and all algorithms which are due are requested by turning on representative bits in RMCTALR. SRM Control now branches to step 1 above. Continuing with step 2, SRM Control will route control to those time-driven algorithms which were requested.	IRARMCTL	IRARMCET
4 This SRM Control entry point receives control under an SRB which was scheduled by another component of SRM. The SRB was scheduled on behalf of routines not holding the SRM lock to execute critical actions and algorithms. Upon receiving control under the SRB, SRM Control makes a branch entry into the interface module, IRARMINT, to execute SYSEVENT 48. The SYSEVENT processor will in turn branch to SRM Control at step 1. Control will then be routed to the critical actions and algorithms which were requested.	IRARMCTL	IRARMCED

This blank leaf represents pages 3-26 - 3-27 which were deleted by Supervisor Performance #2.

Diagram 6-5. Deferred Action Processor (IRARMCEN) (Part 1 of 2)



1 Verify that more users remain on the deferred action queue.

2 Remove the next OUCB from the deferred action queue.

3 Route control to the action routines requested for this user.

Diagram 6-5. Deferred Action Processor (IRARMCEN) (Part 2 of 2)

Extended Description	Module	Label
The Deferred Action Processor routes control to each requested routine for all OUCBs on the deferred action queue. The entry point descriptors for all possible action routines are contained in RMCTEPDT,	IRARMCTL	IRARMCEN
1 If the action queue header is pointing to the dummy pre-assembled OUCB (that is, RMCTAQHD= RMCTOUCB), then the action queue is empty.	IRARMCTL	IRARMCEN
2 The top OUCB is dequeued via compare-and-double-swap, to prevent multi-processing interaction problems. OUCBACT is set to zero.	IRARMCTL	IRARMCEN
3 IRARMCRT scans the EPDT entry point table looking for entry point blocks (RMEPs) whose invocation flags match "one" bits in the input bit pattern. For each successful match, the corresponding entry point is invoked. The invocation bit of each routine invoked is set to zero in the input bit pattern. It is possible for an action routine to call another action routine. In this case, the new routine request is inserted into the OUCBACN field, to be picked up during the processing of this OUCB.	IRARMCTL	IRARMCRT

Diagram 6-6. Algorithm Processor (IRARMCEL) (Part 1 of 2)

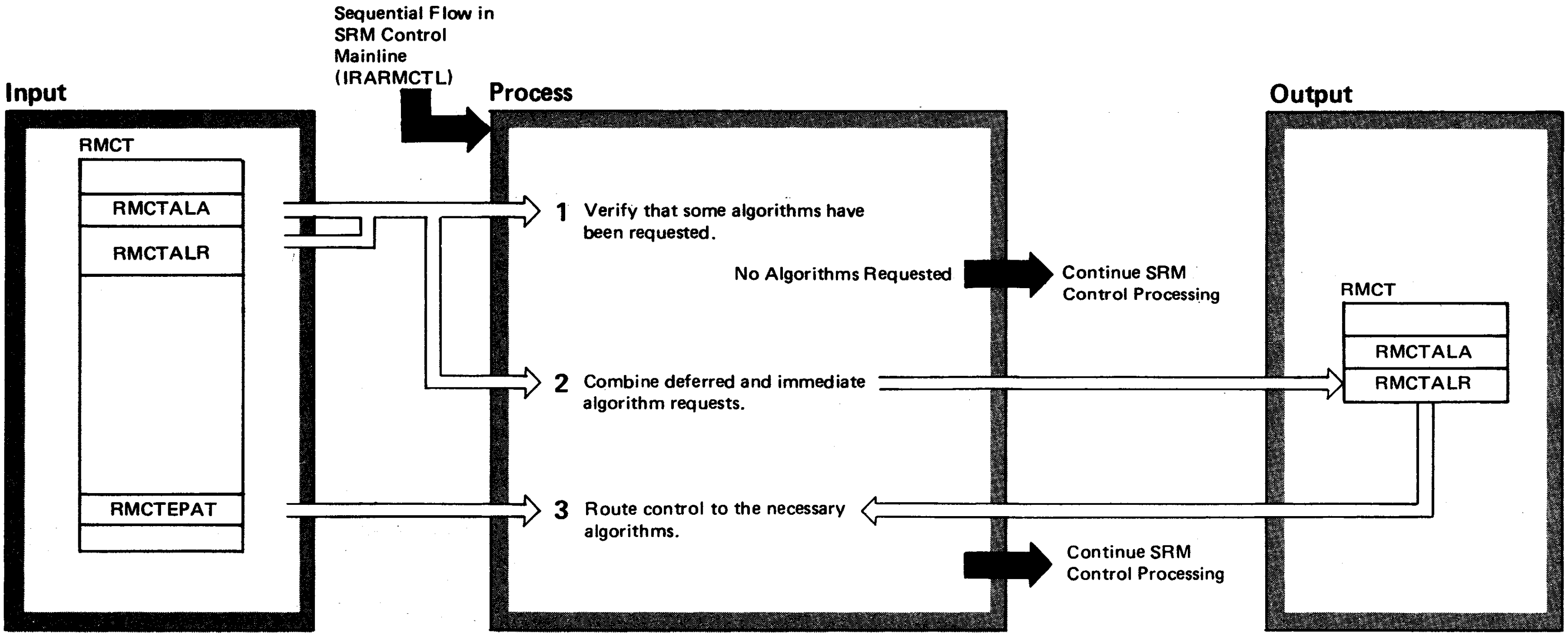
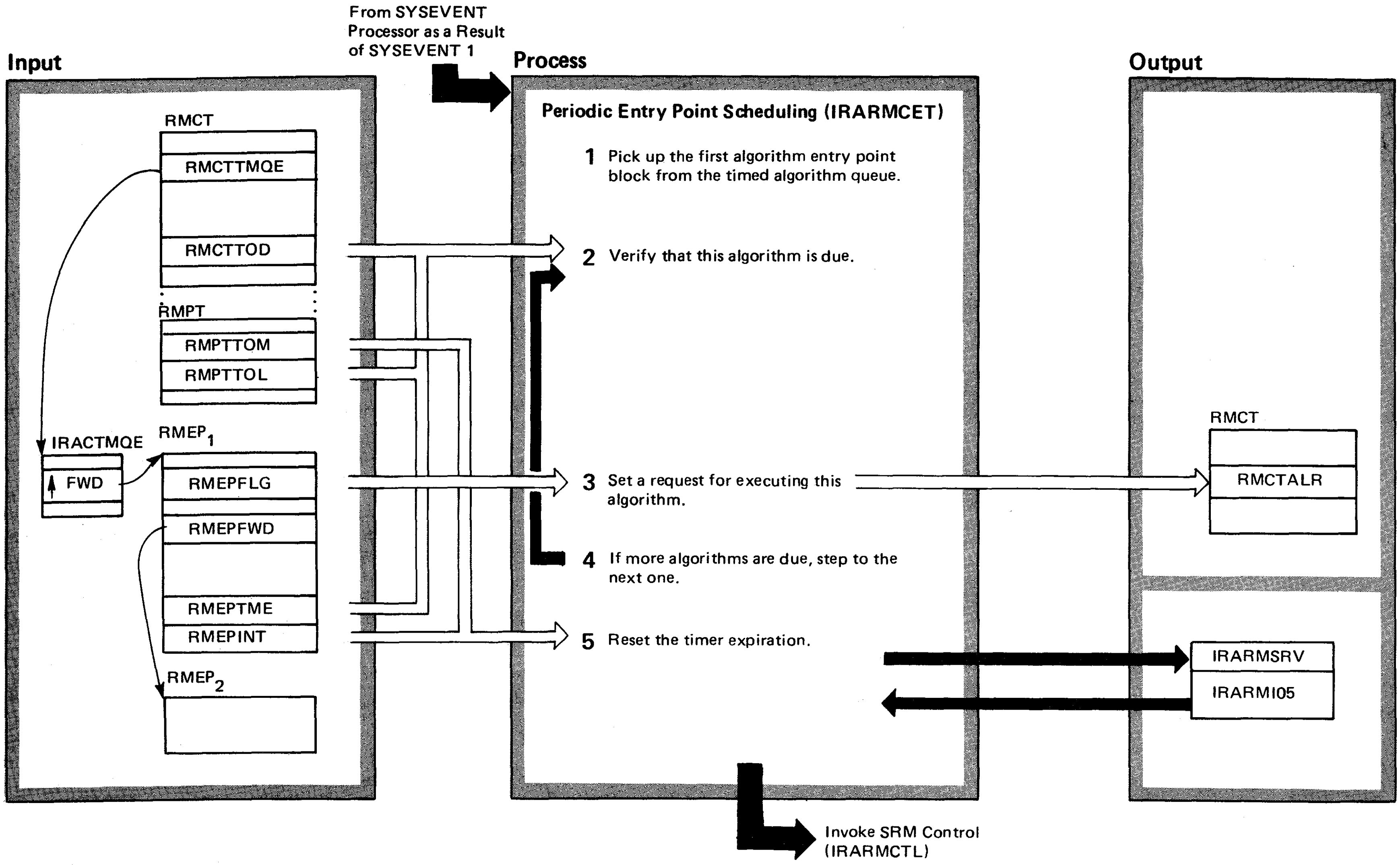


Diagram 6-6. Algorithm Processor (IRARMCEL) (Part 2 of 2)

Extended Description	Module	Label
Algorithm request routes control to all algorithms that have been requested and can now be executed. The entry point descriptors for all possible algorithm routines are contained in RMCTEPAT.	IRARMCTL	IRARMCEL
1 Some algorithms have been requested if RMCTALA and RMCTALR are not both zero. Algorithm requests are stored in RMCTALR by SRM locked routines, and in RMCTALA by SRM unlocked routines.	IRARMCTL	IRARMCEL
2 Compare and swap logic is used to insure that all current requests are obtained for a multiprocessing environment.	IRARMCTL	RMCELL1
3 IRARMCRT scans the EPAT entry point table looking for entry point blocks (RMEPs) whose invocation flags match "one" bits in the input bit pattern. For each successful match, the corresponding entry point is invoked. For each algorithm called, the invocation bit is set to "zero" in the request bit pattern. Input parameters: • reg. 1 – address of first entry point block (RMEP) in the EPAT chained table • reg. 6 – address of input bit pattern (RMCTALR)	IRARMCTL	IRARMCRT

Diagram 6-7. Periodic Entry Point Scheduling (IRARMCET) (Part 1 of 2)



Periodic Entry Point Scheduling (IRARMCET)

1 Pick up the first algorithm entry point block from the timed algorithm queue.

2 Verify that this algorithm is due.

3 Set a request for executing this algorithm.

4 If more algorithms are due, step to the next one.

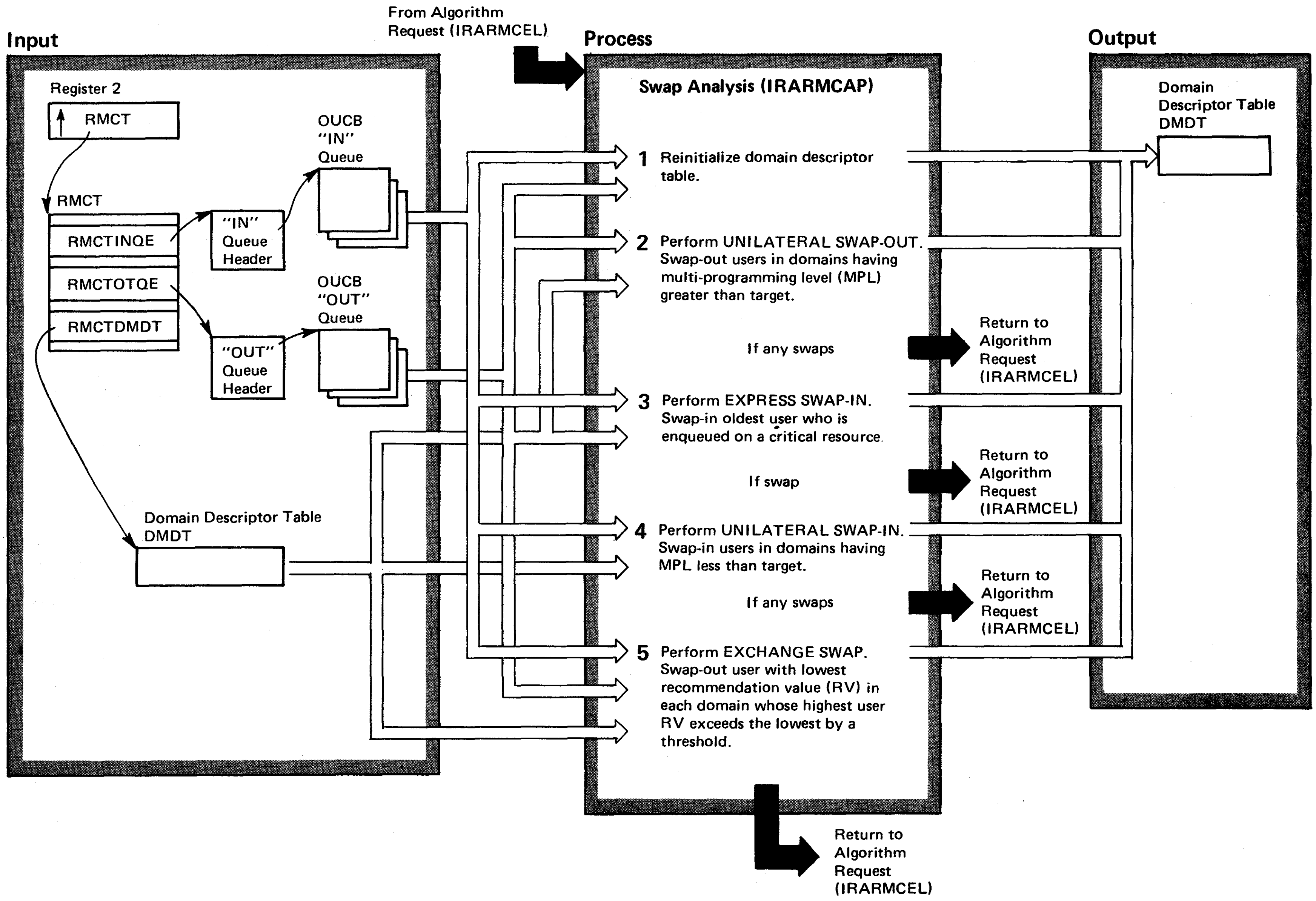
5 Reset the timer expiration.

Diagram 6-7. Periodic Entry Point Scheduling (IRARMCET) (Part 2 of 2)

Extended Description	Module	Label
Periodic Entry Point Scheduling is invoked following an SRM TQE timer expiration. It sets up requests for all SRM periodically scheduled algorithms which are then due. It also requests the resetting of the SRM TQE to cause an interruption when next required.	IRARMCTL	IRARMCET
1 The timer algorithm queue is ordered by the RMEPTME value of the RMEP blocks on the queue.	IRARMCTL	IRARMCET
2 An algorithm on the time-driven queue is "due" if the RMEPTME value is less than the current time (RMCTTOD) + an allowable tolerance (RMPTTOL).	IRARMCTL	IRARMCET
3 The algorithm request field is set up for later action by algorithm control routing (IRARMCEL).	IRARMCTL	IRARMCET
4 The next RMEP block is obtained from the queue.	IRARMCTL	IRARMCET
5 A new timer interruption is requested for the greater of: the minimum scheduling period (RMPTTOM), and the smallest time due of a scheduled routine (see SRM Interface M.O.).	IRARMSRV	IRARMI05

This blank leaf represents pages 3-34 - 3-35 which were deleted by Supervisor Performance #2.

Diagram 6-9. Swap Analysis (IRARMCAP) (Part 1 of 2)



1 Reinitialize domain descriptor table.

2 Perform UNILATERAL SWAP-OUT. Swap-out users in domains having multi-programming level (MPL) greater than target.

3 Perform EXPRESS SWAP-IN. Swap-in oldest user who is enqueued on a critical resource.

4 Perform UNILATERAL SWAP-IN. Swap-in users in domains having MPL less than target.

5 Perform EXCHANGE SWAP. Swap-out user with lowest recommendation value (RV) in each domain whose highest user RV exceeds the lowest by a threshold.

Diagram 6-9. Swap Analysis (IRARMCAP) (Part 2 of 2)

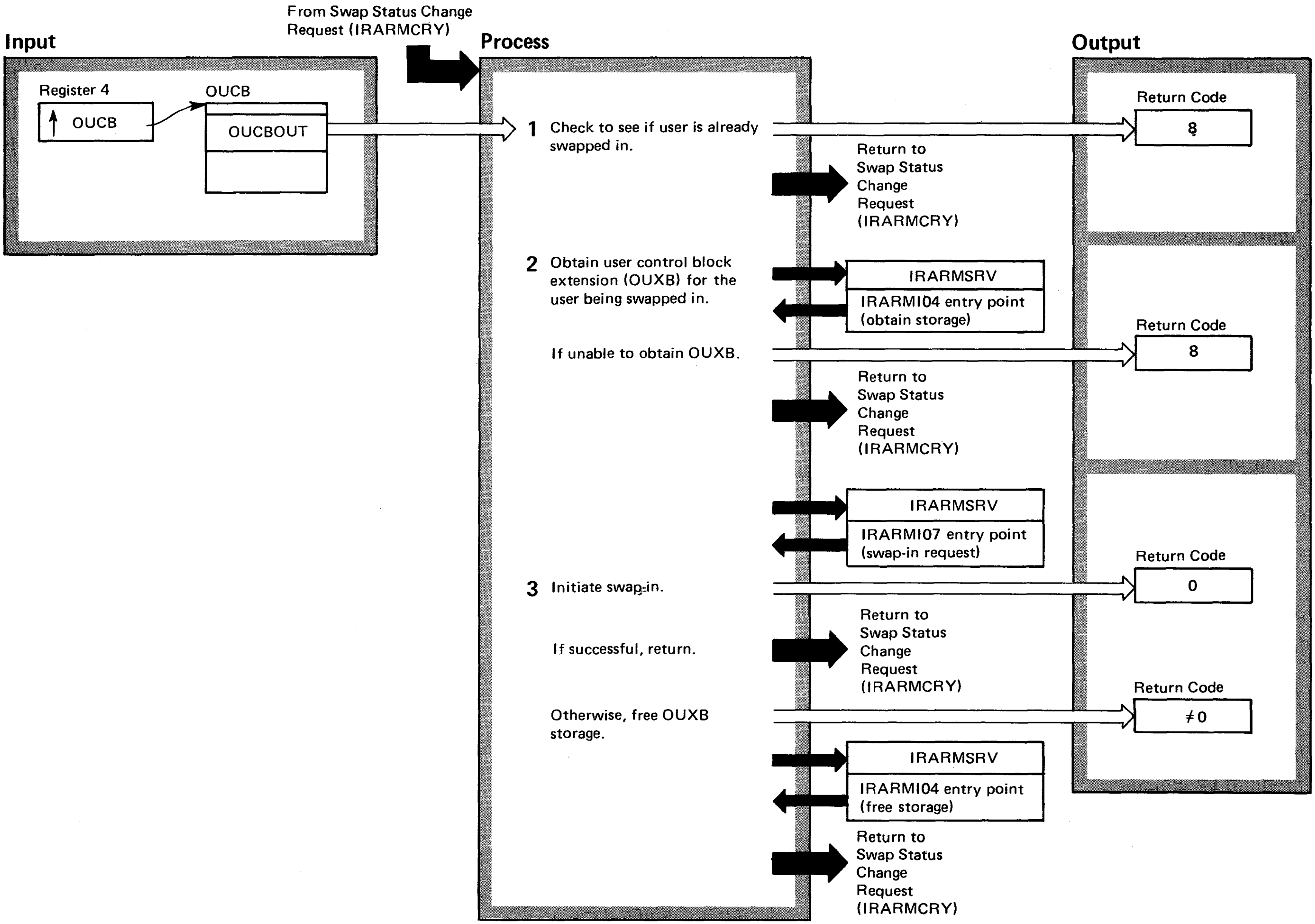
Extended Description	Module	Label (or Segment)
Swap Analysis is performed on a time-driven basis. It is an algorithm activated by IRARMCET. It is also activated by the processing of two SYSEVENTS: USERRDY (4) and SWOUTCMP (15).	IRARMCTL	IRARMCAP
1 The Domain Descriptor Table has one entry for each domain defined by the IPS. Each OUCB on the IN and OUT queues is examined. Swappable, valid users on the IN queue which are not in the process of being swapped out or moving from one SRM queue to another are counted in the current multiprogramming level (MPL) for a domain, as well as users on the OUT queue which are going in or moving from one SRM queue to another. Fields in each domain descriptor table entry are reinitialized with the above MPL count information.	IRARMCTL	IRARMCAP
2 Search the domain descriptor table entries for a domain with an MPL higher than the target value and swap out the user with lowest recommendation value (RV). Repeat until the MPL reaches the target value in every domain. If at least one swap is performed in this step, swap analysis ends here. Otherwise, continue at step 3.	IRARMCTL	IRARMCPO IRARMCSO
3 If there is a user on the OUT queue enqueued on a critical resource, attempt to swap the user in. If MPL in that domain is less than the target, swap that user in. Otherwise, make room for it by a swap out of the user with the lowest RV. Repeated calls to swap analysis may be necessary to reduce MPL below target value to allow the enqueued user to be swapped in. If there is no enqueued user, continue to step 4. Otherwise swap analysis ends here.	IRARMCTL IRARMCTL	IRARMCSI IRARMCPO IRARMCSO
4 Search the domain descriptor table entries for a domain with an MPL less than target and swap in user with highest RV. Repeat until the MPL (plus users in the process of being swapped out) reaches the target in each domain. If at least one swap is done in this step, swap analysis ends here.	IRARMCTL	IRARMCPI IRARMCSI
5 Search the domain descriptor table entries for a MPL that equals the target for that domain. In each of these domains, find the out-of-storage user with the largest RV to come in, and the in-storage user with the smallest RV to remain in. If the difference of their RVs exceeds a threshold (RMPTXCHT), swap out the user with the lower RV.	IRARMCTL	IRARMCPO IRARMCPI IRARMCSO

Error Processing:

- IRARMERR handles all unexpected errors.
- Any non-zero return codes from called routines causes Swap Analysis (IRARMCAP) to end without finishing its processing.

This blank leaf represents pages 3-38 - 3-39 which were deleted by Supervisor Performance #2.

Diagram 6-10. Control Swap-In (IRARMCSI) (Part 1 of 2)



1 Check to see if user is already swapped in.

2 Obtain user control block extension (OUXB) for the user being swapped in.

If unable to obtain OUXB.

3 Initiate swap-in.

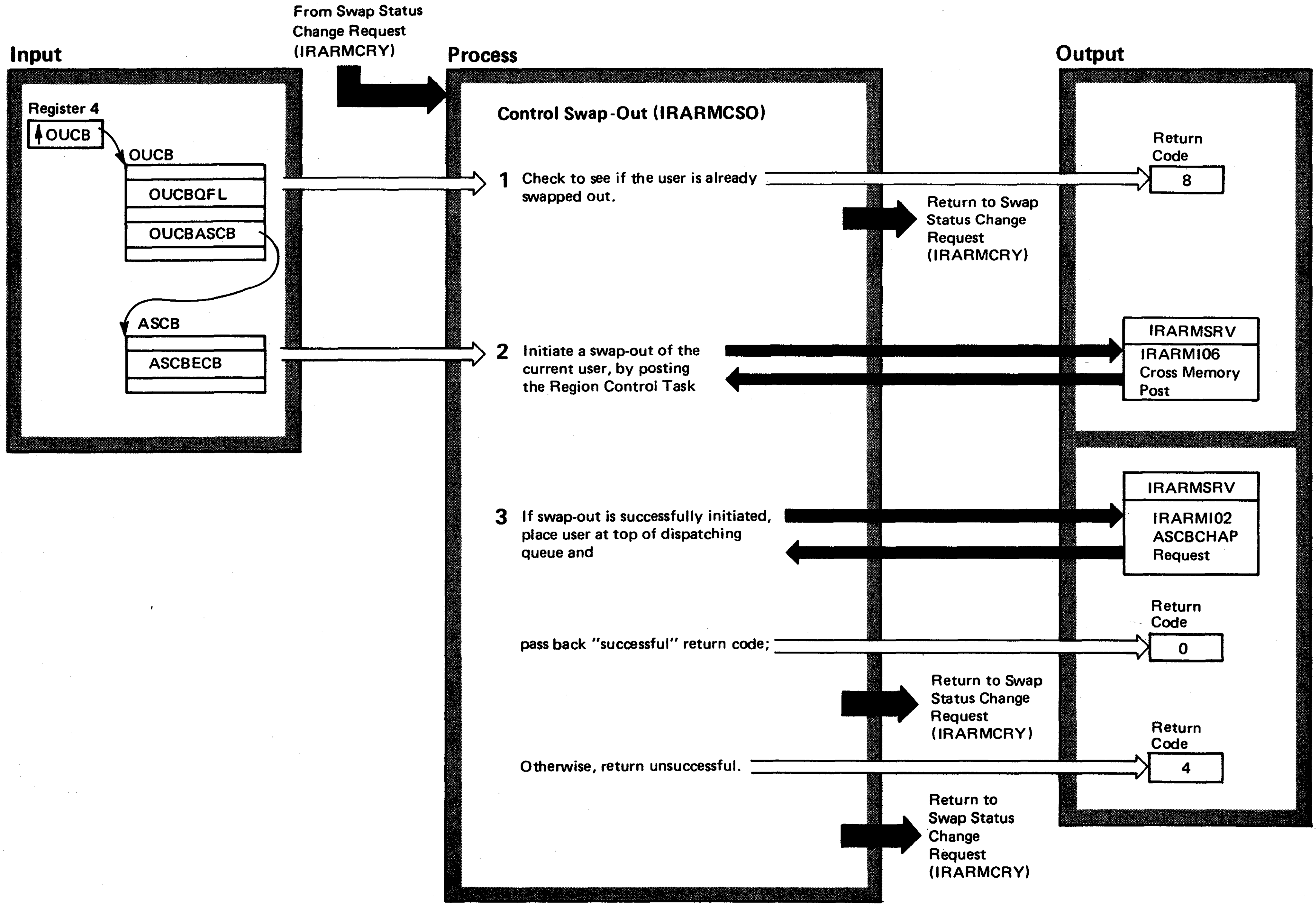
If successful, return.

Otherwise, free OUXB storage.

Diagram 6-10. Control Swap-In (IRARMCSI) (Part 2 of 2)

Extended Description	Module	Label (or Segment)
Control Swap-In accepts a request that an address space be swapped in. If the address space is already swapped in, this is indicated by a return code; if not, control swap-in initiates a swap-in of the address space.	IRARMCTL	IRARMCSI
1 Control swap-in returns to the calling routine with a return code of 8 if the user for which a swap-in has been requested has already been swapped-in. Otherwise, control goes to step 2.	IRARMCTL	IRARMCSI
2 The user control block extension (OUXB) is obtained. It remains in existence as long as the user is swapped in and is released at swap-out.	IRARMSRV	IRARMI04
3 If the swap-in is successfully initiated (return code from IRARMI07 equals 0), the OUXB is cleared, the address of the OUXB is placed into the ASCB (ASCBOUXB), and the OUCB going-in bit is set (OUCBGOI).	IRARMSRV	IRARMI07
Otherwise, the storage for the OUXB is freed.	IRARMSRV	IRARMI04
Error Processing:		
If an attempt to obtain storage for an OUXB fails (step 2), or an attempt to initiate a user swap-in fails (step 3), the user remains on the OUT queue, and Control Swap-in returns to the caller with an error return code.	IRARMCTL	IRARMCSI

Diagram 6-11. Control Swap-Out (IRARMCSO) (Part 1 of 2)



1 Check to see if the user is already swapped out.

2 Initiate a swap-out of the current user, by posting the Region Control Task

3 If swap-out is successfully initiated, place user at top of dispatching queue and

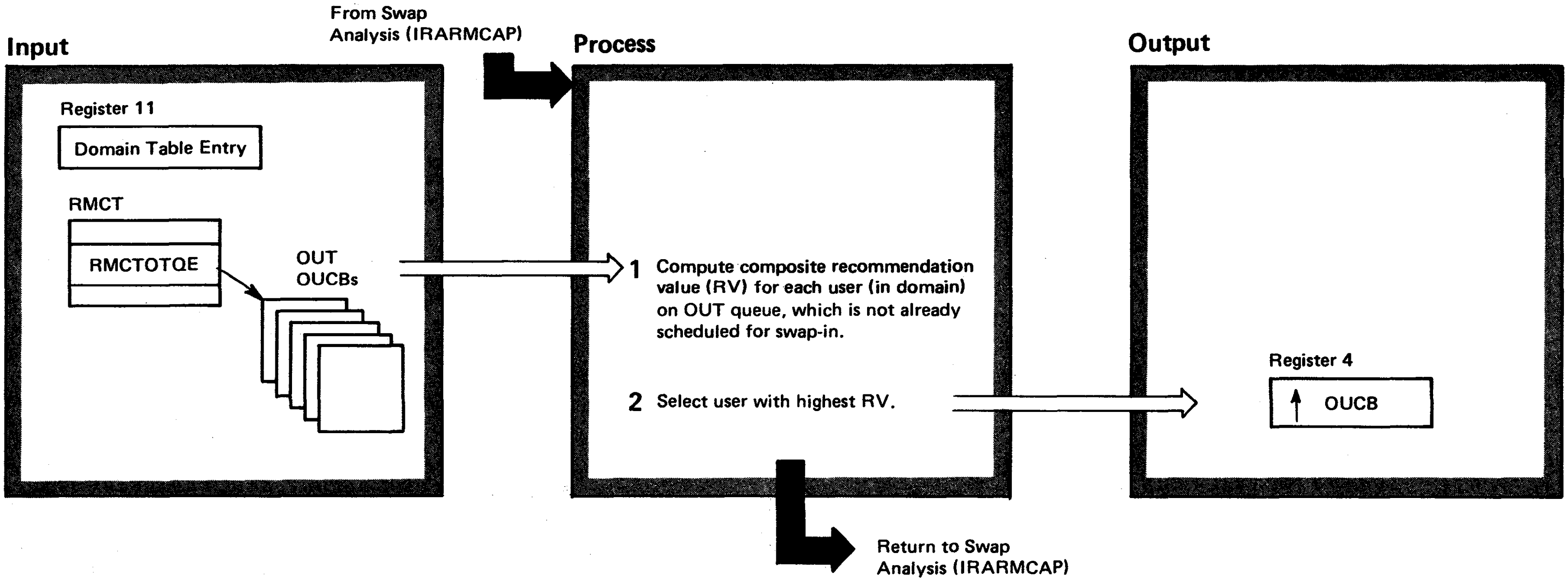
pass back "successful" return code;

Otherwise, return unsuccessful.

Diagram 6-11. Control Swap-Out (IRARMCSO) (Part 2 of 2)

Extended Description	**Module**	**Label (or Segment)**
Control Swap-Out accepts a request that an address space be swapped out. If the address space is already swapped-out, this is indicated by a return code; if not, control swap-out initiates the swap-out of the address space.	IRARMCTL	IRARMCSO
1 Control swap-out returns to the calling routine if the user for which a swap-out has been requested has already been swapped out. Otherwise, control goes to step 2.	IRARMCTL	IRARMCSO
2 The supervisor service request routine requests the initiation of quiesce processing for the user to be swapped out. This request results in the posting of an ECB for the indicated address space, so that the RCT will begin quiesce processing.	IRARMSRV	IRARMI06
3 To expedite quiesce processing, request that the user's ASCB be moved.	IRARMSRV	IRARMI02
A successful return indicates that the post of quiesce processing has been scheduled for the address space. The progress of quiesce processing will be indicated to the SRM by future SYSEVENTs (typically, quiesce started, followed by quiesce completed, followed by swap-out complete).		

Diagram 6-11A. Select User for Swap-In (IRARMCPI) (Part 1 of 2)



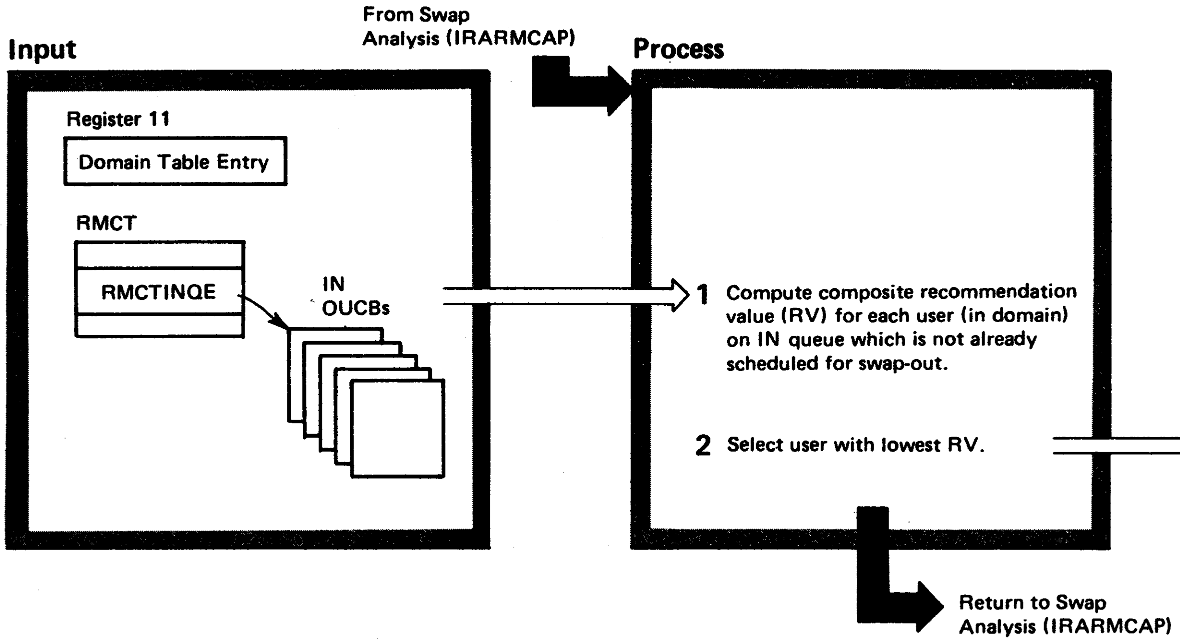
1 Compute composite recommendation value (RV) for each user (in domain) on OUT queue, which is not already scheduled for swap-in.

2 Select user with highest RV.

Diagram 6-11A. Select User for Swap-In (IRARMCPI) (Part 2 of 2)

Extended Description	Module	Label
This routine chooses the user with the highest RV in a particular domain on the OUT queue. If one of the users represented by an OUCB in this domain is assigned to a different domain, for example, because of a period change, return a code of zero indicating no user found. In this case, swap analysis (IRARMCAP) is rescheduled to ensure that the domain descriptor table is initialized to reflect this domain change. The following two steps are performed in a loop until all OUCBs on the OUT queue have been evaluated.	IRARMCTL	IRARMCPI
1 Examine each OUCB on the OUT queue for users in the specified domain. Use the user evaluation subroutine to compute the composite RV for each user.	IRARMCTL IRARMCTL	IRARMCPI IRARMCVL
2 Compare the computed RV to that of the highest RV found up till now. Save this OUCB as the best candidate for a swap-in if its RV is greater. Otherwise, continue until all OUCBs on the OUT queue in this domain have been evaluated.	IRARMCTL	IRARMCPI

Diagram 6-11B. Select User for Swap-Out (IRARMCPO) (Part 1 of 2)



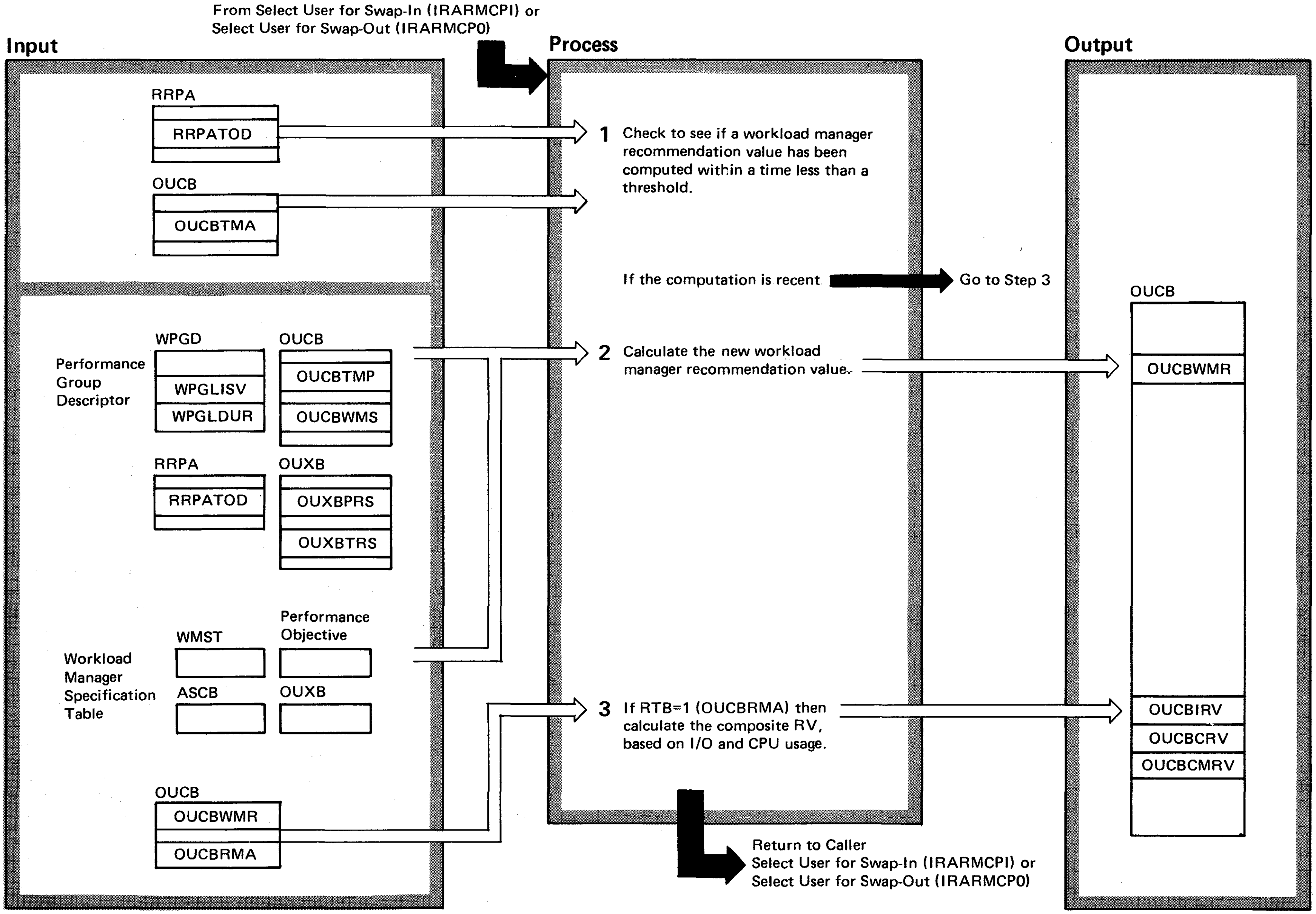
1 Compute composite recommendation value (RV) for each user (in domain) on IN queue which is not already scheduled for swap-out.

2 Select user with lowest RV.

Diagram 6-11B. Select User for Swap-Out (IRARMCPO) (Part 2 of 2)

Extended Description	Module	Label
This routine chooses the user with the lowest RV in a particular domain on the IN queue. If one of the users represented by an OUCB in the domain is assigned to a different domain, for example, because of a period change, return a code of zero indicating no user found. In this case swap analysis (IRARMCAP) is rescheduled to ensure that the domain descriptor table is initialized to reflect this domain change.	IRARMCTL	IRARMCPO
The following two steps are performed in a loop until all OUCB's on the IN queue have been evaluated.		
1 Examine each OUCB on the IN queue for users in the specified domain. Use the user evaluation subroutine to compute the composite RV for each user.	IRARMCTL IRARMCTL	IRARMCPO IRARMCVL
2 Compare the computed RV to that of the lowest RV up till now. Save this OUCB as the best candidate for a swap-out if its RV is lower.	IRARMCTL	IRARMCPO
Otherwise, continue until all OUCBs in this domain on IN queue have been evaluated.		

Diagram 6-11C. User Evaluation (IRARMCVL) (Part 1 of 2)



1 Check to see if a workload manager recommendation value has been computed within a time less than a threshold.

If the computation is recent → Go to Step 3

2 Calculate the new workload manager recommendation value.

3 If RTB=1 (OUCBRMA) then calculate the composite RV, based on I/O and CPU usage.

Diagram 6-11C. User Evaluation (IRARMCVL) (Part 2 of 2)

Extended Description	**Module**	**Label**
User evaluation computes a recommendation value (RV) for one user based on its workload manager recommendation value and its I/O and CPU recommendation values.	IRARMCTL	IRARMCVL
1 A new value is calculated for the workload manager RV only if sufficient time has elapsed since its previous calculation. This time is called threshold 2. (Swap Analysis evaluating threshold RMPTSAET).	IRARMCTL	IRARMCVL
2 Compute the workload manager recommendation value (the normalized workload level) representing the desirability of a swap of this user. This value is based on the rate at which he has recently been receiving service and on the IPS.	IRARMWLM	IRARMWM3
3 If the applicable RTB is 1, add to the workload level an I/O manager recommendation value (for significant users of I/O) and add a CPU manager recommendation value (for significant users of the CPU resource). A positive RV favors the swap-in of a user to correct a CPU or I/O imbalance and a negative RV favors the swap-out of a user.	IRARMIOM IRARMCPM	IRARMIL3 IRARMCL3

IRARMCTL Module Entry Point Summary

IRARMCTL - Mainline Control Processing. Transfers to deferred user action processing (IRARMCEN) and then to the algorithm request routine (IRARMCEL).

IRARMCEN - Deferred User Action Processing. Examines the OUCBACN field of the OUCB to determine the users on the action queue and routes control to all routines whose request bits have been set in the OUCBACN field. Dequeues each OUCB after its indicated actions have been performed.

IRARMCEL - Algorithm Request Routine. Examines the RMCTALR and RMCTALA fields in the RMCT and routes control (via IRARMCRT) to each algorithm whose request bit has been set in either of the two fields. Resets the individual request bit after each algorithm completes.

IRARMCET - Periodic Entry Point Scheduler. Accepts timer interrupts, schedules the algorithm currently due for execution and requeues the SRM timer element to permit interrupts again when the next algorithm is due for execution.

IRARMCED - SRB Dispatched Original Entry Processor. Receives control under an SRB scheduled by the dispatcher and sets up an entry to the mainline of SRM(IRARMCEN) by issuing SYSEVENT 48.

IRARMCQT - Periodically-Invoked Entry Point Rescheduler. Accepts a request to reschedule the execution of a periodically invoked algorithm and requeues the corresponding RMEP block on the timed entry queue.

IRARMCRD - SRB Scheduling Routine. Accepts a request to schedule the SRM SRB which if available is scheduled to obtain the SRM lock.

IRARMCRL - Algorithm Scheduling Routine. Accepts requests for an algorithm to be run. Turns on the bit associated with the algorithm in the RMCTALA or RMCTALR.

IRARMCRN - Action Request Routine. Accepts requests for an action which must run under the SRM lock. If the SRM lock is held, control passes immediately to the action via a routing routine. If the SRM lock is not held, the bit is set in the OUCBACN field of the OUCB associated with the requesting user that identifies that the action requested is deferred.

IRARMCRT - Entry Point Table Scanner. Accepts an invocation bit pattern and an entry point table address. Compares the bit pattern to invocation flags in the entry point table entries. Invokes the routine identified by the entry point when a match is found between the bit pattern and the invocation flags.

IRARMCRY - User Swap Request Receiving Routine. Accepts a request for a user swap and checks to see if such a swap is already in progress. Routes control to IRARMCSO or IRARMCSI if a swap is not in progress and the SRM lock is held.

IRARMCSI - User Swap-In Request. Accepts a swap-in request, allocates an OUXB for the user and initiates the swap-in.

IRARMCSO - User Swap-out Request. Accepts a swap-out request and posts the region control task's quiesce routine to initiate the swap-out.

IRARMRPS - OUCB Repositioning Routine. Dequeues an OUCB and requeues it at the end of the queue specified in its OUCBQFL field.

IRARMWMY - Periodic Entry Point Requeuing Routine. Requeues all of the members on the Timed Algorithm Queue and adjusts all the time-due fields.

IRARMCAP - Swap Analysis Algorithm. Attempts to keep the multiprogramming level (MPL) at its target level in each domain by performing user swaps.

IRARMCPI - Select Swap-In Candidate Subroutine. Scans the OUT queue for the user in a particular domain with the highest recommendation value.

IRARMCPO - Select Swap-Out Candidate Subroutine.
Scans the IN queue for the user in a particular domain with the lowest recommendation value.

IRARMCVL - User Swap Evaluation Routine.
Computes a numerical value representing the recommendation of a user to be swapped in. This recommendation value is the sum of the user's workload level and the recommendations of the I/O and CPU resource managers.

Resource Use Algorithms

The resource management algorithms are concerned with improving overall system resource utilization. These include:

Storage Management

- Page Replacement - This function maintains an up-to-date indication of which frames have gone unreferenced for the longest period of real time and the age of the oldest unreferenced frame in the system. This is accomplished by invoking the real storage manager's (RSM) routine IEAVRFR, real frame replacement (RFR), at periodic real time intervals so that RFR may increment the unreferenced interval count (UIC) for those unreferenced since the last RFR invocation. If RFR finds that an allocated frame was referenced in the last interval it resets the UIC to zero. When the page replacement algorithm completes updating the UIC's for all allocated frames it then saves the highest UIC in the system for use by the real page shortage prevention algorithm and the resource monitor algorithm.
- Real Page Shortage Prevention - This function is invoked by SRM when the available frame queue falls below the available frame queue LOW threshold (PVTAFCLO) so that SRM can take action to remedy the existing real page shortage. When the real page shortage prevention algorithm is notified of a real page shortage it will steal frames from all users and the system pageable area (SPA). It steals the oldest unreferenced allocated frames in the system starting with the highest UIC as saved by the page replacement algorithm until the count of available frames plus the count of the pages stolen exceeds the available frame queue OK threshold. (PVTAFCOK).
- Auxiliary Slot Shortage Prevention - This function is invoked at periodic intervals to check for two levels of auxiliary slot shortages. Reaching the first level threshold causes the creation of memories to be prevented, the swap-out of the batch user who is acquiring auxiliary storage slots at the greatest rate, the delay of newly initiated jobs, and the setting of all domains to their minimal MPL. Messages are written to the operator indicating the occurrence of either of the shortages and the jobnames of the users swapped out because of the shortage. Additionally, when this function determines that the auxiliary slot shortage is relieved a message is written to the operator indicating the alleviation of the slot shortage. Creation of memories is again allowed and those memories that were swapped out are again made eligible for swap-in.
- SQA Shortage Prevention - This function is invoked by the virtual storage manager (VSM) when a shortage of system queue area (SQA) space is detected. This function will then prohibit the creation of memories for the duration of the SQA shortage and notifies the operator of the existence and severity of the shortage. Also, a message is written to the operator when VSM informs this function that the SQA shortage has been relieved and the creation of memories is again permitted.
- Pageable Real Storage Shortage Prevention - This function is invoked by the real storage manager (RSM) when the percentage of fixed frames to total frames exceeds a predefined limit. This function will then prohibit the creation of memories, initiate a swap-out for the swappable user who has allocated the greatest number of fixed frames, delay newly initiated jobs, and set all domains to their minimal MPL. The operator is notified of the existence and severity of the pageable storage shortage and of the identity of the swapped users. Additionally, when this function determines that the shortage has been relieved, a message is written to the operator indicating the alleviation of the shortage. Creation of memories is again allowed, and those memories that were swapped-out are again made eligible for swap-in.

I/O Management

- Device Allocation - This function makes device allocation decisions, based on I/O load balancing considerations when a choice must be made from more than one device candidate. The device allocation decision is made by applying the following rules to the list of candidates (in the order indicated):

  1. If the request is for tape, eliminate all candidates on ready devices (eliminate premounted tape drives) and on devices that contain passed volumes.

2. Choose the candidates on the logical channel with the lowest utilization. The utilization takes into account any datasets previously allocated to the logical channel.
3. Choose the direct access candidates with the lowest allocated user counts.
4. From a list of equal candidates, choose one at random.
5. Insure that the selected candidate device has not been previously allocated to the same user.

I/O Load Balancer Swap Analysis - Consists of a set of routines that monitor I/O logical channel usage of certain users. Users are recommended for swapping based on the extent to which the swap-in or swap-out of a user would correct a detected I/O system imbalance. The I/O load balancer recommendation is scaled so that it will never be greater than one-fifth the highest possible workload level possible with the IPS currently in effect. This recommendation will then be multiplied by the IOC resource factor coefficient (RFC) as specified in parmlib member, IEAOPTXX.

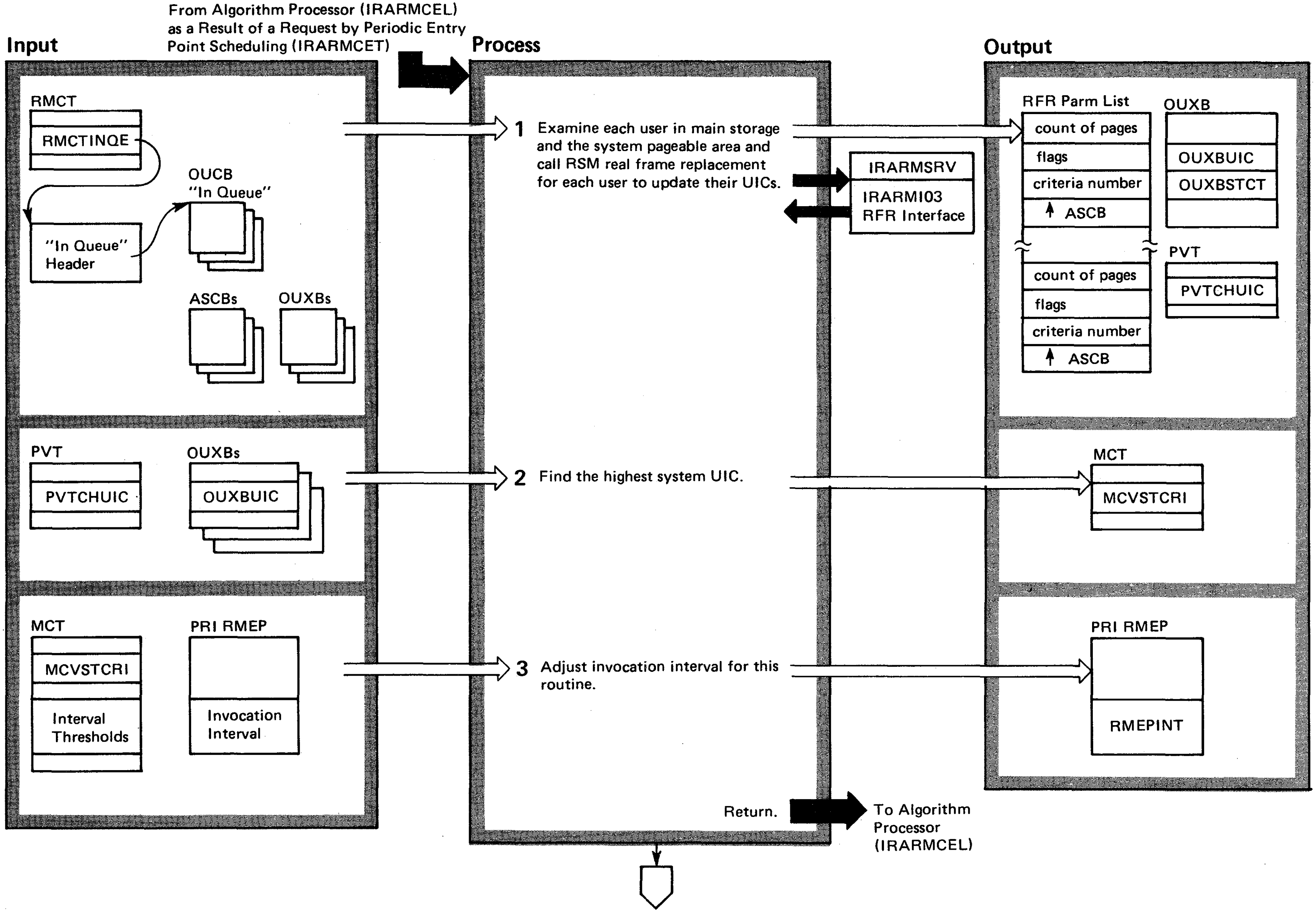
CPU Management

- Automatic Priority Group (APG) Management - Records a subset (16 dispatching priorities) of the ASCB dispatching queue. The APG contains three groups. Users in one group will have their dispatching priorities based on the user's mean execution time before entering a wait state. This wait is defined as any time a task issues WAIT, goes into page wait, or enters terminal wait and there are no other ready tasks in this address space. Users who quickly release the CPU are given a high dispatching priority within the subset. A second group contains users having fixed priorities. A third group consists of users at one rotate priority where each user at this priority is periodically moved to the top of the priority group.
- ENQ/DEQ Algorithm - Inhibits the swap-out of users who are in control of (enqueued upon) resources in demand by other system users. Swap-out is prevented until the resource is released or the user has executed for the interval specified in the enqueue residence value (ERV) installation tuning parameter.
- CPU Load Balancer Swap Analysis - Consists of a set of routines that monitor the system-wide CPU load. They recommend users for swapping when the system is under-or over-utilized, and when users exist who would improve the imbalance by being swapped in or out. The CPU load balancer's recommendation is scaled so that it will never be greater than one-fifth the highest possible workload level possible with the IPS currently in effect. This recommendation will then be multipled by the CPU resource factor coefficient (RFC) as specified in the parmlib number, IEAOPTXX.

Resource Monitor

- Resource Monitor - This function monitors several system resource usage indicators. After a number of sample intervals have completed, the resource monitor will average the resource usage over the sample intervals, and based on resource usage thresholds, will recommend to the domain MPL adjustment routine that the system multi-programming level (MPL) be either raised, lowered, or remain the same.
- Domain MPL Adjustment Routine - This routine will raise or lower individual domain multi-programming levels based on input from the resource monitor and the domain weight factor as provided in the domain descriptor in the IPS.

Diagram 6-12. Storage Management (IRARMSTM) (Part 1 of 8)



1 Examine each user in main storage and the system pageable area and call RSM real frame replacement for each user to update their UICs.

2 Find the highest system UIC.

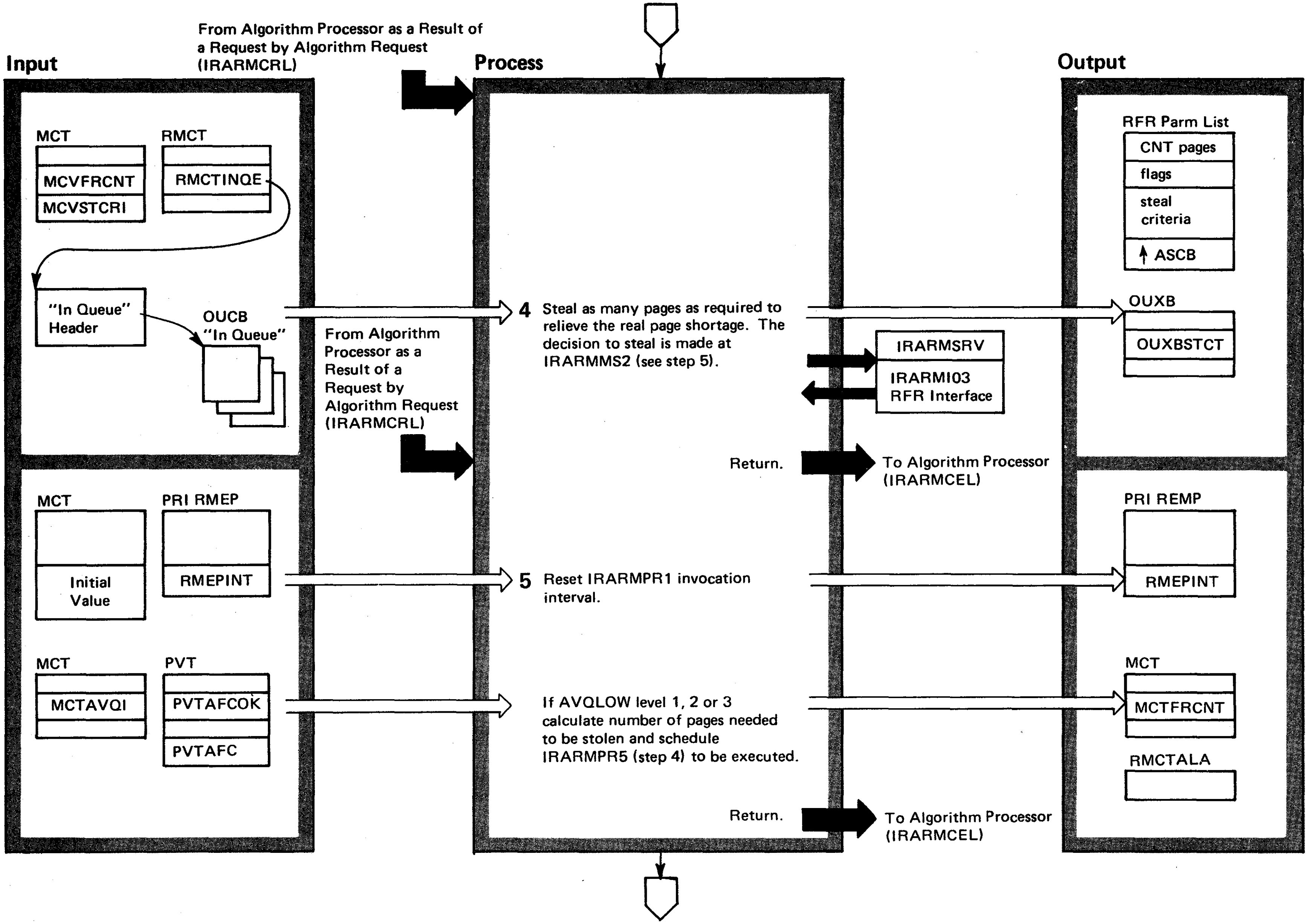
3 Adjust invocation interval for this routine.

Return.

Diagram 6-12. Storage Management (IRARMSTM) (Part 2 of 8)

Extended Description	Module	Label
Storage Management consists of essentially independent routines that are invoked by SRM Control or by the SYSEVENT Processor to control the usage of main and auxiliary storage by all users. For non-swappable users (users whose fixed storage must remain in real storage), the mechanism of page stealing (freeing non-fixed pages for other use) is used for storage management control. For swappable users, both page-stealing and swapping provide the necessary control.	IRARMSTM	
1 For each user in main storage and for the system pageable area build an entry in the parameter list for RSM's real frame replacement routine, IEAVRFR. In these entries indicate that the unreferenced interval counts are to be updated. RSM examines the UICs associated with each of the user's pages. If the page reference bit is on, the UIC for this page is set to zero, and the reference bit reset. If the reference bit is off, the corresponding UIC is incremented by one. RSM then saves the highest UIC count for each user in the corresponding OUXB. The highest UIC for the system pageable area is saved in PVTCHUIC.	IRARMSTM IRARMSRV	IRARMPR1 STEAL IRARMI09
2 The highest UIC found in any memory or system pageable area is identified and saved in MCVSTCRI. This value will be used by the force steal routine (Step 4) as the criteria at which RSM will begin stealing pages.	IRARMSTM	IRARMPR1
3 If the highest UIC found in any memory or system pageable area, MCVSTCRI, is greater than a threshold (MCCUICBD), the invocation interval for IRARMPR1 is incremented.	IRARMSTM	IRARMPR1

Diagram 6-12. Storage Management (IRARMSTM) (Part 3 of 8)



4 Steal as many pages as required to relieve the real page shortage. The decision to steal is made at IRARMMS2 (see step 5).

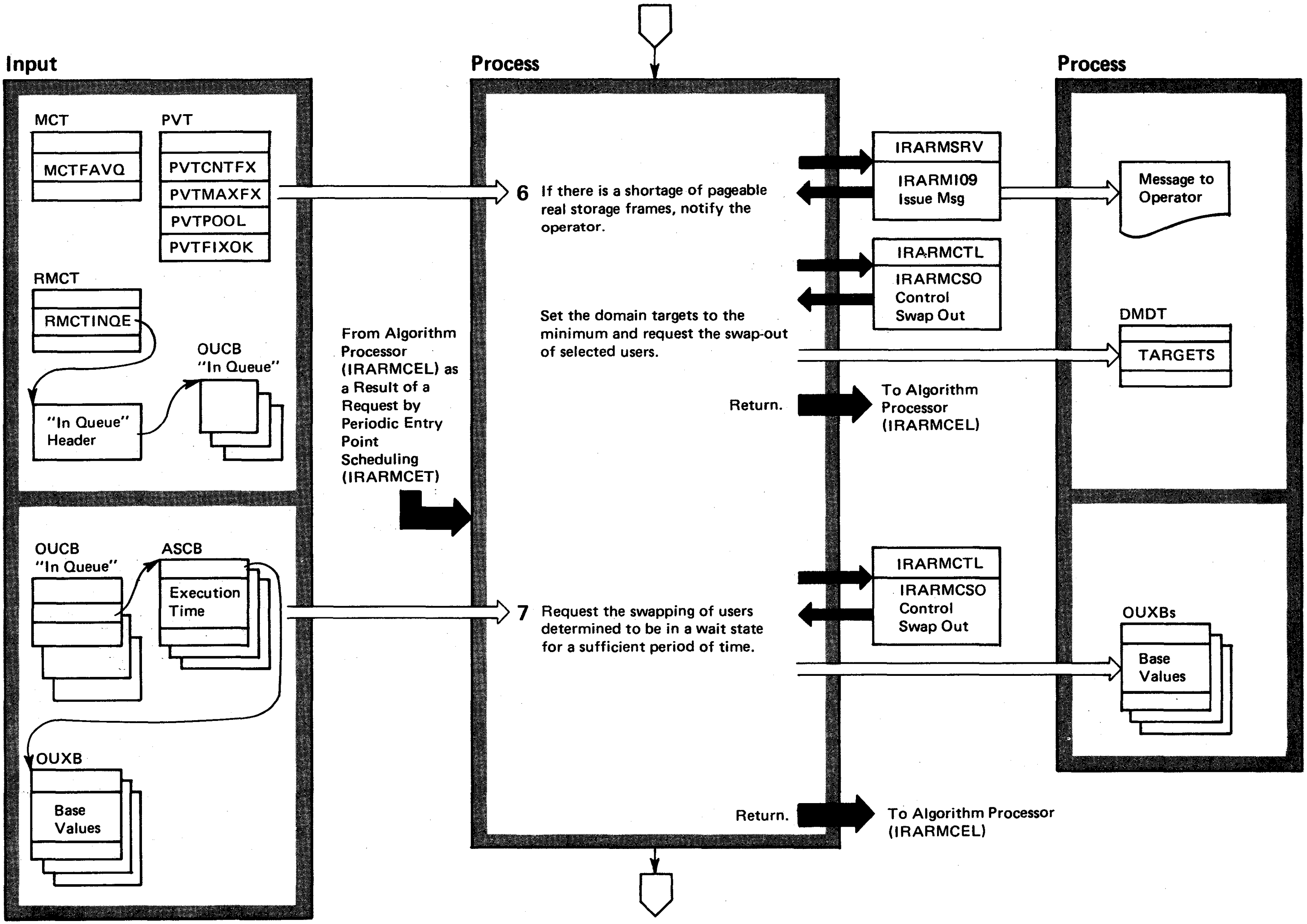
5 Reset IRARMPR1 invocation interval.

If AVQLOW level 1, 2 or 3 calculate number of pages needed to be stolen and schedule IRARMPR5 (step 4) to be executed.

Diagram 6-12. Storage Management (IRARMSTM) (Part 4 of 8)

Extended Description	Module	Label
4 The Real Page Shortage force steal routine is a co-operative effort between the SRM and RSM. SRM calls the RSM routine, IEAVRFR, passing a parameter list identifying an in storage user, the number of pages needed and a steal criteria number, MCVSTCRI. All pages associated with this user that have a UIC greater than the steal criteria are no longer considered part of the user's working set and are stolen. If not enough pages were stolen, another user is identified and IEAVRFR will steal all pages associated with this user that have a UIC greater than the steal criteria. If, after all eligible users have been stolen from at this criteria, pages are still required, the steal criteria will be decremented by one and the process repeated until no further pages are required. The result of this procedure is that the oldest unreferenced system wide pages are stolen first. Pages in the common system area and link pack area are not associated with any specific user. RSM examines these pages when SRM page replacement calls it with an ASCB address of zero.	IRARMSTM IRARMSRV	IRARMPR5 STEAL IRARMI03
5 The SRM will have received an AVQLOW SYSEVENT if there is a shortage of real page frames. Reset the IRARMPR1 interval back to its original value. If the invocation is due to AVQLOW level 1, 2 or 3 (real page shortage) calculate the number of pages needed to get the available frame queue back to the OK threshold and invoke the forced steal algorithm via the algorithm request mechanism.	IRARMSTM IRARMCTL	IRARMMS2 IRARMCRL

Diagram 6-12. Storage Management (IRARMSTM) (Part 5 of 8)



From Algorithm Processor (IRARMCEL) as a Result of a Request by Periodic Entry Point Scheduling (IRARMCET)

6 If there is a shortage of pageable real storage frames, notify the operator.

Set the domain targets to the minimum and request the swap-out of selected users.

Return. To Algorithm Processor (IRARMCEL)

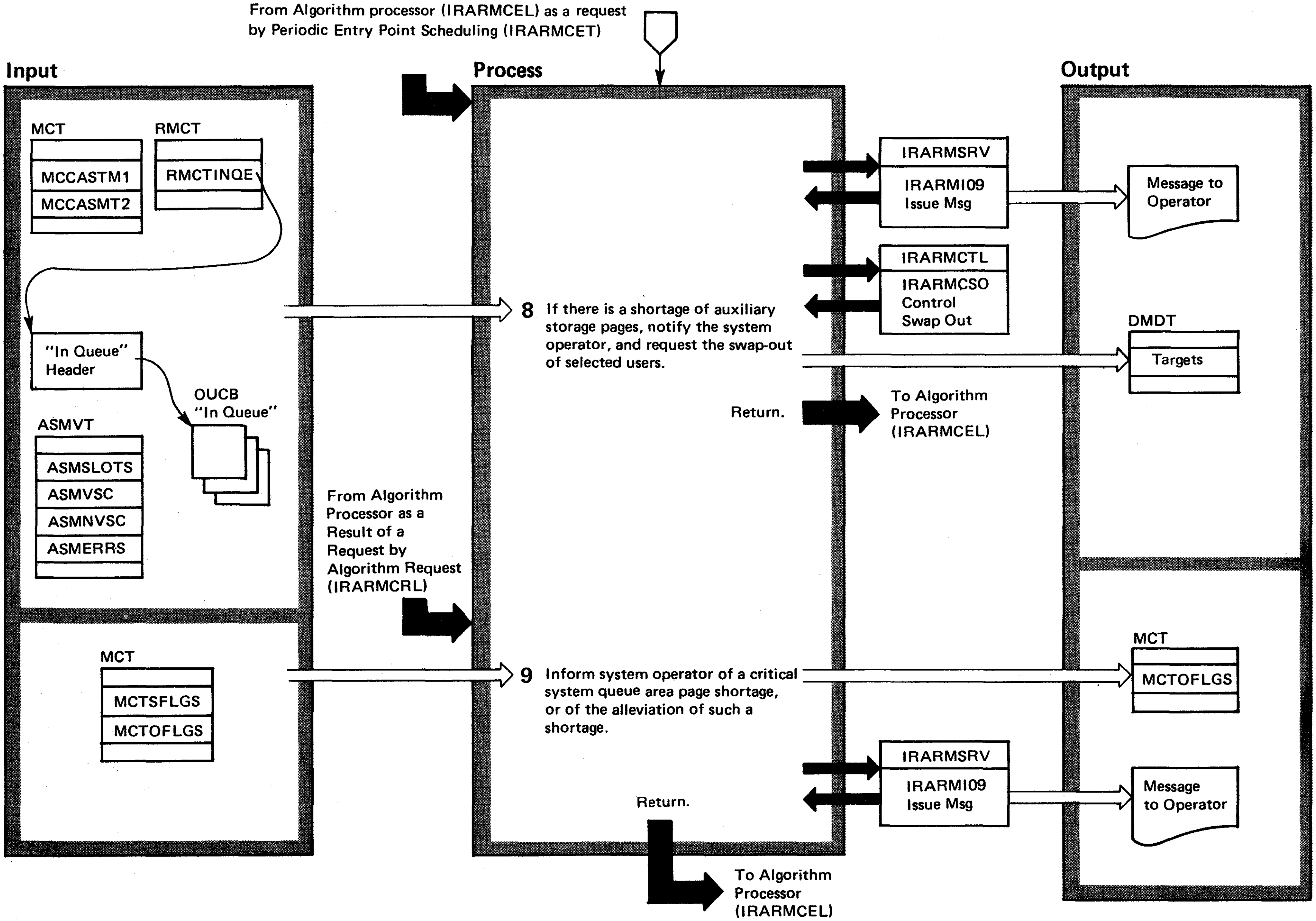
7 Request the swapping of users determined to be in a wait state for a sufficient period of time.

Return. To Algorithm Processor (IRARMCEL)

Diagram 6-12. Storage Management (IRARMSTM) (Part 6 of 8)

Extended Description	Module	Label
6 Pageable Real Frame Shortage, indicated by AVQLOW Level 4, checks for two levels of shortages. A first level shortage causes the prevention of further memory creates, the setting of domain targets to minimums, the delay of newly-initiated jobs and the swap out of the user which has the greatest number of fixed frames. When the second level is reached, another swappable user with the greatest number of fixed frames is also swapped-out. Messages indicating the occurrence of both levels and a message identifying the users swapped are written to the console. A message is also written indicating the alleviation of the shortage.	IRARMSTM IRARMSRV	IRARMMS2 IRARMI09
7 Users who issue a long wait macro instruction will be detected by the SRM when the wait macro processor issues the NIOWAIT SYSEVENT. Users who do not issue a long wait macro instruction to notify the SRM that they will be in the wait state for a "long" time will be detected when they have gone without executing for a sufficient period. At this time, swappable users will be swapped-out.	IRARMSTM	IRARMMS6

Diagram 6-12. Storage Management (IRARMSTM) (Part 7 of 8)



8 If there is a shortage of auxiliary storage pages, notify the system operator, and request the swap-out of selected users.

9 Inform system operator of a critical system queue area page shortage, or of the alleviation of such a shortage.

Diagram 6-12. Storage Management (IRARMSTM) (Part 8 of 8)

Extended Description	Module	Label
8 Auxiliary Storage Shortage Monitoring checks for two levels of auxiliary storage page shortages. The first level shortage causes: the prevention of memory creates, the setting of domain targets to minimums, the swapout of the swappable user who is acquiring auxiliary storage pages at the greatest rate, and the delay of newly-initiated jobs. Messages indicating the occurrence of either shortage level and the users swapped due to the shortage are written to the console; likewise messages are written indicating the alleviation of shortages.	IRARMSTM IRARMCTL IRARMSRV	IRARMASM IRARMCSO IRARMI09
9 The system queue area message writer is invoked by SYSEVENT SQALOW or SQAOK to write shortage messages to the system operator. The messages cannot be written directly by the invoking routines since the SRM lock must be held. The SRM will not permit the creation of new address spaces when an SQA shortage exists.	IRARMSTM IRARMEVT IRARMEVT	IRARMSQA IRARME25 IRARME26

IRARMSTM Module Entry Point Summary

IRARMPR1 - Page Replacement Normal Processing.
Examine each user in main storage and the system pageable area and call RSM real frame replacement for each user to update UICs.

IRARMPR5 - Page Replacement Real Page Shortage Force Steal.
Steal as many pages as required to relieve a real page frame shortage. The steal decision is made at entry IRARMMS2. The oldest unreferenced system-wide pages are stolen first.

IRARMMS2 - Real Page Shortage Prevention.
Calculate the number of frames necessary to reach the O.K. threshold and schedule IRARMPR5 processing (if a real page shortage exists). Inform the operator of users which have the greatest number of fixed frames and direct the swaps of these users (if a pageable real page shortage exists).

IRARMMS6 - Main Storage Occupancy Long Wait Detection.
Discover users who have gone into long wait without notifying SRM. Swapout such users, if swappable.

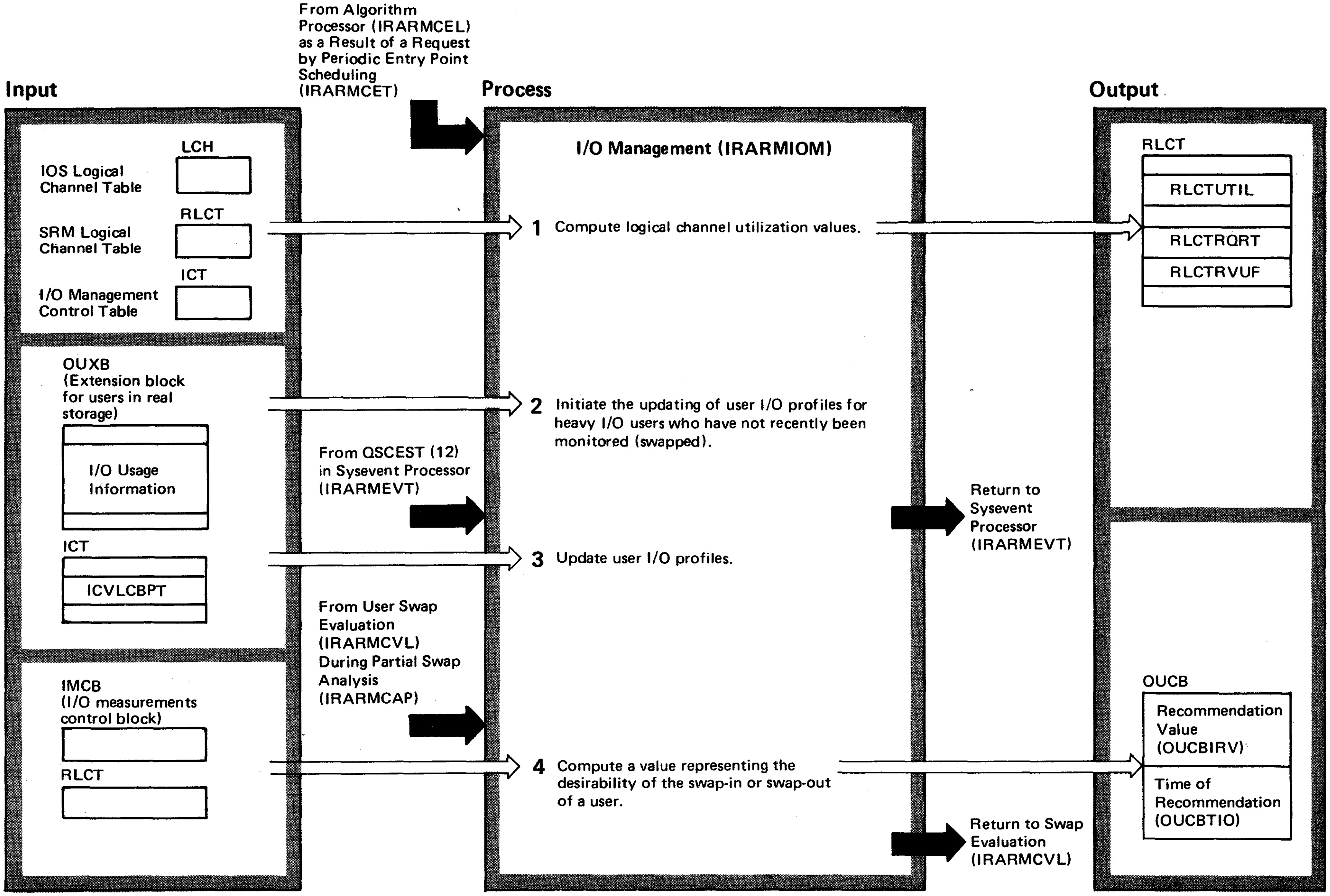
IRARMASM - Auxiliary Storage Shortage Monitoring.
Monitor extent of auxiliary shortage allocation. If auxiliary pages are in short supply, inform operator and direct swaps of users who are most rapidly acquiring auxiliary storage slots.

IRARMSQA - SQA Shortage Message Writer.
Inform operator of system queue area shortages.

STEAL - Internal STM Steal Subroutine.
Add users to RFR interface list until full, then call RSM Real Frame Replacement (RFR) routine (via IRARMI03) and record the number of pages stolen.

This blank leaf represents pages 3-52 - 3-53 which were deleted by Supervisor Performance #2.

Diagram 6-14. I/O Management (IRARMIOM) (Part 1 of 2)



1 Compute logical channel utilization values.

2 Initiate the updating of user I/O profiles for heavy I/O users who have not recently been monitored (swapped).

3 Update user I/O profiles.

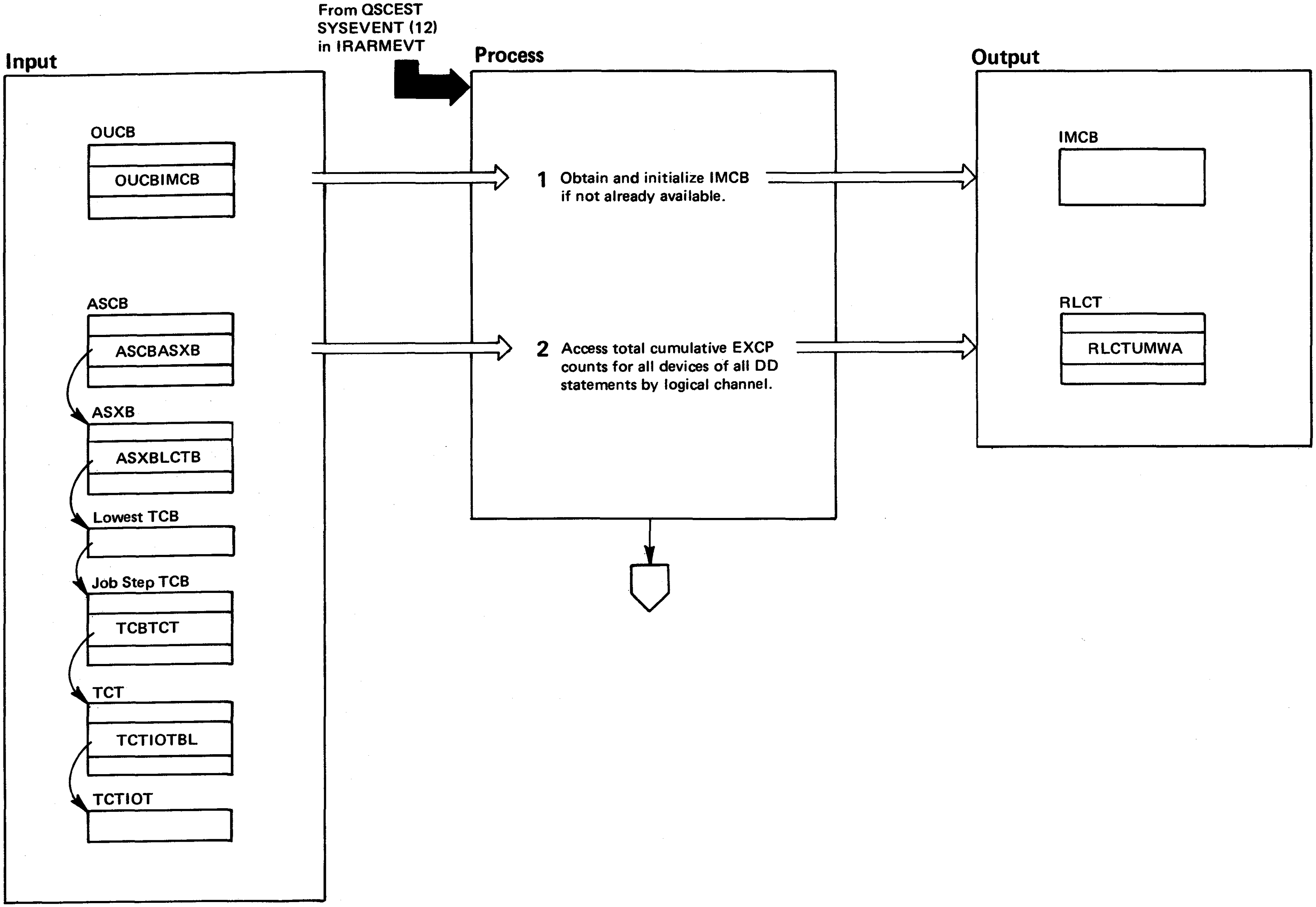
4 Compute a value representing the desirability of the swap-in or swap-out of a user.

Diagram 6-14. I/O Management (IRARMIOM) (Part 2 of 2)

Extended Description	Module	Label
I/O Management consists of a set of routines that monitor the I/O logical channel usage of certain users. They recommend users for swapping based upon the extent to which the swap-in or swap-out of a user would correct a detected I/O system imbalance. One I/O management function is described elsewhere:	IRARMIOM	
• I/O load balancing IMCB (I/O measurement control block) deletion is performed due to execution of the INITDET(11) SYSEVENT and is described in the SYSEVENT Processor M.O.	IRARMIOM	IRARMIL4
1 For each logical I/O channel in the system, the following are calculated:	IRARMIOM	IRARMIL1
• Logical channel utilization (the percentage of recent I/O requests for the channel that encountered a busy condition.		LCHUSE
• Logical channel request rate (rate of recent I/O requests per second).		LCHUSE
• Logical channel utilization factor (difference between a threshold utilization and actual utilization, squared, with a sign indicating whether the logical channel is over-utilized or under-utilized; for logical channels with a balanced I/O utilization, the factor equals zero).		LCHUSE
2 I/O management generates a request that the heavy I/O user, who has not recently been monitored, be swapped out (via Control Swap-Out IRARMCSO), solely for the purpose of obtaining addressability to the user's TCTIOT (I/O timing control table). When the quiesce started SYSEVENT is received by the SRM, the measurements will be obtained, and quiesce processing will be told to "turn the user around" (i.e., do not continue with quiesce processing). See SYSEVENT Processor M.O., SYSEVENT QSCEST.	 IRARMEVT	IRARMIL1 IRARME12
3 See I/O load balancing user I/O monitoring M.O.	IRARMIOM	IRARMILO
4 The I/O swap recommendation value for a user varies with the extent to which the user makes use of out-of-balance logical channels and the degree to which the channels are out of balance. The maximum for this recommendation value is one-fifth of the largest work load level. The I/O resource factor coefficient (RMPTIOC) is factored in to produce the final user swap recommendation value.	IRARMIOM IRARMCTL	IRARMIL3 IRARMCVL

This blank leaf represents pages 3-56 - 3-57 which were deleted by Supervisor Performance #2.

Diagram 6-16. I/O Load Balancing User I/O Monitoring (IRARMIL0) (Part 1 of 4)



1 Obtain and initialize IMCB if not already available.

2 Access total cumulative EXCP counts for all devices of all DD statements by logical channel.

Diagram 6-16. I/O Load Balancing User I/O Monitoring (IRARMILO) (Part 2 of 4)

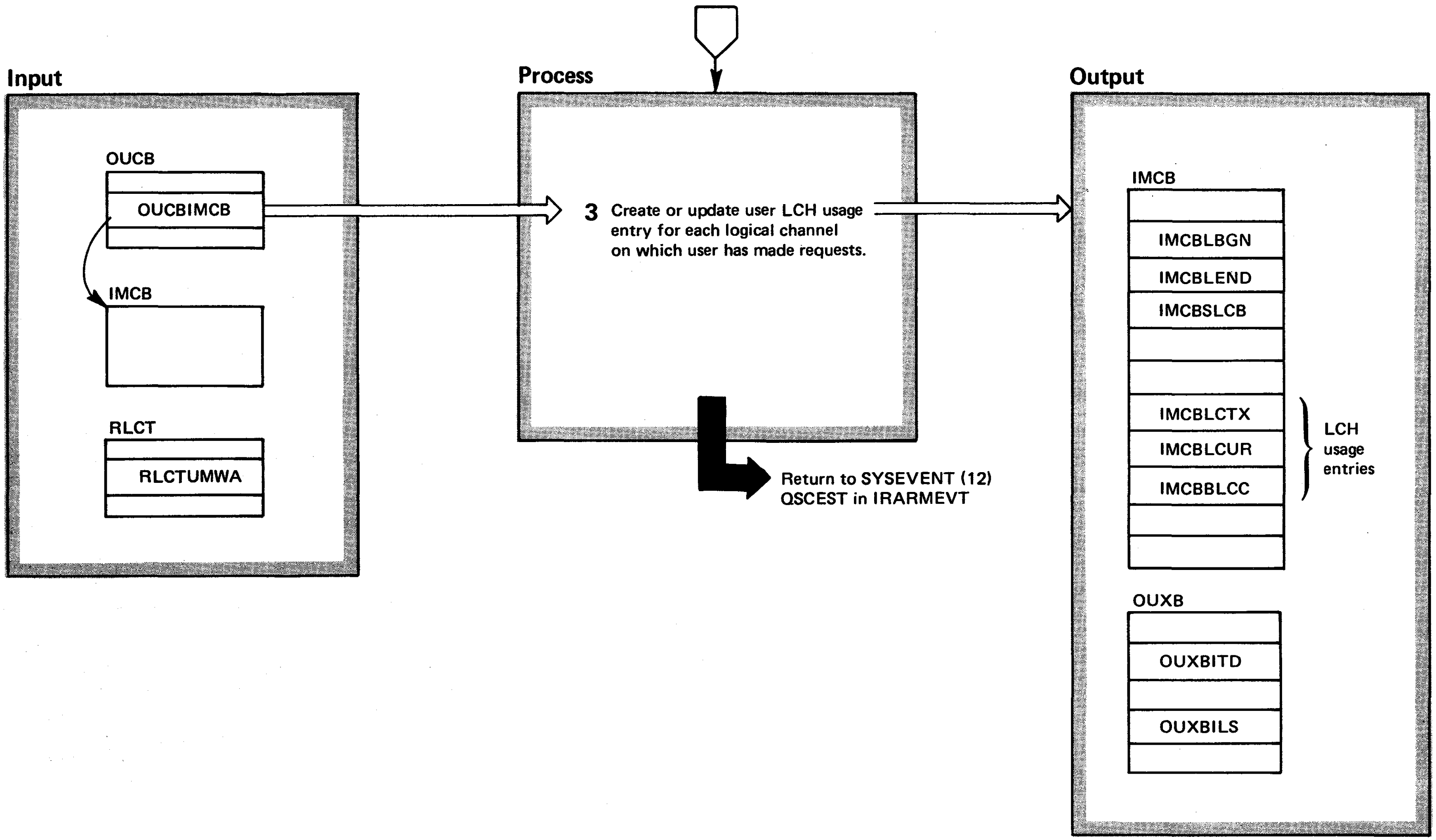
Extended Description	Module	Label
I/O load balancing user I/O monitoring maintains detailed information on logical channel (LCH) utilization for heavy I/O users. This LCH information is used by other I/O load balancing functions to influence swapping decisions when heavy users are using out-of-balance logical channels (over-utilized or under-utilized).	IRARMIOM	IRARMILO

This monitoring is done at the time of the quiesce-started SYSEVENT (SYSEVENT 12). At this time, the I/O Timing Control Table (TCTIOT), which contains monitoring source data, is addressable. Entry through the quiesce-started SYSEVENT may be forced for a user who has not been monitored recently (see I/O Management (IRARMIOM) M.O.).

1 If I/O load balancing is active and the user does not have an IMCB, obtain an IMCB if the user's total I/O request rate is high enough (that is, higher than ICCMNIOR).

2 Access TCTIOT, looking at all user data set declarations, and access all devices allocated to each data set. Through the UCB, associate the device to a logical channel, and sum the user's EXCPs by logical channel using RLCTUMWA as a working variable for the summation.

Diagram 6-16. I/O Load Balancing User I/O Monitoring (IRARMIL0) (Part 3 of 4)



3 Create or update user LCH usage entry for each logical channel on which user has made requests.

Return to SYSEVENT (12) QSCEST in IRARMEVT

Diagram 6-16. I/O Load Balancing User I/O Monitoring (IRARMILO) (Part 4 of 4)

Extended Description	**Module**	**Label**
3 Step through all logical channels (RLCT entries), and determine if the user has been monitored for LCH utilization. That is, determine if an LCH usage entry is established in the IMCB. Update various utilization fields. If an IMCB LCH usage entry is not established, make room for an entry by relocating other LCH usage entries so that entries are kept in RLCT order. Various IMCB fields are initialized for the new entry.	IRARMIOM	

Fields updated or created in the LCH usage entries are:

- IMCBBLCC – previous cumulative number of EXCP requests made on the channel
- IMCBLCTX – pointer to corresponding RLCT entry
- IMCBLCUR – logical channel usage rate

Fields updated in the OUXB are:

- OUXBITD – I/O load balancing base time
- OUXBILS – I/O service base count

Additionally, the IMCBSLCB field is maintained as a summary flag field that indicates all logical channels on which this user is a heavy user.

IRARMIOM Module Entry Point Summary

IRARMIL0 - I/O Load Balancing User I/O Monitoring.
Compute I/O Usage Profile for all swappable problem state users.

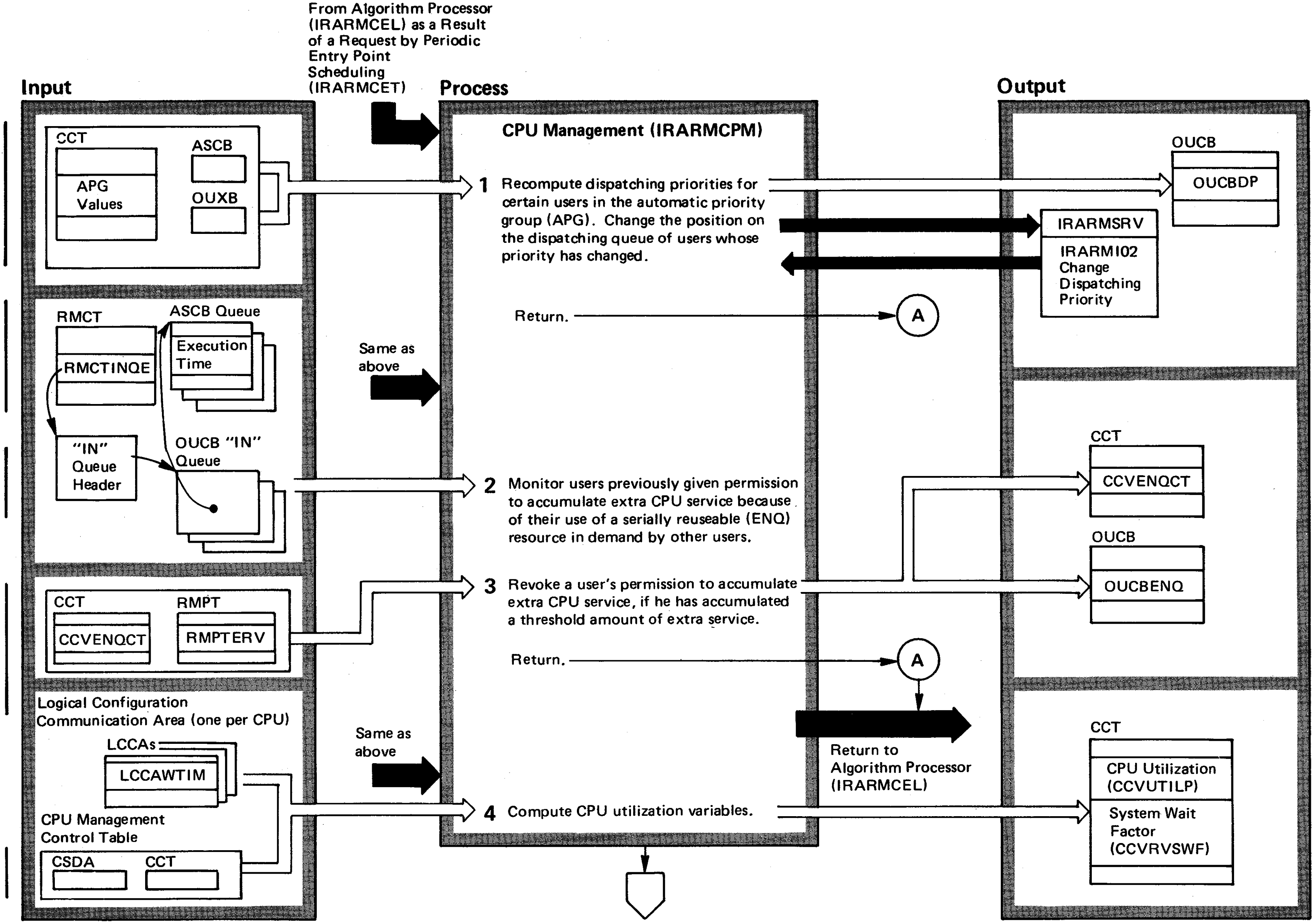
IRARMIL1 - I/O Load Balancing Logical Channel Utilization Monitoring.
Compute channel utilization values for I/O load balancing, page replacement algorithms, and the device allocation SYSEVENT.

IRARMIL3 - I/O Load Balancing User Swap Evaluation.
Compute numerical recommendation value which reflects desireability of swapping a user based on its Logical Channel utilization.

IRARMIL4 - I/O Load Balancing IMCB Deletion Routine.
Clean up control blocks used in monitoring a heavy I/O user at the end of the user job step.

LCHUSE - Internal IOM Subroutine.
Compute logical channel utilization, request rate, and I/O load balancing recommendation value computation factor.

Diagram 6-17. CPU Management (IRARMCPM) (Part 1 of 4)



1 Recompute dispatching priorities for certain users in the automatic priority group (APG). Change the position on the dispatching queue of users whose priority has changed.

2 Monitor users previously given permission to accumulate extra CPU service because of their use of a serially reuseable (ENQ) resource in demand by other users.

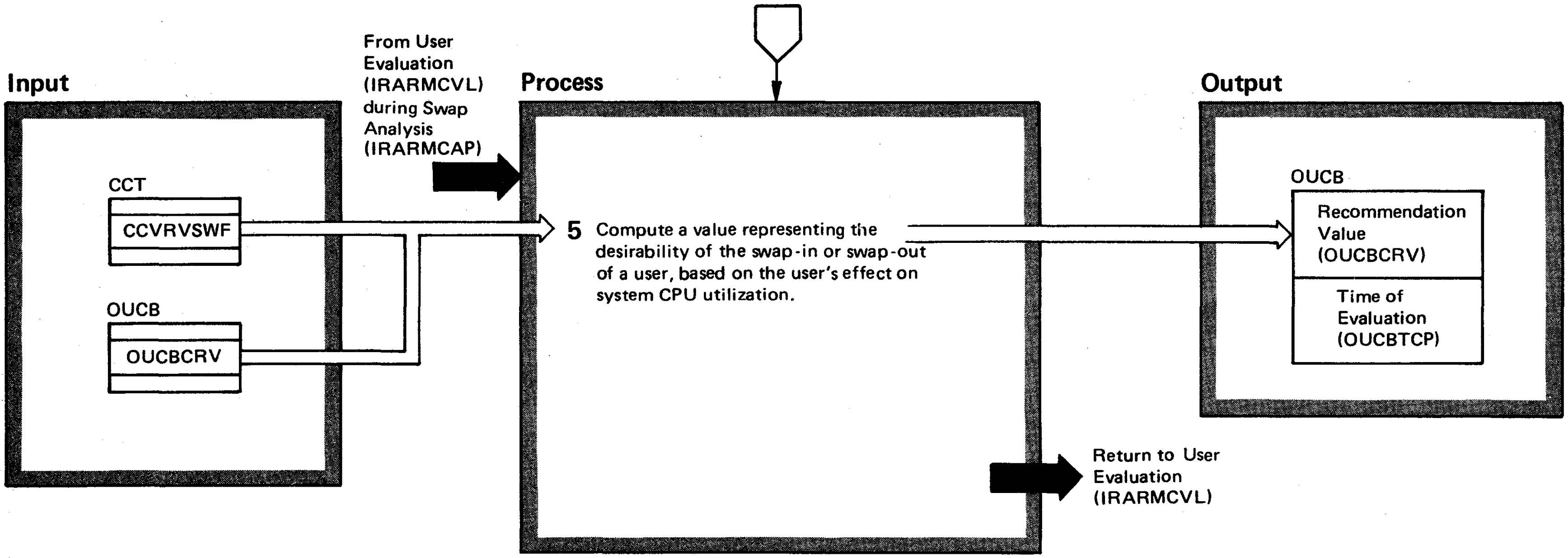
3 Revoke a user's permission to accumulate extra CPU service, if he has accumulated a threshold amount of extra service.

4 Compute CPU utilization variables.

Diagram 6-17. CPU Management (IRARMCPM) (Part 2 of 4)

Extended Description	Module	Label (or Segment)
CPU Management consists of a set of routines that monitor the system-wide CPU load. They recommend users for swapping when the system is under or over-utilized. One CPU management function is described elsewhere:	IRARMCPM	
• CPU load balancing system profile adjustment is performed when the SRM receives a QSCECMP SYSEVENT (13); it is described in the SYSEVENT Processor table.	IRARMCPM	IRARMCL0
1 An optional parameter of the period definition in the IPS has the following format: APG=I where I is an integer between 0 and 15, this parameter applies only to those transactions whose beginning dispatching priority (from the job or step JCL) lies within the APG range defined in IEASYSxx. It is ignored for those transactions which lie outside the range. The effect of this parameter is that the initial dispatching priority (at transaction start and initiator attach) is set to the I value plus the lowest APG dispatching priority. If the parameter is not coded, it defaults to the highest mean time to wait priority in the APG: 6. If the value I is 6 or less, subsequent dispatching priorities will be calculated based on the address space's mean time to wait; that is, the average time he was in execution before entering the wait state. The lower his mean time to wait, the higher will be a user's priority within the APG.	IRARMCPM	IRARMAP1
This computation is performed for all APG users in main storage who have executed for greater than a threshold of time (CCCAPMET) since their last computation. If the value I is 11 the address space will enter a rotate group. At a timed interval, SRM will examine all address spaces in the rotate group. If there is more than one, SRM will move the first dispatchable address space in the group to become the last address space of the group on the dispatching queue.		NEWDP
If the value I is any other valid integer, the dispatching priority will be unchanged until the APG parameter is changed on the IPS period specification. Since it is also possible for a user's dispatching priority to be recalculated while he is being swapped out (See SYSEVENT QSCECMP (13) – profile adjustment, in SYSEVENT Processor table), periodically both "IN" and "OUT" users are checked to see if their order must be changed on the dispatching queue. This function re-sets its time of next invocation, based upon the percentage of APG users that had their dispatching priorities changed.		
2 A user is given permission to accumulate extra CPU service when an ENQHOLD SYSEVENT (20) is received by the SRM, indicating that he holds a resource in demand by other system users. The mechanism for giving him this extra service is the prevention of his swap out by the SRM because of service rate considerations.	IRARMCPM	IRARMEQ1
3 The Enqueue Residence Value (ERV), an OPT parameter, specifies the length of the privileged "spurt" of service for a user for whom an ENQHOLD SYSEVENT (20) has been issued (see 2). When this time is exceeded, the user is made eligible for swap-out, and his OUCB is so flagged.	IRARMCPM	IRARMEQ1
The individual user evaluation routine is called to assign a current workload manager recommendation value to this user.	IRARMWLM	IRARMWM3
4 The CPU utilization is the average percentage of time any CPU in the system was not in the wait state. It is computed by the following formula: $\text{utilization} = 100 - \left[\frac{\text{(sum of wait routines on each CPU)*100}}{\text{(elapsed time since last entry)*(number of CPUs)}}\right]$	IRARMCPM	IRARMCL1
CPU utilization is artifically set to 101% if actual utilization is 100% and one or more users have not been dispatched. This allows the CPU to be considered over-utilized even if the CPU threshold is 100%. The system wait factor is calculated for use in determining the swap recommendation value for a user (see CPU Management M.O., step 5); it is a multiple of the square of the difference between a threshold value and the utilization, with the sign indicating the direction of the imbalance (over- or under-utilized). If the CPU utilization falls between the high and low thresholds, the factor equals zero.		CPUWAIT CPLRVSWF

Diagram 6-17. CPU Management (IRARMCPM) (Part 3 of 4)



5 Compute a value representing the desirability of the swap-in or swap-out of a user, based on the user's effect on system CPU utilization.

Diagram 6-17. CPU Management (IRARMCPM) (Part 4 of 4)

Extended Description	Module	Label (or Segment)
5 The CPU swap recommendation value for a significant CPU user varies with the degree to which the CPU load is out of balance. The recommendation value can not be greater than one-fifth the highest workload level. For insignificant CPU users, the recommendation value is zero.	IRARMCPM	IRARMCL3
The time of this evaluation and the swap recommendation are saved in the OUCB. The user swap evaluation routine, IRARMCVL, then multiplies the recommendation value by the CPU resource factor coefficient (RMPTCPU) to produce the final CPM swap recommendation value.	IRARMCTL	IRARMCVL

IRARMCPM Module Entry Point Summary

IRARMAP1 - Automatic Priority Group Reorder Processing.
Recompute dispatching priorities for all APG users in main storage. Invoke ASCBCHAP for each user whose dispatching priority has changed.

IRARMEQ1 - ENQ/DEQ Algorithm ENQ Time Monitoring.
Stop giving extra CPU service to users with ENQHOLD SYSEVENTs outstanding who have already received their quaranteed CPU service.

IRARMCL0 - CPU Load Balancing User Swap Processing.
Compute user CPU usage profile at QSCECMP SYSEVENT.

IRARMCL1 - CPU Utilization Monitoring.
Compute CPU utilization variables for CPU load balancing and resource management algorithms.

IRARMCL3 - CPU Load Balancing User Swap Evaluation.
Produce a numerical recommendation value which reflects the desireability of swapping a user based on its CPU utilization.

CHAP - IRARMCPM Internal Chapping Subroutine.
Search queue for APG users with changed dispatching priorities, put them in a list and call ASCBCHAP.

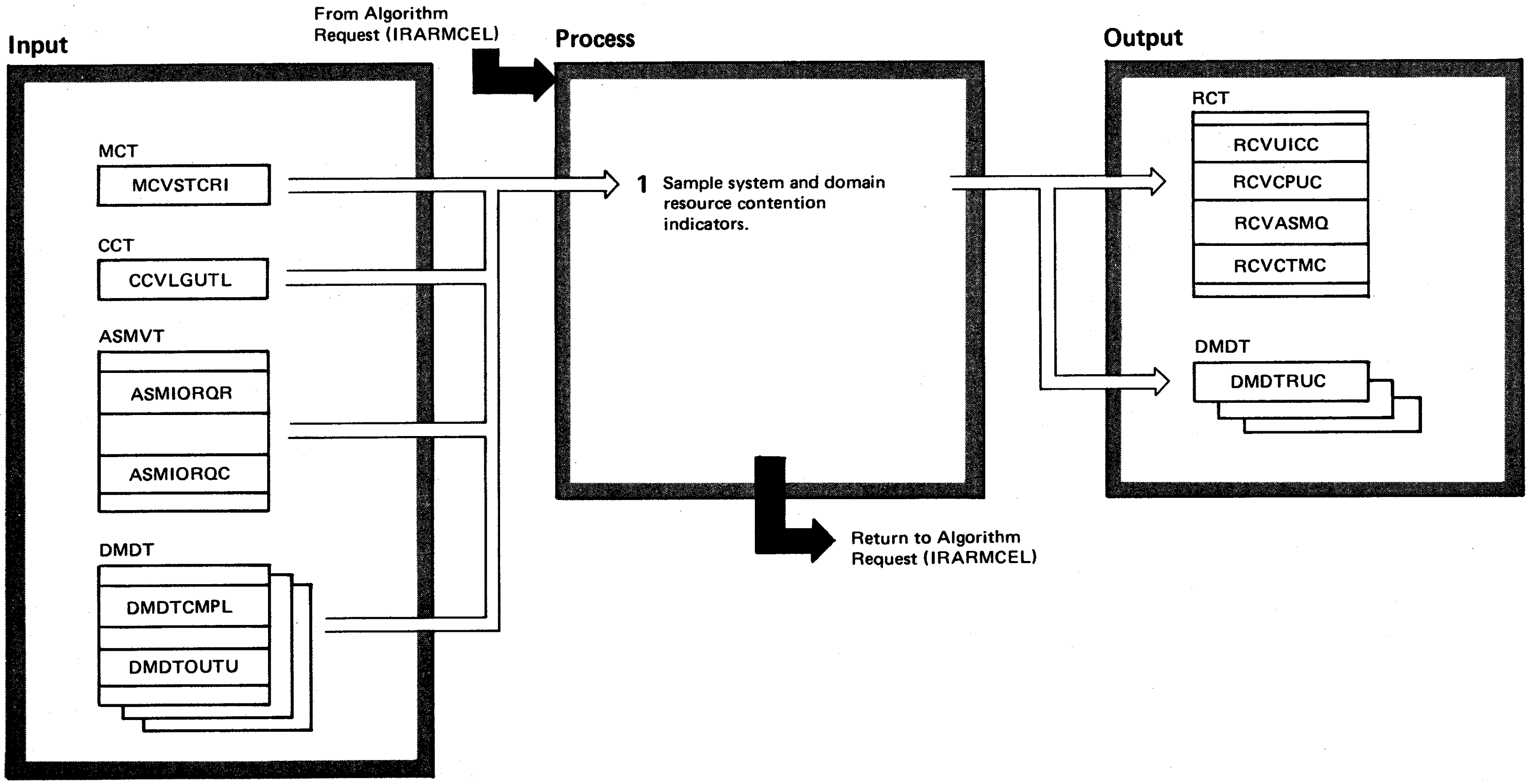
CPLRVSWF - IRARMCPM Internal Wait Factor Computation Subroutine.
Compute system wait factor for CPU load balancing recommendation value computation.

CPUWAIT - IRARMCPM Internal Wait Time and CPU Utilization Compute. Subroutine
Compute accumulated system wait time total for all CPUs and compute recent CPU utilization.

CPUTLCK - IRARMCPM Internal CPU Utilization Checking Routine.
Insure that the computed CPU utilization percentage falls between 0 and 100 percent. If 100 percent and lowest priority user has not been dispatched, set it to 101 percent.

NEWDP - IRARMCPM Internal APG Computation Routine.
Compute mean time to wait and a new dispatching priority for the APG user.

Diagram 6-18. Resource Monitor Periodic Monitoring (IRARMRM1) (Part 1 of 2)

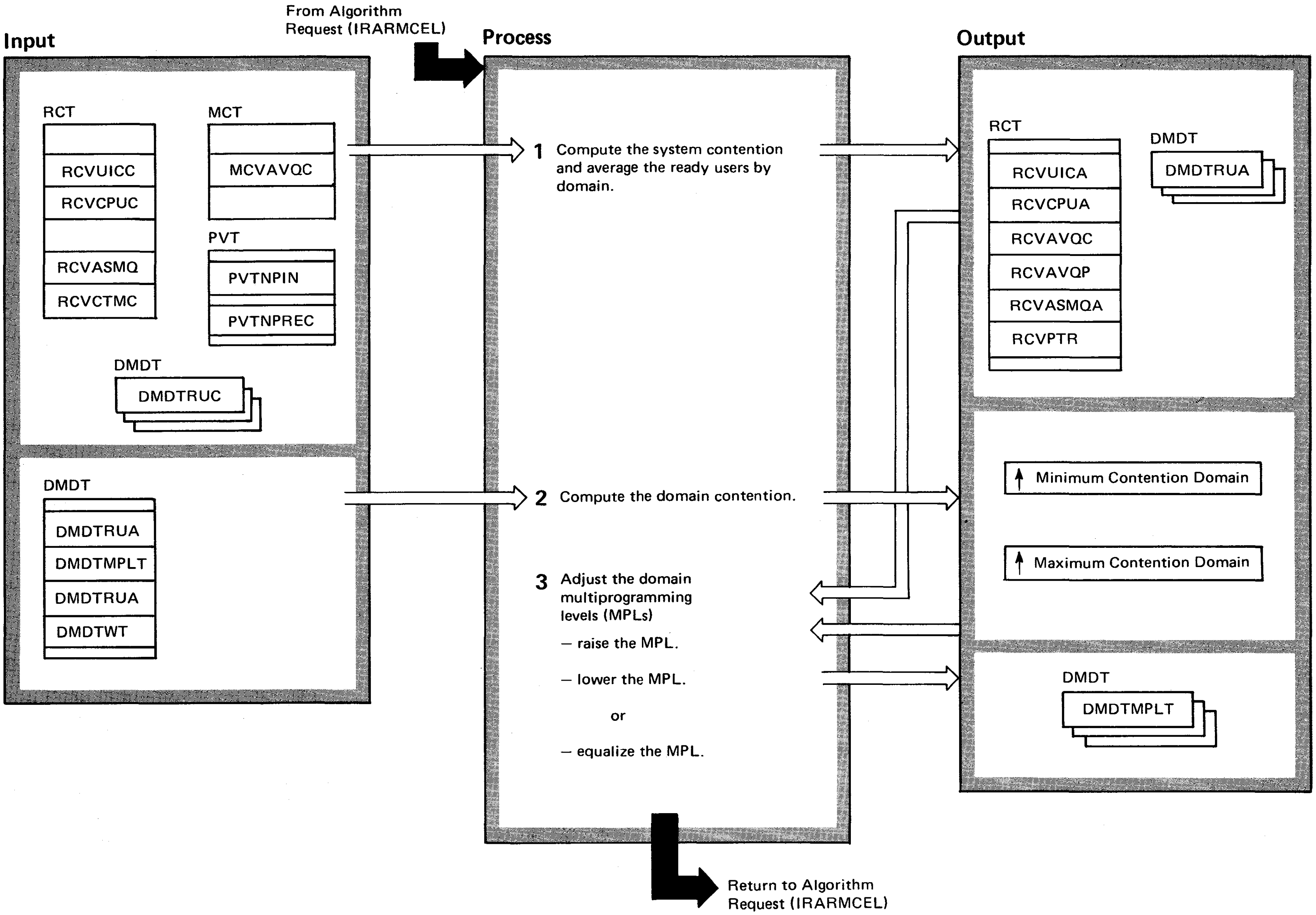


1 Sample system and domain resource contention indicators.

Diagram 6-18. Resource Monitor Periodic Monitoring (IRARMRM1) (Part 2 of 2)

Extended Description	Module	Label
1 This routine is invoked at one second intervals and accumulates the highest system unreferenced frame interval count (MCVSTCRI), the current CPU utilization (CCVLGUTL), and the number of un-completed ASM requests (ASM requests minus ASM completed requests). Additionally the number of ready users (the number of users on the IN queue plus the number of users on the OUT queue) for each domain is accumulated.	IRARMRMR	IRARMRM1

Diagram 6-18A. Resource Monitor MPL Adjustment Processing (IRARMRM2) (Part 1 of 2)



1 Compute the system contention and average the ready users by domain.

2 Compute the domain contention.

3 Adjust the domain multiprogramming levels (MPLs)

— raise the MPL.

— lower the MPL.

or

— equalize the MPL.

Diagram 6-18A. Resource Monitor MPL Adjustment Processing (IRARMRM2) (Part 2 of 2)

Extended Description	Module	Label
This routine is invoked at 30 second intervals and processes the data accumulated by IRARMRM1 to compute the average unreferenced frame interval count (RCVUICA), the number of "AVQ Lows" over the last RM2 interval (for tracking only), the average ASM queue length (RCVASMQA), the system page fault rate per second (RCVPTR), and the average number of ready users for each domain (DMDTRUA). IRARMRM2 also computes the system-wide logical channel utilization. If the average logical channel utilization is above a threshold value or if an individual logical channel has a high utilization and request rate above threshold values, a contention indicator, RCVIOUSE, is set. The above system and domain contention factors are used to adjust the domain target MPLs as follows:	IRARMRMR	IRARMRM2

2 Each used domain contention index is computed by the formula:

$$\frac{\text{average ready users} \times \text{weight}}{\text{max (current target MPL or one)}}$$

This yields a measure of contention for this domain weighted by the user specified importance factor (weight) for the domain.

3 The Resource Monitor will then determine if the system MPL should be raised or lowered by comparing the system contention indicators against predefined limits. All positive conditions must be met to increase and only one condition need be met to force a decrease in the MPL.

	LIMITS			
	INCREASE MPL		DECREASE MPL	
UIC*	GT	1	LT	1
CPU utilization	LT	95%	GT	99%
PAGE FAULTS	LT	30/sec	GT	40/sec
ASMQ	LT	10 requests	GT	20 requests
Average logical channel utilization	LT	20%	GT	20%
Logical channel utilization	LT	30%	GT	30%
			and	
Logical channel request rate	LT	50 requests	GT	50 requests

*unreferenced internal count

Extended Description	Module	Label

- If any domain is unused (average number of ready users less than target minus one) that domain's MPL is decreased by one if the decrease does not drop it below the minimum MPL or one.

- If the system MPL should be raised, the Resource Monitor will pick the domain that has the highest contention index and has not yet reached its maximum MPL and increase this domain's MPL by one.

- If the system MPL should be decreased, the Resource Monitor will pick the domain with the lowest contention index which has not yet reached its minimum MPL and decrease this domain's MPL by one.

- If the system MPL should not be increased or decreased, the Resource Monitor will attempt to equalize the domain's contention index; such that if the highest domain contention index is greater than the lowest, the Resource Monitor will increase the MPL for the high contention domain and decrease the MPL for the lowest contention domain.

IRARMRMR Module Entry Point Summary

IRARMRM1 -	Resource Monitor Periodic Monitoring. Accumulate several system resource contention indicators and the number of ready users for each domain at periodic sample intervals.
IRARMRM2 -	Resource Monitor MPL Adjustment Processing. Compute the average system resource utilization and determine if the system MPL should be raised or lowered.

Workload Management

The workload manager is a collection of subroutines which perform three main functions:

- Monitoring the rate at which system resources are being provided to individual address spaces.
- Providing swapping recommendations (based on installation specifications and resource usage) requested by SRM Control (IRARMCTL).
- Collecting data for MF/1 workload activity reporting.

Subroutines that support the first two functions above are packaged in the workload manager module (IRARMWLM), and the data collecting subroutines are in the workload activity recording module (IRARMWAR).

Nonswappable address spaces and certain privileged system control program functions are not under the control of the workload manager.

In providing swapping recommendations to SRM Control, the workload manager affects the relative rates at which processing resources are provided to active address spaces. By comparing an address space's resource usage (service rate) against the installation performance specifications, the workload manager computes the address space's workload level (also called workload manager recommendation value) which is used by SRM Control as a swapping recommendation.

The workload activity recording facility (IRARMWAR) collects data for MF/1 when MF/1 workload activity reports have been requested. This facility is invoked periodically by the workload manager and the SYSEVENT processor to collect data, that is placed in the workload activity measurement table (WAMT). The workload activity reports may be analyzed by an installation and used to determine the appropriate installation performance specification parameters to meet their needs.

(See the MF/1 and SRM sections of the *OS/VS2 (MVS) Initialization and Tuning Guide* on using workload activity reports).

Several workload management functions are of a housekeeping nature, and are triggered by the receipt of certain SYSEVENTS. These are described in the SYSEVENT Processor M.O., and include:

- Service calculation routine - invoked by SYSEVENTs WKLDINIT(45) and REQSERVC (38).

Module	Label
IRARMWLM	IRARMWM1
IRARMEVT	IRARME45, E38

- Workload level calculation - invoked by SYSEVENT WKLDCOLL(46).

Module	Label
IRARMWLM	IRARMWM4
IRARMEVT	IRARME46

- Start new transaction - invoked by SYSEVENTS RESETPG(31), TGETTPUT(34) and INITATT(10), and module IRARMSET after a NEWIPS(32) SYSEVENT is received.

Module	Label
IRARMWLM	IRARMWMN
IRARMEVT	IRARME31, E34,E10
IRARMSET	
IRARMEVT	IRARME32

- Swap status change request - invoked by SYSEVENTS DONTSWAP(41) and OKSWAP(42).

Module	Label
IRARMWLM	IRARMWMK
IRARMEVT	IRARME41, E42

- Stop old transaction - invoked by SYSEVENTs JOBTERM(9), INITDET(11) and JOBSELECT(8).

Module	Label
IRARMWLM	IRARMWMO
IRARMEVT	IRARME09, E11,E8

- Restore completed processing - invoked by SYSEVENTs RSTORCMP(19) and INITATT(10).

Module	Label
IRARMWLM	IRARMWMR

IRARMEVT	IRARME19, E10

- Quiesce completed processing - invoked by SYSEVENT QSCECMP(13).

Module	**Label**
IRARMWLM	IRARMWMQ
IRARMEVT	IRARME13

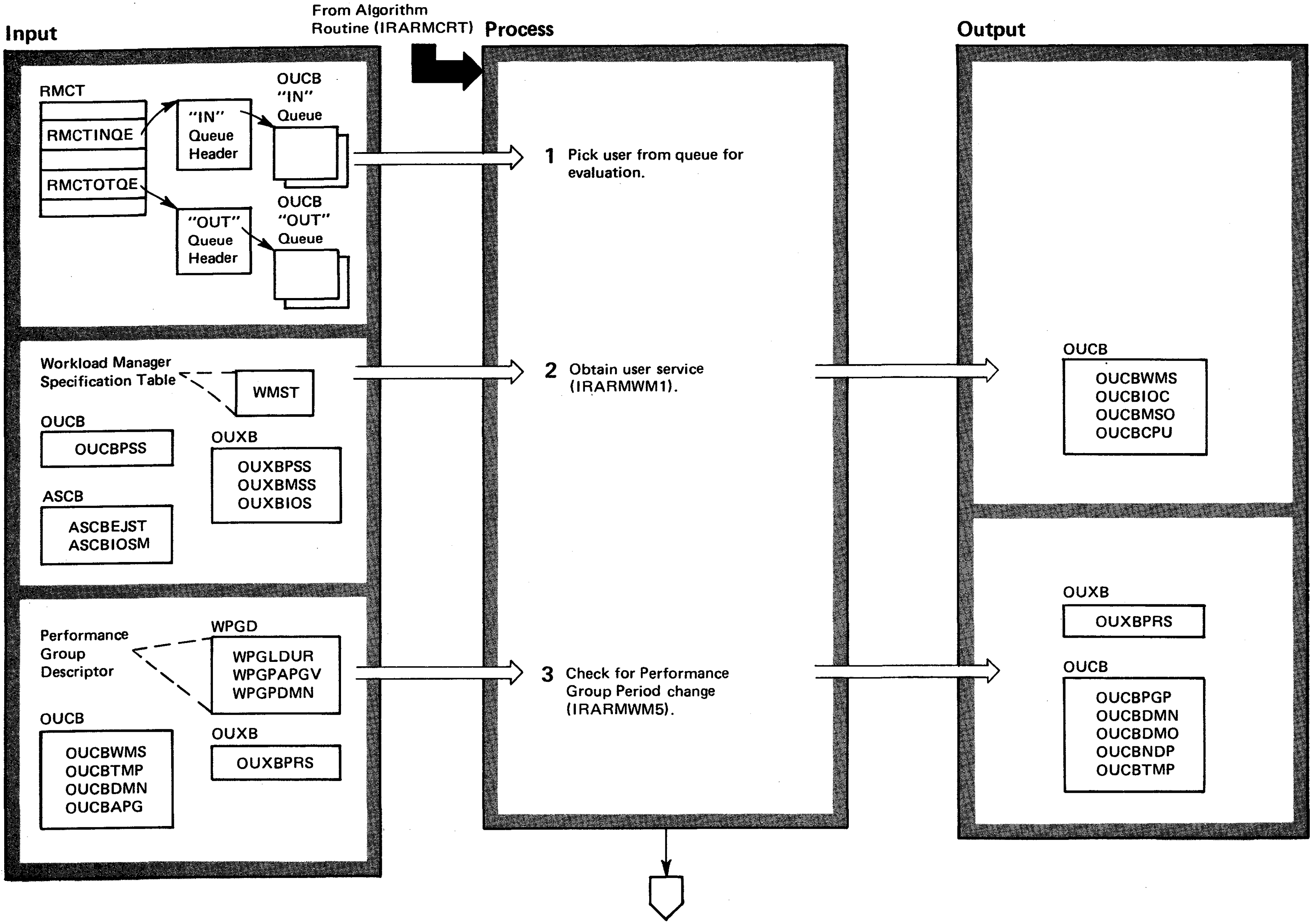
The following workload manager M.O.s describe 3 major functions performed by the IRARMWLM module:

- Swappable user evaluation.
- Scanning the IN queue and OUT queue, evaluating each non-privileged, swappable user, and assigning a current workload level.
- Individual user evaluation - evaluating a (one) non-privileged, swappable user, and assigning a current workload level.
- User ready processing - initializing user control blocks and repositioning the user from the WAIT queue to the OUT queue so the user is eligible for swap-in.

The following workload activity recording M.O.s describe 2 major functions performed by the IRARMWAR module:

- WAMT initialization
  - updating the workload activity measurement table (WAMT) with information from the workload manager specification table (WMST).
- WAMT reinitialization
  - copying the WAMT data to a temporary buffer and then updating service values and workload levels.

Diagram 6-19. Swappable User Evaluation (IRARMWM2) (Part 1 of 4)



1 Pick user from queue for evaluation.

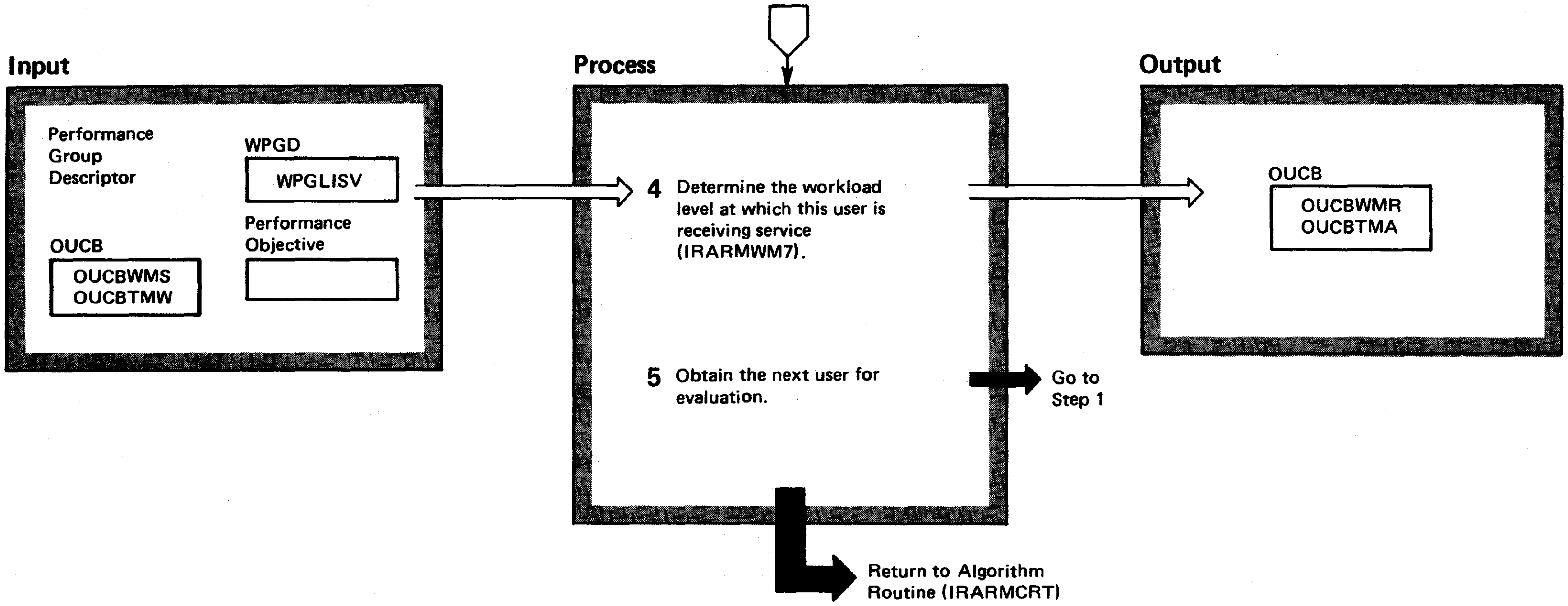
2 Obtain user service (IRARMWM1).

3 Check for Performance Group Period change (IRARMWM5).

Diagram 6-19. Swappable User Evaluation (IRARMWM2) (Part 2 of 4)

Extended Description	Module	Label
The WM2 routine is invoked by IRARMCET approximately every second to evaluate all users that have not been evaluated during the past second and whose period duration is specified in real time. If no periods have real time specified anywhere in the IPS, IRARMWM2 will not be invoked. This ensures that users with period durations specified in real time units are evaluated for period change even though they may be in an inactive domain.		
1 Both the IN and OUT queues are scanned, evaluating non-privileged swappable users.	IRARMWLM	IRARMWM2
2 WM1 is invoked to calculate the service accumulated during the "in real storage interval" for users currently in storage.	IRARMWLM	IRARMWM1
3 Depending on the units code in the IPS (service units or time units), the transaction's accumulated service or time units are checked to determine whether a period has ended. If a period has ended, the current period indication is updated.	IRARMWLM	IRARMWM5
If workload reporting is also active, IRARMWR4 is invoked to communicate the period change.	IRARMWAR	IRARMWR4
If in switching periods, the user also changes domains, he will be repositioned at the end of the appropriate queue. The user dispatching priority is also updated, if applicable.	IRARMCTL	IRARMRPS

Diagram 6-19. Swappable User Evaluation (IRARMWM2) (Part 3 of 4)



4 Determine the workload level at which this user is receiving service (IRARMWM7).

5 Obtain the next user for evaluation.

Diagram 6-19. Swappable User Evaluation (IRARMWM2) (Part 4 of 4)

Extended Description	Module	Label
4 The workload level is a means of comparing users to other users in the same domain. If a user has not yet received enough service to be controlled by the workload manager (that is, his service is less than his interval service value-ISV) or if the user is between job steps, a workload level corresponding to a zero service rate is returned. In calculating his recent service rate, a user's accumulated service is reset to zero when he is swapped-in; his accumulated time is reset to zero when he is swapped-out.	IRARMWLM	IRARMWM7
5 Processing continues until all users on the IN and OUT queues are evaluated.		

Diagram 6-20. Individual User Evaluation (IRARMWM3) (Part 1 of 2)

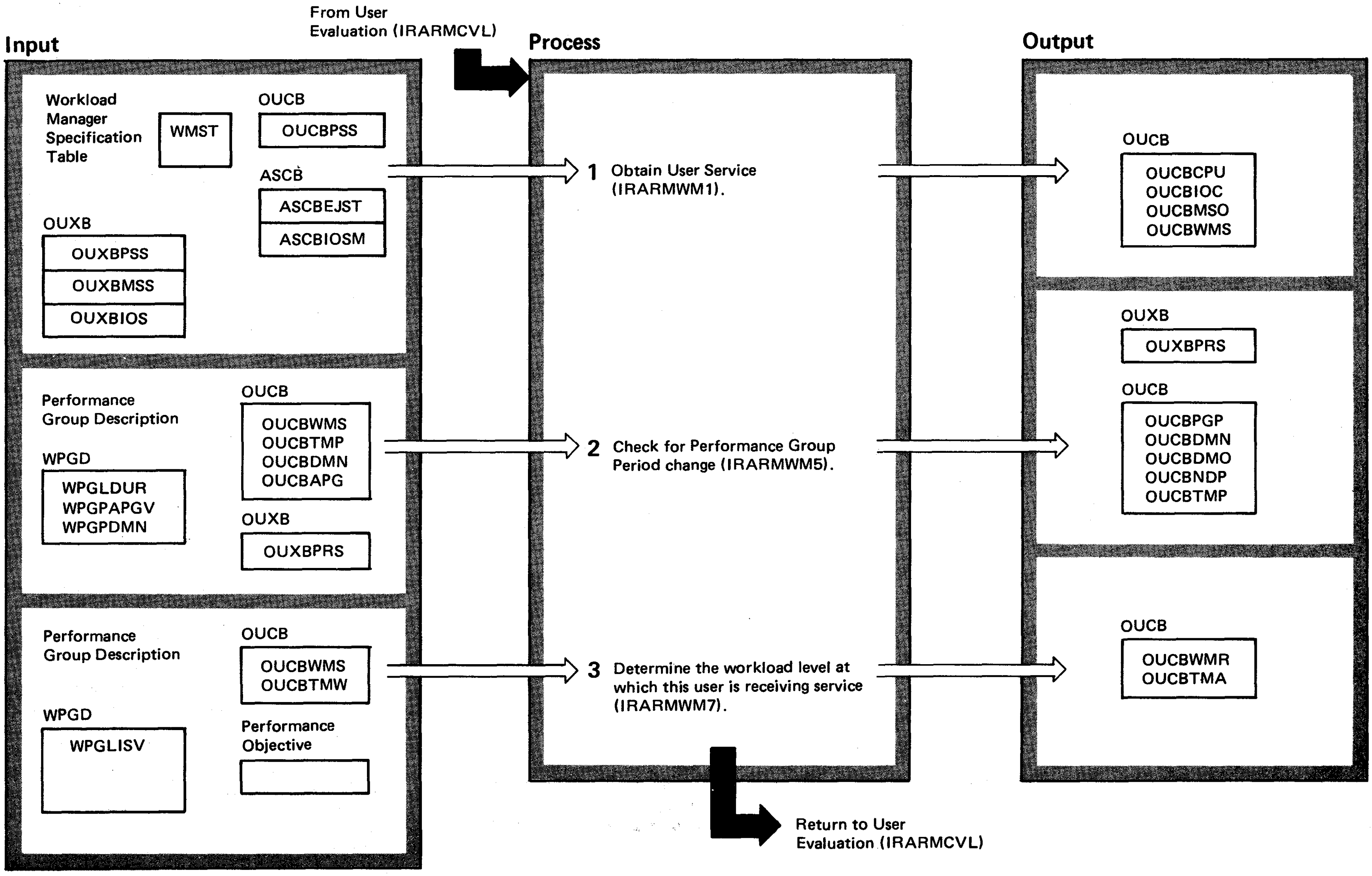
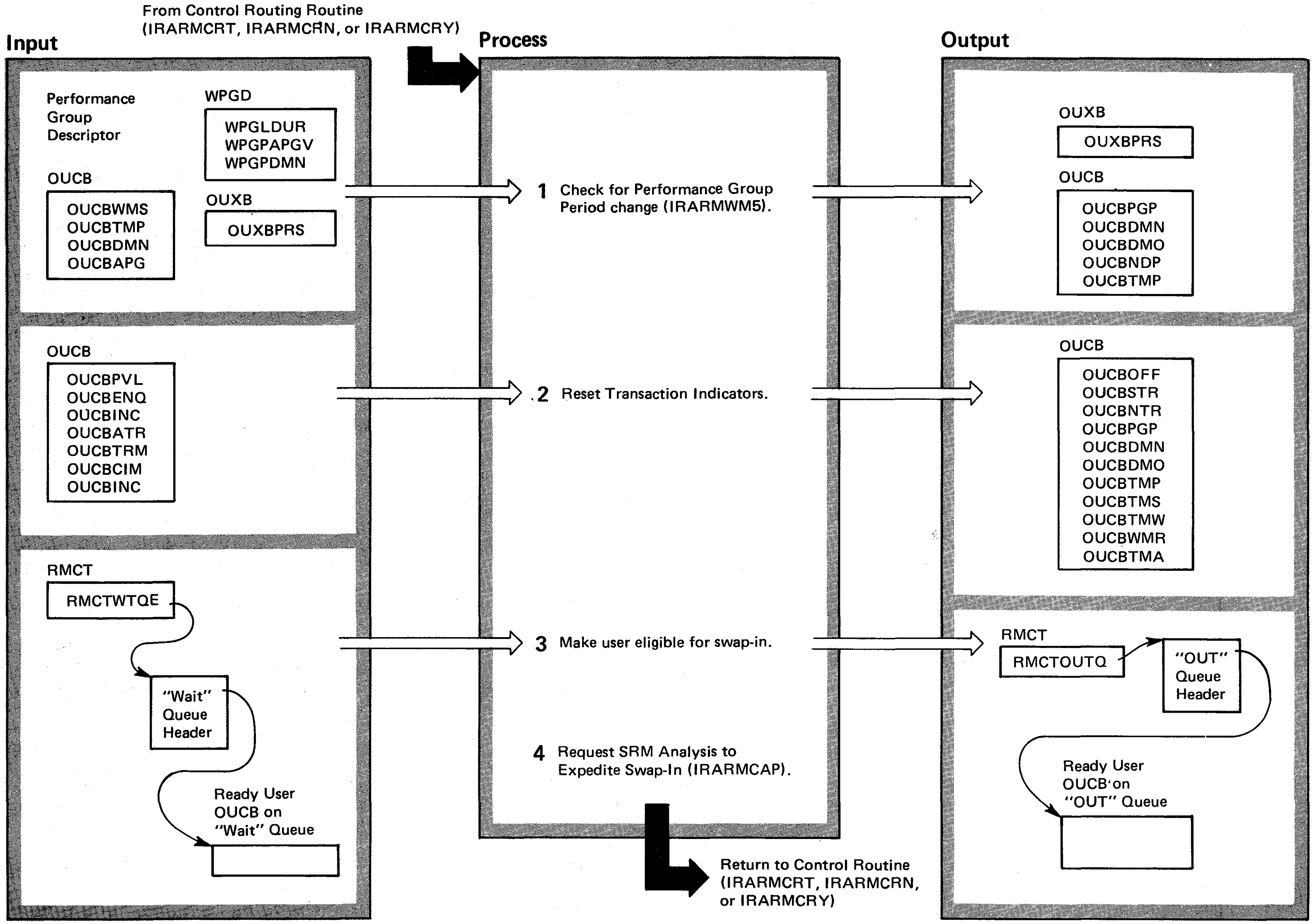


Diagram 6-20. Individual User Evaluation (IRARMWM3) (Part 2 of 2)

Extended Description	Module	Label
The IRARMWM3 routine is invoked by the user evaluation routine (IRARMCVL) during analysis of users in a particular domain. The major output of the routine is the workload level (recommendation value) of the user being evaluated. Non-swappable and privileged users are not evaluated.		
1 WM1 is invoked to calculate the service accumulated during the "in real storage interval" for users currently in storage.	IRARMWLM	IRARMWM1
2 Depending on the units code in the IPS (service units or time units), the transaction's accumulated service or time units is checked to determine whether a period has ended. If a period has ended, the current period indication is updated. If workload reporting is also active, IRARMWR4 is invoked to communicate the period change. If in switching periods, the user also changes domains, he will be repositioned at the end of the appropriate queue. The user dispatching priority is also updated, if applicable.	IRARMWLM IRARMWAR IRARMCTL	IRARMWM5 IRARMWR4 IRARMPRS
3 The workload level is a means of comparing users to other users in the same domain. If a user has not yet received enough service to be controlled by the workload manager (that is, his service is less than his interval service value-ISV) or if the user is between job steps, a workload level corresponding to a zero service rate is returned. In calculating his recent service rate, a user's accumulated service is reset to zero when he is swapped-in; his accumulated time is reset to zero when he is swapped-out.	IRARMWLM	IRARMWM7

Diagram 6-21. User Ready Processing (IRARMHIT) (Part 1 of 2)



1 Check for Performance Group Period change (IRARMWM5).

2 Reset Transaction Indicators.

3 Make user eligible for swap-in.

4 Request SRM Analysis to Expedite Swap-In (IRARMCAP).

Diagram 6-21. User Ready Processing (IRARMHIT) (Part 2 of 2)

Extended Description	Module	Label
IRARMHIT is requested by IRARMEVT when it receives a user ready SYSEVENT(4) from the dispatcher. The major function of this routine is to make users eligible for swap-in by repositioning them from the WAIT queue to the OUT queue.		
1 Depending on the units code in the IPS (service units or time units), the transaction's accumulated service or time units are checked to determine whether a period has ended. If a period has ended, the current period indication is updated. If workload reporting is also active, IRARMWR4 is invoked to communicate the period change. If in switching periods, the user also changes domains, he will be repositioned at the end of the appropriate queue. The user dispatching priority is also updated, if applicable.	IRARMWLM IRARMWAR IRARMCTL	IRARMWM5 IRARMWR4 IRARMRPS
2 The transaction indicators are reset based on the type of user and the user's transaction status when swapped-out. That is:		
a) OUCBs for users between job steps remain effectively unchanged.		
b) OUCBs for Terminal wait users are updated to reflect transaction. Indicators are set to the first period characteristics.		
A workload level is assigned which is equal to the first period objective "zero point".	IRARMWLM	IRARMWM4
c) OUCBs for users which have suspended transactions (may be due to issuing "long wait") are updated so that they look as if the swap-out had just ended.		
3 The "ready" user OUCB is repositioned from the WAIT queue to the end of the OUT queue.	IRARMCTL	IRARMRPS
4 The SRM analysis function is requested in order to have the user swapped in as soon as possible.	IRARMCTL	IRARMCAP

IRARMWLM Module Entry Point Summary

IRARMWM1 - Workload Manager Service Calculator Routine.
The IRARMWM1 routine calculates the amount of service provided to an user since the beginning of the current workload manager reasurement interval for that user. Service is calculated using the following equation:

Service = (MP)/K)+(CT)/K)+EI WHERE:

T = The job step time elapsed in the current interval.
K = The time required to execute 10,000 instructions. (Dependent on the CPU Model)
M = The MSO service coefficient scaled by 1/50.
P = The number of Page-Seconds used by the user.
C = The CPU service coefficient.
E = The EXCP count for this interval.
I = The I/O service coefficient.

This routine calculates each of the three service factors and the total service for the user for the interval.

IRARMWM2 - Swappable User Evaluation Routine.
The IRARMWM2 routine scans the in-storage queue and the out-of-storage-but-ready queue, and evaluates each swappable user assigning each his current workload level.

IRARMWM3 - Individual User Evaluation Routine.
The IRARMWM3 routine evaluates swappable users on the IN and OUT queue, assigning each a current workload level.

IRARMWM4 - Workload Manager Workload Level Calculator Subroutine.
The IRARMWM4 routine accepts a service rate and a performance objective, and calculates the corresponding workload level.

IRARMWM5 - Workload Manager Update Performance Group Period Subroutine.
The IRARMWM5 subroutine tests whether an user has accumulated enough service/time to be assigned to a new performance group period. If so, IRARMWM5 adjusts the pointers which indicate the performance group period, performance objective and domain applicable to the transaction current for the user. Note that the frequency (resolution) at which the test for period end is made depends on how often IRARMWM5 is called for any given user.

IRARMWM7 - WLM Recommendation Calculation Routine.
The IRARMWM7 routine calculates a workload manager recommendation value for a user based on the service which was received and on the performance objective currently associated with the user. Users which have not yet received an amount of service equal to their interval service value (ISV) specification while in core are given a recommendation value boost. The boost gives preferential treatment to users in their ISV as compared to users not in their ISV and users between job steps.

IRARMHIT - Workload manager User Ready SYSEVENT Swap-In Scheduling Routine.
The IRARMHIT routine receives control as the result of a decision to apply swapin processing to a now ready user. It repositions the now ready user from the WAIT queue to the OUT queue.

IRARMWMI - Workload Manager In Storage Interval Change Subroutine.
The IRARMWMI subroutine updates the transaction accumulators with the service and the time received by the user during the preceding in-storage interval.

IRARMWMJ - Routine To Determine The Scope of Applicability of Analysis Processing to a User.
The IRARMWMJ routine examines the current swap status and the performance specification for a user. It indicates if the resource manager algorithms are applicable to this user.

IRARMWMK - WLM Dontswap/Okswap User Analysis Routine.
The IRARMWMK routine calculates the current service and ensures that the user is in the correct performance group period. Applicable algorithm indicators are set based on the new swap status of the user.

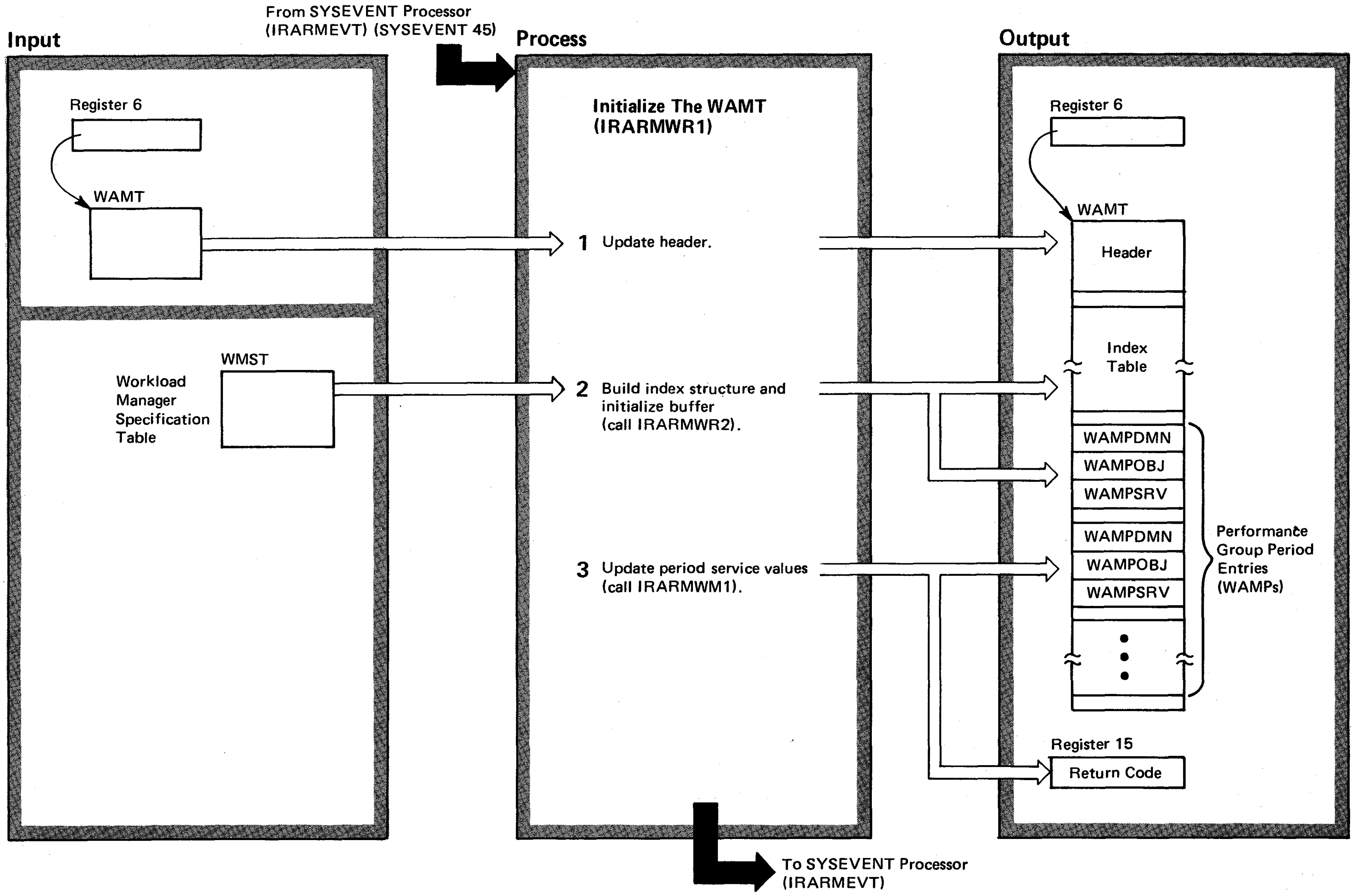
IRARMWMO - Workload Manager Transaction Start Routine.
The IRARMWMN routine receives control as the result of a SYSEVENT that has been defined by the workload manager to signify that a new transaction should be started for that user. If the user is not in storage, a flag is set to cause the IRARMWMN routine to be reentered during the swap-in of the user. Otherwise, any existing transaction is stopped by calling IRARMWMO, and the user transaction fields are reset to reflect the new transaction being started.

IRARMWMO - Workload Manager Transaction Stop Routine.
The IRARMWMO routine receives control as the result of a SYSEVENT that has been specified by the workload manager as defining the end of any current user transaction. If a new transaction is to be created for the user, IRARMWMO indicates the end of the current transaction. If the next user event is not known, IRARMWMO leaves the transaction accumulated values for later resumption of the transaction. In any case, IRARMWMO causes the preceding time and service to be properly recorded for the current transaction.

IRARMWMQ - Workload Manager Quiesce Completed SYSEVENT Processing Routine.
The IRARMWMQ routine receives control when a user has stopped executing, and is being swapped out, so that the workload manager may record the service given that user while he was in storage. The workload manager determines if a user event caused the swap-out, and flags the user to indicate whether such previous service is to be considered when the user is next swapped-in.

IRARMWMR - Workload Manager Restore Completed SYSEVENT Processing Routine.
The IRARMWMR routine receives control when a user has been swapped in, and is ready to begin executing, so that the workload manager can set up the fields used to calculate the service rate received by the user during the forthcoming in-storage residency period.

Diagram 6-22. Initialize for MF/1 (IRARMWR1) (Part 1 of 2)



1 Update header.

2 Build index structure and initialize buffer (call IRARMWR2).

3 Update period service values (call IRARMWM1).

Diagram 6-22. Initialize for MF/1 (IRARMWR1) (Part 2 of 2)

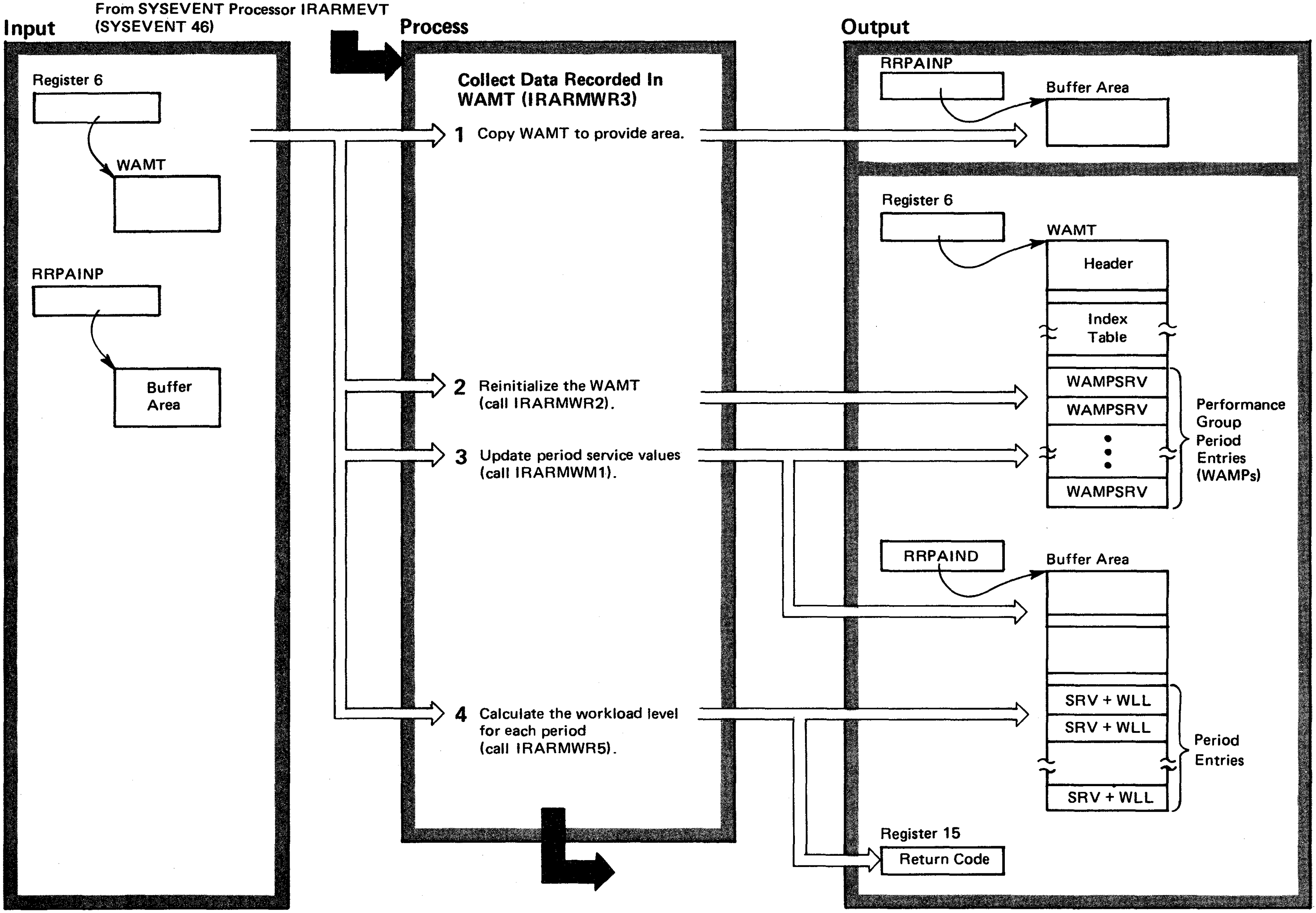
Extended Description	Module	Label
IRARMWR1 constructs and initializes the workload activity measurement table (WAMT) in the buffer (Storage from SQA) obtained by MF/1 and input with Sysevent 45.	IRARMWAR	IRARMWR1
1		
2 The index is used to locate the period entries in WAMT which correspond to a particular performance group. The period entries are updated with their respective domain and performance objective numbers. All other period entry values are zeroed.	IRARMWAR	IRARMWR2
3 Service values in the period entries are initialized such that service already received by active user transactions will not be included in the MF/1 interval service totals.		

Return Codes in Register 15 byte 3

X'00' – Data area accepted and initialized.

X'08' – Length of data area incorrect.

Diagram 6-23. Collect Data for MF/1 (IRARMWR3) (Part 1 of 2)



Collect Data Recorded In WAMT (IRARMWR3)

1 Copy WAMT to provide area.

2 Reinitialize the WAMT (call IRARMWR2).

3 Update period service values (call IRARMWM1).

4 Calculate the workload level for each period (call IRARMWR5).

Diagram 6-23. Collect Data for MF/1 (IRARMWR3) (Part 2 of 2)

Extended Description	Module	Label
IRARMWR3 copies the contents of the WAMT into a collection buffer. The buffer is obtained by MF/1 from LSQA and is fixed in core.	IRARMWAR	IRARMWR3
1 The WAMT is copied into the buffer.		
2 If a set to new IPS occurred, workload collection was terminated and the WAMT was updated to reflect the statistics at that point in time. If the IPS has not been changed, the WAMT is updated for a new collection interval.	IRARMWAR	IRARMWR2
3 Both the WAMT and the collection buffer are updated to reflect the actual service (SRV) received within each resp. interval.	IRARMWLM	IRARMWM1
4 The Workload Levels (WLL) are updated in the collection buffer for MF/1.		

Return codes in Register 15 byte 3

X'00' – Successful Data collection

X'04' – Successful Data collection, but an IPS change occurred terminating workload collection

X'40' – Data collection not active, or data buffer non-existent or copy buffer incorrect size.

IRARMWAR Module Entry Point Summary

IRARMWR1 - Workload Activity Recording Initialization Subroutine.

Constructs and initializes the Workload Activity Measurement Table (WAMT) in the buffer (storage from SQA obtained by MF/1 and input with SYSEVENT 45).

IRARMWR2 - Workload Activity Recording WAMT Initialization Subroutine.

Builds the WAMT in a format suitable for updating by the SRM.

IRARMWR3 - SRM Workload Activity Recording Data Collection Subroutine.

Moves the contents of the WAMT into a collection buffer capable of containing the data. Note: The buffer is obtained by MF/1 from LSQA, storage key 0, and must be fixed in storage.

If the IPS has not been changed, then add to the collected data the transaction data for the current in-storage interval for each in-storage memory with an active transaction re-initialize the data collection buffer for the next collection interval, and calculate the workload level for each performance group period that contains transaction data.

IRARMWR4 - SRM Workload Activity Recording Transaction Data Update Subroutine.

Adds the service and transaction active time to the appropriate WAMT performance group period accumulator in the data collection buffer.

IRARMWR5 - SRM Workload Activity Recording Workload Level Calculation Subroutine.

Calculates the workload level for each WAMT performance group period entry in which transaction data has been accumulated during the last data collection interval.

Note: For those WAMT entries in which the service rate calculated can be associated with multiple workload levels or is zero even though at least one transaction has been active during the data collection interval, the negative value of the workload level will be calculated to indicate to MF/1 an estimated value.

IRARMWR6 - SRM Workload Activity Recording Transaction End Update Subroutine.

Adds to the appropriate WAMT performance group period accumulator the transaction elapsed time and counts the number of transactions that terminated during the current data collection interval.

IRARMWR7 - SRM Workload Activity Recording WAMT Entry Determination Subroutine.

Obtains addressability to the WAMT performance group period entry in which to accumulate user transaction information.

IRARMWR8 - SRM Workload Activity Recording.

Terminates workload activity data collection whenever an IPS change occurs.

System Activities Measurement Facility (MF/1)

System Activities Measurement Facility (MF/1) collects information about system activity in order to produce System Management Facilities (SMF) data records or printed reports or both. MF/1 can monitor the following five classes of system activity:

1. CPU
2. Paging
3. Workload
4. Channel
5. Input/Output Device

MF/1 information collection can be initiated by the issuing of a START command and can be terminated either by the expiration of a specified collection period or by an operator STOP command. MF/1 is always generated with the system, but its execution is completely optional. When it is not operating, it causes little or no performance or storage overhead. When it is executing, storage and performance overhead depends on the set of control options under which it is running.

Options are available to specify the reporting of any of the five classes of system activity. In addition, the time interval for gathering and reporting measurements is an option. Channel and device data are sampled more frequently than once per measurement gathering interval, and the frequency of this sampling rate is an input option. Printed reports and/or SMF records can be obtained once per gathering interval or at the end of the period of MF/1 operation.

MF/1 has three major components:

1. Measurement Facility Control (MFC), which controls the collection, recording, and reporting of system activity measurements, in compliance with specified options.
2. System Activity Measurement Gathering (SAMG), which obtains measurements of system activity, by collecting data at timer interruptions and remote-pending IPC (interprocessor communication) interruptions, and which records measurements on the SMF data set.
3. System Activity Report Generation (SARG), which produces formatted, printed reports from system activity measurements.

Figure 3-11 in the Program Organization section illustrates the flow of control among the major components and main modules of MF/1.

The operator's START command causes MFC Mainline, the system task controlling MF/1 execution, to receive control. MFC Mainline establishes the operating parameters for MF/1 execution from specified options and loads the MG (measurement gathering) routines required by these parameters; it then enables the routing of control to these routines.

MFC Mainline passes control to interval-driven MG routines to gather measurements at a parameter-specified time interval. These routines move collected data into SMF-record-formatted areas and optionally record the data on the SMF data set.

System components outside MF/1 maintain data that is obtained by SAMG at measurement gathering intervals. These include:

1. System Measurement Facility (SMF), which maintains CPU wait time.
2. Real Storage Management (RSM), which maintains VIO paging statistics.
3. Auxiliary Storage Manager (ASM), which maintains auxiliary page statistics.
4. System Resources Manager (SRM), which maintains workload activity data.
5. Input/Output Supervisor (IOS), which maintains I/O activity statistics.

The SARG function is given control at reporting intervals to produce summary reports of the measurements obtained by SAMG and routed to it by MFC. These reports are routed to a SYSOUT data set, which is made available for printing at a parameter-specified time (either immediately or after MF/1 termination).

MFC allows the operator to use the STOP command to terminate MF/1, overriding any parameter-specified duration of execution. Otherwise, MFC terminates measurement gathering, recording, and reporting at the end of the specified duration.

Operational Considerations

MF/1 operation is controlled by input parameters. These parameters are obtained from four sources during MF/1 initialization:

1. START command PARM field.
2. EXEC statement PARM field (MF/1 cataloged procedure).
3. Partitioned data set number (the partitioned data set is specified in the cataloged

procedure — normally it will be SYS1.PARMLIB); the member name is of the form IRBMF1nn, where nn is an input parameter.
4. Program default values.

The parameters can be grouped into three classes:

1. Parameters affecting the initial parameter merge.
2. Parameters causing the loading of MG (measurement gathering) routines.
3. Parameters affecting the mechanics of MF/1 operation.

Class 1 consists of the following merge control keywords:

MEMBER (nn) - the value to be appended to IRBMF1 to name the member of the partitioned data set from which parameters are to be obtained during the input merge. (The default is 00, indicating member IRBMF100.

OPTIONS/NOOPTIONS - specifies whether or not the results of the input merge will be printed on the operator's console, to permit changes to be made. The default value is OPTIONS.

Class 2 consists of the following measurement gathering keywords. (Program default values are underlined.)

<u>CPU</u>/NOCPU
<u>PAGING</u>/NOPAGING
<u>CHAN</u>/NOCHAN
<u>WKLD</u>(<u>PERIOD</u>/GROUP/SYSTEM)/NOWKLD
<u>DEVICE</u>(device report list)/NODEVICE
where the device report list may consist of the following elements:
<u>UNITR</u>/NOUNITR
<u>TAPE</u>/NOTAPE
<u>DASD</u>/NODASD
<u>COMM</u>/NOCOMM
<u>GRAPH</u>/NOGRAPH
<u>CHRDR</u>/NOCHRDR

Specification of the first of any of the above sets of two measurement gathering keywords (CPU, PAGING, and so on.) causes the loading, during MF/1 initialization, of the interval MG routine associated with the keyword, so that reports and/or record copies are produced for that measurement type. If the second of any of the above sets of two is chosen (NOCPU, NOPAGING, and so on.), then no MG routines are loaded for this measurement type, and little or no performance or storage overhead is caused by these routines.

Class 3 consists of the remaining MF/1 keywords: (Program default values are underlined.)

<u>REPORT</u> (REALTIME/<u>DEFER</u>)/NOREPORT - specifies whether formatted reports are to be written to SYSOUT, and, if they are, whether they are to be printed when available, or at MF/1 termination.

SYSOUT (class) - specifies the SYSOUT class for all printed reports. The default is class A.

RECORD/<u>NORECORD</u> - specifies whether or not data records are to be written to the SMF data set at specified intervals.

INTERVAL (value/valueM) - specifies the length of time in minutes between gathering measurements and (optionally) preparing records and printing reports.

CYCLE (value) - specifies the frequency in milliseconds with which channel and device statistics are to be obtained within the specified interval.

<u>STOP</u> (value/valueM/valueH)/NOSTOP - specifies a length of time after which MF/1 will automatically terminate or, alternatively, that MF/1 can only be terminated by an operator's STOP command. The default value is 15M.

Measurement Facility Control (MFC)

MFC is the system task controlling MF/1 operation. It performs the input merge to establish the parameters controlling MF/1, initializes for MF/1 data collection by loading the appropriate MG routines, issues SVC MF DATA A at reporting intervals to initiate data collection for active report printing, and controls MF/1 termination.

See the following method-of-operation diagrams:
Measurement Facility Control (MFC) Mainline (IRBMFMFC)
MFSTART Mainline (IGX00013))
Input Merge Control (IRBMFINP)
Syntax Analyzer (IRBMFANL)
List Option Module (MFLISTOP)
Initialization Mainline (MFIMAINL)
CPU Activity Initialization (IRBMFICP) or Paging Activity Initialization (IRBMFIPG)
Workload Initialization (IRBMFIWK)
Channel Initialization (IRBMFIHA)
Device Initialization (IRBMFIDV)
Data Control (IRBMFDTA)
Termination Processor (IRBMFTMA)
MF/1 Message Processor (IRBMFMPR)

System Activity Measurement Gathering (SAMG)

SAMG consists of a set of measurement gathering (MG) routines whose function is to collect data from the various system components for eventual reporting through SARG, and to copy the information to the SMF data set if so required by the RECORD/NORECORD option. There are two classes of MG routines—interval MG routines and cycle MG routines. There is one interval MG routine associated with each active reporting class; it is activated at reporting intervals (as determined by the INTERVAL keyword to collect interval measurements and, optionally, copy the SMF record. Cycle MG routines are associated with the device and channel reporting classes. If active, they are entered at periods determined by the CYCLE keyword to sample queue lengths and maintain other intermediate device and channel-related data that the related interval MG routines collect at reporting intervals.

See the following method-of-operation diagrams:

MFDATA SVC Mainline (IGX00014)
Interval MG Routine for CPU (IRBMFDCP)
Interval MG Routine for Paging (IRBMFDPP)
Interval Routine for Workload (IRBMFDWP)
Interval MG Routine for Channels (IRBMFDHP)
Interval MG Routine for Devices (IRBMFDDP)
MFROUTER SVC Processor (IRBMFEVT)
Channel Sampling Module (IRBMFECH)
Second CPU Test Channel Sampling Module (IRBMFTCH)
Device Sampling Module (IRBMFEDV)

System Activity Report Generating (SARG)

SARG produces all the formatted reports about the activities being monitored. These reports are made available for printing at a time specified in the REPORT parameter.

See the following method-of-operation diagrams:

Report Generator Control (IRBMFRGM)
Report Generators for CPU, Paging, Workload, Channels, and Devices (IRBMFRCR, IRBMFRPR, IRBMFRWR, IRBMFRHR, and IRBMFRDR)

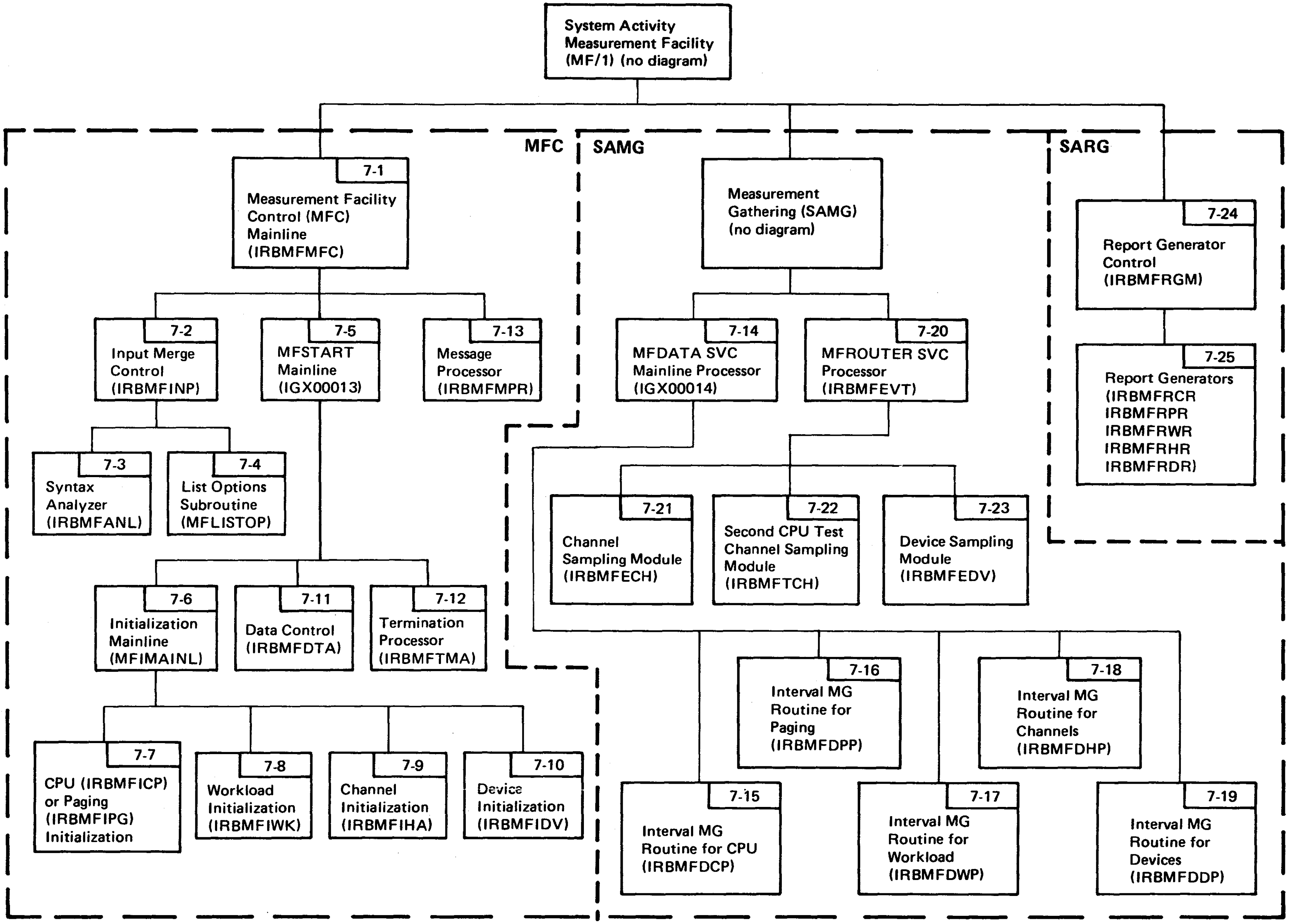
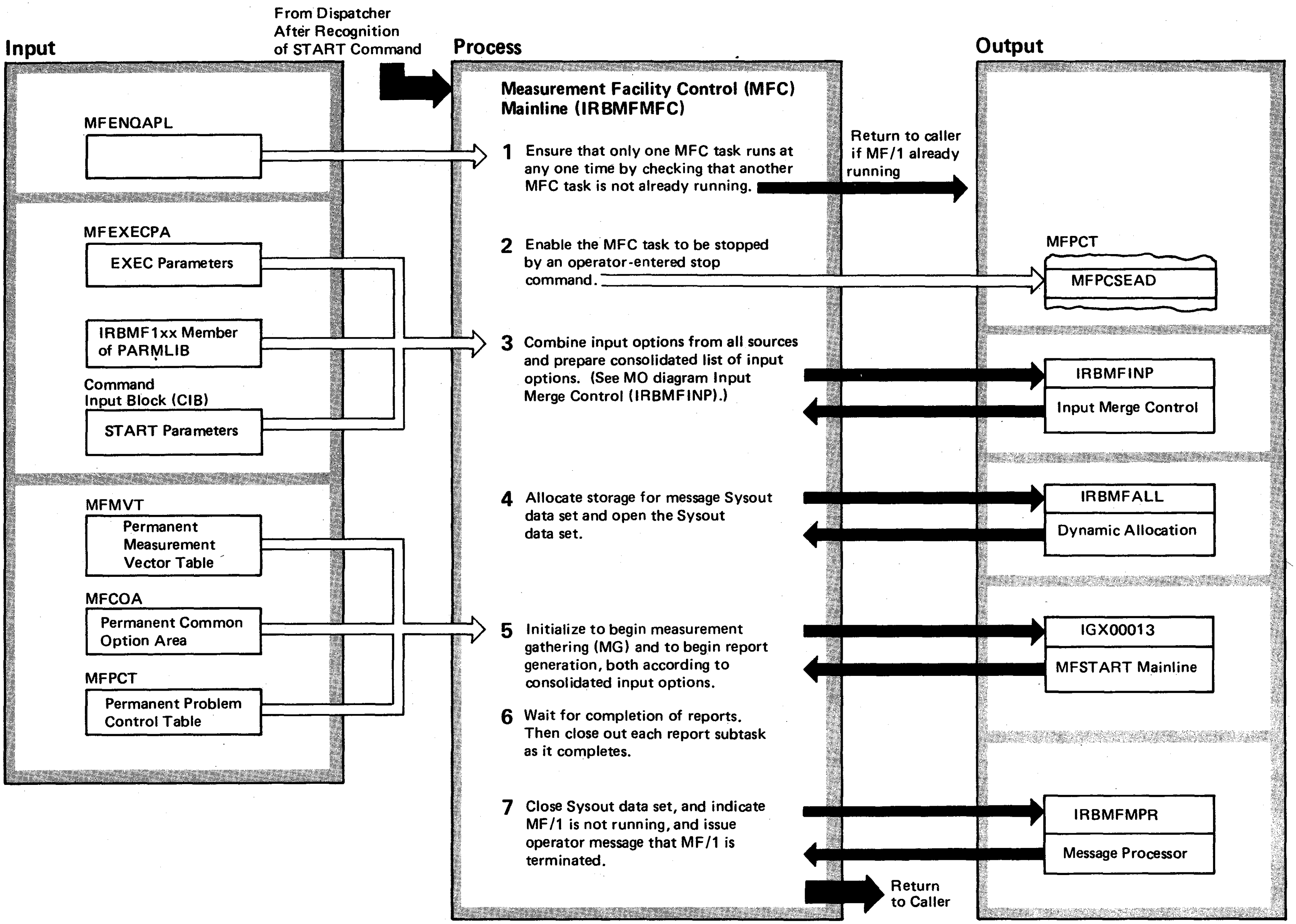


Figure 2-10. System Activity Measurement Facility (MF/1) Visual Contents

Diagram 7-1. Measurement Facility Control (MFC) Mainline (IRBMFMFC) (Part 1 of 2)



Measurement Facility Control (MFC) Mainline (IRBMFMFC)

1 Ensure that only one MFC task runs at any one time by checking that another MFC task is not already running.

2 Enable the MFC task to be stopped by an operator-entered stop command.

3 Combine input options from all sources and prepare consolidated list of input options. (See MO diagram Input Merge Control (IRBMFINP).)

4 Allocate storage for message Sysout data set and open the Sysout data set.

5 Initialize to begin measurement gathering (MG) and to begin report generation, both according to consolidated input options.

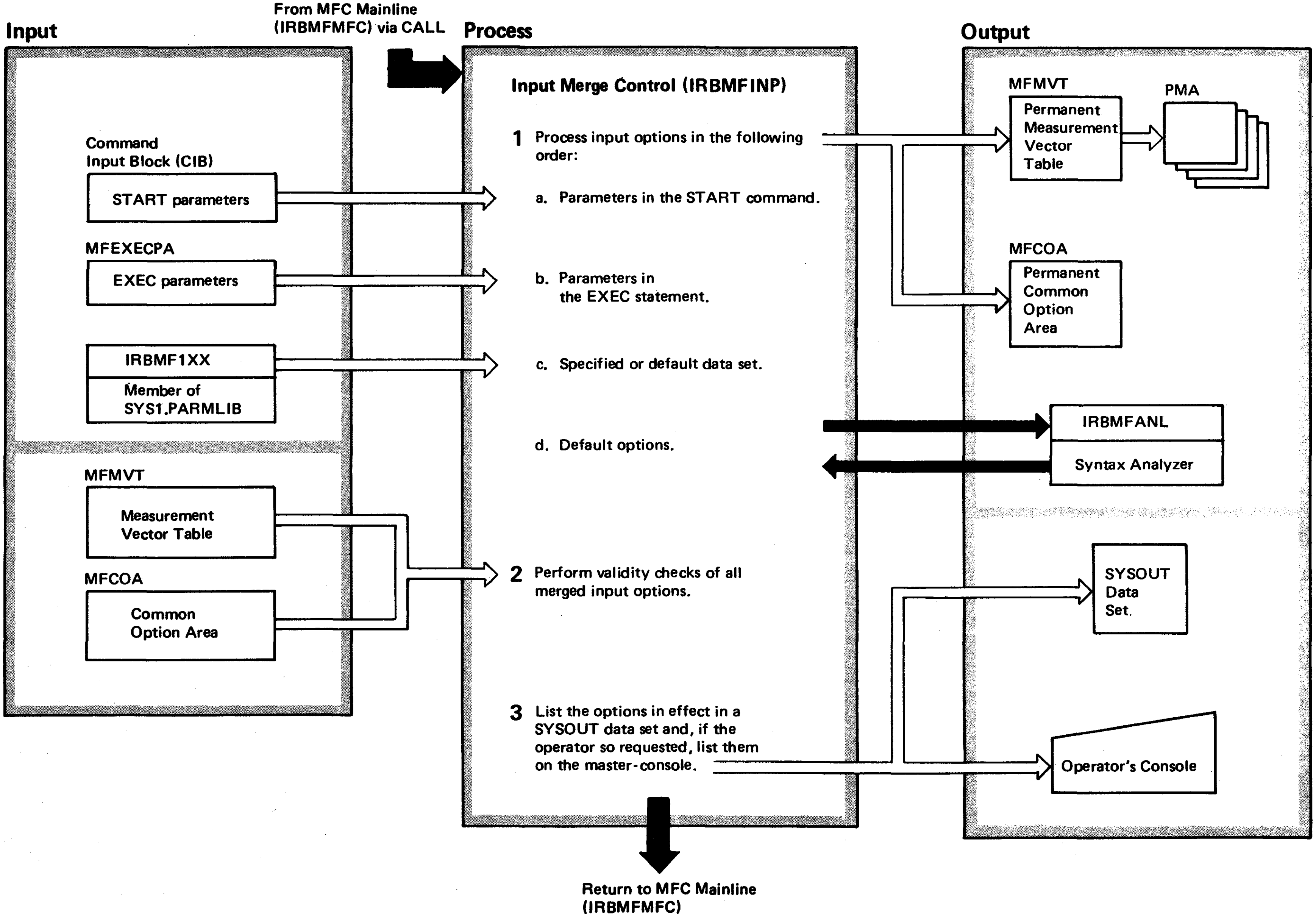
6 Wait for completion of reports. Then close out each report subtask as it completes.

7 Close Sysout data set, and indicate MF/1 is not running, and issue operator message that MF/1 is terminated.

Diagram 7-1. Measurement Facility Control (MFC) Mainline (IRBMFMFC) (Part 2 of 2)

Extended Description	Module	Label
The Measurement Facility Control (MFC) Mainline module (IRBMFMFC) is the first MF/1 module to receive control as a result of the operator starting MF/1. Its main functions are to initialize MF/1 option control blocks, to issue an SVC to cause monitoring of system variables by MF/1, and to assist in terminating MF/1.	IRBMFMFC	
1 The return from an enqueue on a global name indicates whether another MFC Mainline task is dispatched, even if it has been dispatched in another virtual area. If so, this MFC Mainline task issues an operator message and then terminates.	IRBMFMFC	
2 MFC Mainline obtains an ECB address so Data Control (IRBMFDTA) can later accept a stop command from the operator.		
3 MFC Mainline loads and calls Input Merge (IRBMFINP) to merge input options, analyze the syntax of these options, and indicates the option values specified in the PMA and COA.	IRBMFINP	
4 MFC Mainline calls the Dynamic Allocation module (IRBMFALL) to allocate storage for the Sysout data set and then opens the data set.	IRBMFALL	
5 Issue SVC MFSTART to initialize the monitoring, reporting and recording of system measurements by MF/1. Control will not return to this point until MF/1 is ready to terminate (when the specified duration is reached, or MF/1 is stopped by the operator).	IGX00013	
6 Detach each SARG subtask, and free all associated control blocks.	IRBMFTMA	
7 Issue CLOSE to close the message SYSOUT data set, and issue DEQ to indicate that MF/1 is not active. Call the message processor (IRBMFMPR) to indicate termination to the system operator.	IRBMFMFC	

Diagram 7-2. Input Merge Control (IRBMFINP) (Part 1 of 2)



1 Process input options in the following order:

a. Parameters in the START command.

b. Parameters in the EXEC statement.

c. Specified or default data set.

d. Default options.

2 Perform validity checks of all merged input options.

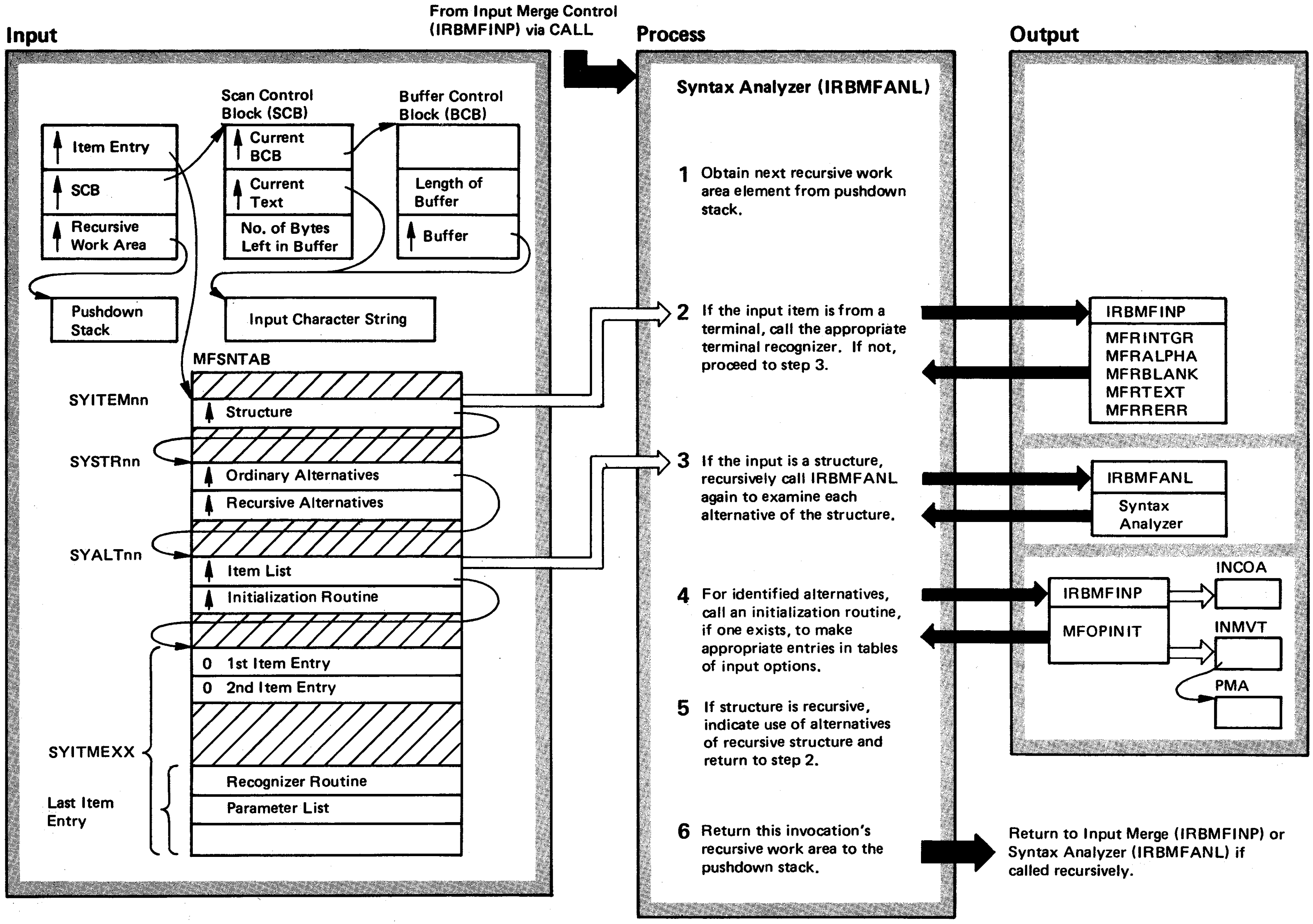
3 List the options in effect in a SYSOUT data set and, if the operator so requested, list them on the master-console.

Diagram 7-2. Input Merge Control (IRBMFINP) (Part 2 of 2)

Extended Description	**Module**	**Label**
The Input Merge Control module (IRBMFINP) receives control from MFC Mainline (IRBMFMFC) early in the execution of IRBMFMFC. IRBMFINP controls the preparation of tables that represent consolidated input parameters and controls the printing of this set of input parameters if required either by an input request or because of invalid or conflicting inputs. Inputs to IRBMFINP are from Start command parameters, EXEC statement parameters, input-specified data set parameters, or from default options. IRBMFINP controls the interpretation of inputs, the merging of these inputs into a set of control blocks that indicate the requested options, and the communication with the console operator to obtain approval or correction of the list of options prepared in response to inputs. The four input sources have the following priority, from highest to lowest: START command, EXEC statement, member data set, and default.	IRBMFINP	

Extended Description	**Module**	**Label**
1 The processing of input options from each input source is essentially similar: for each source, the input data is separated into recognized fields and initialized in temporary control blocks, and then these fields are merged into permanent control blocks that represent input options combined from all sources of input. Routines check for invalid values in input data, for mutually exclusive options, and for syntax errors. Such errors are reported to the operator, who may request new options or instruct the program to ignore the inputs (if ignore is requested, lower priority inputs or defaults are used).	IRBMFINP IRBMFMPR	MFMERGE MFSRCPRO
2 Routine MFVALCHK performs the following validity checks between different option types: a) The report option must not be set to DEFER if NOSTOP is requested. b) Options NOREPORT and NORECORD must not both be set. c) The STOP value must be such that the time in operation is not less than the INTERVAL value.	IRBMFINP	MFVALCHK
3 The set of options is written to a SYSOUT data set and if so requested is printed on the console. Subroutine MFLISTOP writes to SYSOUT, and to the console if required. (See List Option Subroutine (MFLISTOP) diagram.)	IRBMFINP	MFLISTOP

Diagram 7-3. Syntax Analyzer (IRBMFANL) (Part 1 of 2)



1 Obtain next recursive work area element from pushdown stack.

2 If the input item is from a terminal, call the appropriate terminal recognizer. If not, proceed to step 3.

3 If the input is a structure, recursively call IRBMFANL again to examine each alternative of the structure.

4 For identified alternatives, call an initialization routine, if one exists, to make appropriate entries in tables of input options.

5 If structure is recursive, indicate use of alternatives of recursive structure and return to step 2.

6 Return this invocation's recursive work area to the pushdown stack.

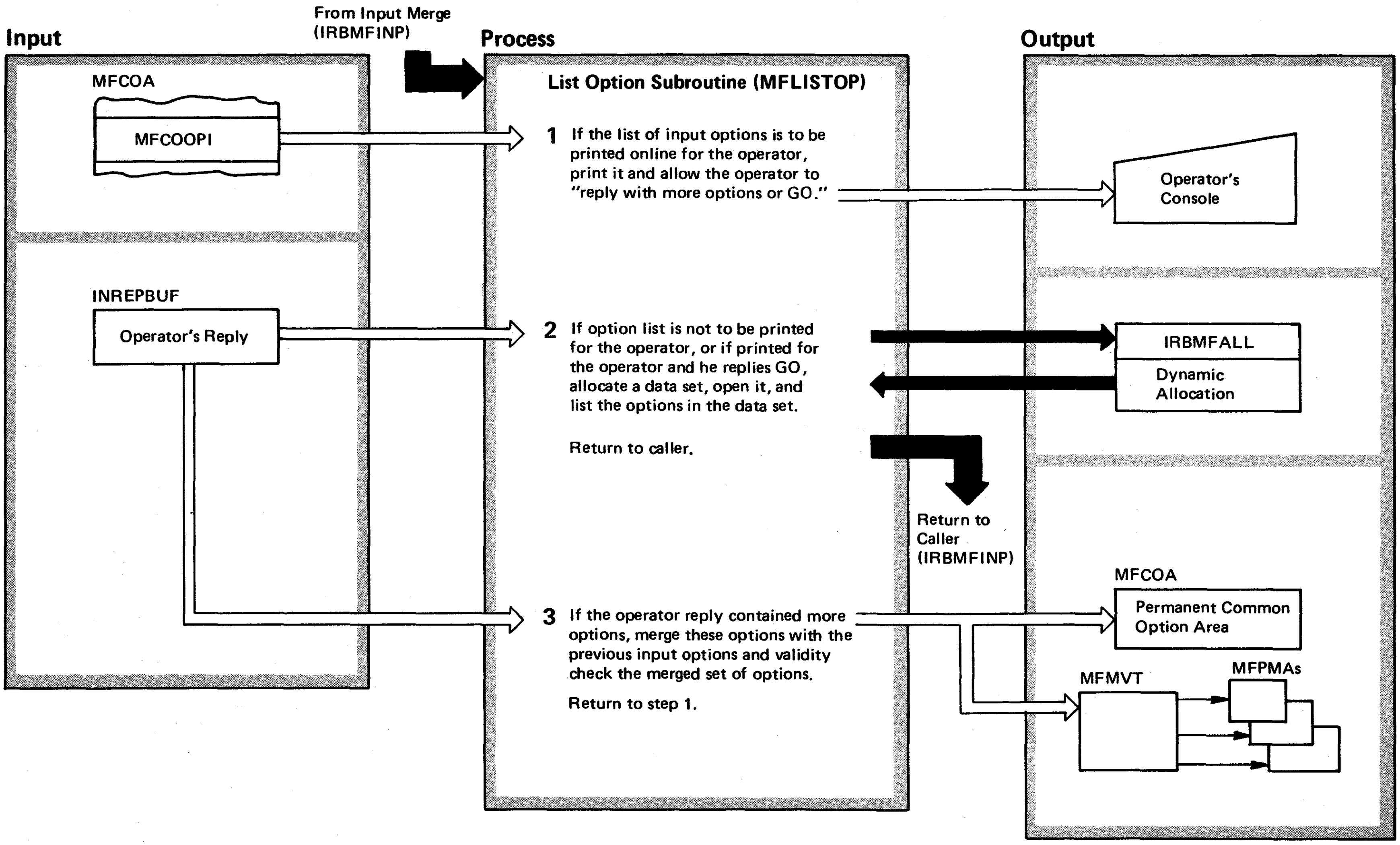
Return to Input Merge (IRBMFINP) or Syntax Analyzer (IRBMFANL) if called recursively.

Diagram 7-3. Syntax Analyzer (IRBMFANL) (Part 2 of 2)

Extended Description	Module	Label
The Syntax Analyzer parses the string of characters from an input source and attempts to identify the intended set of input options. If successful, the Syntax Analyzer builds these options into control blocks to be merged with input options from other sources. The general logic flow of the Syntax Analyzer is to scan the source of input and attempt to establish correspondence between the input and one of the valid inputs described in the Syntax table (MFSYNTAB). Essential to the recognition of valid options is MFSYNTAB, which defines all possible valid inputs in terms of structures, alternatives, items, and terminals. A structure is a high-level description of an option. A structure points to a list of alternatives. Each alternative describes a possible way the input could appear in a character string. Each alternative is made up of one or more items. An item is a terminal item (called simply a terminal) if the item is a string of characters to be matched in the input. Or, an item may point to another structure, which in turn, is made up of alternatives, each of which has one or more items. The Syntax Analyzer examines each item one by one, and, if all items of an alternative consist of valid terminals, appropriate entries are made in control blocks that represent input options from this source. If any item points to a structure, the alternatives of that structure are examined, item by item, until terminals can be compared on a character for character basis to establish valid items. Valid items are used to establish valid alternatives. When a valid alternative is established no other alternative of the structure needs to be examined. The Syntax Analyzer uses a Scan Control Block (SCB) to keep track of where in the input string the current comparison is being made.	IFBMFANL	

Extended Description	Module	Label
1 The address of the pushdown stack is provided as an input to the syntax analyzer on each invocation.	IRBMFANL	
2 The input item entry identifies the appropriate recognizer to call.	IRBMFINP	MFRINTGR MFRALPHA MFRBLANK MFRTEXT MFRRERR
3 The initial word of each item entry contains a bit that identifies the item as a terminal (that is, a character string capable of verification by a terminal recognizer) or as part of a structure. If the item is part of another structure, the second word of the item entry contains the address of the structure entry (SYSTRnn). The structure entry points to lists of alternatives (SYALTnn), each of which point to item lists (SYLTMExx).	IRBMFANL	
4 The third word in each alternative entry contains the address of the Initialization routine MFOPINIT, if initialization is to be done.	IRBMFINP	MFOPINIT
5 A recursive bit is located in the first word of each structure entry. If the bit is on, a recursive list of alternatives is examined.		
6 On return to the SYNTAX ANALYZER on a recursive invocation, this invocation's recursive workarea is returned to the pushdown stack.	IRBMFANL	

Diagram 7-4. List Option Subroutine (MFLISTOP) (Part 1 of 2)



List Option Subroutine (MFLISTOP)

1 If the list of input options is to be printed online for the operator, print it and allow the operator to "reply with more options or GO."

2 If option list is not to be printed for the operator, or if printed for the operator and he replies GO, allocate a data set, open it, and list the options in the data set.

Return to caller.

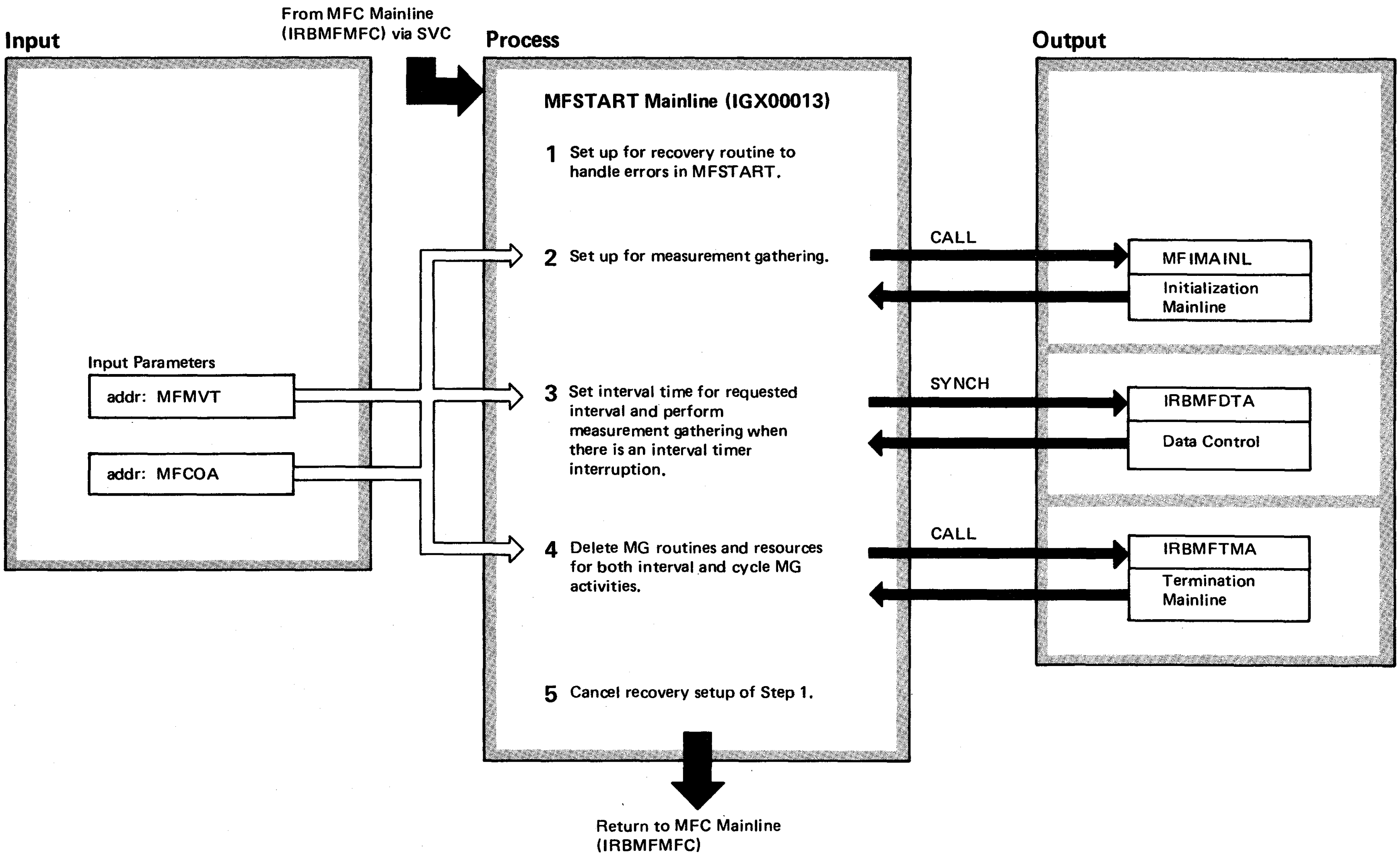
3 If the operator reply contained more options, merge these options with the previous input options and validity check the merged set of options.

Return to step 1.

Diagram 7-4. List Option Subroutine (MFLISTOP) (Part 2 of 2)

Extended Description	Module	Label
The List Option module (MFLISTOP) lists options, as requested by the input source or in response to errors in specifying the MF/1 options.	IRBMFINP	MFLISTOP
1 If the OPTN option (list input options on the console) is set on, either by direct operator request as an input option or as a result of an error in input options, the list is printed.	IRBMFINP IRBMFMPR	MFLISTOP
2 Routine IRBMFALL is used to allocate space for listing the SYSOUT data set. If the opening of the data set is successful, the final list of options in effect is written to the data set.	IRBMFALL	
3 If the operator replied with more options in response to REPLY WITH MORE OPTIONS OR GO, his reply is scanned for errors and control blocks (permanent) are initialized. This cycle of actions continues until he replies GO to the message, then a final list of options is written to the SYSOUT data set.	IRBMFINP IRBMFANL	MFSRCPRO MFVALCHK

Diagram 7-5. MFSTART Mainline (IGX00013) (Part 1 of 2)



1 Set up for recovery routine to handle errors in MFSTART.

2 Set up for measurement gathering.

3 Set interval time for requested interval and perform measurement gathering when there is an interval timer interruption.

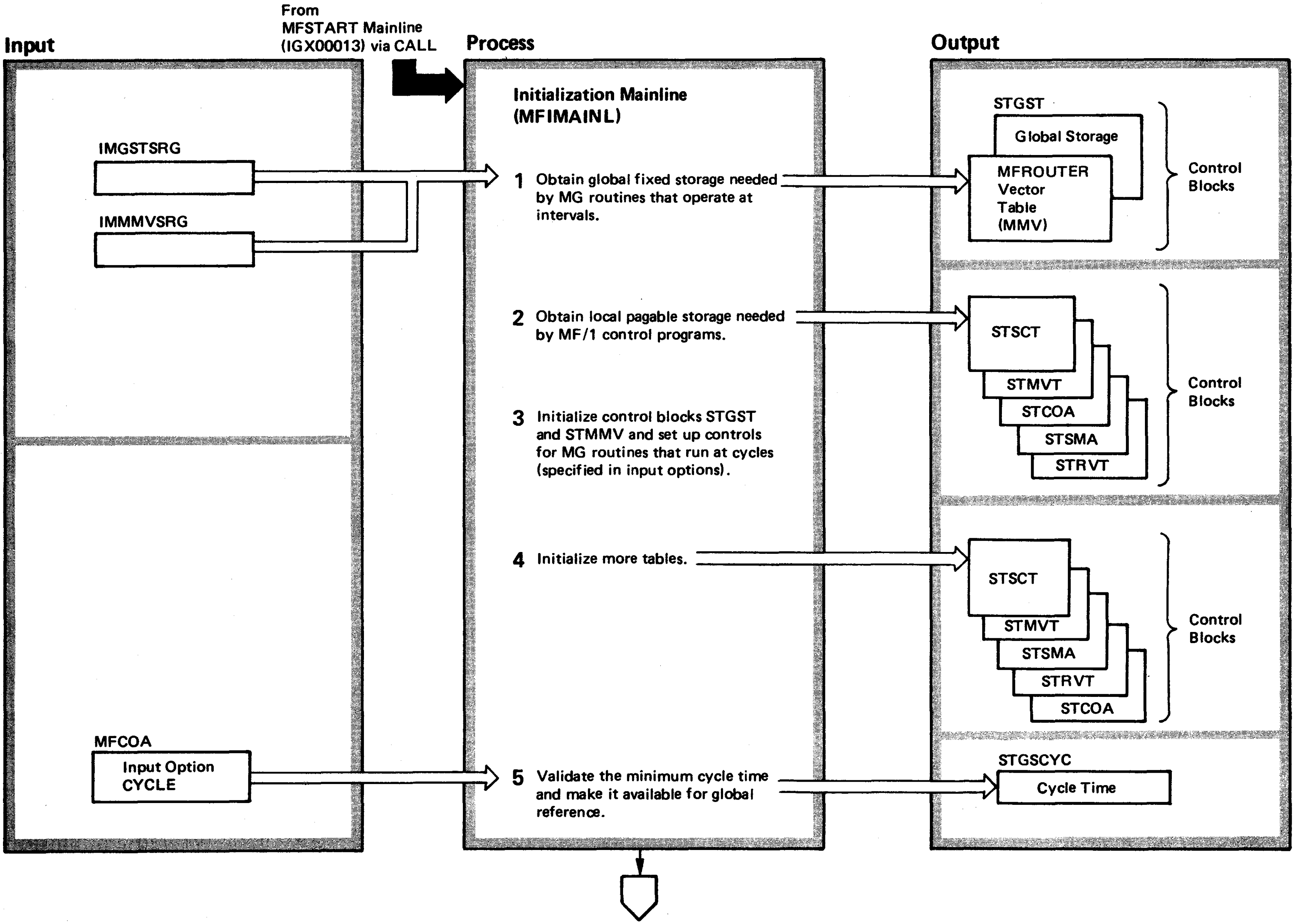
4 Delete MG routines and resources for both interval and cycle MG activities.

5 Cancel recovery setup of Step 1.

Diagram 7-5. MFSTART Mainline (IGX00013) (Part 2 of 2)

Extended Description	Module	Label
The MFSTART Mainline (IGX00013) processor controls the initialization and termination of routines that perform MF/1 functions.	IGX00013	
1 Issue an ESTAE macro instruction to provide entry to routine IRBMFSDE, which receives control in event of MF/1 errors.	IGX00013 IRBMFSDE	
2 Call the Initialization routine (MFIMAINL), which, in turn, calls other initialization routines (see the first paragraph in the MFC Mainline (IRBMFMFC) M.O. diagram).		MFIMAINL
3 Use SYNCH macro instruction to change to problem state and to transfer control to the Data Control routine (IRBMFDTA), which sets the interval timer and initiates measurement gathering after each interval.	IRBMFDTA	
4 After the last interval, Data Control returns control to MFSTART Mainline, which calls Termination Mainline (IRBMFTMA).	IRBMFTMA	
5 MFSTART Mainline cancels the ESTAE routine entry.		

Diagram 7-6. Initialization Mainline (MFIMAINL) (Part 1 of 6)



1 Obtain global fixed storage needed by MG routines that operate at intervals.

2 Obtain local pagable storage needed by MF/1 control programs.

3 Initialize control blocks STGST and STMMV and set up controls for MG routines that run at cycles (specified in input options).

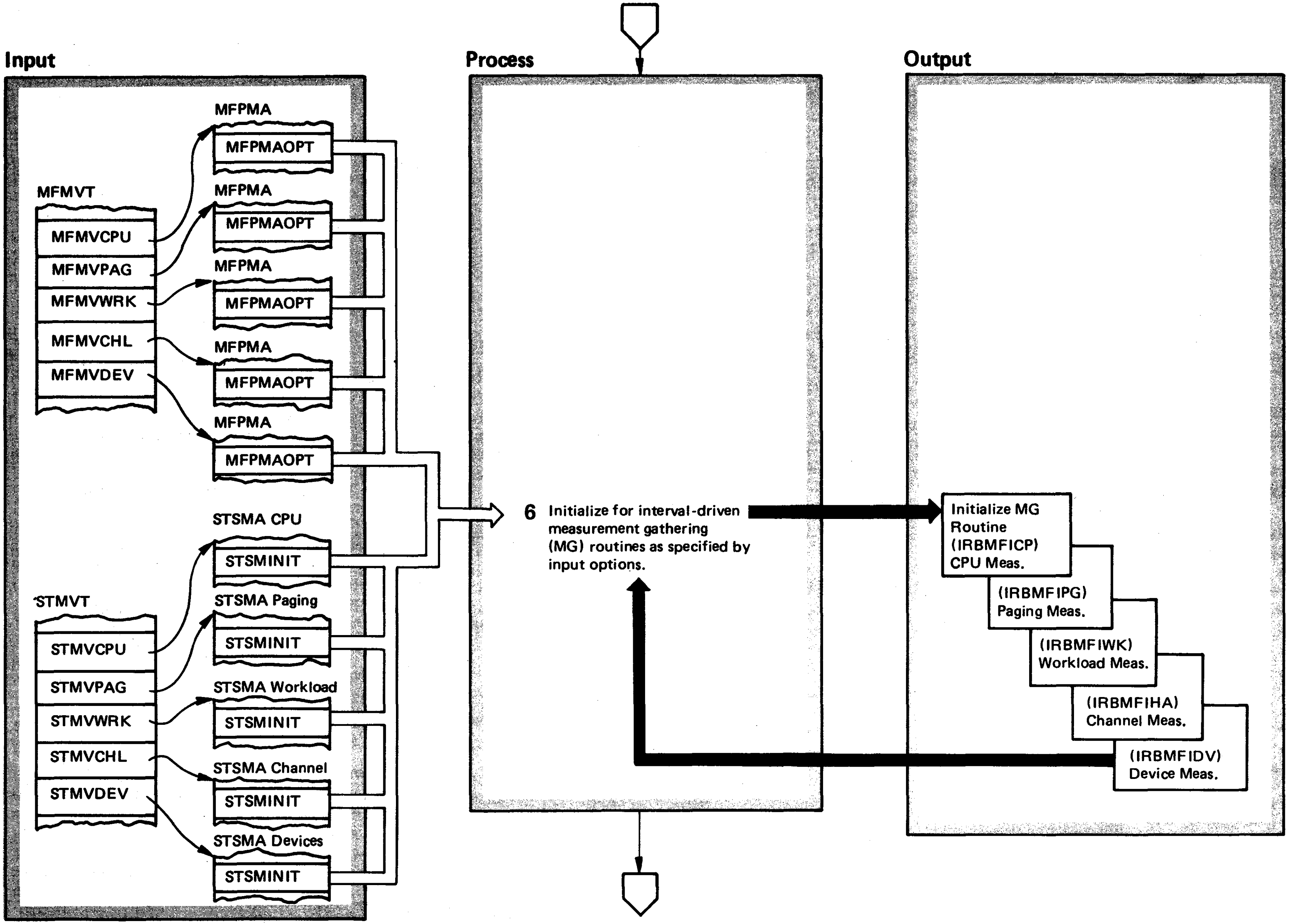
4 Initialize more tables.

5 Validate the minimum cycle time and make it available for global reference.

Diagram 7-6. Initialization Mainline (MFIMAINL) (Part 2 of 6)

Extended Description	**Module**	**Label**
The Initialization Mainline (MFIMAINL) procedure controls the allocation of space for and the initialization of control blocks. It also calls routines whose purposes are to initialize different functions essential to measurement gathering (MG). Finally, it issues the MFDATA SVC to collect initial values of requested measurements.	IGX00013	MFIMAINL
1 MFIMAINL uses the GETMAIN macro instruction to obtain storage for the Global Storage Table (STGST) and for the MFROUTER (control routine for sample collecting routines) Vector Table (STMMV).	IGX00013	MFIMAINL
2 MFIMAINL uses the GETMAIN macro instruction to obtain storage for the Supervisor Control Table (STSCT), Measurement Vector Table (STMVT), the Common Option Area (STCOA), Supervisor Measurement Area (STSMA), and the Resource Vector Table (STRVT).	IGX00013	
3 MFIMAINL places initial values into the control blocks for which space was obtained in step 1.	IGX00013	
4 MFIMAINL places initial values into the control blocks for which space was obtained in step 2.	IGX00013	
5 The time specified by the cycle input option must not be less than 50 milliseconds.	IGX00013	

Diagram 7-6. Initialization Mainline (MFIMAINL) (Part 3 of 6)

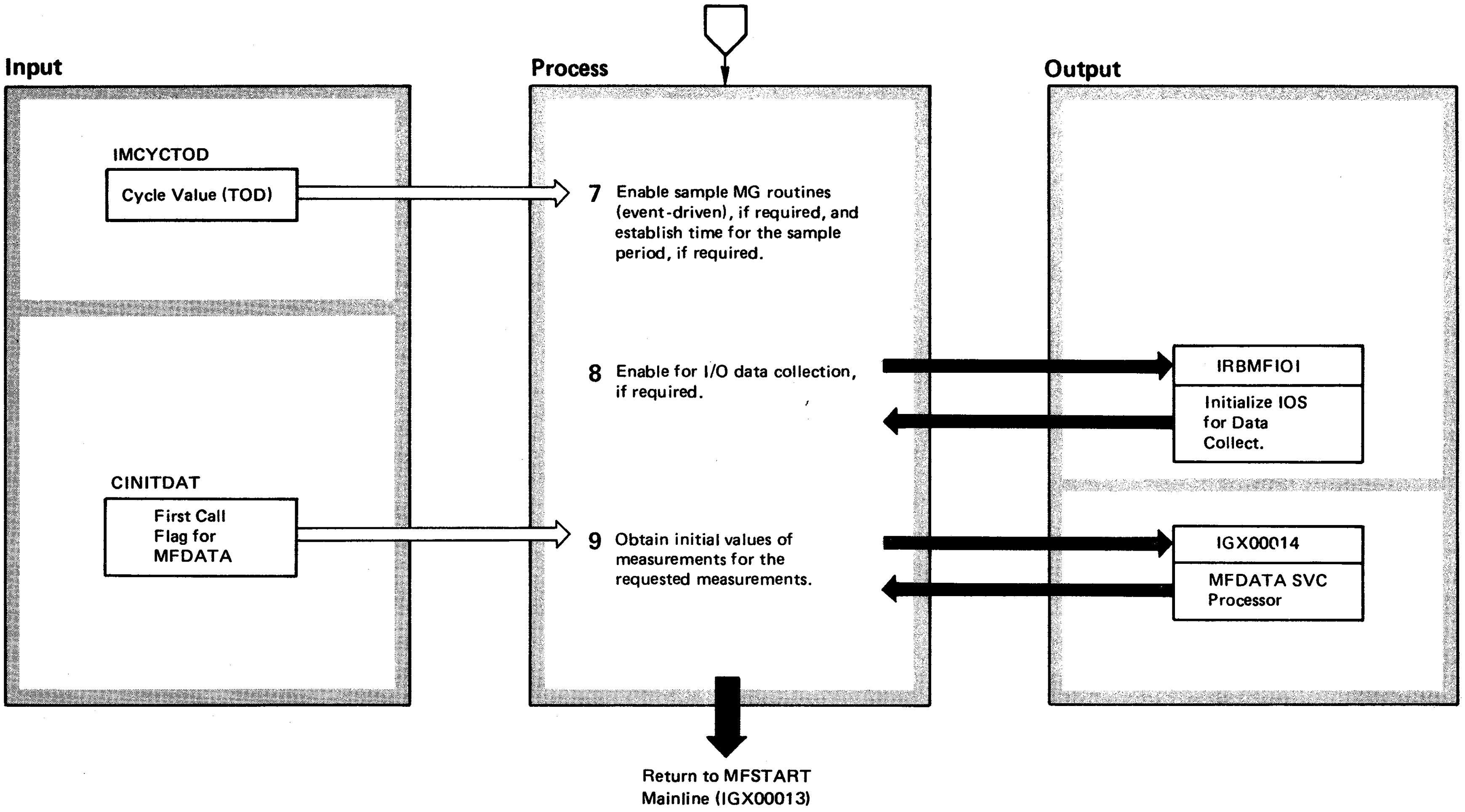


6 Initialize for interval-driven measurement gathering (MG) routines as specified by input options.

Diagram 7-6. Initialization Mainline (MFIMAINL) (Part 4 of 6)

Extended Description	Module	Label
6 MFIMAINL calls the routines that initialize the MG routines. Only those MG routines required for the requested kinds of reports are called. For example, if CPU is the only requested report, then IRBMFICP is the only MG routine called.	IRBMFICP	
	IRBMFIPG	
	IRBMFIWK	
	IRBMFIHA	
	IRBMFIDV	

Diagram 7-6. Initialization Mainline (MFIMAINL) (Part 5 of 6)



7 Enable sample MG routines (event-driven), if required, and establish time for the sample period, if required.

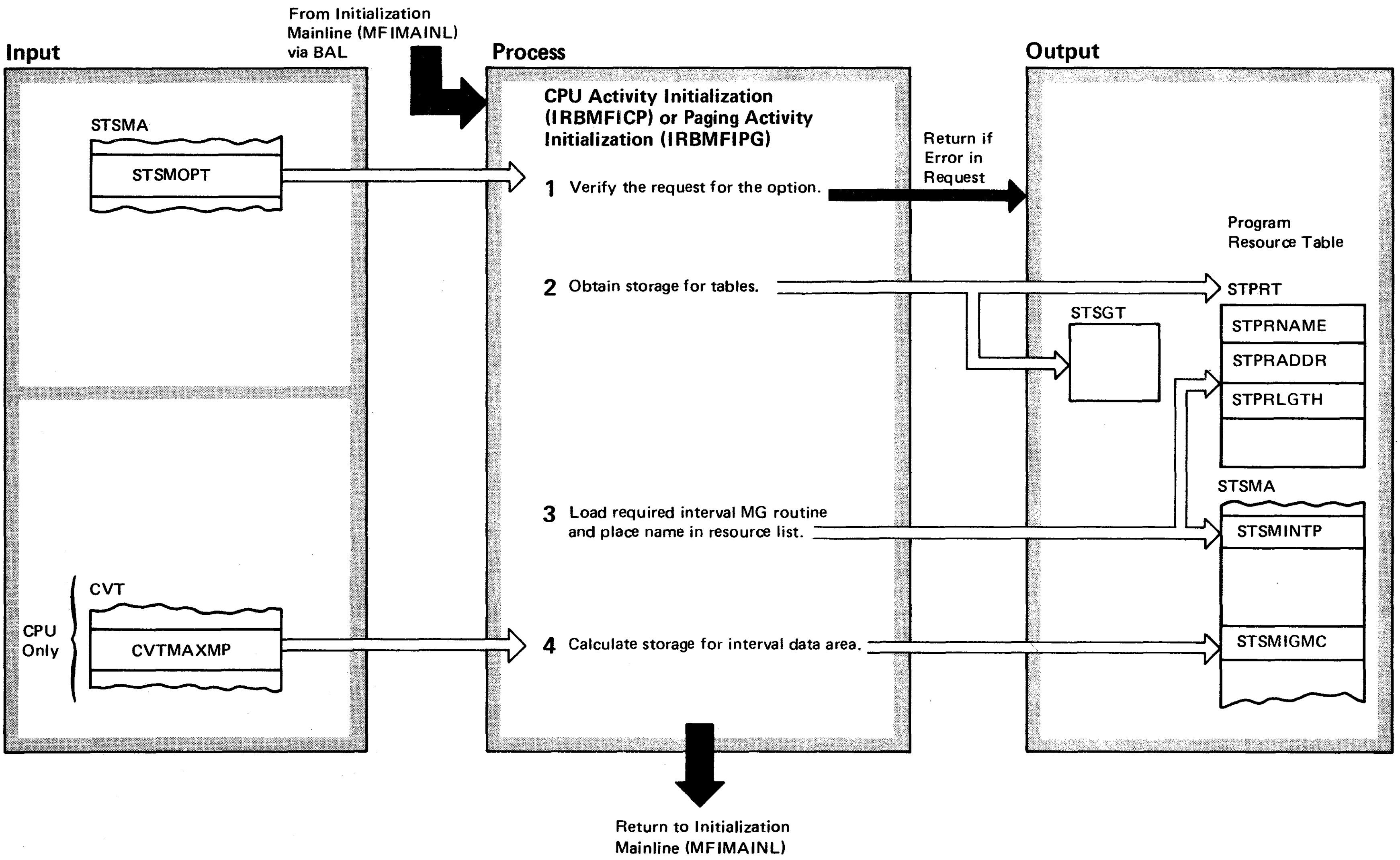
8 Enable for I/O data collection, if required.

9 Obtain initial values of measurements for the requested measurements.

Diagram 7-6. Initialization Mainline (MFIMAINL) (Part 6 of 6)

Extended Description	Module	Label
7 If channel or device reports are requested, MFIMAINL sets a flag in the Communications Vector Table (CVT) in CVT item CVTMFACT. MFIMAINL also puts the time of the next sample into MF/1's Timer Quene Element (TQE). Before calling Routine IEAQTE00 to enqueue the TQE on the timer queue, MFIMAINL obtains the dispatcher lock and establishes a Functional Recovery Routine (FRR) exit; after setting the TQE, these actions are reversed.	IGX00013 IEAQTE00	
8 MFIMAINL calls routine IRBMFIOI to change instructions in the system IOS functions so that channel and device measurements are collected as IOS operates.	IRBMFIOI	
9 MFIMAINL issues the MFDATA SVC (SVC 109, code 14) to collect data as requested by input options. This first call to each, however, is indicated as the initial call and results in taking initial values against which later values are compared.	IGX00014	

Diagram 7-7. CPU Activity Initialization (IRBMFICP) or Paging Activity Initialization (IRBMFIPG) (Part 1 of 2)



CPU Activity Initialization (IRBMFICP) or Paging Activity Initialization (IRBMFIPG)

1 Verify the request for the option.

2 Obtain storage for tables.

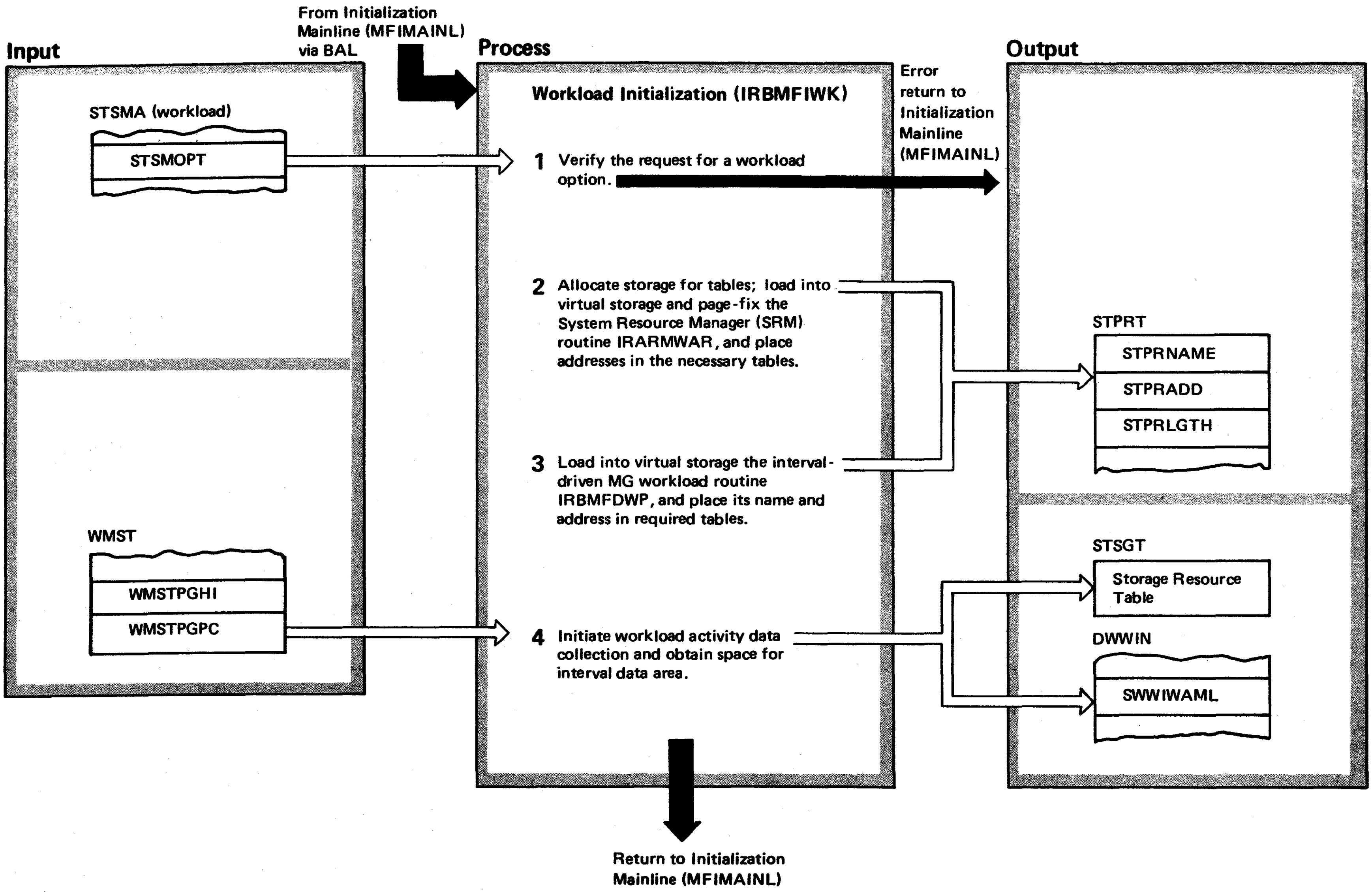
3 Load required interval MG routine and place name in resource list.

4 Calculate storage for interval data area.

Diagram 7-7. CPU Activity Initialization (IRBMFICP) or Paging Activity Initialization (IRBMFIPG) (Part 2 of 2)

Extended Description	Module	Label
The CPU Initialization (IRBMFICP) and the Paging Initialization (IRBMFIPG) both have very similar functions, inputs, and outputs. Therefore, one M.O. diagram is used to describe the functions of both. IRBMFICP and IRBMFIPG allocate storage space for control blocks, ensure that copies of the required interval MG routine are in the virtual storage space, and calculate the length of the required data area.	IRBMFICP or IRBMFIPG	
1 The CPU or Paging Initialization routine ensures that the input option has been specified by checking the STSMSTA bit in the STSMOPT word of the Supervisor Measurement Area (STSMA).	IRBMFJCP or IRBMFIPG	
2 The CPU or Paging Initialization routine uses the GETMAIN macro instruction to obtain the necessary storage.	IRBMFJCP or IRBMFIPG	
3 After adding the entry to the Program Resource Table (STPRT), the initialization routine indicates in the Resource Vector Table (STRVT) the next available entry in the STPRT. The entry point address is placed in the System Measurement Area (STSMA) for use by the MFDATA SVC Processor (IGX00014).	IRBMFJCP or IRBMFIPG	
4 The storage length for CPU data is: 4 + length of (SMFRCD70) + length of (SMF70A) + (CVTMAXMP + 1) times length of (SMF70B). The storage length for paging data is: 4 + length of (SMFRCD71) + length of (SMF71A) + length of (SMF71B).		

Diagram 7-8. Workload Initialization (IRBMFIWK) (Part 1 of 2)



1 Verify the request for a workload option.

2 Allocate storage for tables; load into virtual storage and page-fix the System Resource Manager (SRM) routine IRARMWAR, and place addresses in the necessary tables.

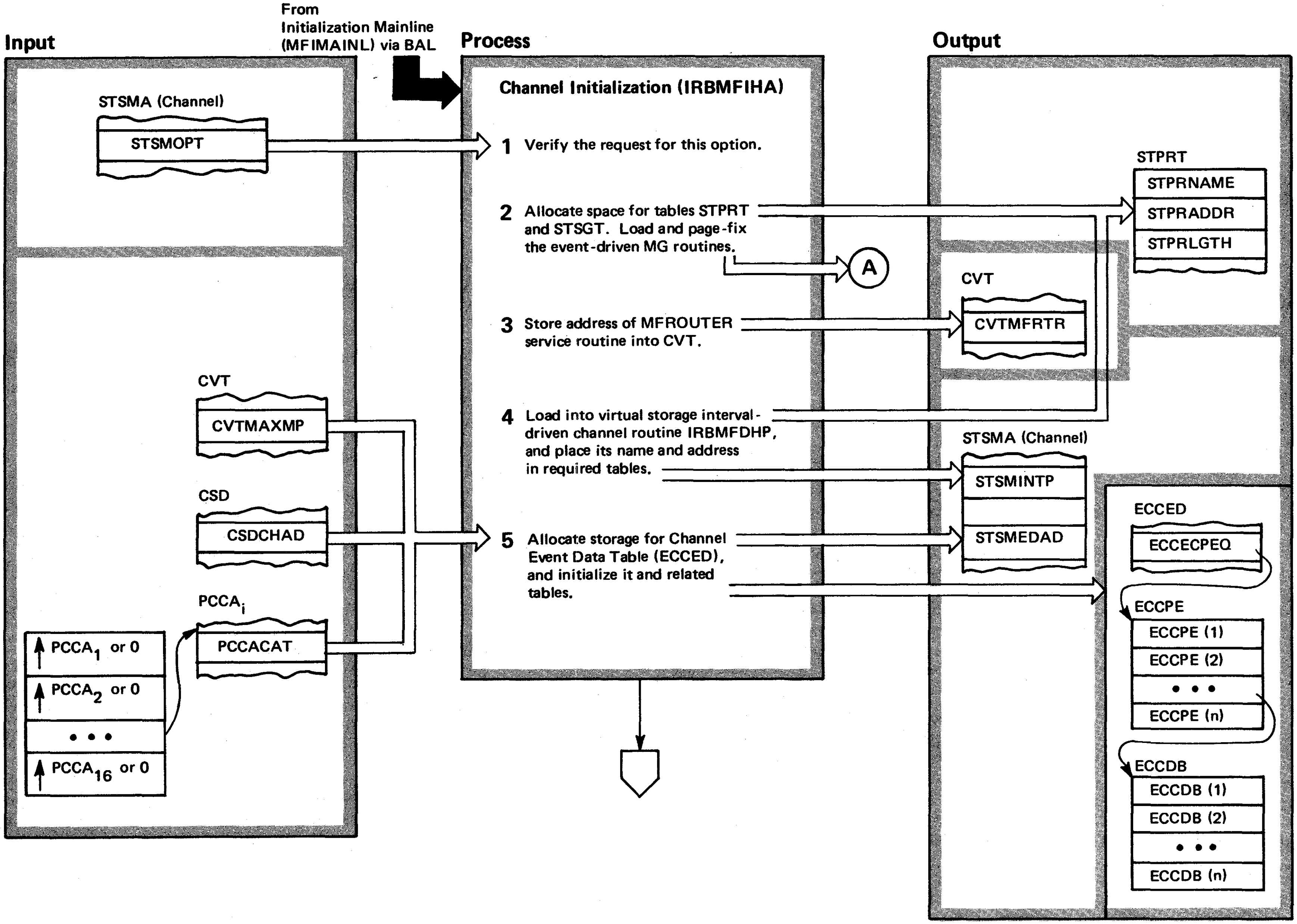
3 Load into virtual storage the interval-driven MG workload routine IRBMFDWP, and place its name and address in required tables.

4 Initiate workload activity data collection and obtain space for interval data area.

Diagram 7-8. Workload Initialization (IRBMFIWK) (Part 2 of 2)

Extended Description	Module	Label
The Workload Initialization (IRBMFIWK) routine allocates storage for control blocks, ensures that a copy of the Interval MG routine for Workload (IRBMFDWP) is in storage, and calculates the length of the data area.	IRBMFIWK	
1 IRBMFIWK ensures that the workload option has been selected as an input option by checking the STSMSTA bit of the STSMOPT of the Supervisor Measurements Area (STSMA).	IRBMFIWK	
2 IRBMFIWK uses the GETMAIN macro instruction to obtain the required storage. IRBMFIWK also uses the PGFIX macro instruction to fix IRARMWAR. Then, IRBMFIWK issues a WAIT macro instruction for page fix completion. The name and address of IRARMWAR are placed in the Program Resource Table (PRT) and the Resource Vector Table (RVT) is marked to indicate the next entry in the PRT.	IRBMFIWK	
3 The name of the Interval MG Routine for Workload (IRBMFDWP) is placed into the STPRT, and its address into STSMINTP of the System Measurement Area (STSMA).	IRBMFIWK	
4 IRBMFIWK calls routine MFIIPSWA, which uses a GETMAIN macro instruction to obtain storage for the interval workload data area. The length of this area is: length (WAMT) + (highest performance group number times (length (WAMTNDX entry)) + (total number of performance group periods) times (length of WAMP). (A performance group is a term of the System Resources Manager (SRM).) The length and address of the area are inserted into Storage Resource Table (STSGT). The address of IRARMWAR is inserted into the gotten area, and IRBMFIWK issues a SYSEVENT WKLDINIT macro instruction to initiate SRM workload data collection. Return code 00 from the SYSEVENT is the good return. Return code 08 indicates that the installation performance specification (IPS) was changing when the SYSEVENT macro instruction was issued; another SYSEVENT is therefore issued. Return code 20 from the SYSEVENT indicates that MF/1 data collection is already active; therefore a bad return is made to IRBMFIWK.	IRBMFIWK	MFIIPSWA

Diagram 7-9. Channel Initialization (IRBMFIHA) (Part 1 of 4)



Channel Initialization (IRBMFIHA)

1 Verify the request for this option.

2 Allocate space for tables STPRT and STSGT. Load and page-fix the event-driven MG routines.

3 Store address of MFROUTER service routine into CVT.

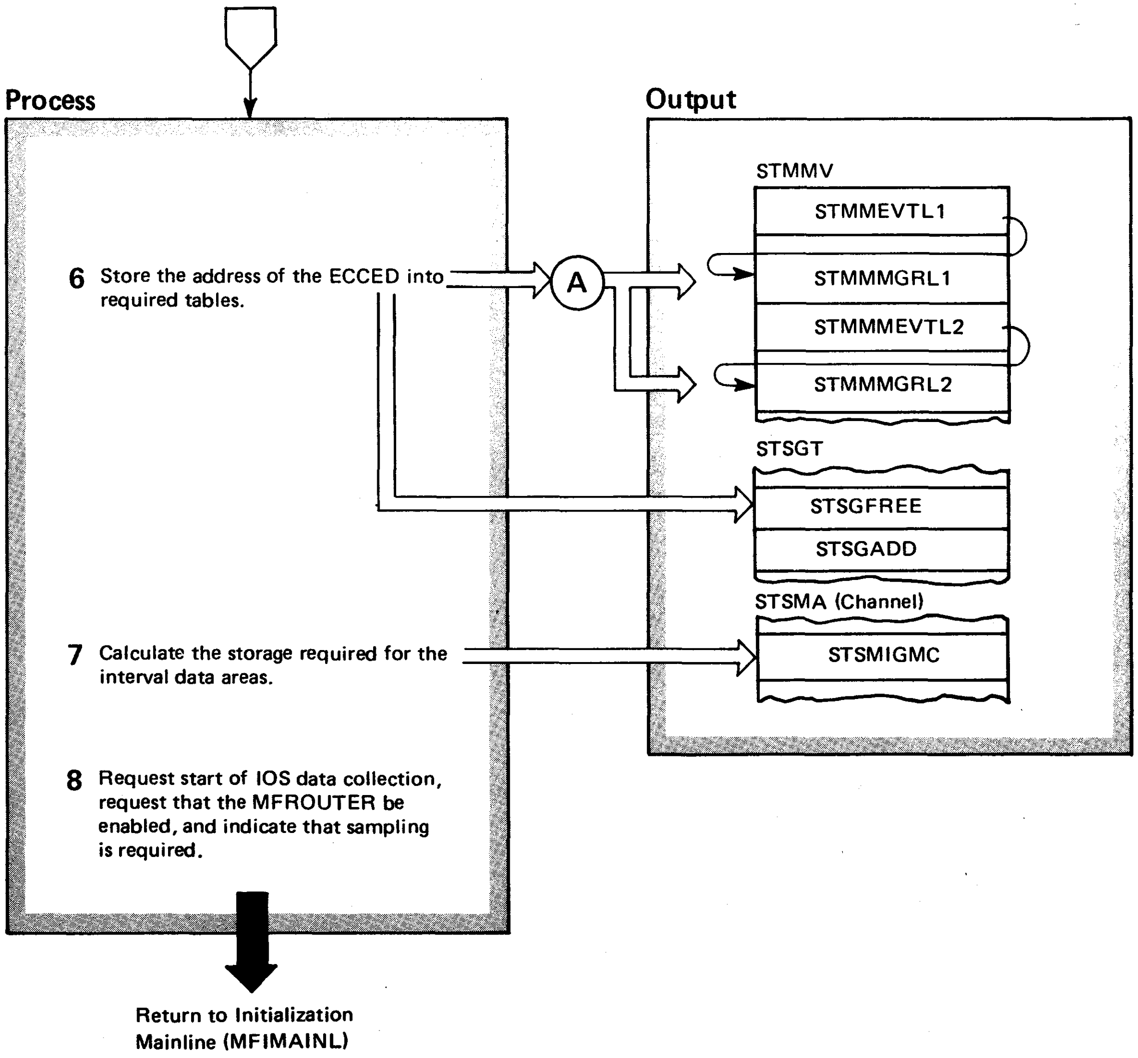
4 Load into virtual storage interval-driven channel routine IRBMFDHP, and place its name and address in required tables.

5 Allocate storage for Channel Event Data Table (ECCED), and initialize it and related tables.

Diagram 7-9. Channel Initialization (IRBMFIHA) (Part 2 of 4)

Extended Description	Module	Label
The Channel Initialization (IRBMFIHA) performs the initialization functions required to cause MF/1 to begin collecting channel data. These functions include initializing both event-driven and interval-driven MG routines.	IRBMFIHA	
1 IRBMFIHA checks bit STSMSTA of SYSMOPT in the System Measurement Area (STSMA) to ensure that channel data has been specified as an input option.	IRBMFIHA	
2 IRBMFIHA activates modules IRBMFEVT (to respond to MFROUTER requests), IRBMFECH (to collect event-driven sample data on the channels of the CPU that executes the instructions when IRBMFEVT receives control), and IRBMFTCH (to collect event-driven sampled data on the channels of any CPU not executing the instructions when IRBMFEVT assumes control). The activation consists of adding the modules to the Program Resource Table (STPRT) and adding IRBMFECH and IRBMFTCH to IRBMFEVT routing table entries, STMMMGRL1 and STMMMGRL2.	IRBMFIHA	IHLOADM1 IHPAGFX1 IHLOADM2 IHPAGFX2 IHLOADM3 IHPAGFX3
3 Set the MF/1 MFROUTER pointer (CVTMFRTR) in the Communication Vector Table (CVT) to point to IRBMFEVT.	IRBMFIHA	
4 The name IRBMFDHP is placed into the STPRT and the STRVNPRT is updated to show the addition of IRBMFDHP. The address of IRBMFDHP is placed into STSMINTP of the STSMA for use by IRBMFEVT.	IRBMFIHA	IHLOADM4
5 A CPU element (ECCPE) is allocated and initialized for each possible CPU (MAXMP + 1), and then for each ECCPE, channel Data Block (ECCDB) entries are formed for each possible channel (CSDCHAD + 1). These CDBs are used to store data collected at each sampling event.	IRBMFIHA	IHGETMN3

Diagram 7-9. Channel Initialization (IRBMFIHA) (Part 3 of 4)



6 Store the address of the ECCED into required tables.

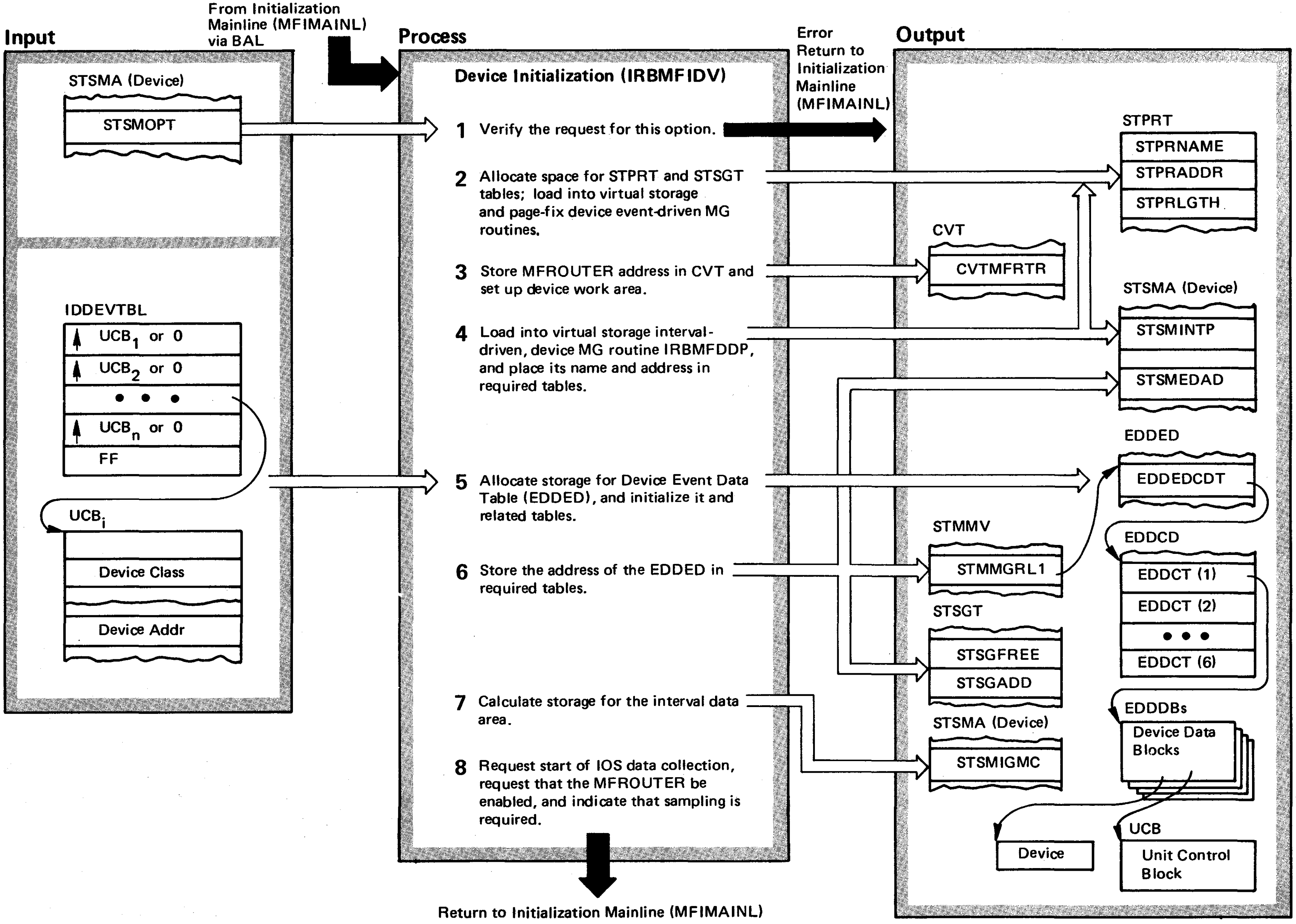
7 Calculate the storage required for the interval data areas.

8 Request start of IOS data collection, request that the MFROUTER be enabled, and indicate that sampling is required.

Diagram 7-9. Channel Initialization (IRBMFIHA). (Part 4 of 4)

Extended Description	Module	Label
6 The address of the Channel Event Data Table (ECCED) is stored in STMMMGRL1 and STMMMGRL2 of the MFROUTER Measurement Vector Table (STMMV) for use by the MFROUTER Processor (IRBMFEVT). The ECCED address is also stored into the Storage Resource Table (STSGT) and the System Measurement Area (STSMA).	IRBMFIHA	
7 The storage length for interval data is: 4 + length of (SMFRCD73) + length of (SMF73A) + (CVTMAXMP + 1) times (CSDCHAD + 1) times length of (SMF73B).		
8 The return code from IRBMFIHA is set to indicate that IOS data collection should be requested, that the MFROUTER should be enabled, and that sampling of channel data is required.	IRBMFIHA	

Diagram 7-10. Device Initialization (IRBMFIDV) (Part 1 of 2)



Device Initialization (IRBMFIDV)

1 Verify the request for this option.

2 Allocate space for STPRT and STSGT tables; load into virtual storage and page-fix device event-driven MG routines.

3 Store MFROUTER address in CVT and set up device work area.

4 Load into virtual storage interval-driven, device MG routine IRBMFDDP, and place its name and address in required tables.

5 Allocate storage for Device Event Data Table (EDDED), and initialize it and related tables.

6 Store the address of the EDDED in required tables.

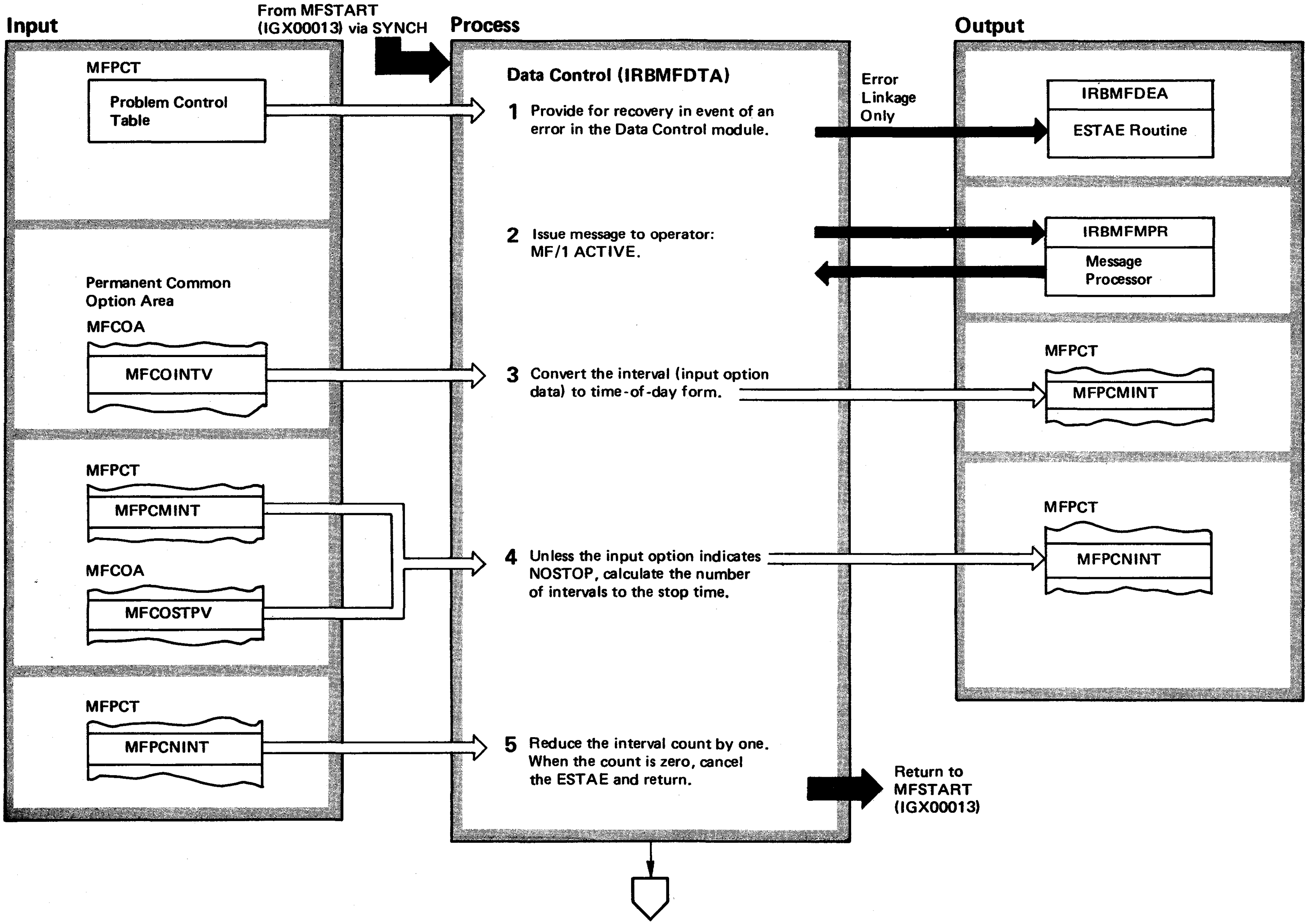
7 Calculate storage for the interval data area.

8 Request start of IOS data collection, request that the MFROUTER be enabled, and indicate that sampling is required.

Diagram 7-10. Device Initialization (IRBMFIDV) (Part 2 of 2)

Extended Description	Module	Label
The Device Initialization (IRBMFIDV) routine activates the MFROUTER Processor (IRBMFEVT) to respond to calls for event-driven sampling of device data. In addition IRBMFIDV initializes the interval-driven device data MG routine IRBMFDDP. Required storage and table initialization are also performed so that device data can be collected and stored.	IRBMFIDV	
1 IRBMFIDV checks that STSMOPT in the System Measurement Area (STSMA) is on. If not, IRBMFIDV returns immediately.	IRBMFIDV	
2 IRBMFIDV adds the MFROUTER Processor (IRBMFEVT) and the event-driven device MG module (IRBMFEDV) to the Program Resource Table (STPRT). The Resource Vector Table (STRVT) is also changed to indicate the next STPRT entry.	IRBMFIDV	
3 Set CVTMFRTR in the Communication Vector Table (CVT) to point to IRBMFEVT.	IRBMFIDV	
4 The module name, IRBMFDDP, is placed into the STPRT, and as in step 2, STRVNPRT is changed. The address of IRBMFDDP is placed in the SMA (specifically, STSMINTP) for use by the MFDATA SVC Processor (IGX00014).	IRBMFIDV	
5 To allocate and initialize the Device Event Data Table (EDDEDT) the following phases are necessary: a) The number of nonzero entries in the IOS UCB lookup table is determined. The result is the maximum number of devices possible. A work table is allocated on the basis of this count. b) For each class of devices to be monitored, the IOS UCB lookup table is used to search for existing devices in the class. As a device is found, its UCB address is put into the work area and a class count is increased by one if that UCB address had not already been processed. c) The preceding phase is repeated without modifying the work area as a check that the lookup table is not in the process of being changed. If an error is found, both the preceding phase (b) and this one (c) is repeated. d) Finally the EDDED, the Device Class Data Table (EDDCD), and the Device Data Blocks (EDDDBS) are allocated and initialized, based on the work area information. The EDDCD entries consist of one entry for each of the following device classes in the order listed: • tape • communication equipment • direct access • graphics • unit record • character reader Each entry is zero if no device exists for that class; otherwise it contains the address of the EDDDB table and number of DDBS for the devices that do exist.	IRBMFIDV	
6 The address of the EDDED is stored into STMMMGRL of the MFROUTER Measurement Vector Table (STMMV), into STSGT, and into STMEDAD of the SMA.	IRBMFIDV	
7 The storage length is; $4 + \text{length (DDDVT)} + \sum_{k=1}^{n} b_k$ times [length of (SMFRCD74) + length of (SMF74A) + Ck times (length of (SMF74B))] where, DDDVT is a table of entries for each device class n = number of device classes Ck = number of devices in class K b_k = 1 if Ck ≠ 0 and b_k = 0 if Ck = 0		
8 The return code for IRBMFIDV is set to indicate that IOS data collection should be started, that MFROUTER should be enabled, and that sampling of device data is required.	IRBMFIDV	

Diagram 7-11. Data Control (IRBMFDTA) (Part 1 of 4)



Data Control (IRBMFDTA)

1 Provide for recovery in event of an error in the Data Control module.

2 Issue message to operator: MF/1 ACTIVE.

3 Convert the interval (input option data) to time-of-day form.

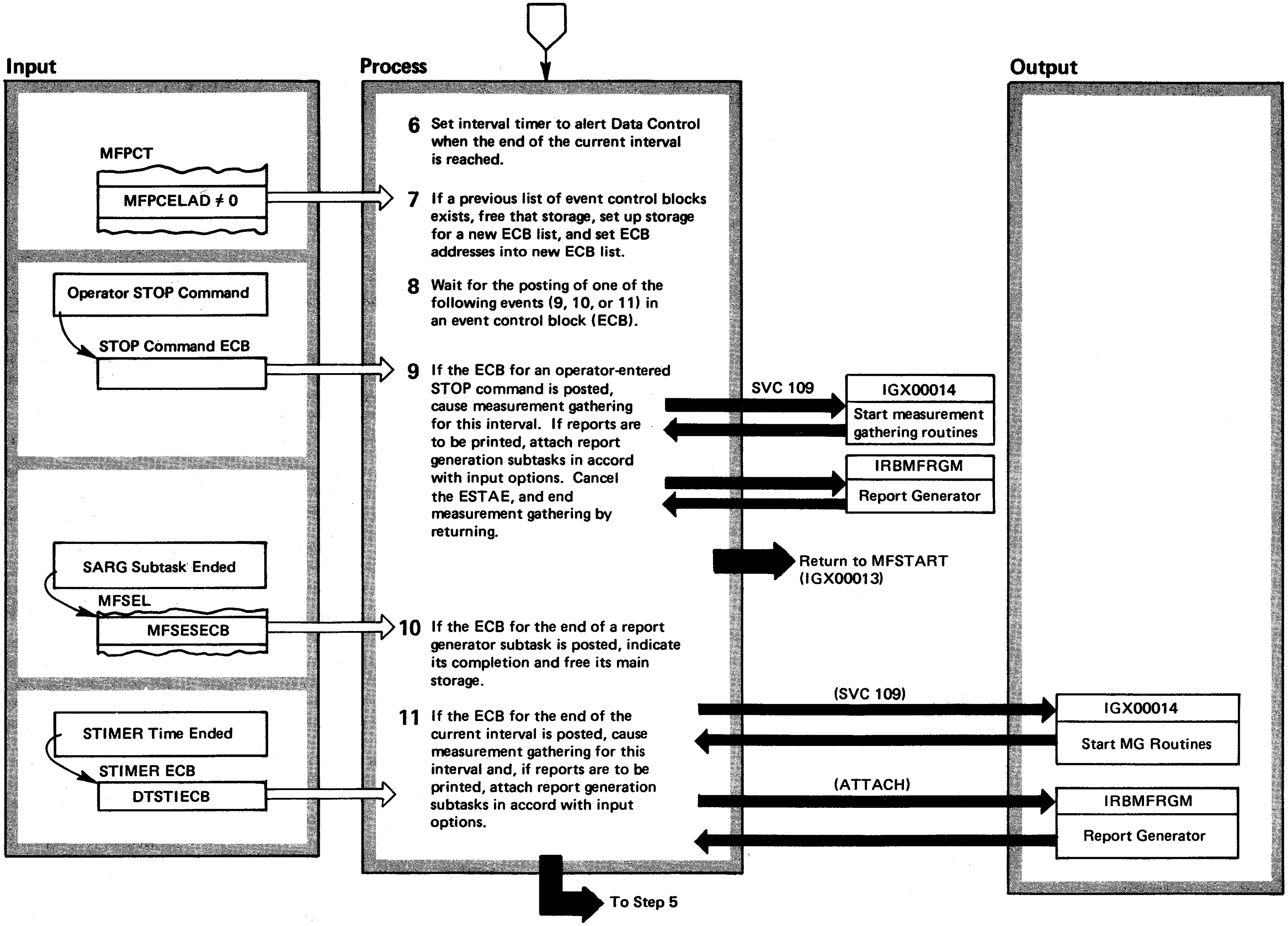
4 Unless the input option indicates NOSTOP, calculate the number of intervals to the stop time.

5 Reduce the interval count by one. When the count is zero, cancel the ESTAE and return.

Diagram 7-11. Data Control (IRBMFDTA) (Part 2 of 4)

Extended Description	Module	Label
Data Control (IRBMFDTA) is executed in problem state in response to a SYNCH macro instruction issued by the MFSTART module. This change from supervisor state in MFSTART represents the entry into the main measurement gathering operations, which are controlled from the Data Control Module. Control includes establishing the interval of measurement gathering, as specified by an input option, and the queueing of report generation subtasks if real time reporting was requested. In addition, Data Control performs a number of event control block and storage control functions.	IRBMFDTA	
1 Establish ESTAE routines.	IRBMFDEA	
2 This message is the first normal operation message to the operator. It is issued after he indicates GO.	IRBMFMPR	
3 Interval time is entered in minutes. This time is converted to microseconds and placed in a doubleword such that a one in bit 51 equals one microsecond.	IRBMFDTA	
4 A stop time (input option) is specified or NOSTOP is specified. If NOSTOP is specified, the stop command is used to stop MF/1 operation. If a stop value is given, the amount of time from the current time until the stop time is divided by the interval length to obtain the number of intervals.	IRBMFDTA	
5 Data Control reduces the number of such intervals each time through this code. When this interval count is zero, MF/1 measurements are ended.	IRBMFDTA	

Diagram 7-11. Data Control (IRBMFDTA) (Part 3 of 4)



6 Set interval timer to alert Data Control when the end of the current interval is reached.

7 If a previous list of event control blocks exists, free that storage, set up storage for a new ECB list, and set ECB addresses into new ECB list.

8 Wait for the posting of one of the following events (9, 10, or 11) in an event control block (ECB).

9 If the ECB for an operator-entered STOP command is posted, cause measurement gathering for this interval. If reports are to be printed, attach report generation subtasks in accord with input options. Cancel the ESTAE, and end measurement gathering by returning.

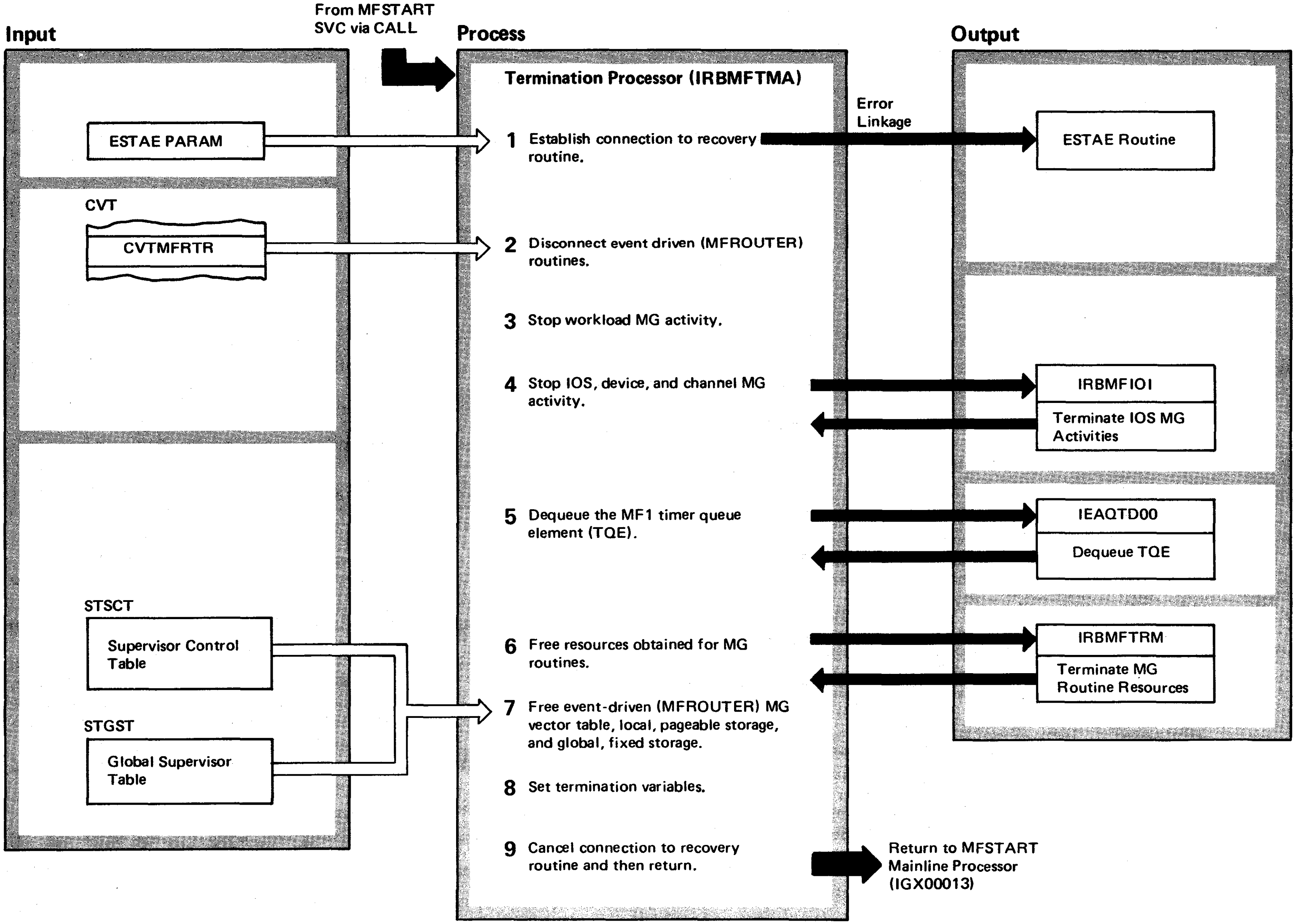
10 If the ECB for the end of a report generator subtask is posted, indicate its completion and free its main storage.

11 If the ECB for the end of the current interval is posted, cause measurement gathering for this interval and, if reports are to be printed, attach report generation subtasks in accord with input options.

Diagram 7-11. Data Control (IRBMFDTA) (Part 4 of 4)

Extended Description	Module	Label
6 The routine sets the STIMER macro instruction for the length of the current interval and compensates for any stop during the interval.	IRBMFDTA	
7 It uses one FREEMAIN macro instruction to free storage of any existing event control blocks (ECBs). Then the routine uses GETMAIN to obtain storage for pointers to ECBs: one ECB for the STOP command, one for the STIMER alert, and one for each report generation (SARG) subtask.	IRBMFDTA	
8 One of three conditions has occurred when an ECB is posted: a) The operator has issued a stop command. If so, create short interval data, and end measurements. Return to caller of Data Control. b) A report generator subtask has ended. If so, detach the subtask, and dequeue its subtask element (SEL) from the subtask queue (SQU). c) The STIMER interval has been reached (the current interval has ended). If so, issue an MFDATA SVC to cause measurement gathering for this interval and attach a report generation subtask unless no report of these measurements was requested. Build a (SARG) subtask queue element (MFSQU) for the subtask.	IRBMFDTA	
9 An EXTRACT macro instruction is used to obtain the command input buffer (CIB) address of the STOP. A short interval results when the STOP command is issued. The MFDATA SVC controls the collection of requested measurement data. Report generation subtasks are called by attaching the Report Generator control (IRBMFRGM).	IRBMFDTA IGX00014 IRBMFRGM	
10 Data Control issues a DETACH macro instruction to remove a completed subtask and then shortens the subtask queue. The subtask's main storage (its element subpool space) is freed by means of a FREEMAIN macro instruction.	IRBMFDTA	
11 The MFDATA SVC controls the collection of requested measurement data. Report generation subtasks are called by attaching the Report Generator control (IRBMFRGM).	IGX00014 IRBMFRGM	

Diagram 7-12. Termination Processor (IRBMFTMA) (Part 1 of 2)



1 Establish connection to recovery routine.

2 Disconnect event driven (MFROUTER) routines.

3 Stop workload MG activity.

4 Stop IOS, device, and channel MG activity.

5 Dequeue the MF1 timer queue element (TQE).

6 Free resources obtained for MG routines.

7 Free event-driven (MFROUTER) MG vector table, local, pageable storage, and global, fixed storage.

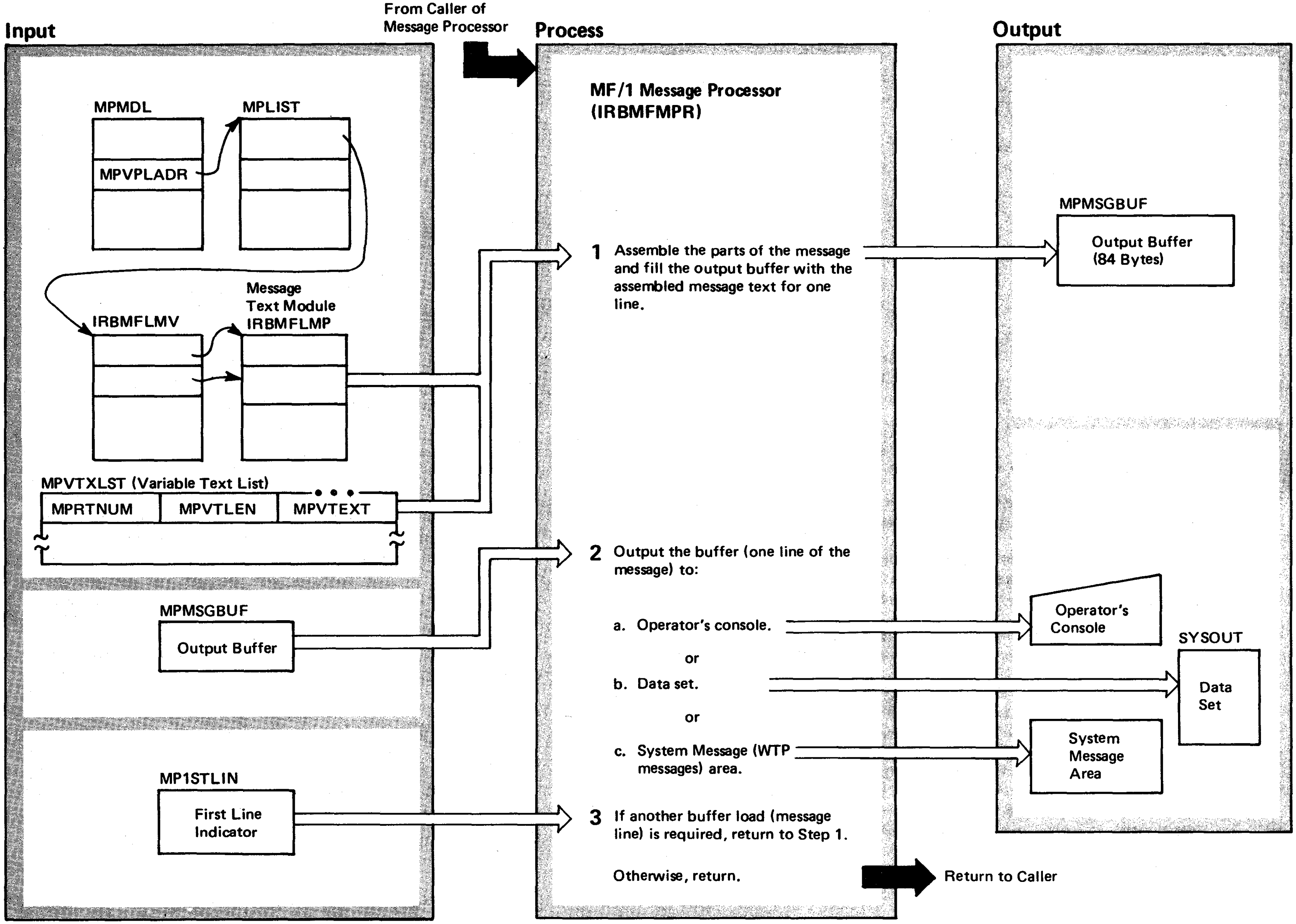
8 Set termination variables.

9 Cancel connection to recovery routine and then return.

Diagram 7-12. Termination Processor (IRBMFTMA) (Part 2 of 2)

Extended Description	Module	Label
The Termination Processor (IRBMFTMA) disconnects MF/1 from the resident nucleus. The Termination Processor dequeues the Timer Queue Element (TQE), disconnects the event driven (cycle) MG routines, disables workload activity data collection, releases global storage, and restores the changes made in the system I/O processor (IOS) to enable channel and device data collection.	IRBMFTMA	
1 The Termination Processor provides ESTAE parameters to provide for retrying while releasing resources.	IRBMFTMA	
2 The linkage to the MFROUTER service routine (IRBMFEVT) is changed so that if an attempt is made to transfer control to IRBMFEVT, immediate return will be made by a BR 14. The Termination Processor also ensures that no CPU is currently executing event-driven MG code when this code is disconnected.	IRBMFTMA	
3 The Termination Processor causes the workload manager to stop workload activity data collection.	IRBMFTMA IRARMWLM	
4 The Termination Processor calls the IOS Initiation/Termination Module (IRBMFIOI) to restore the changes it made to IOS.	IRBMFIOI	
5 The Termination Processor dequeues the MF/1 timer queue element (TQE) by disabling (using the SETLOCK macro instruction); providing a functional recovery routine (FRR) link (because of having disabled); and using the TQE Dequeue routine (IEAQTD00) to dequeue the TQE. The Termination Processor then cancels the FRR link, and enables by means of the SETLOCK macro instruction.	IRBMFTMA IEAQTD00	
6 The Termination Processor calls routine IBBMFTRM to release the resources of each MG routine.	IRBMFTRM	
7 The Termination Processor uses the FREEMAIN macro instruction to release the measurement Vector Table (STMMV), the MF/1 local storage, and MF/1 global storage.	IRBMFTMA	
8 The Termination Processor dequeues the MF/1 enqueue resource by use of the DEQ macro instruction.	IRBMFTMA	
9 The ESTAE connection is canceled by use of the ESTAE macro instruction.	IRBMFTMA	

Diagram 7-13. MF/1 Message Processor (IRBMFMPR) (Part 1 of 2)



1 Assemble the parts of the message and fill the output buffer with the assembled message text for one line.

2 Output the buffer (one line of the message) to:

a. Operator's console.

or

b. Data set.

or

c. System Message (WTP messages) area.

3 If another buffer load (message line) is required, return to Step 1.

Otherwise, return.

Diagram 7-13. MF/1 Message Processor (IRBMFMPR) (Part 2 of 2)

Extended Description	Module	Label
The Message Processor (IRBMFMPR) is called from several places in the MF/1 program to print output messages. (These are: IRBMFDTA, IRBMFINP, IRBMFRGM, IRBMFMFC, and IRBMFMLN.) The Message Processor assembles the required message from parts in the Message Text module (IRBMFLMP), moves the parts into an output buffer, one message line at a time, and writes the message lines to the required output device or data set.	IRBMFMPR	
1 Input parameters define the message in terms of fixed and/or variable text portions. Fixed text portions are obtained from IRBMFLMP through an index in table MPLIST. When an MPLIST entry contains a zero, a variable text entry is obtained from the variable text list (MPVTXLST). If the variable text length (MPVTLEN) is non-zero, the variable text is moved into the buffer. If the variable text length is zero and the MPRTNUM field is non-zero, the MPRTNUM value is used to index into IRBMFLMV, to obtain fixed text from IRBMFLMP. Up to 80 bytes of message text and message identifier are assembled in the buffer.	IRBMFMPR	MFBLDMSG
2 The message Processor calls routine MFOUTMSG to write the buffer to the operator's console or required data set and then returns to the Message Processor as soon as the message is sent.	IRBMFMPR	MFOUTMSG
3 The message Processor controls the assembling of message lines and writing them until the entire message is sent.	IRBMFMPR	

Diagram 7-14. MFDATA SVC Mainline Processor (IGX00014) (Part 1 of 4)

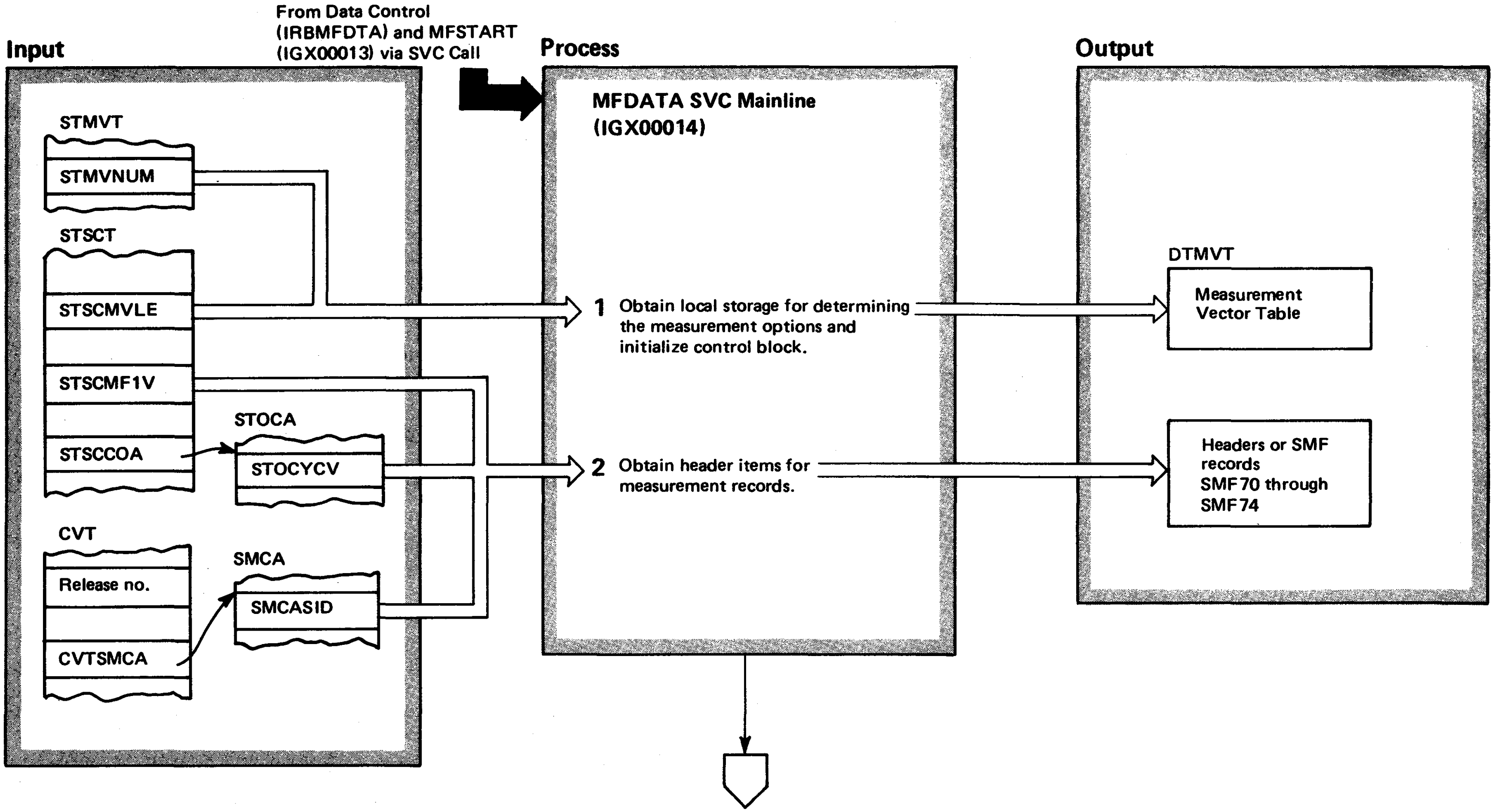
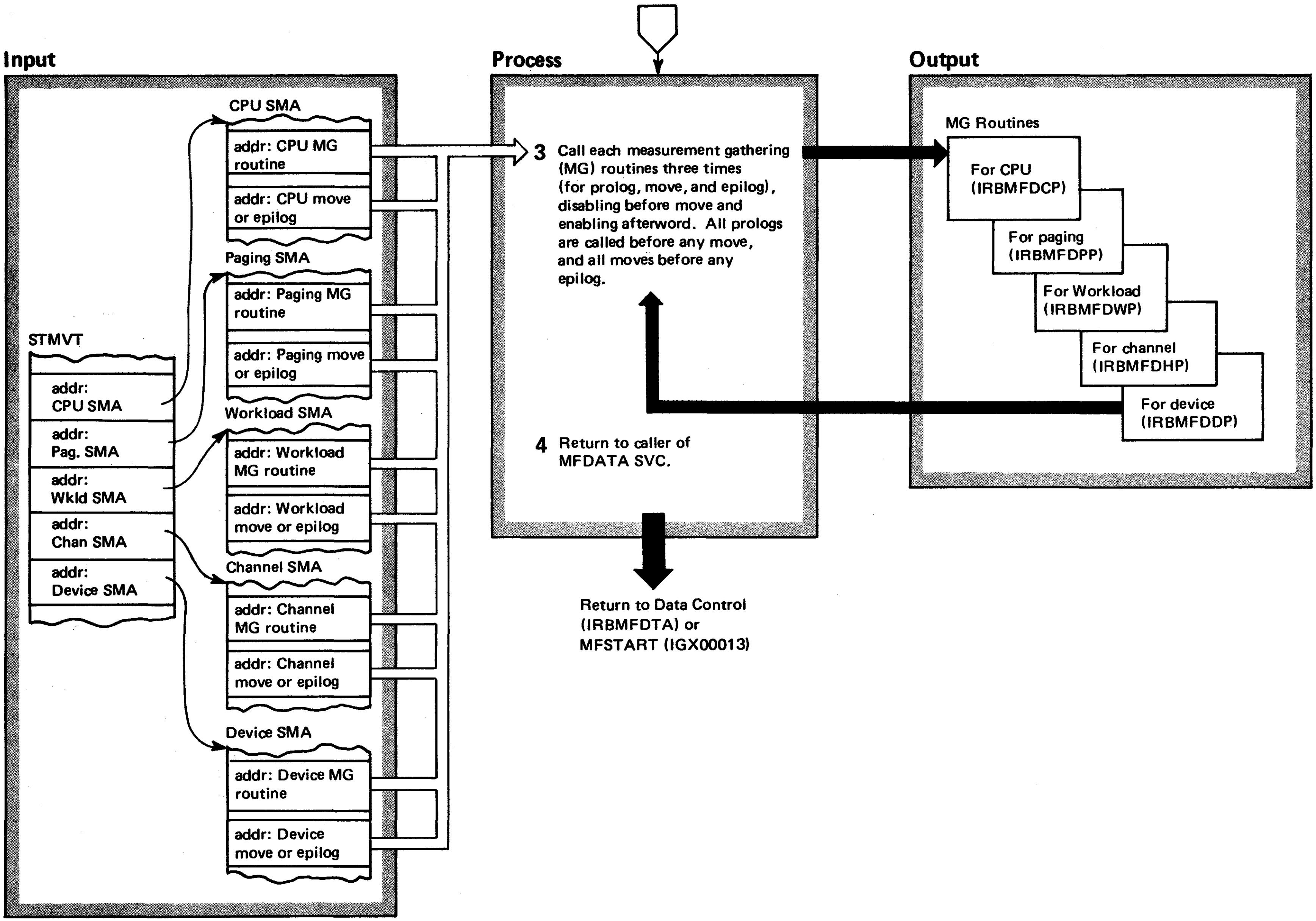


Diagram 7-14. MFDATA SVC Mainline Processor (IGX00014) (Part 2 of 4)

Extended Description	**Module**	**Label**
The MFDATA SVC Mainline (IGX00014) processor executes in response to an MFDATA SVC issued by the Data Control module (IRBMFDTA), once each interval, and by MFSTART (IGX00013) during initialization. When called, IGX00014 controls the operation of measurement gathering routines. Each MG routine collects measurements of one of the following kinds if called for by input option: • CPU wait time • Paging activity • Workload • Channel activity • Device activity The measurements for the interval are placed in records that have the format of System Management Facilities (SMF-70-74). Internal Copies of these records are used by report generation routines (SARG) to provide printed reports specified by input options.	IGX00014	
1 Issue the GETMAIN macro instruction to obtain storage for the Measurement Vector Table (DTMVT) and initialize the table area by setting all option pointers to zero.	IGX00014	
2 Obtain SMF record header items for: a) Identifying the record as an OS/VS2 record. b) System identification. c) MF/1 version number. d) Operating system release and level. e) Cycle length (from input option).		

Diagram 7-14. MFDATA SVC Mainline Processor (IGX00014) (Part 3 of 4)



3 Call each measurement gathering (MG) routines three times (for prolog, move, and epilog), disabling before move and enabling afterword. All prologs are called before any move, and all moves before any epilog.

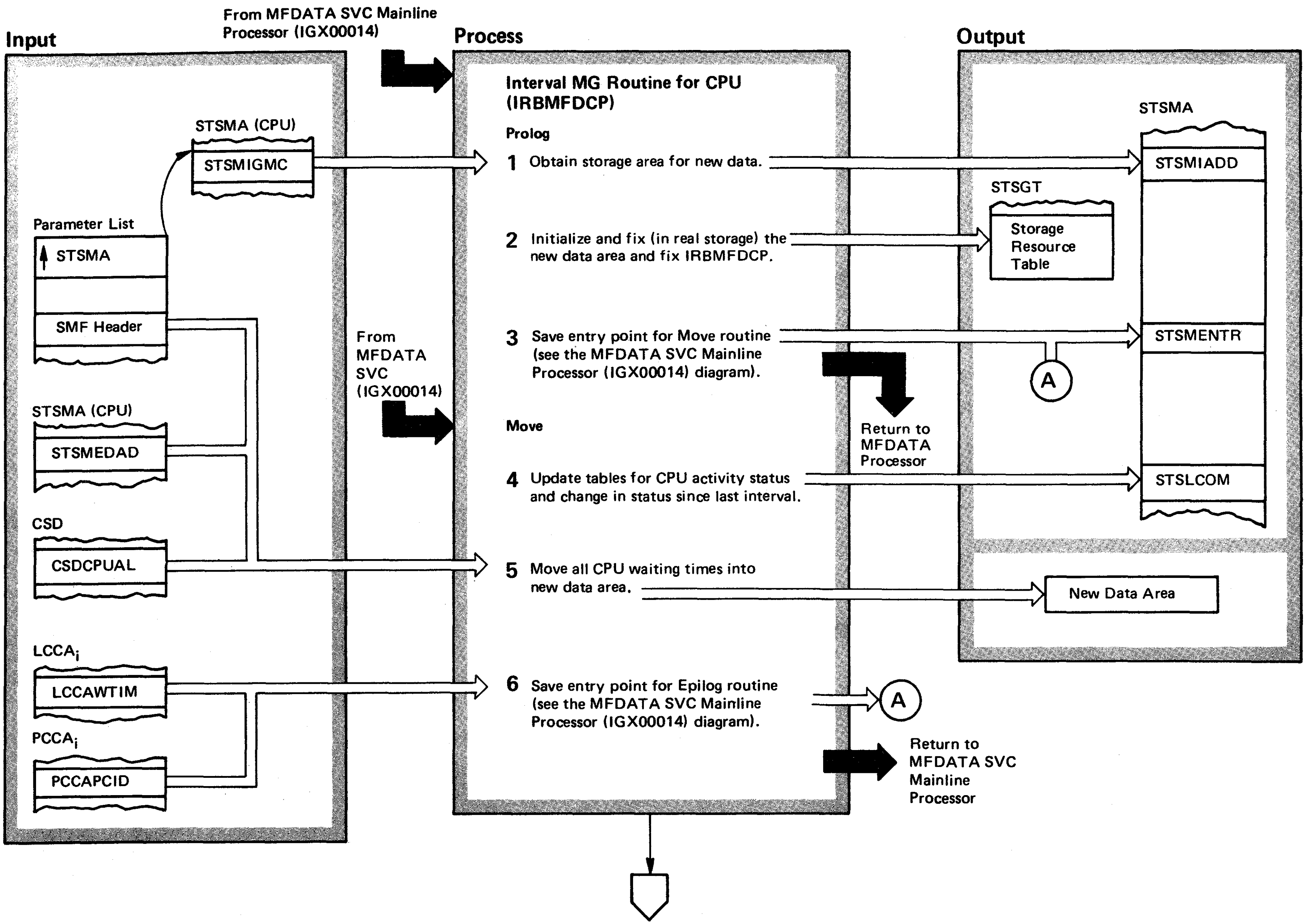
4 Return to caller of MFDATA SVC.

Return to Data Control (IRBMFDTA) or MFSTART (IGX00013)

Diagram 7-14. MFDATA SVC Mainline Processor (IGX00014) (Part 4 of 4)

Extended Description	Module	Label
3 Each MG routine has a prolog, a move part, and an epilog. The prologs for all the required (by input option) MG routines are called first in the order listed in the first paragraph of this explanation. When the prologs have been called, the required move parts are called, and then the epilogs are called. The effect on each MG routine, however, is as though it executed from start to end without interruption. This arrangement is used to allow the move parts of these routines and IGX00014 to execute disabled. Before the move parts of the MG routines, which contain the code to move measurement data into record formats, are executed, interruptions are disabled by obtaining and releasing the dispatcher lock. When the SETLOCK is released, it is released disabled. The reverse technique is used to enable, after all the move parts of the MG routines have been executed.	IRBMFDCP IRBMFDPP IRBMFDWP IRBMFDHP IRBMFDDP	
4 Upon return to the caller, IGX00014 save the Measurement Vector Table (DTMVT) address in register 1.	IGX00014	

Diagram 7-15. Interval MG Routine for CPU (IRBMFDCP) (Part 1 of 4)



Interval MG Routine for CPU (IRBMFDCP)

Prolog

1 Obtain storage area for new data.

2 Initialize and fix (in real storage) the new data area and fix IRBMFDCP.

3 Save entry point for Move routine (see the MFDATA SVC Mainline Processor (IGX00014) diagram).

Move

4 Update tables for CPU activity status and change in status since last interval.

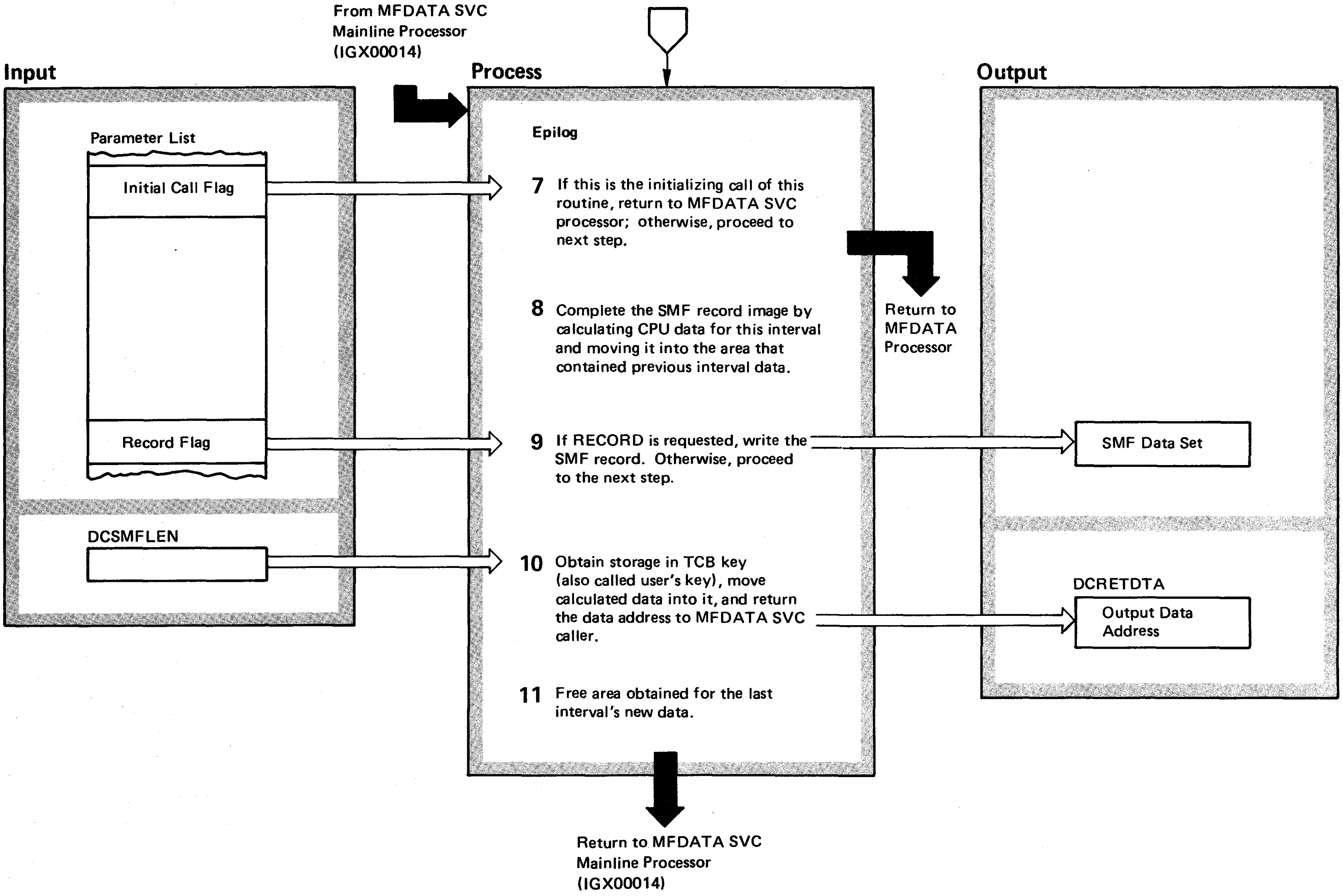
5 Move all CPU waiting times into new data area.

6 Save entry point for Epilog routine (see the MFDATA SVC Mainline Processor (IGX00014) diagram).

Diagram 7-15. Interval MG Routine for CPU (IRBMFDCP) (Part 2 of 4)

Extended Description	**Module**	**Label**
The Interval MG Routine for CPU (IRBMFDCP) receives control from the MFDATA SVC Processor at the end of each interval if CPU activity reports are required. IRBMFDCP copies CPU wait times for all CPUs into a contiguous storage area and builds an internal image of the MF/1 CPU activity record (SMFRCD70) for the SMF data set. IRBMFDCP calculates wait time for each CPU by subtracting the wait time read at the end of the current interval from that read at the end of the previous interval, after adjusting for the possibility of wrap-around readings.	IRBMFDCP	
Prolog		
1 Use the GETMAIN macro instruction to obtain the required storage in key zero.	IRBMFDCP	DCGETMN1
2 Store the subpool and length of the storage obtained into the first word of the area. Use the PGFIX macro instruction to fix the data and IRBMFDCP.		
3 Save the entry point, as described in the M.O. diagram MFDATA SVC Mainline Processor (IGX00014), for use in returning to the Move part of IRBMFDCP.	IRBMFDCP	
Move		
4 If a CPU is now online whose flag is not set in STSMEDAD of the Supervisor Measurement Area (STSMA), set its flag to indicate that it has been online.	IRBMFDCP	DCMOVE
5 Partially initialize the SMF record image, set online status flags for all valid CPUs, and move in wrap-around wait time measurement counters for those CPUs.	IRBMFDCP	
6 See Step 3.	IRBMFDCP	DCEPILOG

Diagram 7-15. Interval MG Routine for CPU (IRBMFDCP) (Part 3 of 4)



7 If this is the initializing call of this routine, return to MFDATA SVC processor; otherwise, proceed to next step.

8 Complete the SMF record image by calculating CPU data for this interval and moving it into the area that contained previous interval data.

9 If RECORD is requested, write the SMF record. Otherwise, proceed to the next step.

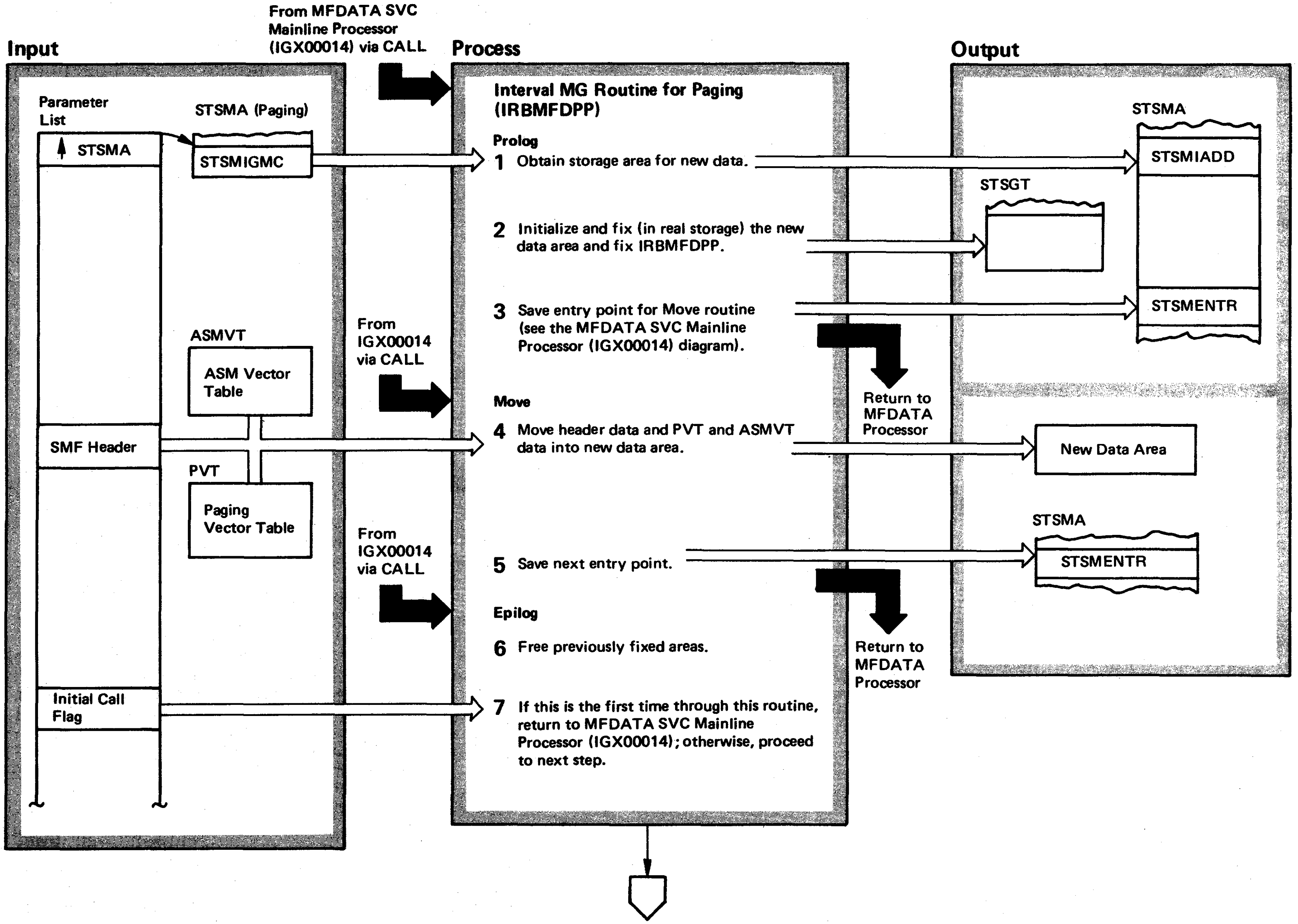
10 Obtain storage in TCB key (also called user's key), move calculated data into it, and return the data address to MFDATA SVC caller.

11 Free area obtained for the last interval's new data.

Diagram 7-15. Interval MG Routine for CPU (IRBMFDCP) (Part 4 of 4)

Extended Description	Module	Label
Epilog		
7 On the first call to the MFDATA SVC, the MFDATA SVC Processor calls the interval MG routines to obtain a first set of wrap-around measurements for later calculations (subtraction).	IRBMFDCP	
8 Move through all possible CPU entries in old and new data areas, and calculate CPU wait times for CPUs active throughout the interval. Allow for wrap-around values when subtracting current from previous values.	IRBMFDCP	
9 Use the SMFWTM macro instruction to write the image of the SMFRCD70 record to the SMF data set.	IRBMFDCP	
10 Use the GETMAIN macro instruction to obtain the required storage in user key; use the MODESET macro instruction to change to the TCB key.	IRBMFDCP	
11 Release the storage of the internal SMF image using a FREEMAIN macro instruction.	IRBMFDCP	

Diagram 7-16. Interval MG Routine for Paging (IRBMFDPP) (Part 1 of 4)



Interval MG Routine for Paging (IRBMFDPP)

Prolog

1 Obtain storage area for new data.

2 Initialize and fix (in real storage) the new data area and fix IRBMFDPP.

3 Save entry point for Move routine (see the MFDATA SVC Mainline Processor (IGX00014) diagram).

Move

4 Move header data and PVT and ASMVT data into new data area.

5 Save next entry point.

Epilog

6 Free previously fixed areas.

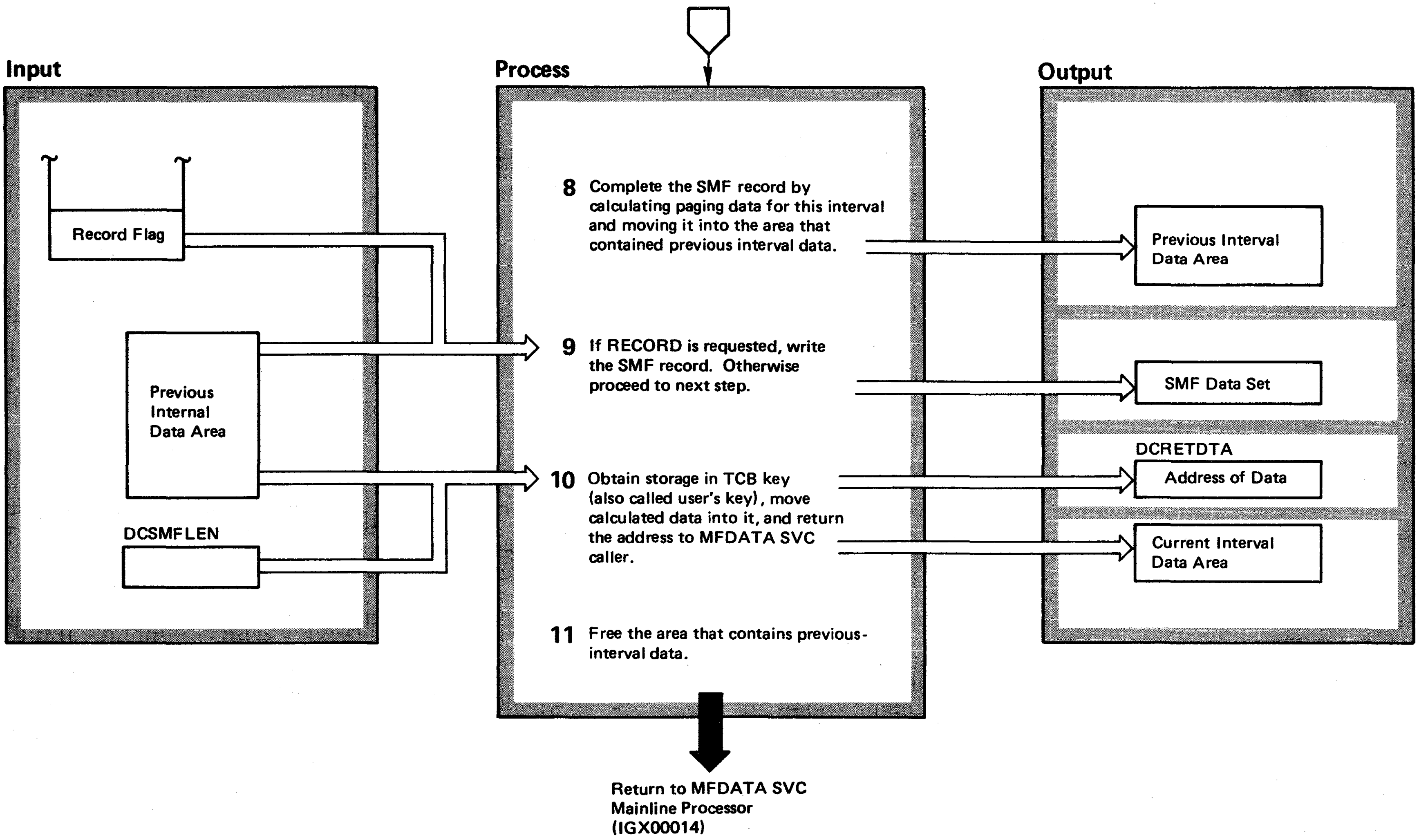
7 If this is the first time through this routine, return to MFDATA SVC Mainline Processor (IGX00014); otherwise, proceed to next step.

Diagram 7-16. Interval MG Routine for Paging (IRBMFDPP) (Part 2 of 4)

Extended Description	Module	Label
The Interval MG Routine for Paging (IRBMFDPP) builds an internal image of an SMF-71 paging record and, optionally, copies this image to the SMF data set. IRBMFDPP uses, for the internal image, data collected by the paging supervisor and the auxiliary storage manager. As described in the M.O. for the MFDATA SVC Processor, IRBMFDPP executes in three parts, PROLOG, MOVE, and EPILOG, but no break in execution is apparent except for the need to save entry points for the MOVE and EPILOG parts.	IRBMFDPP	
1 The GETMAIN macro instruction is used to obtain storage in key zero. The data for this interval is to be moved into this storage.	IRBMFDPP	
2 Use macro instruction PGFIX to inhibit paging of both the data area and routine IRBMFDPP.	IRBMFDPP	
3 The entry point is to be used to enter the Move part of IRBMFDPP. Between the PROLOG and Move a mechanism is used that avoids freeing data that would be freed in a normal return.	IRBMFDPP IRBMFDPP	 DPMOVE

Extended Description	Module	Label
4 IRBMFDPP moves a standard SMF record header and MF/1 control section and then fills in data fields in the internal record image (SMFRCD71).	IRBMFDPP	DPRT0017
5 IRBMFDPP provides entry to its EPILOG.	IRBMFDPP	DPEPILOG
6 IRBMFDPP uses the PGFREE macro instruction to allow paging in previously fixed area.	IRBMFDPP IRBMFDPP	DPRT00018 DPPAGFX4
7 On being called as part of initialization via the Initialization Mainline (MFMAINL) and MFDATA SVC Processor (IGX00014), IRBMFDPP returns to IGX00014, leaving initial-value data in an SMF record to be used at the end of the interval.	IRBMFDPP	

Diagram 7-16. Interval MG Routine for Paging (IRBMFDPP) (Part 3 of 4)



8 Complete the SMF record by calculating paging data for this interval and moving it into the area that contained previous interval data.

9 If RECORD is requested, write the SMF record. Otherwise proceed to next step.

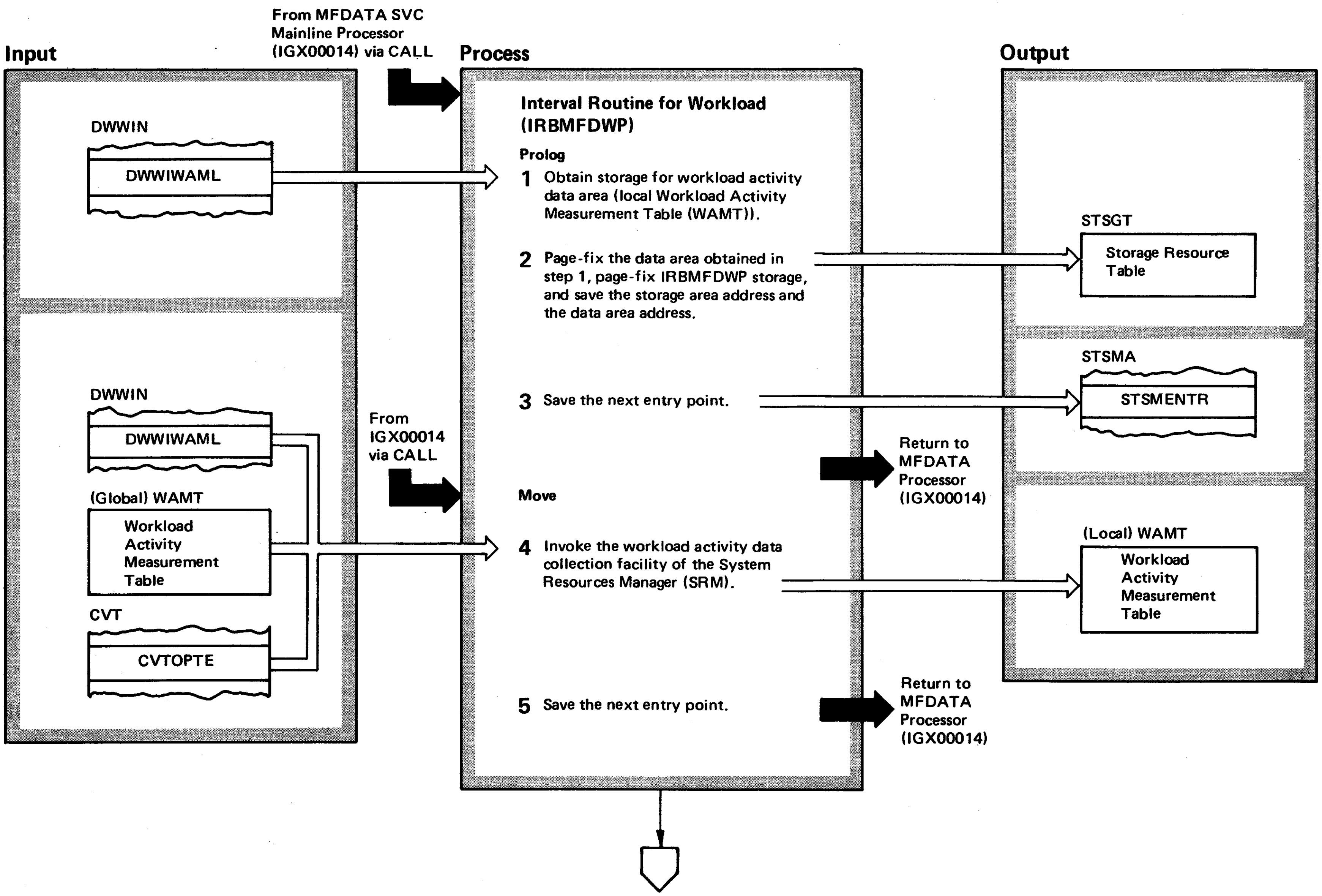
10 Obtain storage in TCB key (also called user's key), move calculated data into it, and return the address to MFDATA SVC caller.

11 Free the area that contains previous-interval data.

Diagram 7-16. Interval MG Routine for Paging (IRBMFDPP) (Part 4 of 4)

Extended Description	Module	Label
8 Values of paging data are calculated by comparing data at the start and end of the interval. Calculated values are placed in the old data area.	IRBMFDPP	
9 If the input option of recording data is requested, IRBMFDPP writes the SMFRCD71 internal image to the SMF data set using the SMFWTM macro instruction.	IRBMFDPP	
10 IRBMFDPP uses the GETMAIN macro instruction to obtain storage in user key. Change to user key by means of the MODESET macro instruction.	IRBMFDPP	
11 IRBMFDPP uses the FREEMAIN macro instruction to free storage.	IRBMFDPP	

Diagram 7-17. Interval MG Routine for Workload (IRBMFDWP) (Part 1 of 4)



Interval Routine for Workload (IRBMFDWP)

Prolog

1 Obtain storage for workload activity data area (local Workload Activity Measurement Table (WAMT)).

2 Page-fix the data area obtained in step 1, page-fix IRBMFDWP storage, and save the storage area address and the data area address.

3 Save the next entry point.

Move

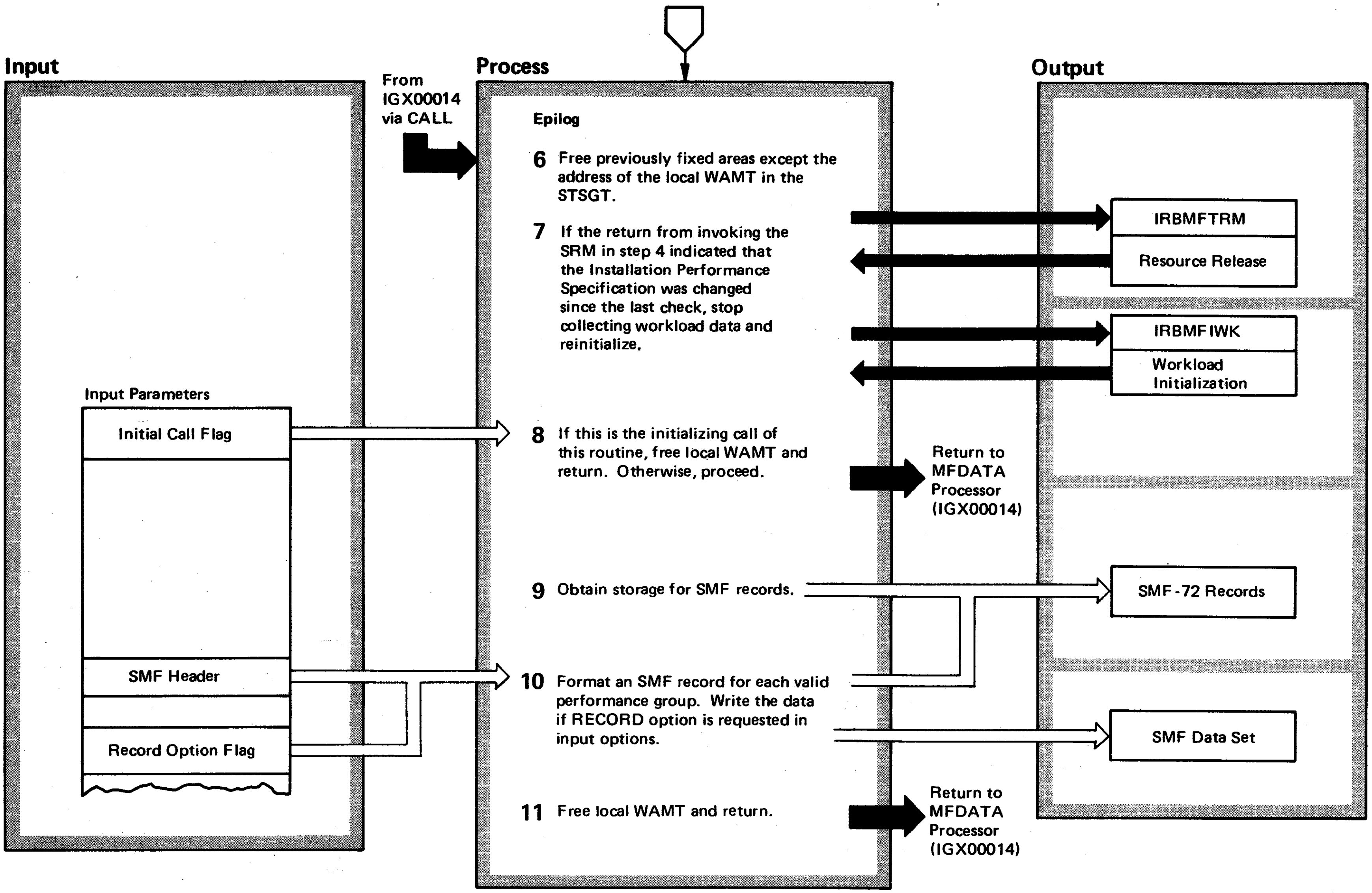
4 Invoke the workload activity data collection facility of the System Resources Manager (SRM).

5 Save the next entry point.

Diagram 7-17. Interval MG Routine for Workload (IRBMFDWP) (Part 2 of 4)

Extended Description	Module	Label
The Interval Routine for Workload (IRBMFDWP) builds the internal image of SMF-72 records from data collected by the Workload manager of the System Resources Manager (SRM). If required by input option selection, IRBMFDWP also copies the SMF record image to the SMF output data set.	IRBMFDWP	
Prolog		
1 The Interval Routine for Workload (IRBMFDWP) uses the GETMAIN macro instruction to obtain storage in supervisor key for the Workload Activity Measurement Table (WAMT).	IRBMFDWP	
2 IRBMFDWP uses the PGFIX macro instructions to page-fix the data area and instructions of IRBMFDWP. Item STRVNSGT is updated to indicate the next available slot in the Storage Resource Table (STSGT).	IRBMFDWP	
3 The entry point of the Move part of IRBMFDWP is saved in the Supervisor Measurement Area (STSMA) to implement a special return sequence, which does not free storage and does not invalidate addressing. The purpose of this return sequence is to separate each interval MG routine into three parts: Prolog, Move, and Epilog. The Prologs of all MG routines are all executed before any Move, and all the Move parts before any Epilog. Because of the special return sequences used, however, each interval MG routine appears to be executed without any break, from start of Prolog through end of Epilog.	IRBMFDWP	DWMOVE
Move		
4 Issue a SYSEVENT WKLDCOLL, which generates a branch entry to the SRM. SRM copies workload data from the global WAMT to the local WAMT.	IRARMINT	
5 Save entry point in STSMA for epilog segment.	IRBMFDWP	DWEPILOG

Diagram 7-17. Interval MG Routine for Workload (IRBMFDWP) (Part 3 of 4)



Epilog

6 Free previously fixed areas except the address of the local WAMT in the STSGT.

7 If the return from invoking the SRM in step 4 indicated that the Installation Performance Specification was changed since the last check, stop collecting workload data and reinitialize.

8 If this is the initializing call of this routine, free local WAMT and return. Otherwise, proceed.

9 Obtain storage for SMF records.

10 Format an SMF record for each valid performance group. Write the data if RECORD option is requested in input options.

11 Free local WAMT and return.

Diagram 7-17. Interval MG Routine for Workload (IRBMFDWP) (Part 4 of 4)

Extended Description	Module	Label
Epilog		
6 Remove the address of the storage, from the Storage Resource Table (STSGT). (The address of the local WAMT in the STSGT is not removed until its storage is freed in step 8 or 9.)	IRBMFDWP	
7 If the IPS changes, issue a SYSEVENT WKLDTERM to terminate the recording of workload data. Call General Resource Release to free the global WAMT and MF/1 workload measurement resources. Then call workload initialization to re-initialize workload activity data collection for the new IPS.	IRARMINT IRBMFTRM IRBMFIWK	
8 IRBMFDWP issues a FREEMAIN macro instruction to release storage for the local WAMT.	IRBMFDWP	MFFREWAM
9 The amount of storage obtained in user's key (the key in the TCB) is determined as follows: No. of bytes required = 8 + (highest performance group no.) times (length of WAMTNDX) + (total number of performance groups) times (length of SMF72B) + (total no. of valid performance group numbers) times (length of SMFRCD72 + length of SMF72A)	IRBMFDWP	

Extended Description	Module	Label
10 Following is the SMF records area format:		

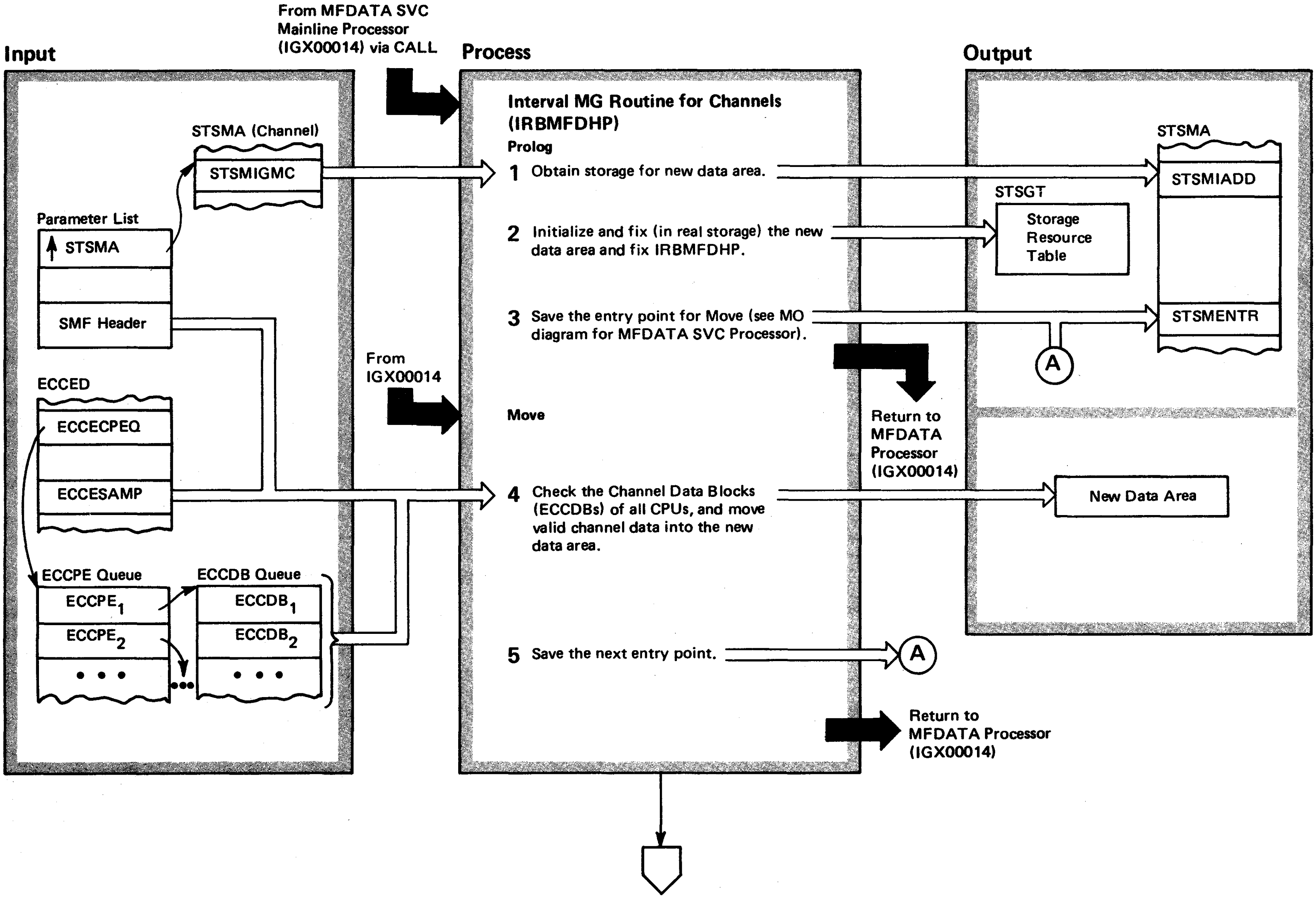
Start Length
highest perf group number
WAMTNDX_1
WAMTNDX_2
. . .
WAMTNDX highest perf group number
SMFRCD72_1
SMF72A_1
SMF72B1_1
SMF72B1_2
SMFRCD72_2
. . .

Where WAMTNDX is the i^{th} index to the SMF72 record associated with PGi (or zero if PGi is not a valid performance group).

IRBMFDWP issues an SMFWTM macro instruction to copy each record to the SMF data set if RECORD was requested.

11 See step 8.	IRBMFDWP	MFFREWAN

Diagram 7-18. Interval MG Routine for Channels (IRBMFDHP) (Part 1 of 4)



Interval MG Routine for Channels (IRBMFDHP)

Prolog

1 Obtain storage for new data area.

2 Initialize and fix (in real storage) the new data area and fix IRBMFDHP.

3 Save the entry point for Move (see MO diagram for MFDATA SVC Processor).

Move

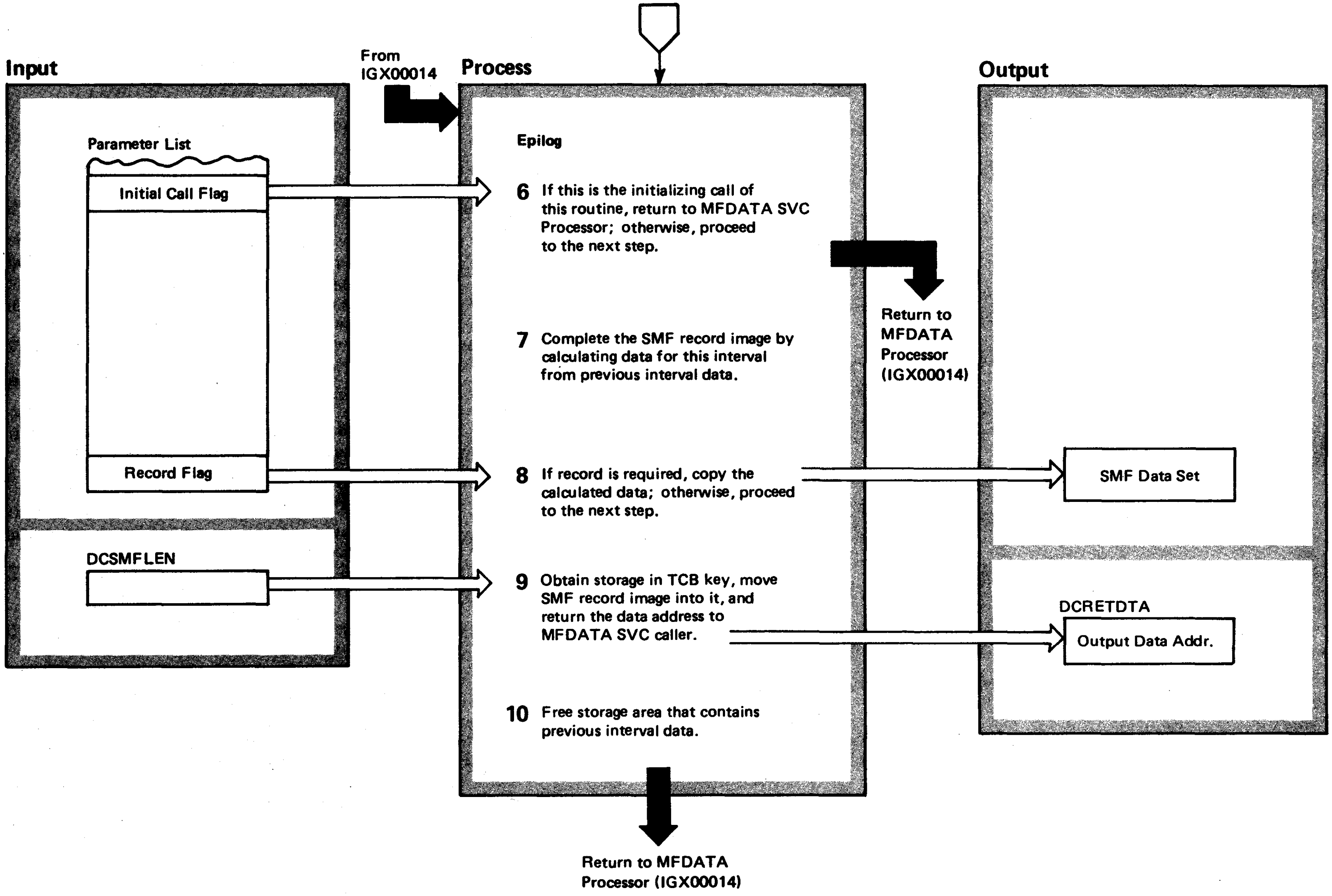
4 Check the Channel Data Blocks (ECCDBs) of all CPUs, and move valid channel data into the new data area.

5 Save the next entry point.

Diagram 7-18. Interval MG Routine for Channels (IRBMFDHP) (Part 2 of 4)

Extended Description	**Module**	**Label**
The Interval MG Routine for Channels (IRBMFDHP) receives control from the MFDATA SVC Mainline Processor at the end of each interval if channel activity reports are required. IRBMFDHP obtains and formats (sample) cycle data collected by the event-driven channel routines IRBMFECH and IRBMFTCH. IRBMFDHP records the data on the SMF data set (via the SMFWTM macro instruction) if RECORD is specified as an input option.	IRBMFDHP	
Prolog		
1 Use the GETMAIN macro instruction to obtain the required storage in key zero.	IRBMFDHP	
2 Store the subpool number and the length of the storage area obtained into the first word of the area. Use the PGFIX macro instruction to fix the data area and IRBMFDHP.		
3 Save the entry point, as described in the M.O. diagram, MFDATA SVC Mainline Processor (IGX00014), for use in returning to the Move part of IRBMFDHP.	IRBMFDHP	DHMOVE
Move		
4 Partially initialize the SMF record image in storage. Then check through the channel Data Blocks (ECCDBs) associated with each CPU. (There is a CPU Element (ECCPE) entry for each CPU; each CPE entry points to one or more ECCDB entries.) Move data from each ECCDB to an associated part of the new data area.	IRBMFDHP	
5 Save the entry point for returning to the Epilog segment.	IRBMFDHP	DHEPILOG

Diagram 7-18. Interval MG Routine for Channels (IRBMFDHP) (Part 3 of 4)



6 If this is the initializing call of this routine, return to MFDATA SVC Processor; otherwise, proceed to the next step.

7 Complete the SMF record image by calculating data for this interval from previous interval data.

8 If record is required, copy the calculated data; otherwise, proceed to the next step.

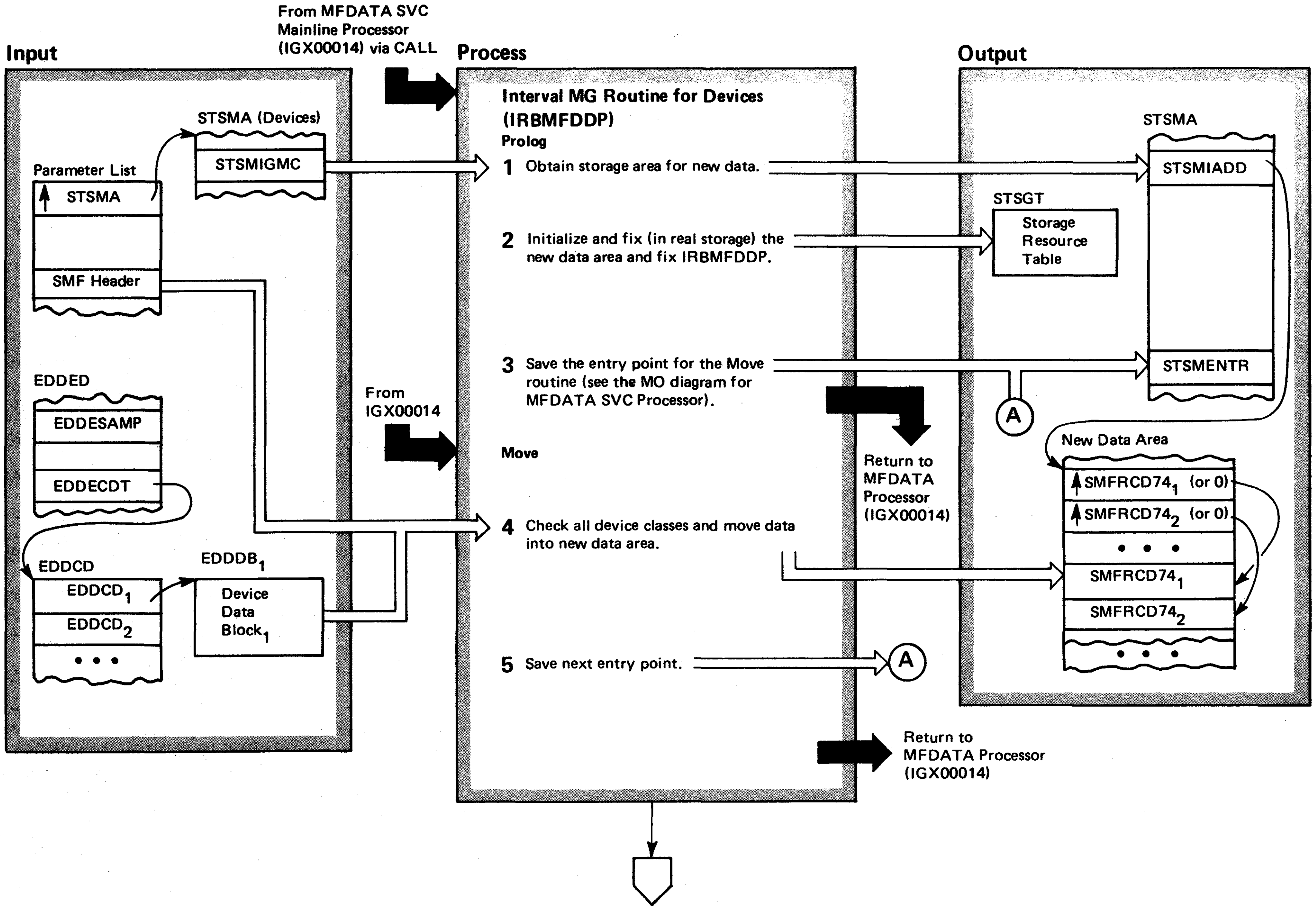
9 Obtain storage in TCB key, move SMF record image into it, and return the data address to MFDATA SVC caller.

10 Free storage area that contains previous interval data.

Diagram 7-18. Interval MG Routine for Channels (IRBMFDHP) (Part 4 of 4)

Extended Description	Module	Label
Epilog		
6 On the initializing call to the MFDATA SVC, the MFDATA SVC Processor calls the interval MG routine to obtain initial values of measurement data, which are required at the end of the measurement interval to calculate data for that interval. Processing ends here on that call.	IRBMFDHP	
7 There is an SMF73B entry for each channel whether or not it was detected online during the interval. At this point, entries for channels not online during the interval are eliminated from the record image and remaining entries are compressed together.	IRBMFDHP	
8 The internal image of the SMF record is copied to the SMF data set by use of the SMFWTM macro instruction.	IRBMFDHP	
9 Use the GETMAIN macro instruction to obtain the required storage in user key. Use the MODESET macro instruction to switch to the user's (TCB) key.	IRBMFDHP	
10 Release the storage used for the internal image of the SMF record, using a FREEMAIN macro instruction.		

Diagram 7-19. Interval MG Routine for Devices (IRBMFDDP) (Part 1 of 4)



Interval MG Routine for Devices (IRBMFDDP)

Prolog

1 Obtain storage area for new data.

2 Initialize and fix (in real storage) the new data area and fix IRBMFDDP.

3 Save the entry point for the Move routine (see the MO diagram for MFDATA SVC Processor).

Move

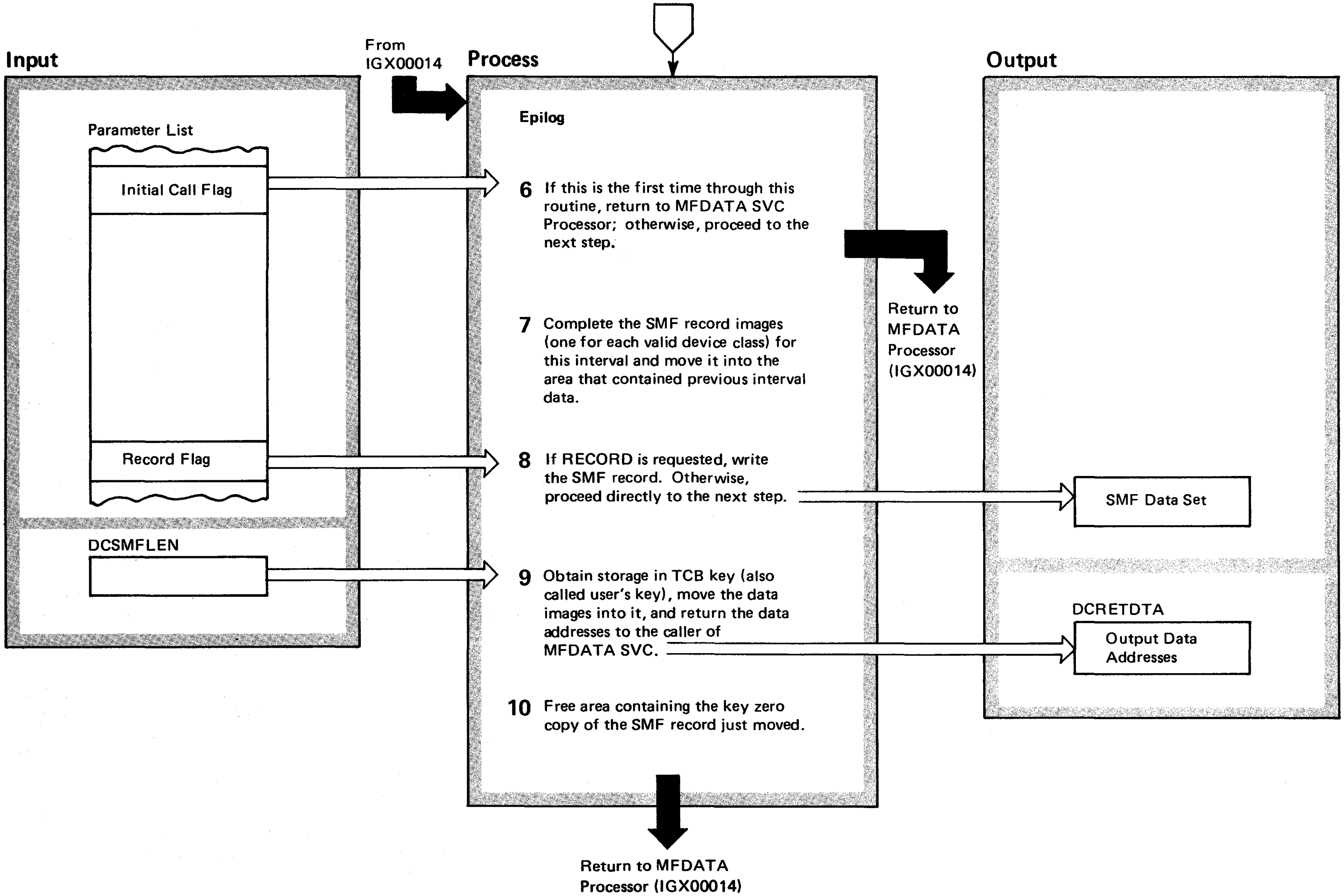
4 Check all device classes and move data into new data area.

5 Save next entry point.

Diagram 7-19. Interval MG Routine for Devices (IRBMFDDP) (Part 2 of 4)

Extended Description	**Module**	**Label**
The Interval Routine for Devices (IRBMFDDP) receives control from the MFDATA SVC Mainline Processor (IGX00014) at the end of each interval if any device reports are required. IRBMFDDP builds the internal image of one or more device data SMF records (SMFRCD74; one record for each class of device report requested) from data collected in event control blocks by the device data event-driven sampling routine (IRBMFEDV). If requested in the input options, IRBMFDDP copies the internal record images to the SMF data set (via the SMFWTM macro instruction).	IRBMFDDP	
Prolog		
1 Use the GETMAIN macro instruction to obtain the required storage in key zero.	IRBMFDDP	
2 Store the subpool and length of the storage obtained into the first word of the area. Use the PGFIX macro instruction to fix the data area and IRBMFDDP.	IRBMFDDP	
3 Save the entry point, as described in the M.O. diagram, MFDATA SVC Mainline Processor (IGX00014), for use in returning to the Move part of IRBMFDDP.	IRBMFDDP	DDMOVE
Move		
4 Initialize the images of the SMF records. Check all device classes (one class is associated with each EDDCD), and move data from the Device Data Block (EDDDB) entries and the Device Event Data Table (EDDED) into the SMF74 record image corresponding to that device class. If no devices exist for a class or if no measurements are required for a class, the pointer for the SMF74 record image is set to zero.	IRBMFDDP	
5 Save entry for returning to Epilog segment.	IRBMFDDP	

Diagram 7-19. Interval MG Routine for Devices (IRBMFDDP) (Part 3 of 4)



Epilog

6 If this is the first time through this routine, return to MFDATA SVC Processor; otherwise, proceed to the next step.

7 Complete the SMF record images (one for each valid device class) for this interval and move it into the area that contained previous interval data.

8 If RECORD is requested, write the SMF record. Otherwise, proceed directly to the next step.

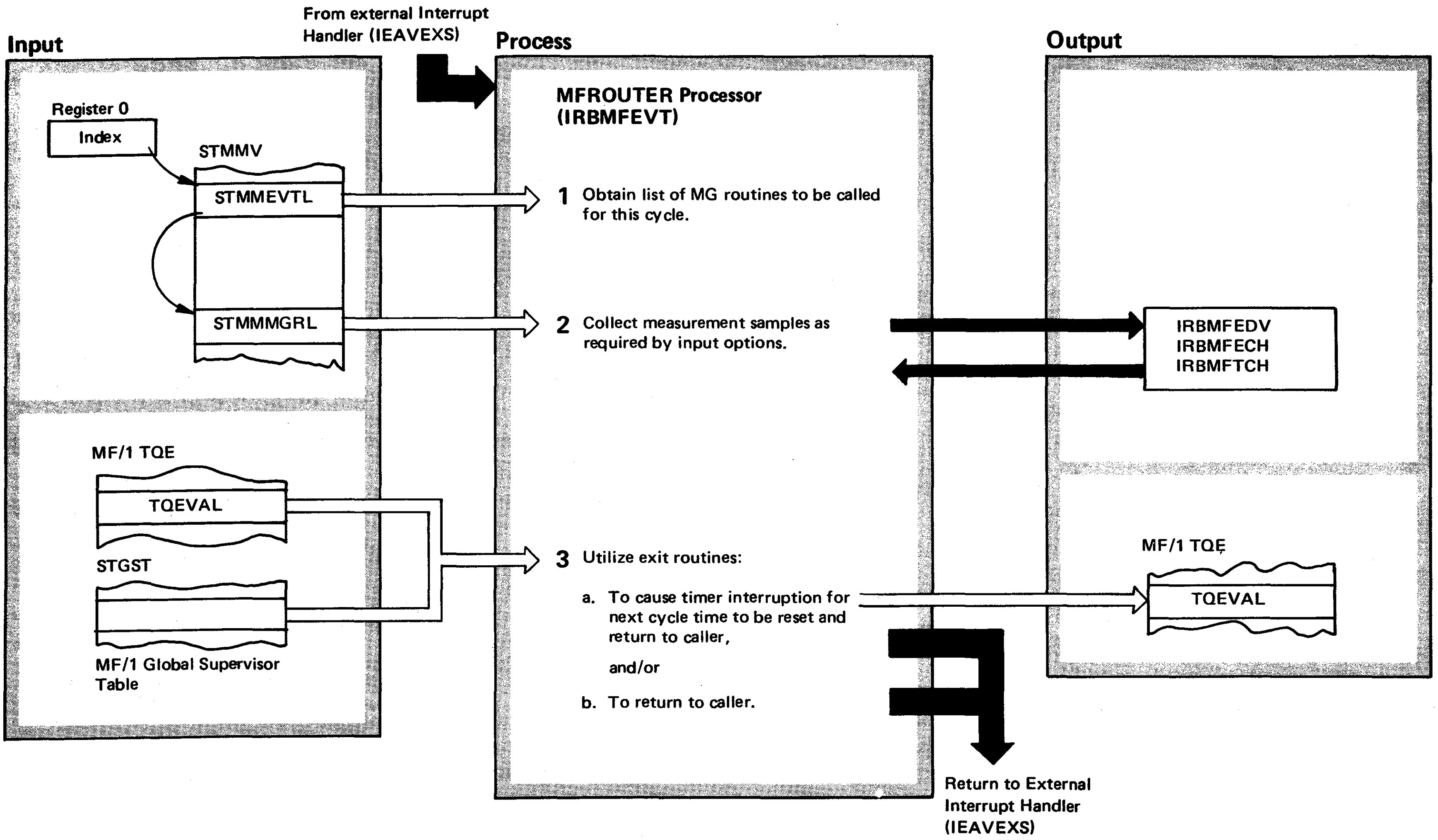
9 Obtain storage in TCB key (also called user's key), move the data images into it, and return the data addresses to the caller of MFDATA SVC.

10 Free area containing the key zero copy of the SMF record just moved.

Diagram 7-19. Interval MG Routine for Devices (IRBMFDDP) (Part 4 of 4)

Extended Description	**Module**	**Label**
Epilog		
6 On the initializing call to the MFDATA SVC, it calls interval-driven MG routines to obtain a first set of wrap-around measurements for use at the end of the first interval in calculating values for that interval. Processing ends here on that call.	IRBMFDDP	DDEPILOG
7 For each device class for which a device exists and for which measurements are required, place the data for the interval just ended in the record for previous interval, overlaying previous data where necessary. For each SMF74B record, which exists for a device whether or not it appears at all online during the interval, the determination is made of whether or not to keep it. If no device measurements are associated with the SMF74B record, the record is eliminated and the other records compressed. All the SMF74 records retained are sorted into order of ascending device address.	IRBMFDDP	
8 Use the SMFWTM macro instruction to copy the internal images to the SMF data set.	IRBMFDDP	
9 Use the GETMAIN macro instruction to obtain the required storage in user key. Use the MODESET macro instruction to change to user key.	IRBMFDDP	
10 Use the FREEMAIN macro instruction to release the storage obtained for the internal images of SMF records.	IRBMFDDP	

Diagram 7-20. MFROUTER SVC Processor (IRBMFEVT) (Part 1 of 2)



MFROUTER Processor (IRBMFEVT)

1 Obtain list of MG routines to be called for this cycle.

2 Collect measurement samples as required by input options.

3 Utilize exit routines:

a. To cause timer interruption for next cycle time to be reset and return to caller,

and/or

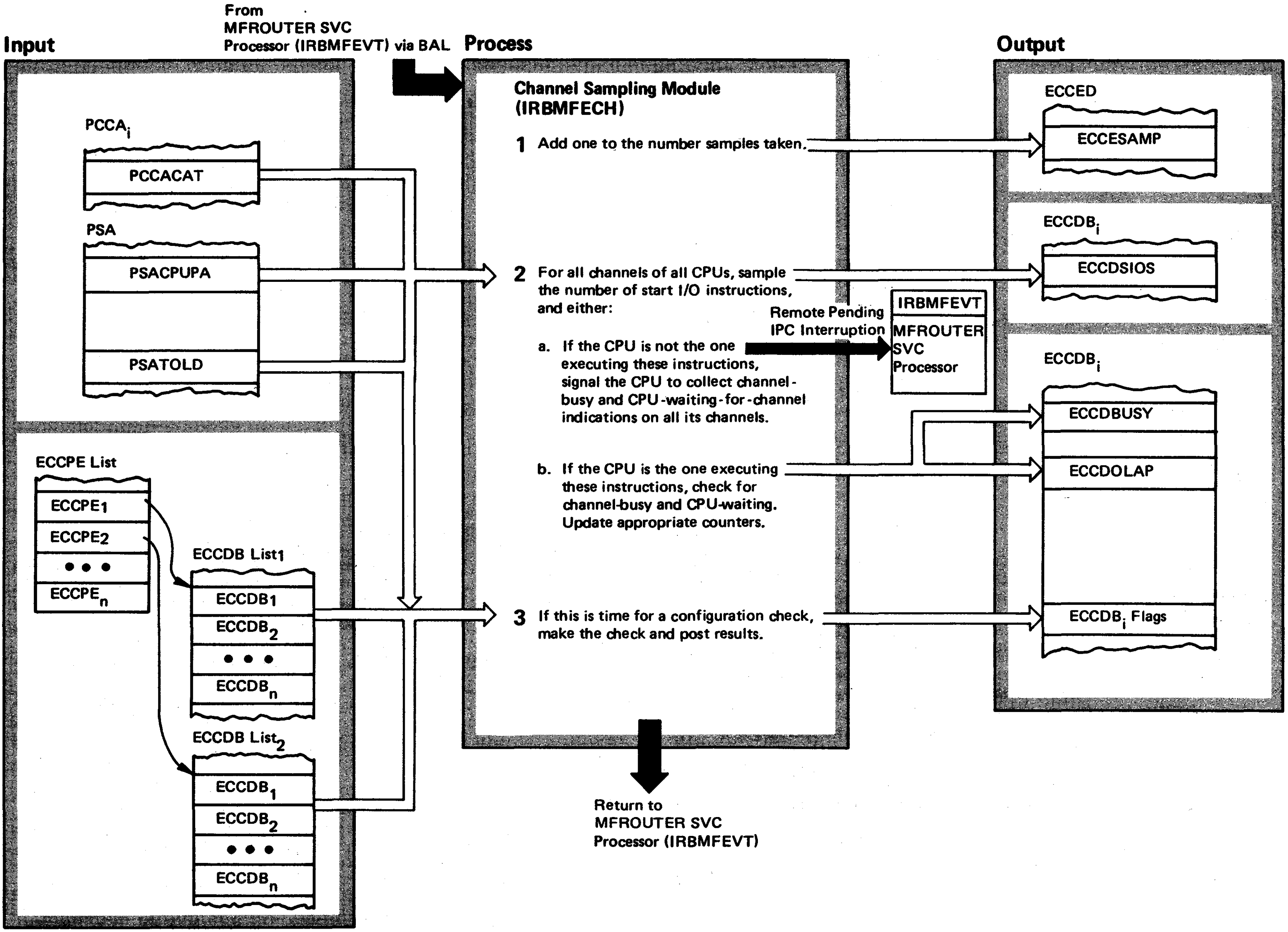
b. To return to caller.

Diagram 7-20. MFROUTER SVC Processor (IRBMFEVT) (Part 2 of 2)

Extended Description	Module	Label
The MFROUTER Processor (IRBMFEVT) calls the event-driven MG routines and a routine to reset the MF/1 Timer Queue Element (TQE) after the time expires in the TQE. The routines called are: • Channel Event MG (IRBMFECH) • Second CPU Channel Event MG (IRBMFTCH) • Device Event MG (IRBMFEDV) • Timer Enqueue (IEAQTE00)	IRBMFEVT	
1 Events are timer interruptions or remote pending interruptions on CPU-to-CPU communications. Timer interruptions are at the rate of sample time periods for device and/or channel sampling. CPU-to-CPU communication interruptions are at the same rate, but are only caused by one CPU requesting another to sample channel data.	IRBMFEVT	

Extended Description	Module	Label
2 The MFROUTER Processor branches to the MG routines in the order set up by their entry addresses in the MFROUTER Vector Table (STMMV), which was set according to entry options by MFSTART and modules connected with MFSTART.	IRBMFECH IRBMFTCH IRBMFEDV	
3a If IRBMFEVT was entered in response to a timer interruption, it branches last to a subroutine (IRBMFEVE) to reset the timer (enque the TQE onto the timer queue). The address of the subroutine is placed in the MG routine loop (MFROUTER Vector Table) by the MFSTART SVC Processor (IGX00013).	IRBMFEVE IGX00013	
3b If IRBMFEVT was entered in response to a CPU-to-CPU interruption, it branches last to a subroutine (IRBMFEVL) to restore status and return to the caller of IRBMFEVT.	IRBMFEVT	IRBMFE

Diagram 7-21. Channel Sampling Module (IRBMFECH) (Part 1 of 2)



Channel Sampling Module (IRBMFECH)

1 Add one to the number samples taken.

2 For all channels of all CPUs, sample the number of start I/O instructions, and either:

a. If the CPU is not the one executing these instructions, signal the CPU to collect channel-busy and CPU-waiting-for-channel indications on all its channels.

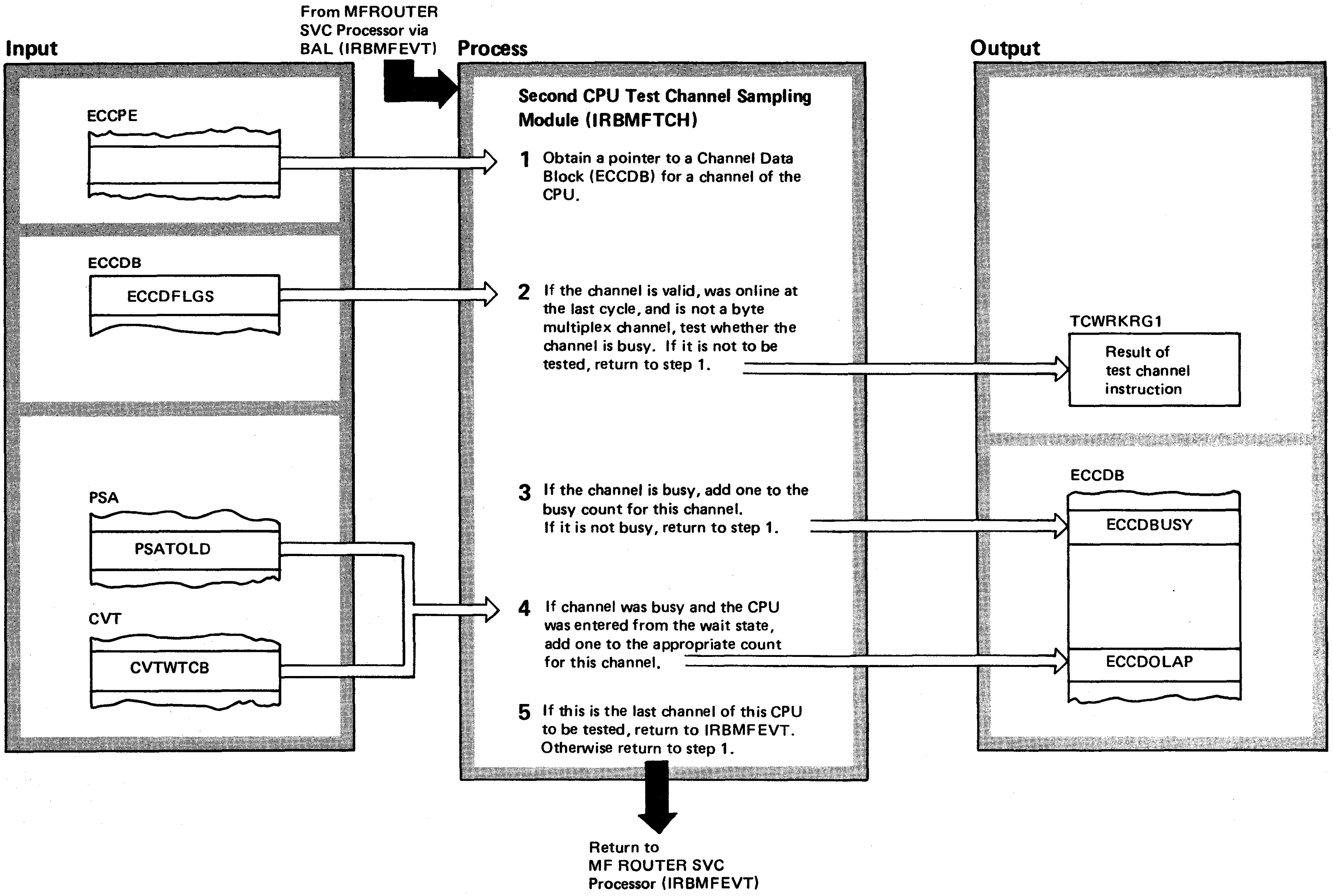
b. If the CPU is the one executing these instructions, check for channel-busy and CPU-waiting. Update appropriate counters.

3 If this is time for a configuration check, make the check and post results.

Diagram 7-21. Channel Sampling Module (IRBMFECH) (Part 2 of 2)

Extended Description	Module	Label
The Channel Sampling Module (IRBMFECH) receives control from MFROUTER SVC Processor at each cycle sample time. IRBMFECH collects the channel measurement samples IOS provides and monitors channel status with regard to the channels being online or offline.	IRBMFECH	
1 The Channel Sampling Module increases counter ECCESAMP in the Channel Event Data Table (ECCED).	IRBMFECH	
2 IRBMFECH checks channels through CPU Entry Tables (ECCFE), which contain a pointer to a Channel Data Block (ECCDB) list for each CPU. If item ECCDVALD in the CDB is on, that CPU was online at one or more configuration checks, and therefore, IRBMFECH obtains the count of Start I/O (SIO) instructions issued to the channel. This count of SIOs is obtained from the Physical Configuration Communication Area (PCCA).	IRBMFECH	
2a IRBMFECH signals the other CPU with the RPSGNL macro instruction, with the PCCA address in register 1.	IRBMFECH	
2b IRBMFECH uses entries from the Channel Availability Table (CAT) to increase the count of Start I/O (SIO) instructions since the last sample. If this channel is not offline and is not a byte multiplexor, IRBMFECH issues a TCH instruction to test whether the channel is now busy. If the channel is busy, the routine adds one to the count of busy samples. If the CPU was waiting when MF/1 was given control, IRBMFECH adds one to the count of CPU waiting and channel busy (ECCDOLAP).	IRBMFECH	
3 The Channel's Configuration is checked if the number of samples taken is multiple of the configuration check field. If the CPU is online, set appropriate flags in the ECCED. If a change in configuration has occurred since the last check, the routine sets appropriate flags.	IRBMFECH	

Diagram 7-22. Second CPU Test Channel Sampling Module (IRBMFTCH) (Part 1 of 2)



Second CPU Test Channel Sampling Module (IRBMFTCH)

1 Obtain a pointer to a Channel Data Block (ECCDB) for a channel of the CPU.

2 If the channel is valid, was online at the last cycle, and is not a byte multiplex channel, test whether the channel is busy. If it is not to be tested, return to step 1.

3 If the channel is busy, add one to the busy count for this channel. If it is not busy, return to step 1.

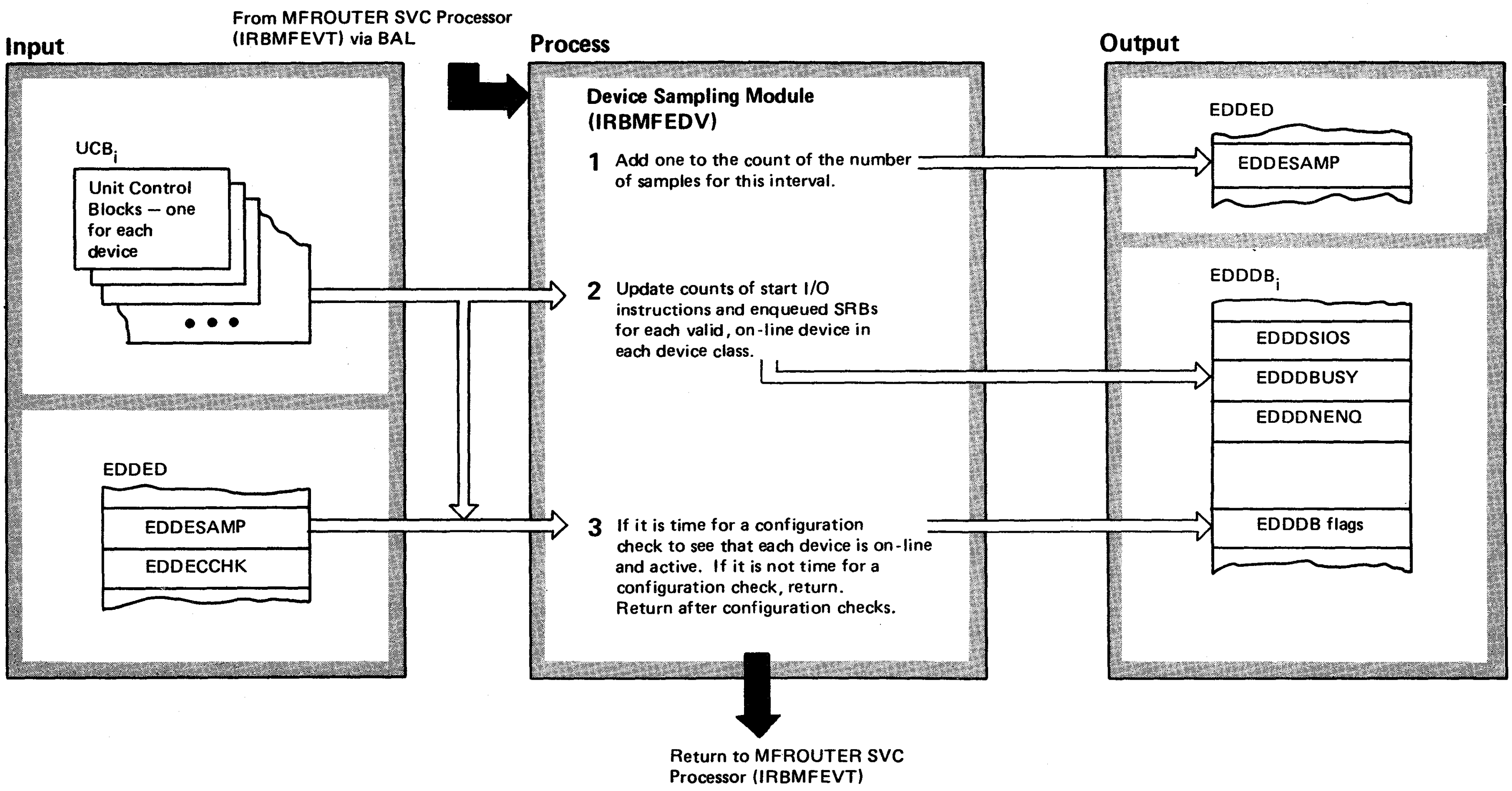
4 If channel was busy and the CPU was entered from the wait state, add one to the appropriate count for this channel.

5 If this is the last channel of this CPU to be tested, return to IRBMFEVT. Otherwise return to step 1.

Diagram 7-22. Second CPU Test Channel Sampling Module (IRBMFTCH) (Part 2 of 2)

Extended Description	Module	Label
The Second CPU Test Channel Sampling Module (IRBMFTCH) collects data for each channel (of one CPU) that has been active during the entire previous cycle period. The data is collected by issuing a Test Channel instruction and noting the response of the channel.	IRBMFTCH	
1 Each Channel Data Block (ECCDB) is storage for data on one channel. An ECCDB is defined for each channel possible to be connected to a CPU.	IRBMFTCH	
2 IRBMFTCH tests bits set by other modules and, if any test fails, passes over this channel.	IRBMFTCH	
3 A count is kept of the times the channel is found busy during a cycle test. The busy status is read as a result of the Test Channel instruction.	IRBMFTCH	
4 A CPU-waiting count is accomplished similarly to the channel-busy count.	IRBMFTCH	
5 A DO loop is used to step through tests for all channels of a CPU.	IRBMFTCH	

Diagram 7-23. Device Sampling Module (IRBMFEDV) (Part 1 of 2)



Device Sampling Module (IRBMFEDV)

1 Add one to the count of the number of samples for this interval.

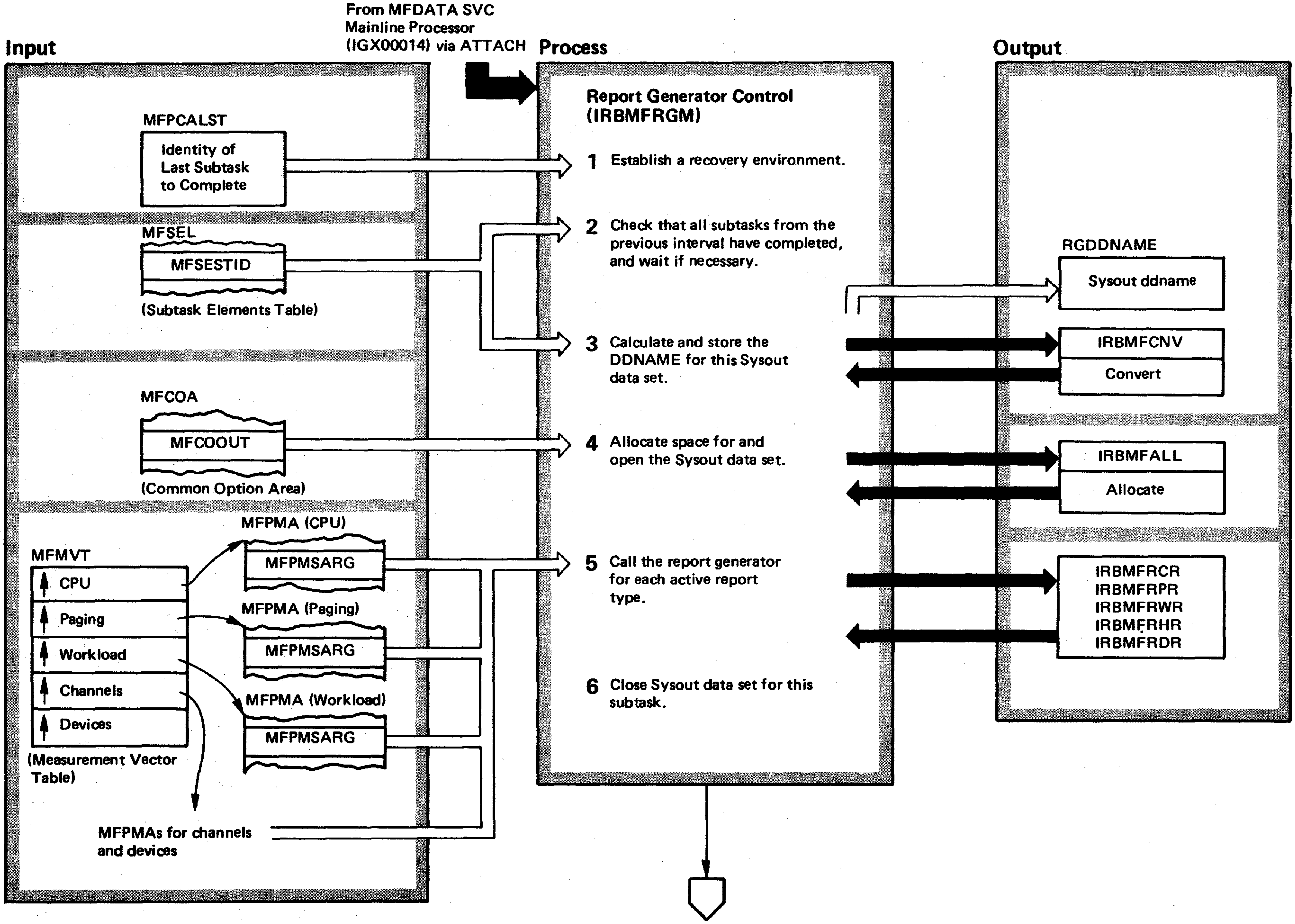
2 Update counts of start I/O instructions and enqueued SRBs for each valid, on-line device in each device class.

3 If it is time for a configuration check to see that each device is on-line and active. If it is not time for a configuration check, return. Return after configuration checks.

Diagram 7-23. Device Sampling Module (IRBMFEDV) (Part 2 of 2)

Extended Description	Module	Label
The Device Sampling Module (IRBMFEDV) receives control from the MFROUTER SVC Processor (IRBMFEVT) at each cycle sample time. IRBMFEDV gathers sample data on the use of I/O devices, as maintained by IOS.	IRBMFEDV	
1 The Device Sampling Module increases counter EDDESAMP in the Device Event Data Table (EDDED).	IRBMFEDV	
2 The Device Sampling Module checks the Device Class Data Table (EDDCD) entry for devices that exist in that class. Nonzero entries in the EDDCD point to one or more Device Data Blocks (EDDDB) for that class. Each EDDDB entry points to a Unit Control Block (UCB), which contains data with which to add to the following wrap-around counts in the EDDDB: a. EDDDSIOS, which is the current number of SIOs for that device in this interval. b. EDDDBUSY, which is the current number of samples, in which the device was busy. c. EDDDNENQ, which is the current number of SRBs enqueued on this device.	IRBMFEDV	
3 The configuration of devices is checked if the number of samples is an even multiple of the configuration check field (EDDECCHK): If the online flag in the UCB (UCBONLI) and the alive flag (EDDDALIV) in the Device Data Block (EDDDB) do not match, turn on the configuration changed flag (EDDDCCHG) in the EDDDB, and record the proper status (EDDDALIV). If the device address in the UCB (UCBNAME) and in the EDDDB (EDDDADDR) do not match, turn on the configuration changed flag in the EDDDB and move the current address of the device into EDDDADDR in the EDDDB.	IRBMFEDV	

Diagram 7-24. Report Generator Control (IRBMFRGM) (Part 1 of 4)



1 Establish a recovery environment.

2 Check that all subtasks from the previous interval have completed, and wait if necessary.

3 Calculate and store the DDNAME for this Sysout data set.

4 Allocate space for and open the Sysout data set.

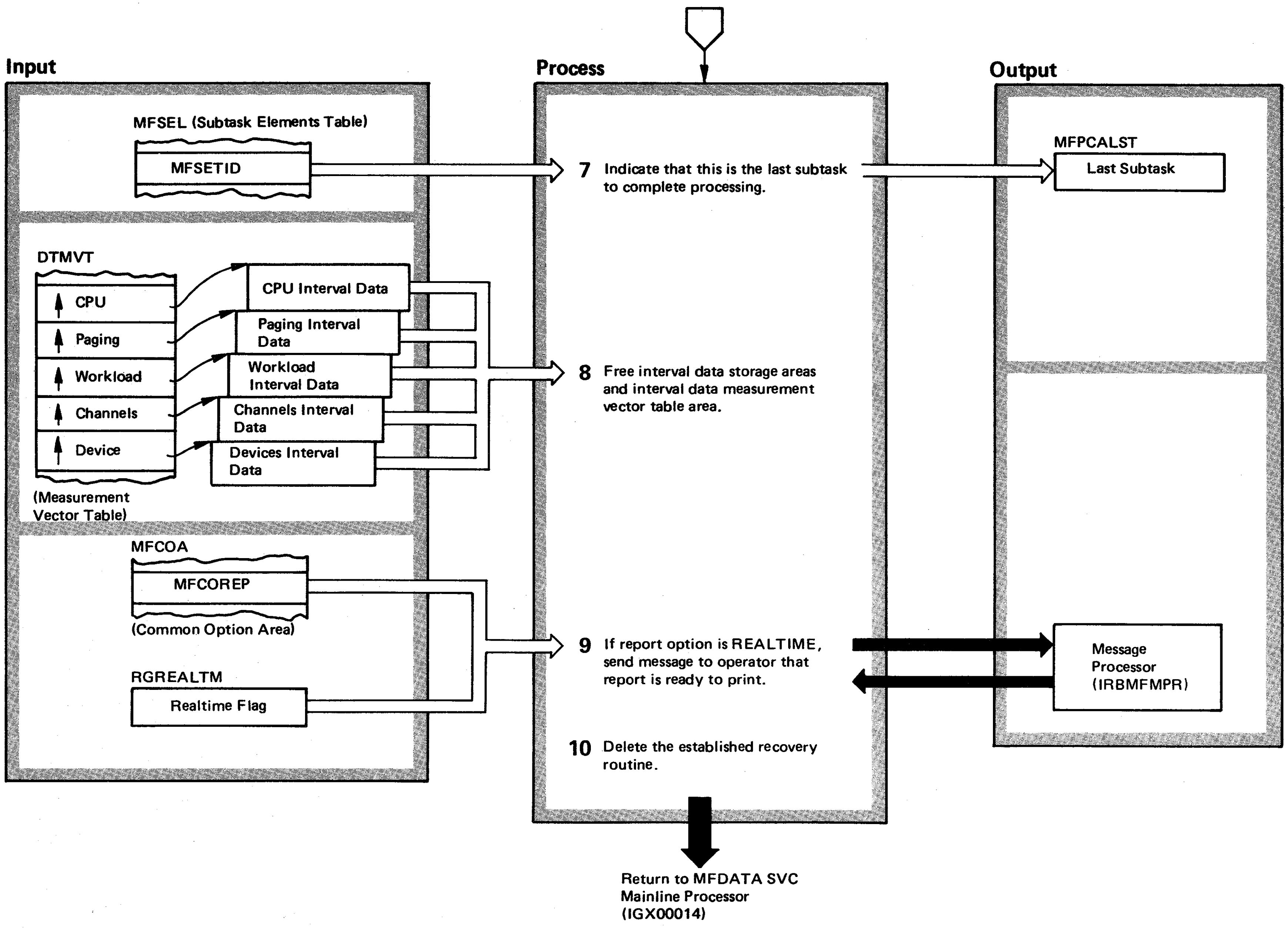
5 Call the report generator for each active report type.

6 Close Sysout data set for this subtask.

Diagram 7-24. Report Generator Control (IRBMFRGM) (Part 2 of 4)

Extended Description	Module	Label
The Report Generator Control (IRBMFRGM) controls the allocation of SYSOUT data space, the calling of report generators for each report type requested, and the freeing of interval measurement data space. IRBMFRGM also informs the operator when reports are ready to print if REALTIME reporting was requested.	IRBMFRGM	
1 Establish an ESTAE recovery routine.	IRBMFSAR	
2 Waiting for all previous subtasks to complete ensures the correct association of reports for each report interval.	IRBMFRGM	
3 The SYSOUT DDNAME is converted into the form MFRnnnnn, where nnnnn are the characters that represent the subtask ident.	IRBMFCNV	
4 The IRBMFALL is called to allocate SYSOUT data space as needed during program execution. It issues SVC 99 to attempt allocation.	IRBMFALL	
5 For each report type, IRBMFRGM loads, calls, and, when the report type has been produced, deletes the required report generator. IRBMFRGM provides an ESTAE data linkage during report generation.	IRBMFRCR IRBMFRPR IRBMFRWR IRBMFRHR IRBMFRDR	
6 If the REALTIME option for report printing is in effect, the SYSOUT data set is printed immediately; otherwise it is printed upon termination of MF/1.	IRBMFRGM	

Diagram 7-24. Report Generator Control (IRBMFRGM) (Part 3 of 4)



7 Indicate that this is the last subtask to complete processing.

8 Free interval data storage areas and interval data measurement vector table area.

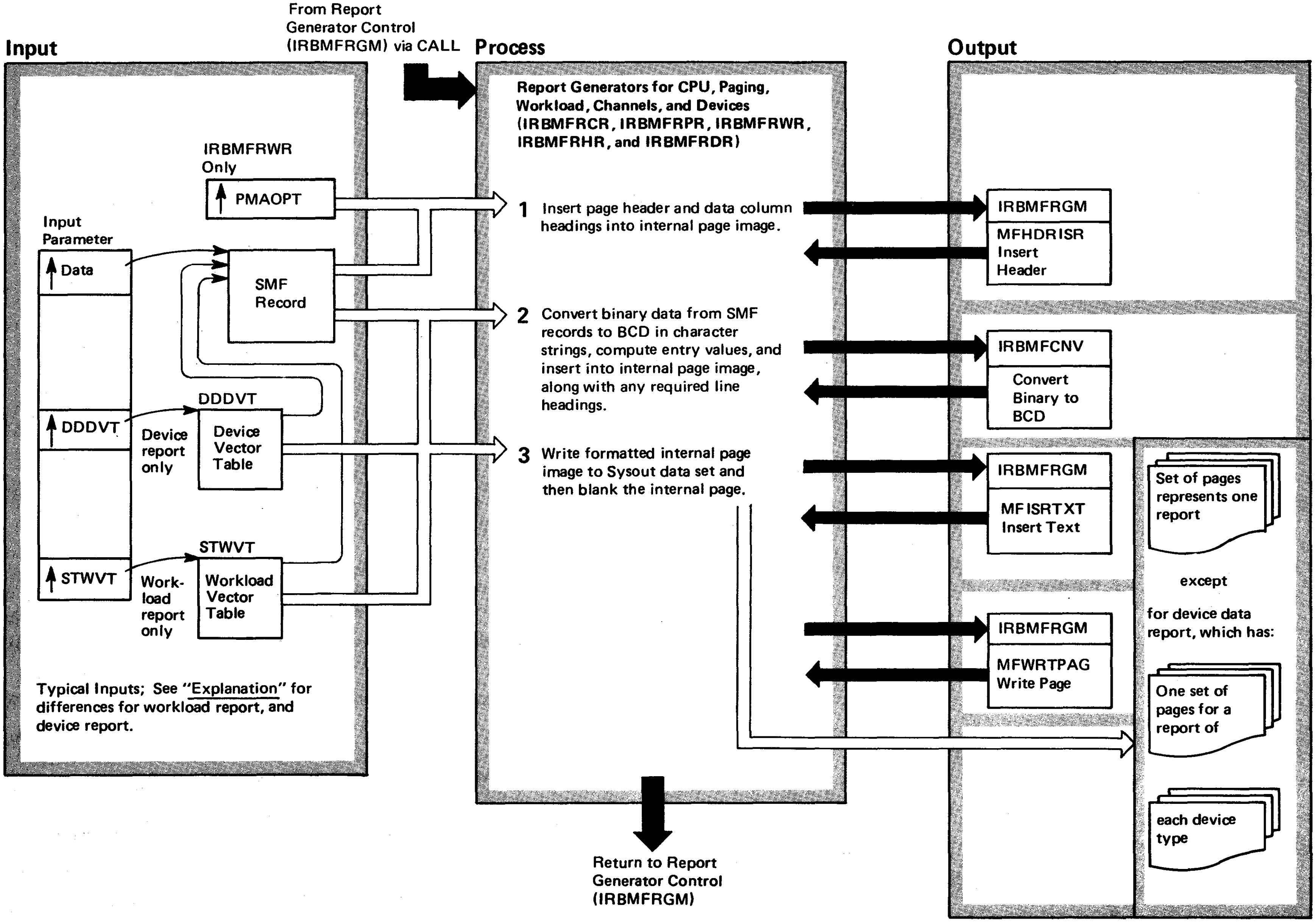
9 If report option is REALTIME, send message to operator that report is ready to print.

10 Delete the established recovery routine.

Diagram 7-24. Report Generator Control (IRBMFRGM) (Part 4 of 4)

Extended Description	**Module**	**Label**
7 The identity of the last subtask to terminate is updated to establish serialization of reports on the printer.	IRBMFRGM	
8 The main storage for this interval's data is not needed after the required reports are written on the SYSOUT data set.	IRBMFRGM	
9 The operator message is MF1 REPORT AVAILABLE FOR PRINTING.	IRBMFMPR	
10 Cancel the previously established ESTAE routine.	IRBMFSAR	
ERROR PROCESSING: If an error occurs while a report is being written, another attempt is made to write all reports. A second error, or an error occurring prior to report writing, will cause an ABEND.		

Diagram 7-25. Report Generators for CPU, Paging, Workload, Channels, and Devices (IRBMFRCR, IRBMFRPR, IRBMFRWR, IRBMFRHR, and IRBMFRDR) (Part 1 of 2)



1 Insert page header and data column headings into internal page image.

2 Convert binary data from SMF records to BCD in character strings, compute entry values, and insert into internal page image, along with any required line headings.

3 Write formatted internal page image to Sysout data set and then blank the internal page.

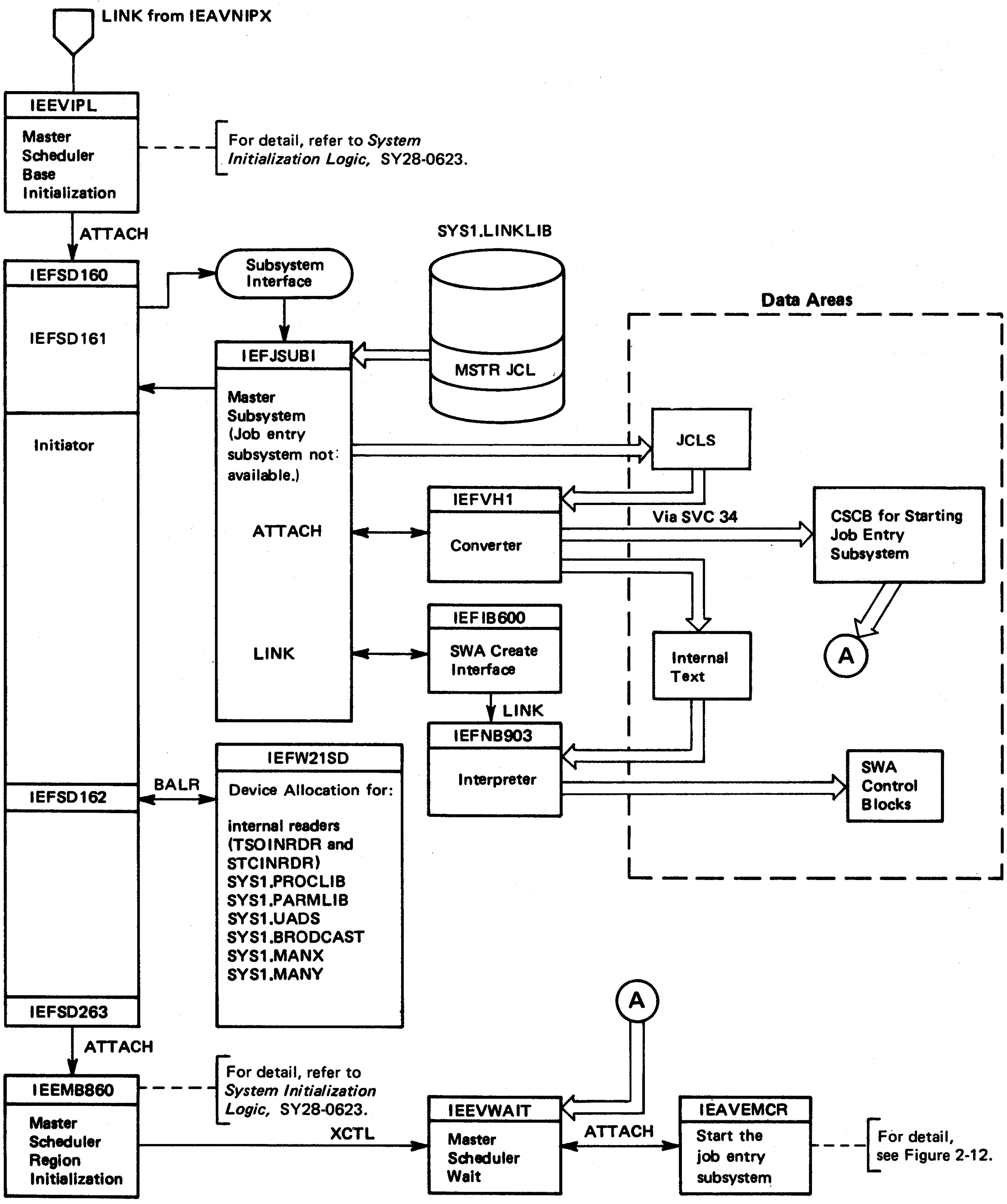
Typical Inputs; See "Explanation" for differences for workload report, and device report.

Diagram 7-25. Report Generators for CPU, Paging, Workload, Channels, and Devices (IRBMFRCR, IRBMFRPR, IRBMFRWR, IRBMFRHR, and IRBMFRDR) (Part 2 of 2)

Extended Description	Module	Label
This M.O. diagram covers the five report generator modules:		
• CPU Activity (IRBMFRCR)	IRBMFRCR	
• Paging Activity (IRBMFRPR)	IRBMFRPR	
• Workload (IRBMFRWR)	IRBMFRWR	
• Channel Activity (IRBMFRHR)	IRBMFRHR	
• Device Activity (IRBMFRDR)	IRBMFRDR	
Each report generator formats interval data for its report type and writes it to a SYSOUT data set for either REAL-TIME or deferred printing.		
1 Each report generator subtask calls procedure MFHDRISR whenever a header is to be written on a new page (see note after step 3).	IRBMFRGM	MFHDRISR
2 After the page and column headers are written, the report generator extracts data from the SMF record image, manipulates it, and writes entries into the internal image of the report page. Parameter MFPMAOPT is used only for the workload report to determine the depth of workload reporting.	IRBMFRGM	MFISRTXT
The report generator routine calls routine IRBMFCNV to convert a signed binary number into its equivalent as a character string. The resulting string is supplied as a fixed length string parameter. The following are provided as input parameters (starting address in register 1) to IRBMFCNV:	IRBMFCNV	
a) the input signed binary value. b) the signed decimal scaling factor for the input value. c) the address of the output string. d) the length of the output string. e) the no. of digits to the right of the decimal pt. f) commas or no commas. g) floating point or no floating point.		
If commas in the output could cause loss of significant digits, they are not inserted. If the output string is *shorter* than necessary, commas are removed first. If the output string still cannot accept the entire value, least significant digits to the right of decimal point are next removed, up to and including the decimal point itself. If this action is sufficient, the return code is 4; otherwise the entire field is filled with asterisks and the return code is 8. If the output string is larger than necessary, it is right justified. The insert text routine is used to put data into the RGM internal page image.	IRBMFRGM	MFISRTXT
3 Subroutine MFWRTPAG writes the internal page image, line by line, to the SYSOUT data set using a QSAM PUT. Blank lines are consolidated, and a single record is written with carriage control characters indicating the number of lines to skip.	IRBMFRGM	MFWRTPAG
Note: Input data formats differ among the five report generators:		
• CPU, paging, and channel report generators each receive a single SMF record image and each produce a single report.	IRBMFRCR IRBMFRPR IRBMFRHR	
The device report generator receives a fixed length list of SMF record images and produces a report for each one. There is input data for each defined device type unless the corresponding Device Vector Table (DDDVT) entry is zero.	IRBMFRDR	
The workload report generator receives a variable length list of SMF record images, preceded by a count, and produces a single report. There is input data for each performance group number (PGN) unless the corresponding Workload Vector Table (STWVT) entry is zero.	IRBMFRWR	

The following four figures illustrate the relationship among some of the job scheduling subcomponents (Details of module-to-module flow within a subcomponent are in 'Section 3: Program Organization.'):

- Figure 2-11 shows the first use of job scheduling code: master scheduler initialization attaches the initiator to start the master scheduler. The initiator attaches IEEMB860, which continues the initialization process and finally passes control to the master scheduler wait module. Starting the master scheduler in this manner allows several system and TSO data sets to be allocated normally. These data sets are then available during the last portion of master scheduler initialization. Note that the master subsystem, rather than a job entry subsystem, converts and interprets the master scheduler's JCL. For more information on master scheduler initialization, refer to *OS/VS2 System Initialization Logic*, SY28-0623.
- Figure 2-12 shows the second use of job scheduling code: the START command for a job entry subsystem is processed. This START command was in the master scheduler JCL that was interpreted during the initiation of the master scheduler. Note that the master subsystem, rather than a job entry subsystem, converts and interprets the job entry subsystem's JCL. The master subsystem starts job entry subsystems and also starts subsystems defined by the installation. For more information on the master subsystem, see 'Master Subsystem' in this section.
- Figure 2-13 depicts general START/LOGON/MOUNT processing. This processing begins with a START, LOGON, or MOUNT command and culminates in the attach of an initiator, a terminal monitor program (TMP), the MOUNT processor, or a started system task (TCAM, for example). A new address space is created for each START, LOGON, or MOUNT command. The value of the new address space ID (ASID) is at least four (master scheduler's ASID is one; auxiliary storage manager's ASID is two; and the primary job entry subsystem's ASID is three).
- Figure 2-14 shows how a normal job enters the system and is attached as a problem program by the initiator. A new address space is not created for each job entering the system; a job executes in the address space of the initiator that attached it. When the job entry subsystem first receives a job's JCL, it stores it on the spool data set. It then passes the JCL through the converter and puts the resultant internal text in another data set. When initiator requests the selection of a job (via the subsystem interface), the job entry subsystem chooses a job and passes the already-existing internal text through the interpreter, creating SWA control blocks. The initiator can now continue with the initiation of the job.



Note: Refer to the index for the page numbers of function diagrams (hipos) that describe the functions of particular modules.

Figure 2-11. Job Scheduling: Initiation of the Master Scheduler

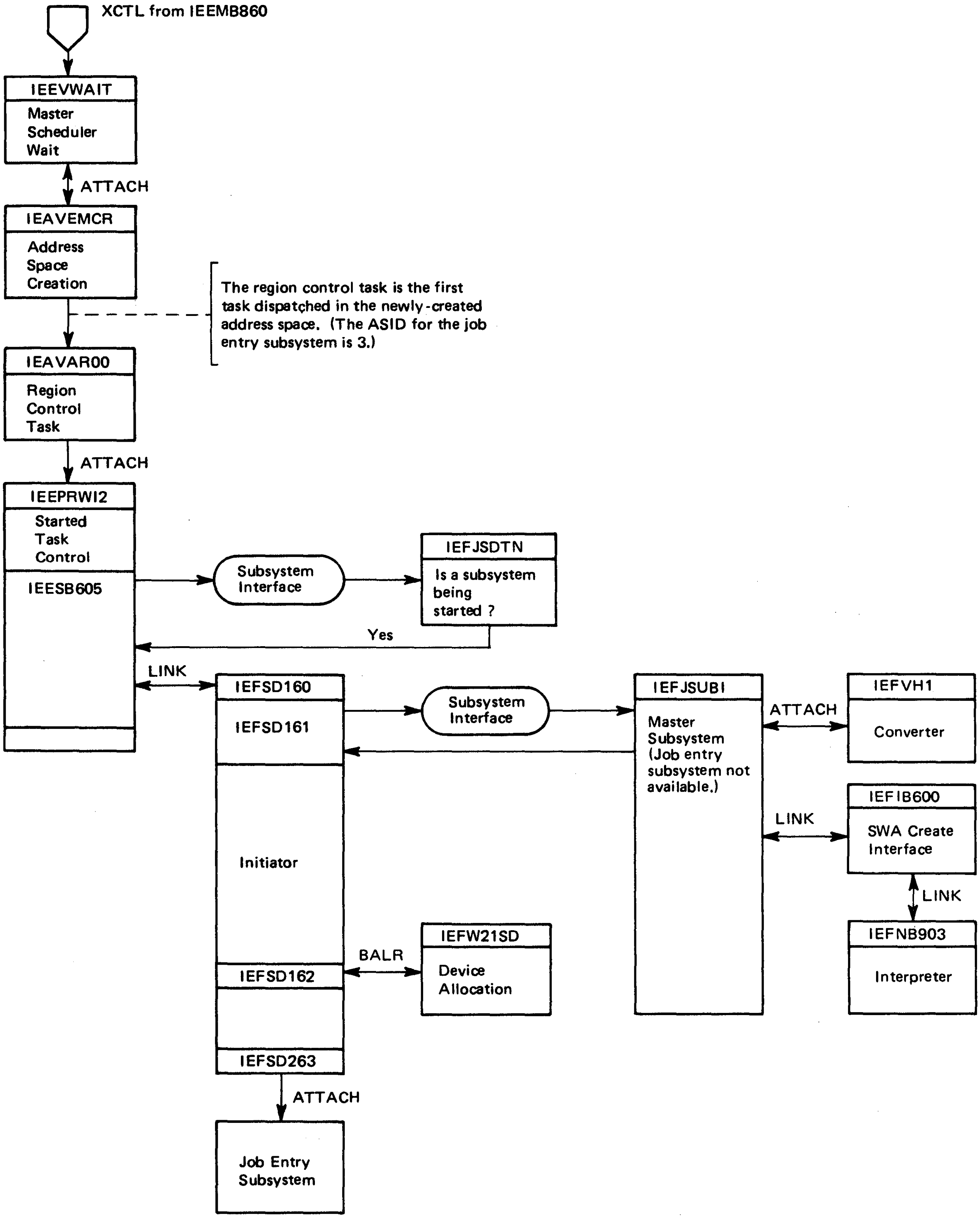


Figure 2-12. Job Scheduling: Initiation of the Job Entry Subsystem

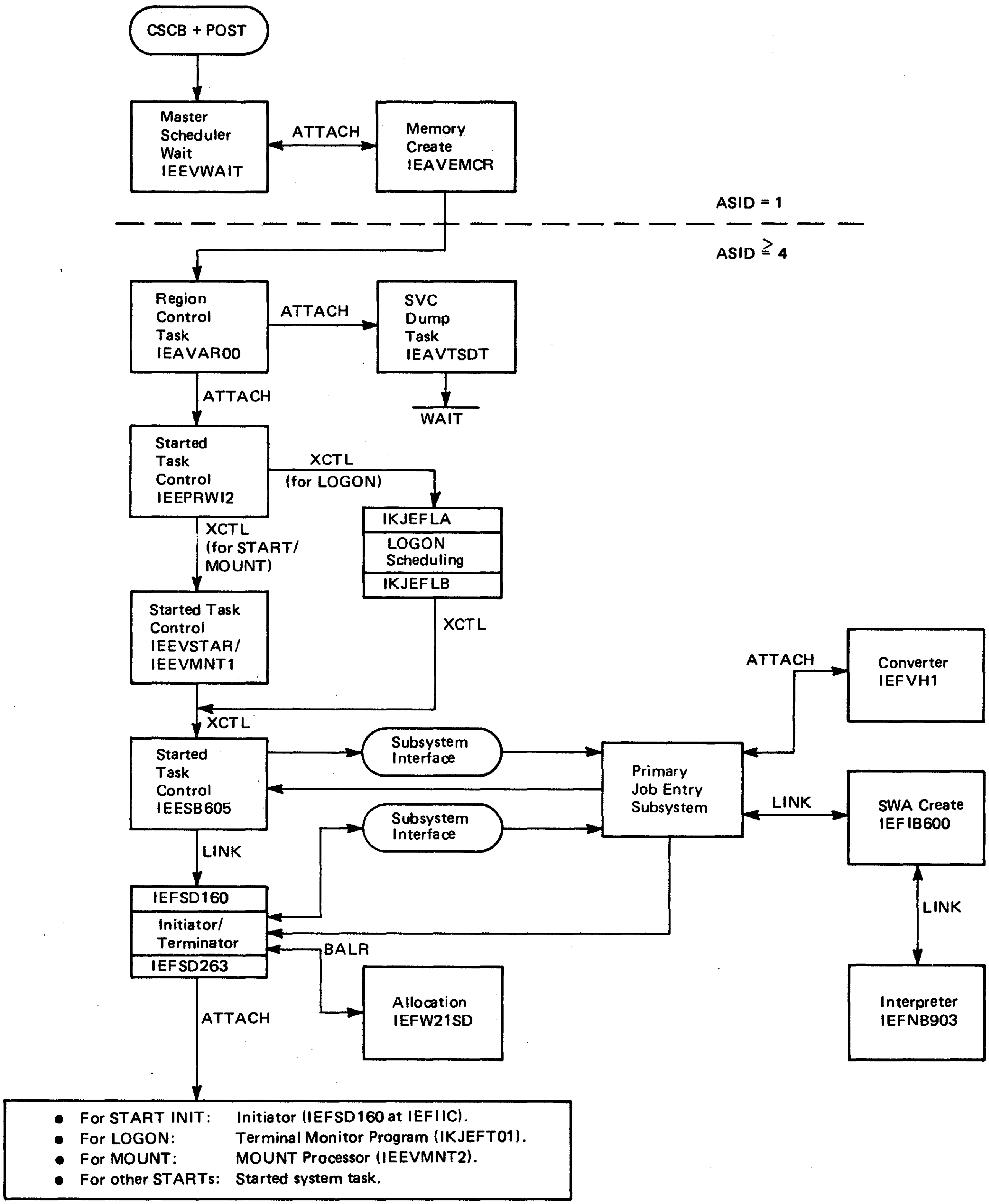


Figure 2-13. Job Scheduling: START/LOGON/MOUNT Initiation

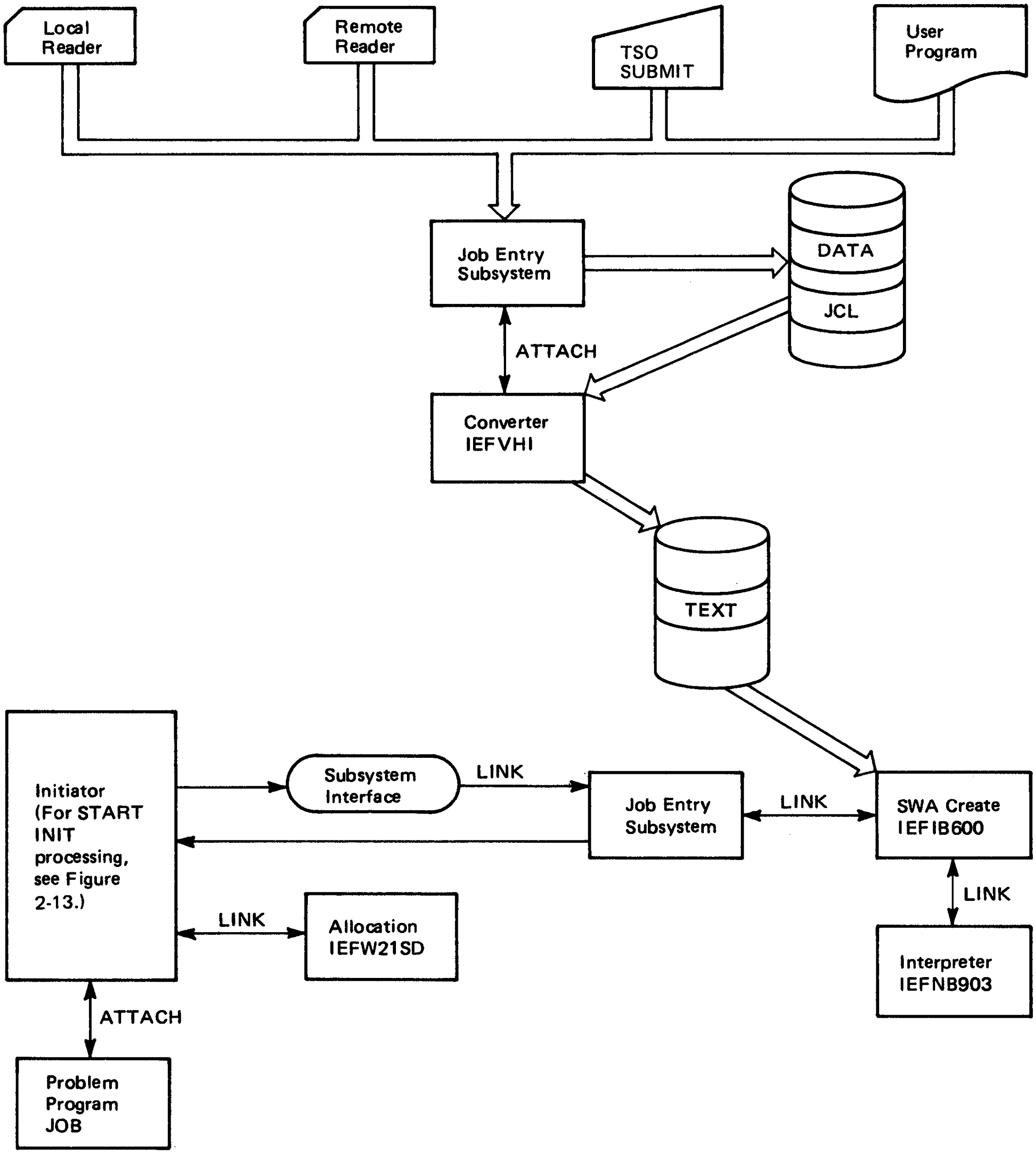


Figure 2-14. Job Scheduling: Normal Job Entry and Initiation

Subsystem Interface

The subsystem interface is the means by which OS/VS2 system routines request services of either the master subsystem or a job entry subsystem. To request subsystem services, a system routine issues the IEFSSREQ macro instruction after placing the correct function code in the SSOB and placing the name of the desired subsystem in the SSIB. The macro instruction causes control to pass to the subsystem interface routine, IEFJSREQ. The specified function code and subsystem name tell the interface routine which subsystem routine gets control. Figure 2-15 lists all existing function codes, their meanings, and the subsystem modules that get control.

A job entry subsystem performs functions related to entering jobs into the system. For example, it handles SYSIN and SYSOUT data sets; it also passes a job's JCL through the converter and interpreter, thus creating SWA control blocks for the job. See 'Job Scheduling' in this section.

On the other hand, the master subsystem does not handle normal jobs. It is used by the system to start the master scheduler and subsystems. A subsystem can be a job entry subsystem (JES2, for example) or another subsystem defined by the installation. Once a job entry subsystem is initiated and ready to accept jobs, the master subsystem is no longer needed for initiation processing. However, if an installation wishes to replace the active job entry subsystem with another version (or to start another subsystem), the master subsystem must be used to start this new version (or the new subsystem).

In addition to starting subsystems, the master subsystem broadcasts requests to all active subsystems. (See note at bottom of Figure 2-15.) For a detailed description of the master subsystem, refer to 'Master Subsystem' in this section. JES2, a job entry subsystem, is described in *OS/VS2 JES2 Logic*, SY28-0622.

Function Code	SSOB Extension ID	Subsystem Function	Subsystem	Module Name	Module Label	Primary Caller
1	SO	Process SYSOUT data sets.	JES2	HASPSSSM	HOSSOUT	TSO OUTPUT
			JES3	IATSIOP	IATSIOP	TSO OUTPUT
2	CS	Cancel a job.	JES2	HASPSSSM	HOSCANC	TSO CANCEL
			JES3	IATSICN	IATSICN	TSO CANCEL
3	CS	Find the status of a job.	JES2	HASPSSSM	HOSSTAT	TSO STATUS
			JES3	IATSIST	IATSIST	TSO STATUS
4	ET	Notify the subsystem of end-of-task.	Master	IEFJRASP*		SVC 87
			JES2	HASPSSSM	HOSEOT	
			JES3	IATSIJS	EOT	
5	JS	Subsystem job selection. (Provides a job that has a complete SWA.)	Master	IEFJJOBS		IEFSD161
			JES2	HASPSSSM	HOSJBSL	IEFSD161
			JES3	IATSIJS	IATSIJS	IEFSD161
6	AL	Allocation of SYSIN/SYSOUT data sets (and internal readers.)	Master	IEFJDSNA		Allocation
			JES2	HASPSSSM	HOSALLOC	Allocation
			JES3	IATSIDM	IATSIDMA	Allocation
7	AL	Unallocation of SYSIN/SYSOUT data sets (and internal readers.)	Master	IEFJDSNA		Unallocation
			JES2	HASPSSSM	HOSUNAL	Unallocation
			JES3	IATSIDM	IATSIDMU	Unallocation
8	EN	Notify subsystem of end-of-address space.	Master	IEFJRASP*		Subsystem interface resource manager
			JES2	HASPSSSM	HOSEOM	
			JES3	IATSIJS	EOM	
9	WT	Notify subsystem of a WTO message.	Master	IEFJRASP*		SVC 35
			JES2	HASPSSSM	HOSWTO	
			JES3	IATSIWO	IATSIWO	
10	CM	Notify subsystems of an operator command.	Master	IEFJRASP*		SVC 34
			JES2	HASPSSSM	HOSCMND	
			JES3	IATSI34	IATSI34	
11	US	Request subsystem to validate a remote destination userid.	JES2	HASPSSSM	HOSUSER	TSO LOGON, Unallocation
			JES3	IATSIVL	IATSIVL	TSO LOGON, Unallocation
12	JT	Notify the subsystem of job termination.	Master	IEFJJTRM		IEFSD166
			JES2	HASPSSSM	HOSTERM	IEFSD166
			JES3	IATSIJS	JOBTERM	IEFSD166

* IEFJRASP broadcasts the indicated request to all active subsystems.
Each active subsystem then performs the requested function.

Figure 2-15. Subsystem Interface Summary (Part 1 of 3)

Function Code	SSOB Extension ID	Subsystem Function	Subsystem	Module Name	Module Label	Primary Caller
13	RQ	Request subsystem to re-enqueue a job.	JES2	HASPSSSM	HOSRENQ	IEFSD166
			JES3	IATSIJS	JOBREQ	IEFSD166
14	DM	Notify all subsystems of a delete operator message (DOM)	Master	IEFJRASP*		Subsystem interface resource manager
			JES3	IATSIDO	IATSIDO	
15	VS	Request master subsystem to verify a subsystem name.	Master	IEFJSDTN		STC
16	DA	Open a subsystem data set.	JES2	HASPSSSM	HOSOPEN	OPEN
			JES3	IATSIDM	IATSIDMO	OPEN
17	DA	Close a subsystem data set.	JES2	HASPSSSM	HOSCLOS	CLOSE
			JES3	IATSIDM	IATSIDMC	CLOSE
18	DA	Checkpoint a subsystem data set.	JES2	HASPSSSM	HOSCKPT	Checkpoint
			JES3	IATSIDM	IATSIDMK	Checkpoint
19	DA	Restart a subsystem data set.	JES2	HASPSSSM	HOSREST	Restart
			JES3	IATSIDM	IATSIDMR	Restart
20	RR	Request job id.	JES2	HASPSSSM	HOSREQID	System Log
			JES3	IATSIJS	REQJBID	System Log
21	RR	Return job id.	JES2	HASPSSSM	HOSRETID	System Log
			JES3	IATSIJS	RETJBID	System Log
22	SI	Notify subsystem of step initiation.	JES3	IATSIBS	IATSIBS	IEFSD162
23	DY	Dynamic allocation.	JES3	IATSICA	IATSIDA	Dynamic allocation
24	CA	Common allocation.	JES3	IATSICA	IATSICA	Allocation
25	CU	Common unallocation.	JES3	IATSICA	IATSICU	Unallocation
26	DD	Change DDNAME.	JES3	IATSICA	IATSIDD	Allocation
27	NQ	Change ENQ use attribute.	JES3	IATSICA	IATSIDQ	Allocation
28	DR	DDR device candidate selection.	JES3	IATSIDR	IATSIRC	DDR
29	DR	DDR device candidate verification.	JES3	IATSIDR	IATSIRV	DDR
30	DR	DDR UCB swap notification.	JES3	IATSIDR	IATSIRS	DDR

*IEFJRASP broadcasts the indicated request to all active subsystems.
Each active subsystem then performs the requested function.

Figure 2-15. Subsystem Interface Summary (Part 2 of 3)

Function Code	SSOB Extension ID	Subsystem Function	Subsystem	Module Name	Module Label	Primary Caller
31	DR	DDR swap completion.	JES3	IATSIDR	IATSIRE	DDR
32	CF	Failing START command.	Master JES3	IEFJRASP* IATSISF	 IATSISF	 SVC 34
33	WT	Notify subsystem of console switch.**	JES3	IATSIWO	IATSIWO	IEAVSWCH
34	WT	Notify subsystem of WTL message.**	JES3	IATSIWO	IATSIWO	SVC 36

*IEFJRASP broadcasts the indicated request to all active subsystems. Each active subsystem then performs the requested function.

**Functions 14, 33, and 34 are not supported by JES2.

Figure 2-15. Subsystem Interface Summary (Part 3 of 3)

Diagram 8-1. Subsystem Interface (IEFJSREQ) (Part 1 of 4)

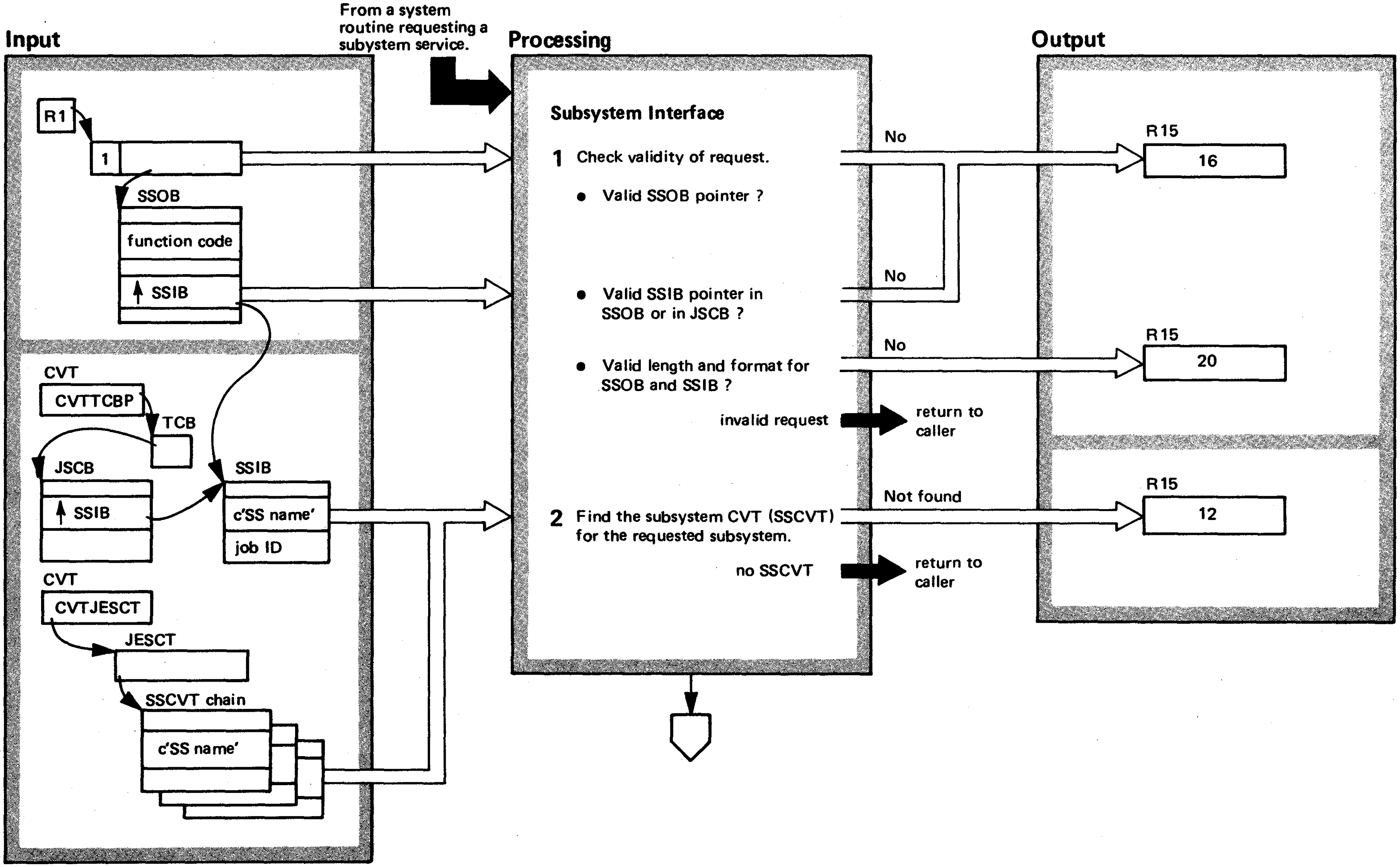


Diagram 8-1. Subsystem Interface (IEFJSREQ) (Part 2 of 4)

Extended Description	Module	Label
The subsystem interface handles requests for services to be performed by a job entry subsystem or the master subsystem. When a system routine issues the macro instruction IEFSSREQ, the subsystem interface gets control. It determines which subsystem is requested and which function routine in that subsystem is to be executed. The initialization of the subsystem interface is described in *OS/VS2 System Initialization Logic,* SY28-0623.	IEFJSREQ	
1 The requestor creates an SSIB and SSOB before invoking the subsystem interface: the SSIB identifies the subsystem requested, and the SSOB identifies the subsystem function routine that is to be executed. The subsystem interface ensures that the requestor made no errors in its request. If the SSOB has a zero SSIB pointer, the subsystem interface uses the SSIB pointer in the current JSCB.	IEFJSREQ	
2 There is one SSCVT for each subsystem defined at system generation time. The four-character subsystem name in each SSCVT is compared to the subsystem name in the SSIB. If a match is found, the subsystem name in the SSIB is valid.	IEFJSREQ	

Diagram 8-1. Subsystem Interface (IEFJSREQ) (Part 3 of 4)

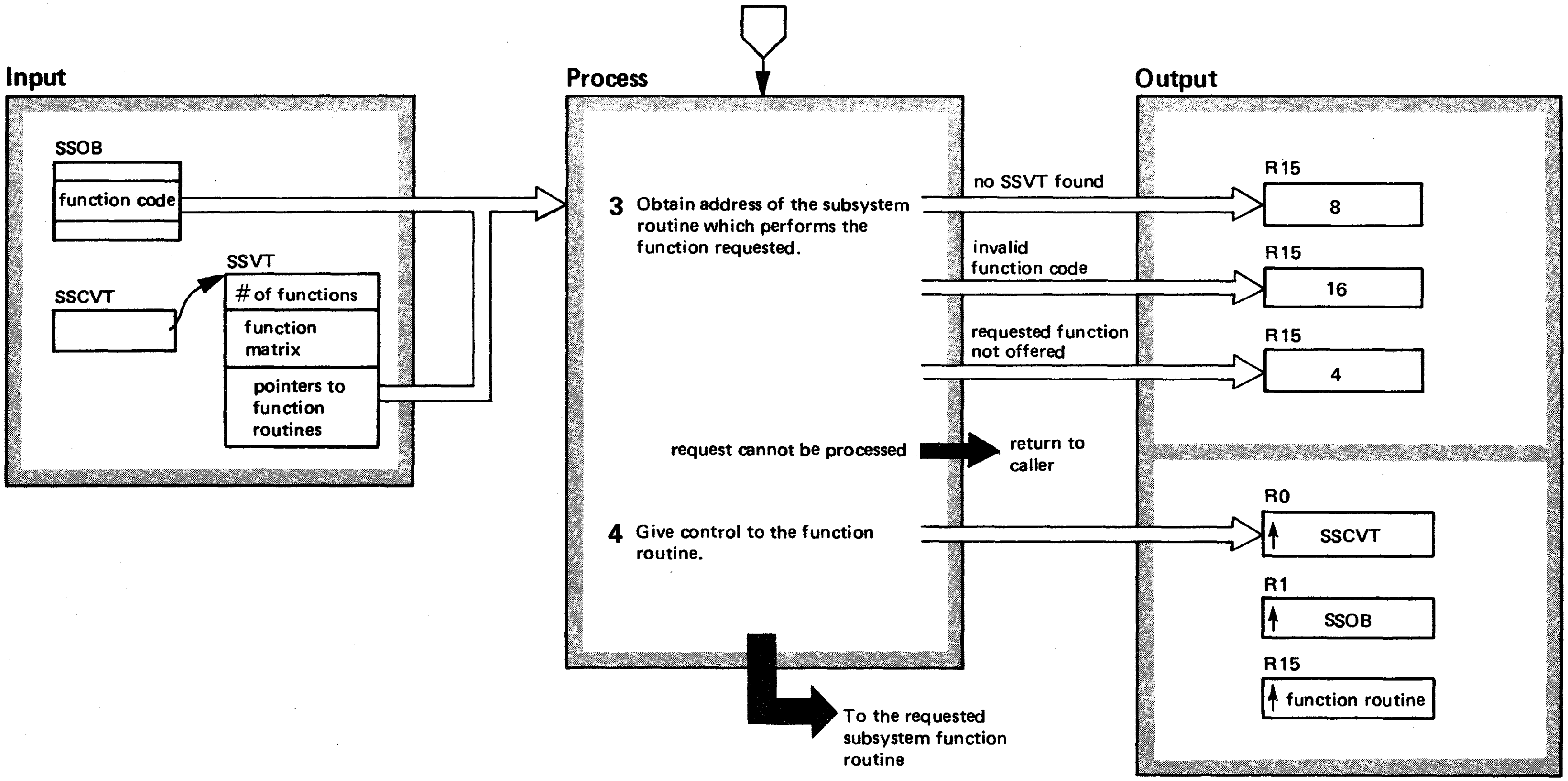


Diagram 8-1. Subsystem Interface (IEFJSREQ) (Part 4 of 4)

Extended Description	Module	Label
3 If the SSVT pointer in the SSCVT (the SSCVT located during Step 2 above) is zero, the subsystem has not been initialized yet and therefore is inactive. If the SSVT exists, it is used to find the address of the subsystem function routine.	IEFJSREQ	
In the SSOB is the function code, a number between 1 and 256, which refers to a single byte in the SSVT's function matrix. The number 1 refers to the 1st byte in the matrix, 2 refers to the 2nd byte, and so on. The matrix byte contains a value that is an index into the list of entry point addresses for the subsystem function routines. A value of 1 refers to the 1st address, a value of two refers to the 2nd address, and so on. A value of 0 in the matrix byte indicates that the function is not supported by this subsystem.		
4 Finally, the subsystem interface passes control to the subsystem routine at the address obtained in Step 3 above. When the subsystem routine completes its processing, it returns directly to the system routine that requested the service.	IEFJSREQ	

The master subsystem is a collection of routines that perform functions required to initiate certain system tasks. Job scheduling normally initiates a task or a user job using the services of a job entry subsystem to obtain and interpret the job's JCL. But, certain system tasks are initiated when a job entry subsystem is not available. These tasks include the master scheduler, which is the first initiated task in the system, and job entry subsystems. In fact, any subsystem defined as such at SYSGEN time is initiated via the master subsystem rather than via a job entry subsystem.

The converter and interpreter, when invoked by the master subsystem to interpret the JCL for a task, do not use the normal access method (VSAM) to read and write the JCLS and internal text chains; rather, they use the pseudo access method. The pseudo access method manipulates data located in real storage, whereas VSAM manipulates data located in external storage. Since the pseudo access method uses the standard RPL/ACB interface, the converter and interpreter can use the pseudo access method as if it were VSAM.

The master subsystem performs additional functions related to initiating subsystems: subsystem determination, common request routing, data set name assignment, and subsystem termination. These functions, as well as subsystem initiation itself, are invoked via the subsystem interface.

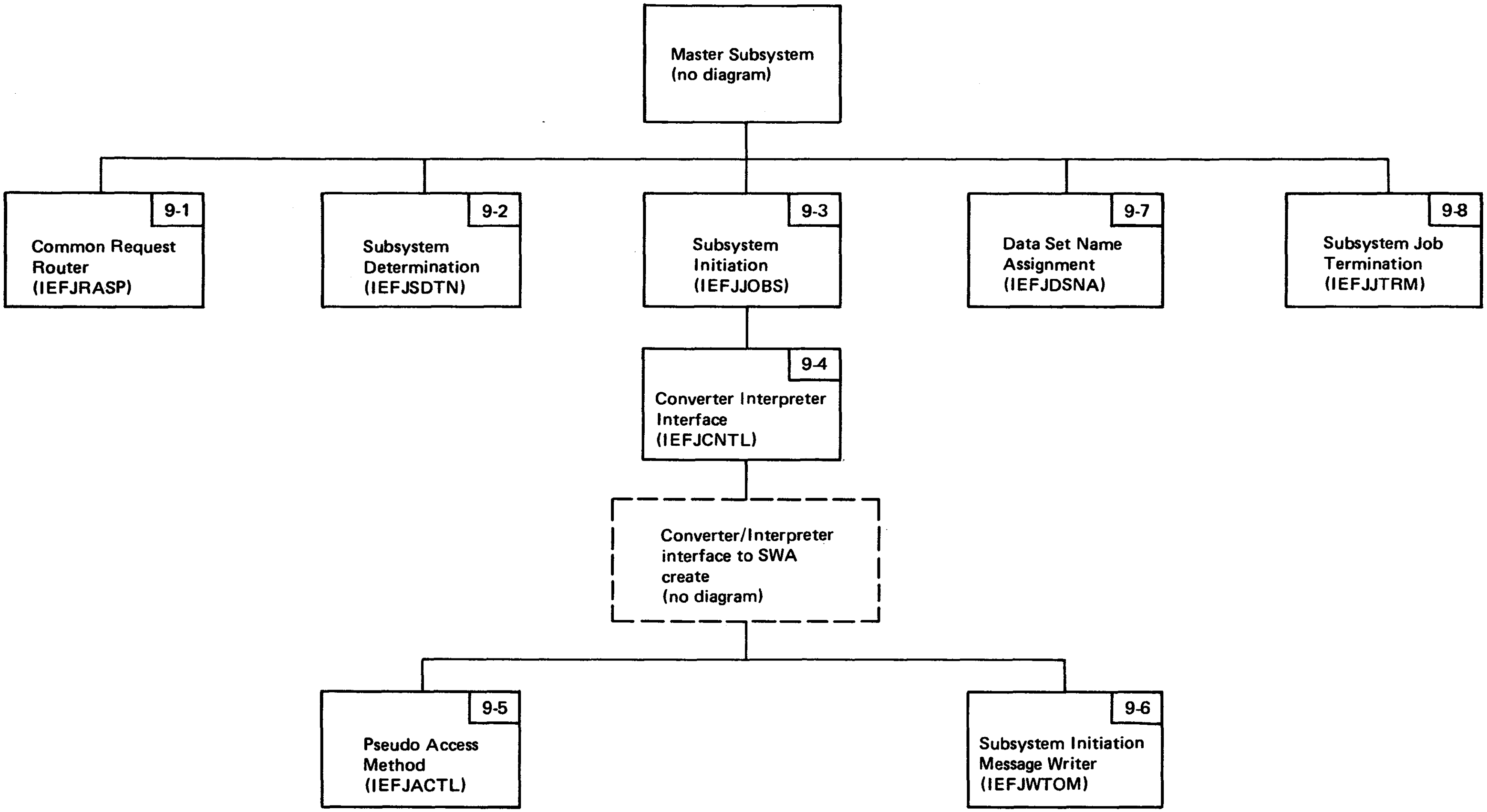
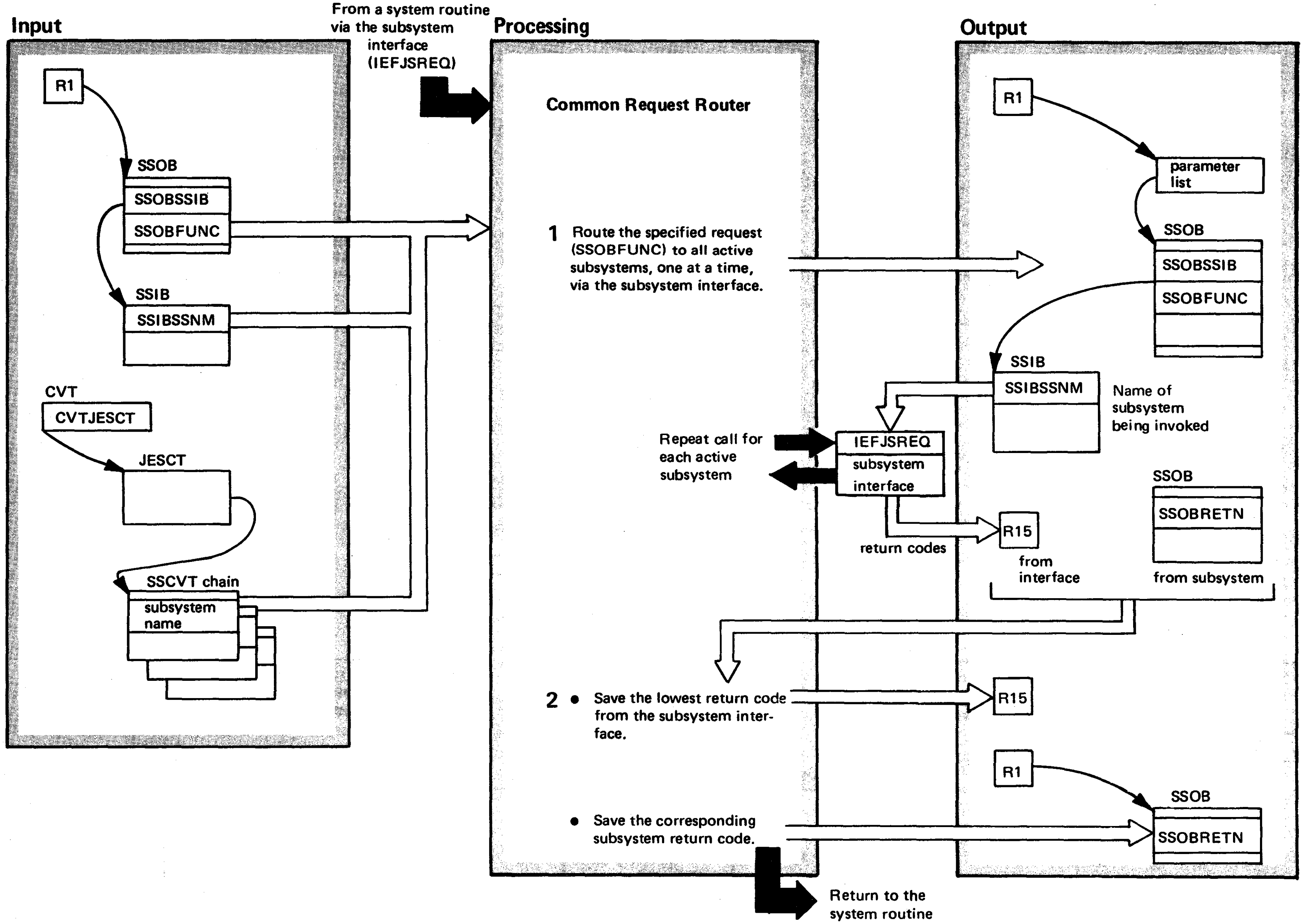


Figure 2-16. Master Subsystem Visual Contents

Diagram 9-1. Common Request Router (IEFJRASP) (Part 1 of 2)



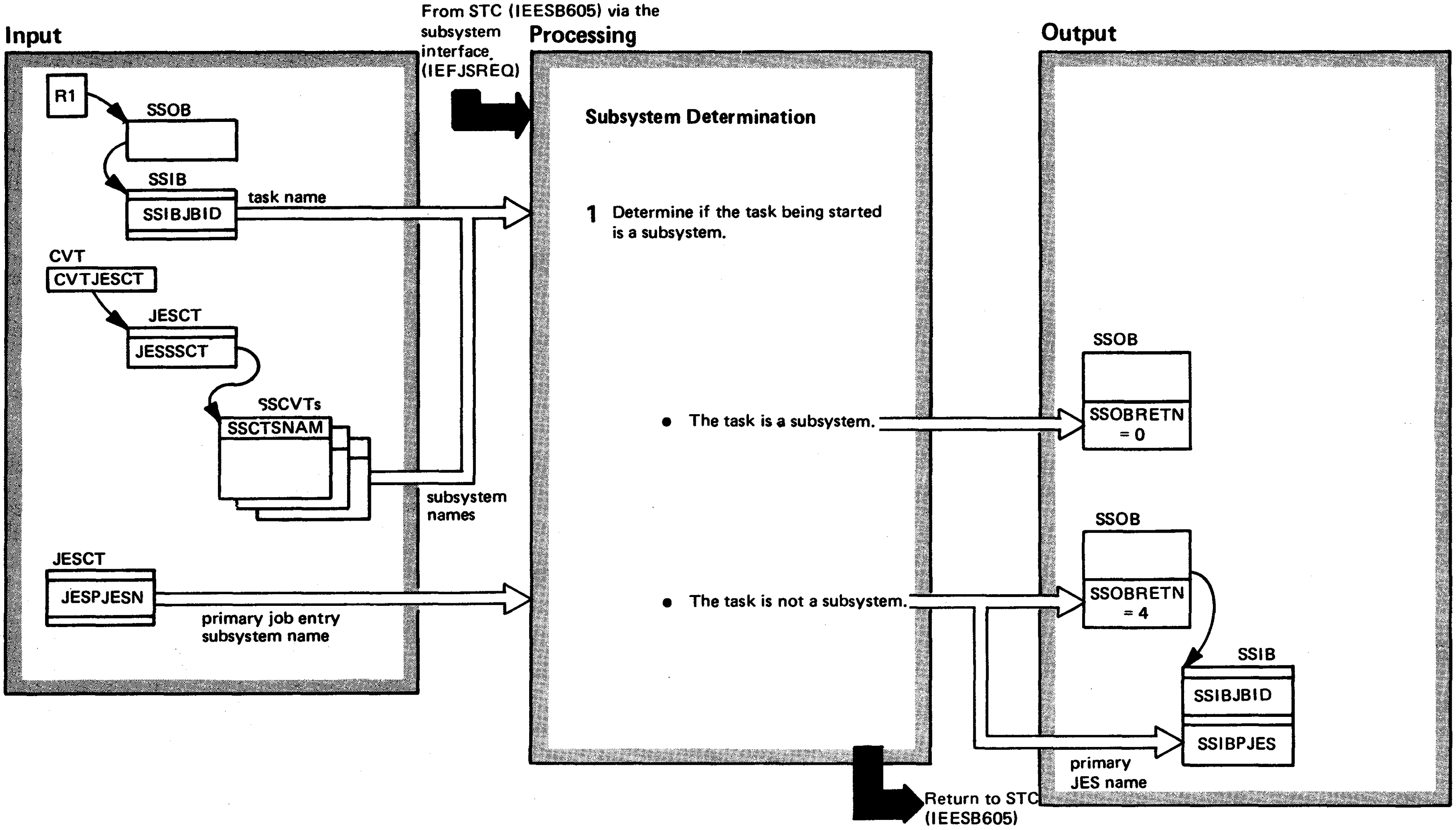
1 Route the specified request (SSOBFUNC) to all active subsystems, one at a time, via the subsystem interface.

2
- Save the lowest return code from the subsystem interface.
- Save the corresponding subsystem return code.

Diagram 9-1. Common Request Router (IEFJRASP) (Part 2 of 2)

Extended Description	Module	Label
The common request router, a function of the master subsystem, routes a single request to all the active subsystems except the master subsystem. This request may be for command processing, for notification of address space or task termination, for WTOs, and for DOMs.	IEFJRASP	
1 The common request router obtains the numerical code of the requested function from the SSOB and notifies each active subsystem to perform the requested function. To accomplish this, the router first places the name of an active subsystem in the SSIB and the function code in the SSOB. Then, the router invokes the subsystem interface which passes control to the routine that performs the function. The router invokes the interface once for each active subsystem, changing the subsystem name in the SSIB each time.	IEFJRASP	
2 Following each invocation of the subsystem interface, the router analyzes the return codes from the subsystem interface and from the subsystem. The router saves the lowest code returned by the interface. It also saves the highest subsystem return code that was passed back with the lowest interface code.	IEFJRASP	

Diagram 9-2. Subsystem Determination (IEFJSDTN) (Part 1 of 2)



Subsystem Determination

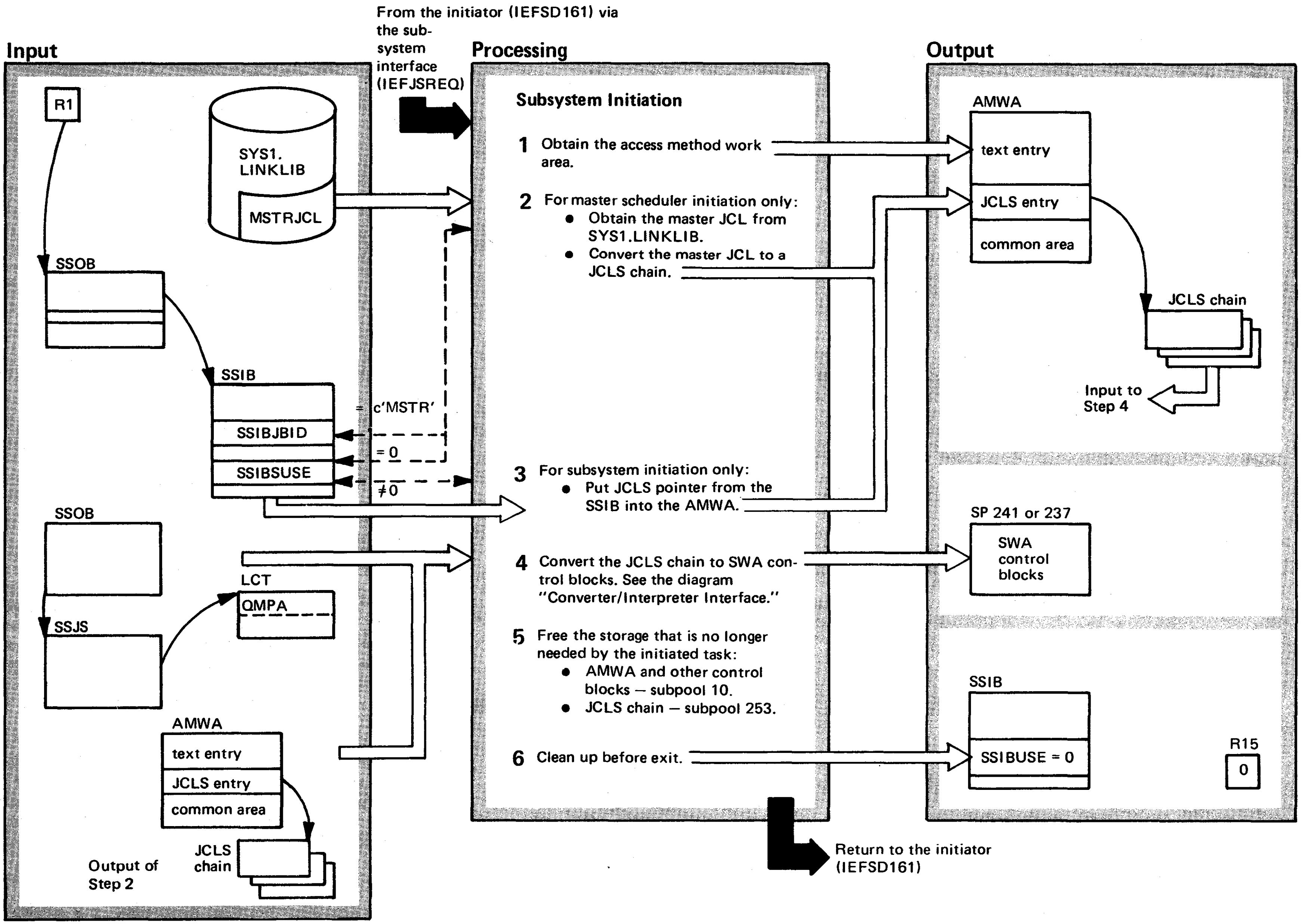
1 Determine if the task being started is a subsystem.

- The task is a subsystem.
- The task is not a subsystem.

Diagram 9-2. Subsystem Determination (IEFJSDTN) (Part 2 of 2)

Extended Description	Module	Label
The master subsystem provides a subsystem determination function. Subsystem determination is used by the initiator during the processing of a START command to determine if a subsystem is being started. A subsystem must be started using the master subsystem, whereas other tasks are started using the primary job entry subsystem.	IEFJSDTN	
1 Subsystem determination compares the task name in the job ID field of the SSIB to the subsystem name in each of the subsystem CVTs. If a match is found, the task being started is a subsystem. In this case, the name of the master subsystem is left in the SSIB. If no match is found, the task is not a subsystem. In this case, the name of the primary job entry subsystem is placed in the SSIB.	IEFJSDTN	

Diagram 9-3. Subsystem Initiation (IEFJJOBS) (Part 1 of 2)



Subsystem Initiation

1 Obtain the access method work area.

2 For master scheduler initiation only:
- Obtain the master JCL from SYS1.LINKLIB.
- Convert the master JCL to a JCLS chain.

3 For subsystem initiation only:
- Put JCLS pointer from the SSIB into the AMWA.

4 Convert the JCLS chain to SWA control blocks. See the diagram "Converter/Interpreter Interface."

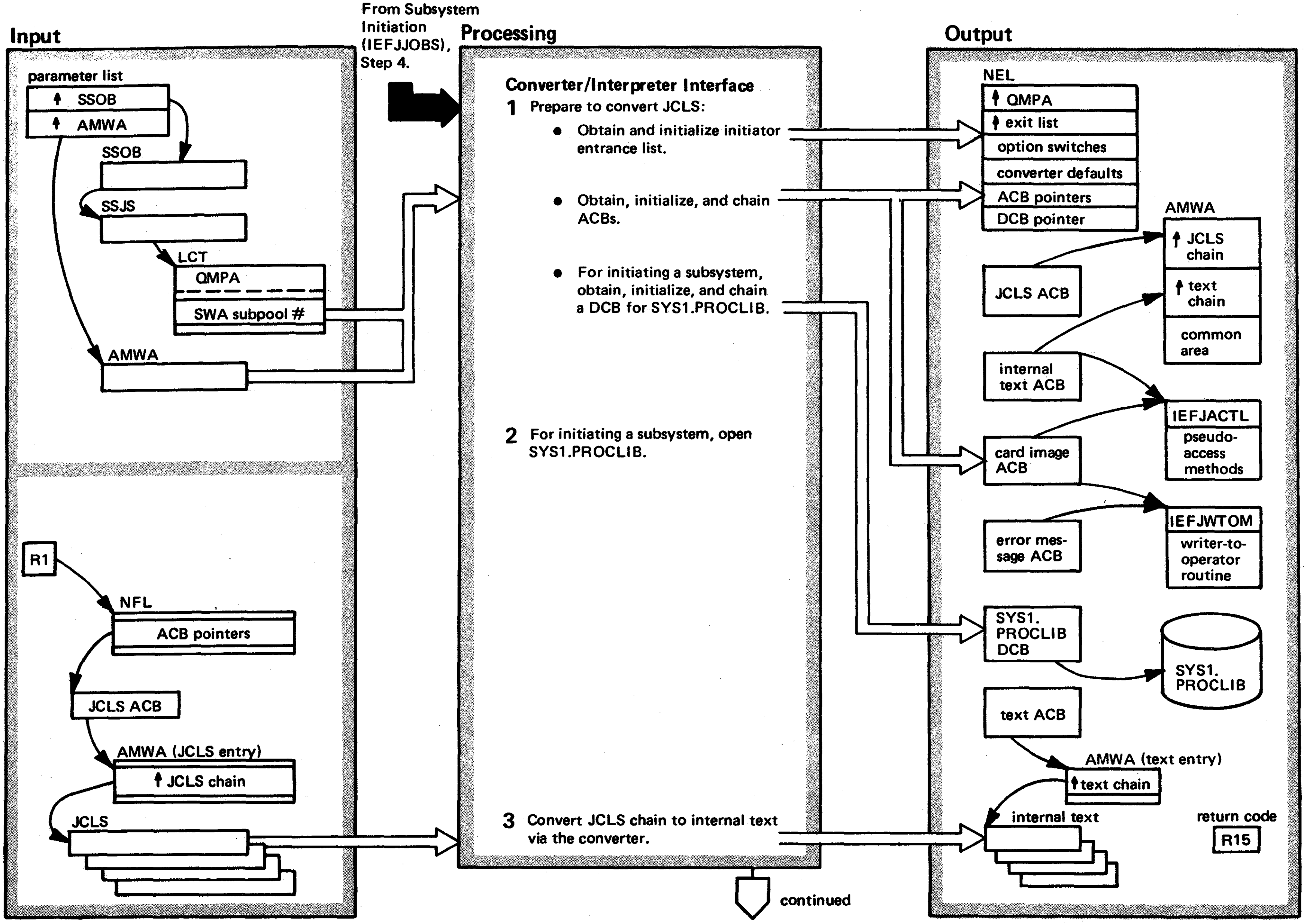
5 Free the storage that is no longer needed by the initiated task:
- AMWA and other control blocks — subpool 10.
- JCLS chain — subpool 253.

6 Clean up before exit.

Diagram 9-3. Subsystem Initiation (IEFJJOBS) (Part 2 of 2)

Extended Description	Module	Label
The initiator issues IEFSSREQ specifying "job select" to invoke subsystem initiation, a function of the master subsystem, to obtain and interpret the JCL for tasks that cannot use the services of a job entry subsystem. These tasks include the master scheduler, which is the first initiated task in the system, the job entry subsystems themselves, and any other subsystems defined at SYSGEN time.	IEFJJOBS	
Subsystem initiation obtains the JCL that defines the resources needed by the master scheduler or by a subsystem and invokes the converter and interpreter to create SWA (scheduler work area) control blocks from that JCL.		
1 The access method work area (AMWA) contains information about the JCLS (JCL set) and internal test chains. AMWA contains information for use by the pseudo access method when it reads and writes these chains of records.	IEFJJOBS	
2 If the address of the JCLS (a set of chained JCL records) in the SSIB is zero, subsystem initiation assumes that the master scheduler is being started and obtains the JCL card images from the MSTRJCL member of SYS1.LINKLIB. A listing of this JCL appears under the topic "Master Scheduler Initialization" in *OS/VS2 System Initialization Logic,* SY28-0623.	IEFJJOBS	
The JCL to JCLS chain conversion routine first checks that each 80-byte JCL card image begins with // or /*. If an error is found, a return code of four is passed back to the caller. One at a time, the card images are stored in 88-byte areas of subpool 253. The first 8 bytes of each area comprise a chaining field and a reserved field.	IEFJJCLS	
Subsystem initiation places the address of the first chained JCLS record into the JCLS entry of the AMWA.	IEFJJOBS	
3 A non-zero JCLS pointer in the SSIB indicates that a subsystem is being started. In this case, subsystem initiation moves the JCLS pointer to the JCLS entry in the AMWA, skipping the JCL to JCLS chain conversion.	IEFJJOBS	
4 The JCLS-chain-to-SWA conversion routine invokes first the converter, then the SWA-create routine. The converter converts the JCLS chain to internal text; the SWA-create routine invokes the interpreter to create SWA control blocks using the internal text as input. For more detail, refer to the diagram, Converter/Interpreter Interface.	IEFJCNTL	
5 The storage deletion routine frees the storage in subpool 253 used by the JCLS chain and frees all control blocks residing in subpool 10. (The control blocks in subpool 10 were obtained in step 4 before the converter was invoked.) The only ACB remaining is the one for error messages located in the SWA subpool (subpool 241 for the master scheduler, subpool 237 for a subsystem). Allocation uses this ACB when issuing its error messages.	IEFJCDLT	
6 The final step of subsystem initiation sets both the JCLS pointer (in the SSIB) and register 15 to zero. Return codes passed back to the initiator indicate whether or not SWA control blocks were created.	IEFJJOBS	
Error Processing		
All ABENDs issued by master subsystem routines cause the caller's ESTAE routine to get control (the caller being the system routine that invoked the master subsystem via the subsystem interface).		
Subsystem initiation issues a 0B1 user ABEND if the initiator passes it either of the following two invalid addresses: • A zero SSOB address passed in register 1. • A zero address for the JCLS chain when a subsystem is being started.	IEFJJOBS	
If subsystem initiation passes a zero SSOB or AMWA pointer to the converter/interpreter interface, the interface issues a 0B1 ABEND. The interface issues a 0B4 ABEND if SYS1.PROCLIB was not opened successfully or if the block size contained in the PROCLIB DCB is not a multiple of 80. If the attach of the converter is unsuccessful, the interface issues a 0B5 ABEND.	IEFJCNTL	

Diagram 9-4. Converter/Interpreter Interface (IEFJCNTL) (Part 1 of 4)



From Subsystem Initiation (IEFJJOBS), Step 4.

Converter/Interpreter Interface

1 Prepare to convert JCLS:

- Obtain and initialize initiator entrance list.
- Obtain, initialize, and chain ACBs.
- For initiating a subsystem, obtain, initialize, and chain a DCB for SYS1.PROCLIB.

2 For initiating a subsystem, open SYS1.PROCLIB.

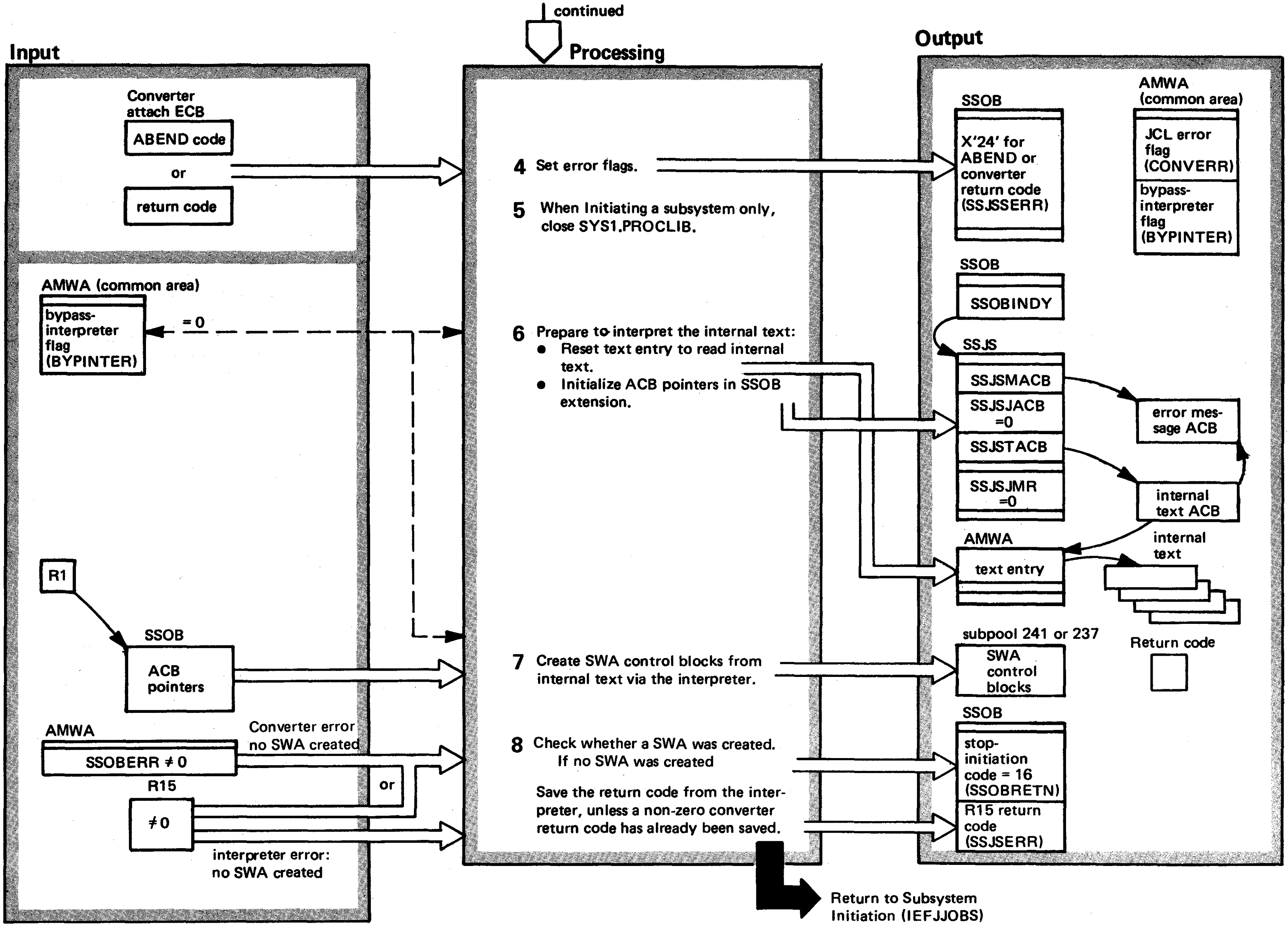
3 Convert JCLS chain to internal text via the converter.

continued

Diagram 9-4. Converter/Interpreter Interface (IEFJCNTL) (Part 2 of 4)

Extended Description	Module	Label
The converter/interpreter interface, a master subsystem routine, controls the conversion of a JCLS chain to SWA control blocks. The JCLS chain, passed from subsystem initiation (IEFJJOBS), defines the resources needed by the started task (that is, by the master scheduler, a job entry subsystem, or any other defined subsystem).	IEFJCNTL	
1 The converter/interpreter interface creates the environment for the converter to operate. It builds the NEL as an interface with the converter. It also builds the ACBs to allow the converter to interface with the pseudo access method as if it were the normal access method (VSAM). Refer to the diagram, Pseudo Access Method.	IEFJCNTL	
2 When starting a subsystem, started task control (STC) builds a JCLS chain which the initiator passes to the master subsystem. This JCLS defines a step that executes a JCL procedure located in SYS1.PROCLIB. The converter/interpreter interface opens SYS1.PROCLIB so that the converter can obtain the procedure and convert it to internal text.	IEFJCNTL	
3 The address of the NEL (interpreter entrance list) is passed to the converter which then proceeds to convert the JCLS chain to internal text. The converter uses the pseudo access method to read the JCLS chain record-by-record and then to write a chain of internal text. The subsystem initiation message writer handles the error messages normally issued by the converter/interpreter. The messages are sent to hardcopy according to the MSGLEVEL specification in the JCL. All the error messages, and card images, in addition to having their usual message ID, will be prefixed by the master subsystem message ID (c'IEF196I'). Refer to the diagram, Subsystem Initiation Message Writer.	IEFVH1	

Diagram 9-4. Converter/Interpreter Interface (IEFJCNTL) (Part 3 of 4)



4 Set error flags.

5 When Initiating a subsystem only, close SYS1.PROCLIB.

6 Prepare to interpret the internal text:
- Reset text entry to read internal text.
- Initialize ACB pointers in SSOB extension.

7 Create SWA control blocks from internal text via the interpreter.

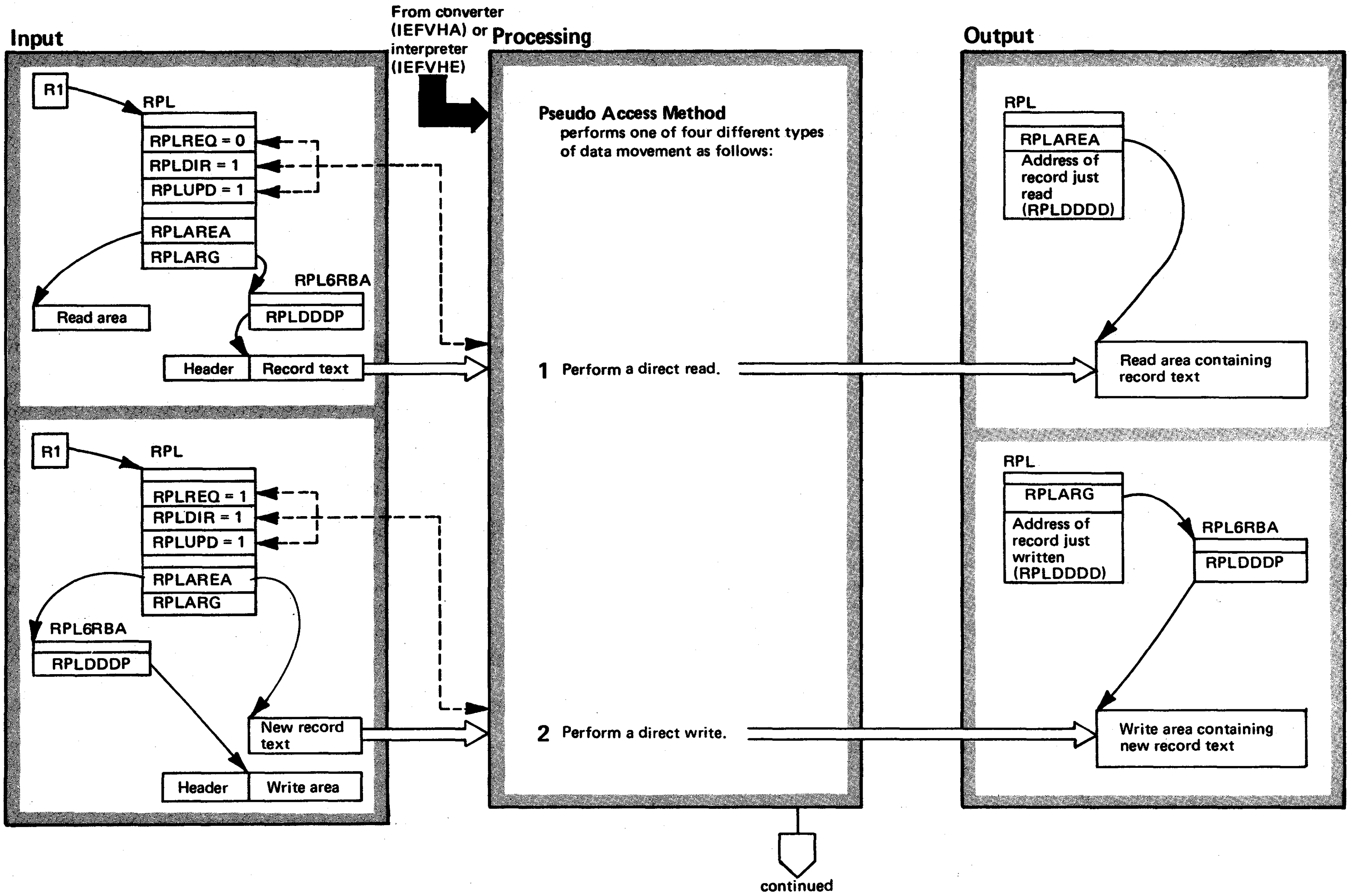
8 Check whether a SWA was created.
If no SWA was created

Save the return code from the interpreter, unless a non-zero converter return code has already been saved.

Diagram 9-4. Converter/Interpreter Interface (IEFJCNTL) (Part 4 of 4)

Extended Description	Module	Label
4 Depending on the success of the converter, flags are set that affect subsequent processing. If the converter abnormally terminated or passed back a return code greater than four, the bypass-interpreter flag is turned on. In this case, the interpreter is not invoked. The converter return code is placed in the SSJSSERR field as a preliminary indication that no SWA control blocks were created.	IEFJCNTL	
5 SYS1.PROCLIB is closed since it is no longer needed by the converter.	IEFJCNTL	
6 The SSJS extension of the SSOB is initialized with the addresses of the ACBs required by the interpreter. The JMR field is set to zero because SMF records are not being collected; the journal ACB address is set to zero because journal records for checkpoint/restart are not being kept.	IEFJCNTL	
7 The converter/interpreter interface passes control to the SWA-create interface (IEFIB600), passing it the address of an SSOB in register one. The SWA-create interface invokes the interpreter to create SWA control blocks from the internal text. (The SWA is located in subpool 241 for the master scheduler and subpool 237 for a subsystem.) The interpreter uses the pseudo access method to read the internal text and, like the converter in Step 3, uses the message writer to issue its error messages. (Refer to the diagrams, Pseudo Access Method and Subsystem Initiation Message Writer.)	IEFIB600 IEFNB903	
8 If a SWA was not created because a converter error or an interpreter error occurred, the SSOBRETN field is set to indicate that the initiation of this task is to be ended. If there was an interpreter error but no converter error, the contents of register 15 are placed in the return code field of the SSOB. If a converter error occurred previously, the SSOBERR field is not changed thus preserving the converter return code.	IEFJCNTL	
Error Processing		
If subsystem initiation passes a zero SSOB or AMWA pointer to the converter/interpreter interface, the interface issues a 0B1 ABEND. The interface issues a 0B4 ABEND if SYS1.PROCLIB was not opened successfully or if the block size contained in the PROCLIB DCB is not a multiple of 80. If the attach of the converter is unsuccessful, the interface issues a 0B5 ABEND.	IEFJCNTL	

Diagram 9-5. Pseudo Access Method (IEFJACTL) (Part 1 of 4)



Pseudo Access Method performs one of four different types of data movement as follows:

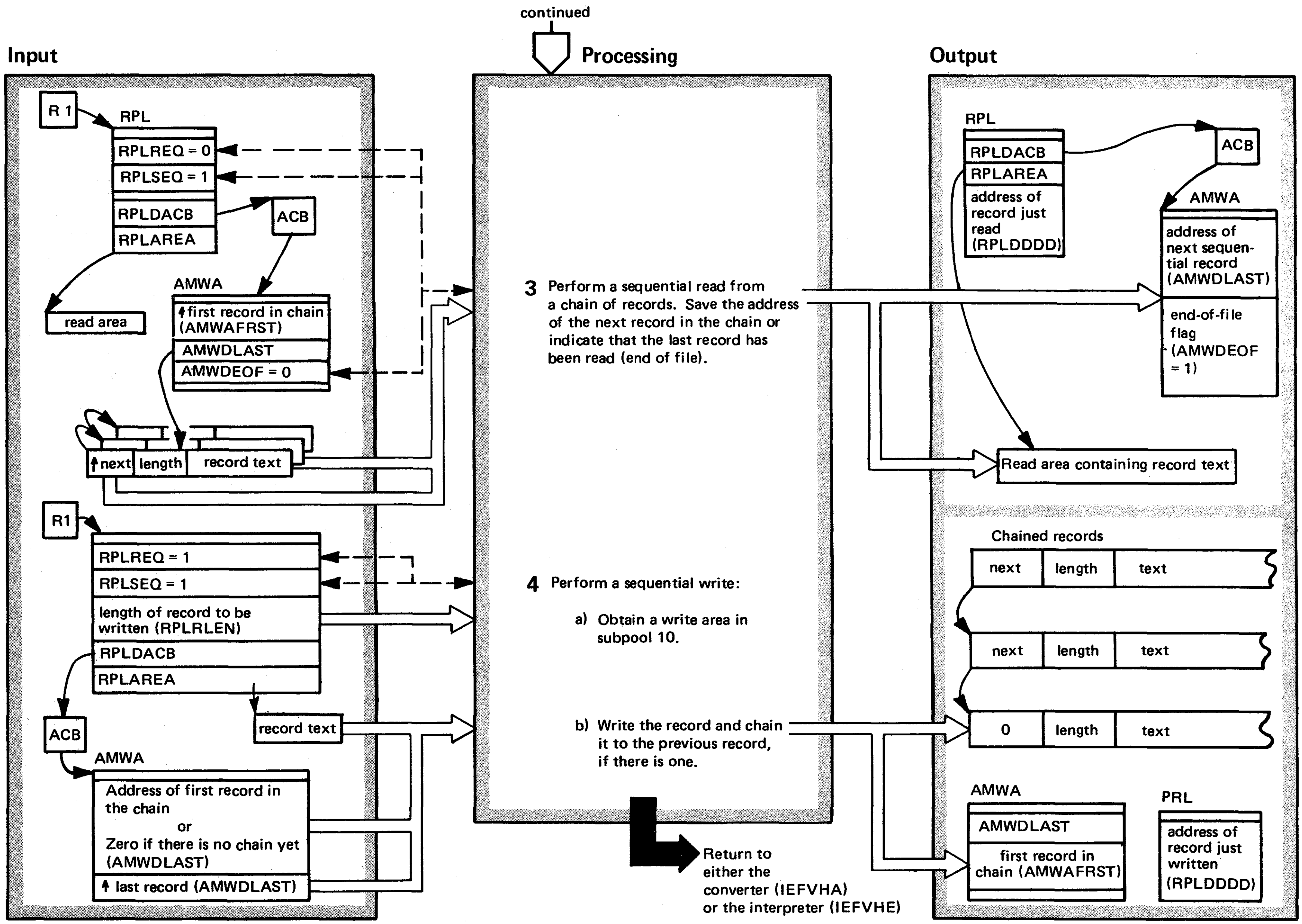
1 Perform a direct read.

2 Perform a direct write.

Diagram 9-5. Pseudo Access Method (IEFJACTL) (Part 2 of 4)

Extended Description	Module	Label
The pseudo access method provides the master subsystem with a data manipulation service at a time when no access methods services are available via the RPL/ACB interface. Rather than accessing data that resides on an external storage device, the pseudo access method manipulates data located in real storage. The converter and interpreter use the pseudo access method when a task is being started via the master subsystem and not by a job entry subsystem.	IEFJACTL	
The subsystem initiation function of the master subsystem sets up the standard RPL/ACB interface for the converter and interpreter but places the address of the appropriate pseudo access method routine in the ACBs instead of the address of the VSAM routines. The switch is not detectable to the converter and interpreter.	IEFJCNTL	
Pseudo access method control determines which of four types of data movement is being requested by checking flags in the RPL and AMWA. Each of the four steps in the diagram indicate the flag settings required for its particular processing.	IEFJACTL	
1 The converter uses the direct read to obtain a particular internal text record for updating. First, the control routine must determine that a direct read is being requested by checking flags in the RPL. The read routine moves the text record to the specified area. The length of the move is specified in the header on the text record being read.	IEFJACTL IEFJDIRD	
2 After the converter has updated the internal text record obtained by a direct read operation, it writes the new record over the original record by requesting a direct write operation. First, the control routine must determine that a direct write is being requested by checking flags in the RPL. The direct write routine moves the new record text to the specified area.	IEFJACTL IEFJDWRT	

Diagram 9-5. Pseudo Access Method (IEFJACTL) (Part 3 of 4)



3 Perform a sequential read from a chain of records. Save the address of the next record in the chain or indicate that the last record has been read (end of file).

4 Perform a sequential write:

a) Obtain a write area in subpool 10.

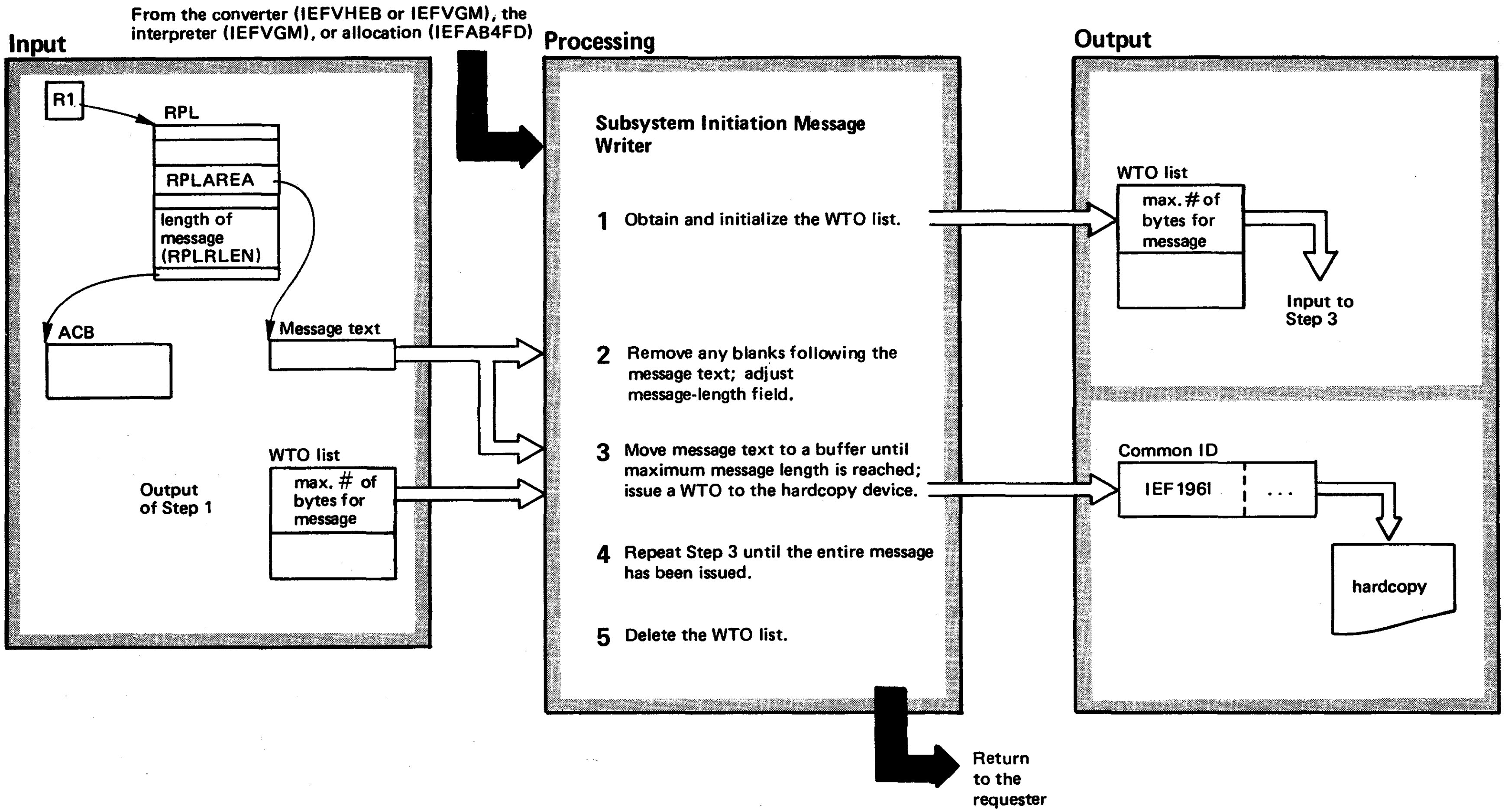
b) Write the record and chain it to the previous record, if there is one.

Return to either the converter (IEFVHA) or the interpreter (IEFVHE)

Diagram 9-5. Pseudo Access Method (IEFJACTL) (Part 4 of 4)

Extended Description	Module	Label
3 The converter and interpreter use the sequential read to read records from in-storage record chains (the JCLS chain and the internal text chain, respectively). First, the control routine must determine that a sequential read is being requested by checking flags in the RPL. If the bit AMWDEOF is on, indicating an end-of-file condition, a return code of 8 is passed back to the caller.	IEFJACTL	
The read is performed by moving a record in the chain to a specified area. The header in the record just read contains a pointer to the next record in the chain. This pointer is saved in preparation for the next sequential read. If the pointer to the next record is zero, the end-of-file flag is turned on to prevent another read operation.	IEFJREAD	
4 The converter uses the sequential write to write and chain together internal text records.		
a) First, the control routine must determine that a sequential write is being requested by checking flags in the RPL.	IEFJACTL	
b) The write is performed by first obtaining an area in subpool 10. The new record is moved to the area just obtained. If the AMWDLAST field indicates that a previous record exists in the chain, that record is chained to the newly-written record.	IEFJWRTE	

Diagram 9-6. Subsystem Initiation Message Writer (IEFJWTOM) (Part 1 of 2)



Subsystem Initiation Message Writer

1 Obtain and initialize the WTO list.

2 Remove any blanks following the message text; adjust message-length field.

3 Move message text to a buffer until maximum message length is reached; issue a WTO to the hardcopy device.

4 Repeat Step 3 until the entire message has been issued.

5 Delete the WTO list.

Diagram 9-6. Subsystem Initiation Message Writer (IEFJWTOM) (Part 2 of 2)

Extended Description	Module	Label
The converter, the interpreter, and allocation normally issue their messages to a SYSOUT data set. The subsystem initiation message writer issues these messages to hardcopy instead. This message writer is used for tasks being started via the master subsystem. These tasks include the master scheduler, job entry subsystems, and other defined subsystems.	IEFJWTOM	
1 The message writer issues the list form of the WTO macro instruction. In this way, it obtains the maximum length allowed for a hardcopy record.	IEFJWTOM	
2 The message text is scanned backwards starting at the end in order to eliminate any trailing blanks.	IEFJWTOM	
3 The writer issues a WTO macro instruction to write the message to hardcopy device. The hardcopy device is defined at system generation time. Each message, in addition to having its usual identifier, is prefixed by the common identifier IEF196I to indicate that the master subsystem issued this message on behalf of a starting task.	IEFJWTOM	
4 If the message is longer than the maximum length allowed for a single hardcopy record, the message is split, and the WTO macro instruction is issued repeatedly until the entire message text has been issued to hardcopy.	IEFJWTOM	
5 The writer deletes the WTO list area after the message is issued.	IEFJWTOM	

Diagram 9-7. Data Set Name Assignment (IEFDSNA) (Part 1 of 2)

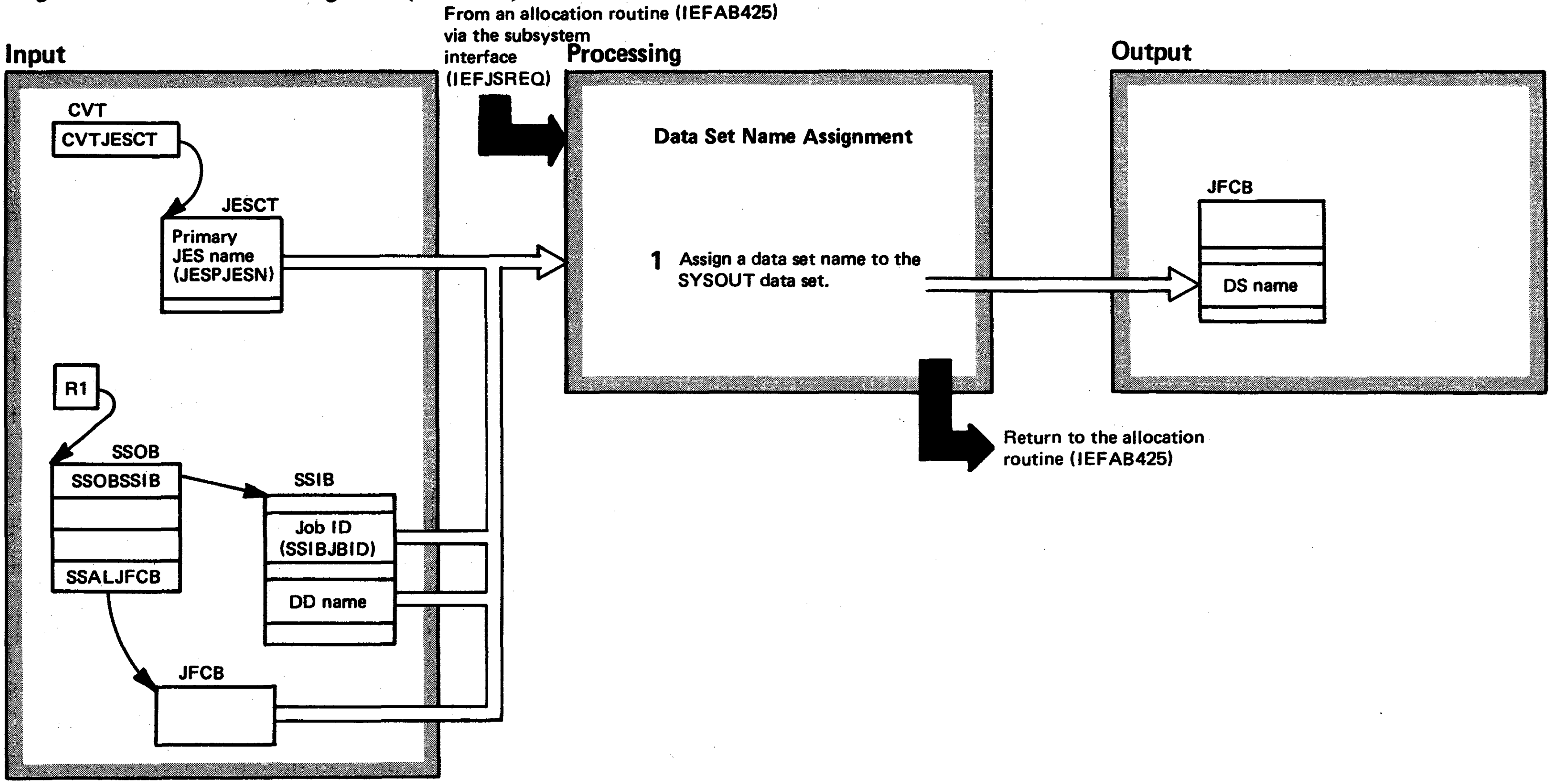
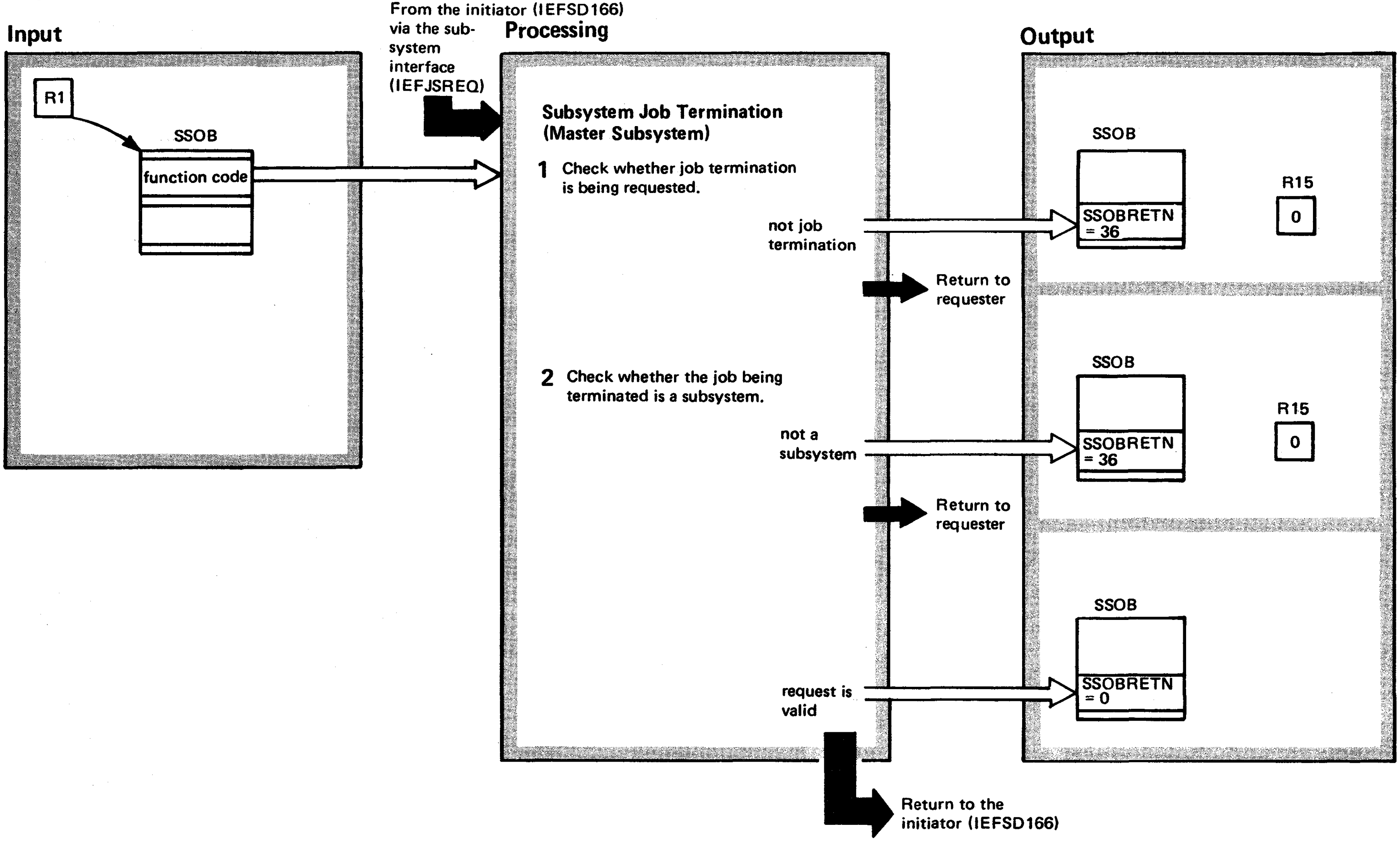


Diagram 9-7. Data Set Name Assignment (IEFDSNA) (Part 2 of 2)

Extended Description	Module	Label
The master subsystem provides a data set name assignment function. Data set name assignment assigns a data set name to each SYSOUT data set specified in the master JCL (that JCL used to start the master scheduler) and in the JCL used to start a job entry subsystem.	IEFJDSNA	
1 The data set name is constructed according to the following format: xxxx.yyyyyyyy.aabbbb.cccccccc where xxxx = primary job entry subsystem name yyyyyyyy = job ID specified in the SSIB aa = c'MS' bbbb = c'0000' cccccccc = DD name of the JCL record for the SYSOUT data set.	IEFJDSNA	

Diagram 9-8. Subsystem Job Termination (IEFJJTRM) (Part 1 of 2)



Subsystem Job Termination (Master Subsystem)

1 Check whether job termination is being requested.

2 Check whether the job being terminated is a subsystem.

Diagram 9-8. Subsystem Job Termination (IEFJJTRM) (Part 2 of 2)

Extended Description	Module	Label
The master subsystem provides a subsystem job termination function. Subsystem job termination is a dummy routine which, when the job being terminated is a subsystem, replaces the normal job termination function.	IEFJJTRM	
1 Subsystem job termination verifies that job termination was actually requested.	IEFJJTRM	
2 Subsystem job termination also verifies that the job being terminated is a subsystem.	IEFJJTRM	

Initiator/Terminator

The purpose of the initiator/terminator is to make all necessary preparations for the execution of a job step/task. A task can be defined either as a unit of work which competes for system resources and is described by a task control block, (TCB), or as a request for the execution of some code.

To prepare a task for execution, the initiator performs the following functions:

- Obtains storage for and initializes the control blocks for a task.
- Assigns special properties to a task.
- Oversees the allocation of data sets and devices for a task.
- Opens any required catalogs and libraries for a task.
- Attaches the task.

When each task has completed execution, the terminator portion of the initiator/terminator performs these functions:

- Deletes the control blocks no longer needed.
- Deletes the RACF accessor environment, if one exists.
- Oversees the freeing of data sets and devices used by the task.
- Detaches the task.

When an entire job is complete, the initiator/terminator clears or deletes the control blocks and data areas the job used and the storage space it occupied.

The initiator provides the above functions for these situations:

- Completing master scheduler initialization.
- Starting a subsystem.
- Processing a START, MOUNT or LOGON command.
- Initiating a normal job.

In the first three situations the initiator is used as a subroutine to initiate a single job. When the job is completed, the initiator subroutine returns to its caller.

In the last case, the initiator itself is a task created as a result of a command to start an initiator. This initiator can, in turn, attach a task. When that task has completed, the initiator requests another job by invoking the job entry subsystem (JES). JES then returns to the initiator either another job or an indicator to stop processing.

Important Considerations

There are two new concepts in MVS that are important to the understanding of initiator/terminator processing: the scheduler work area and SYSEVENT macro instructions.

Scheduler Work Area

In MVS, most scheduler control blocks used by the initiator reside on a pageable portion of virtual address space called the scheduler work area (SWA). The purpose of SWA is to reduce contention for job queue resources. A more thorough discussion of SWA, including a list of resident control blocks, appears in the section of this book entitled SWA Manager.

SYSEVENT Macro Instructions

An entirely new concept for MVS is a SYSEVENT macro instruction. Use of a SYSEVENT macro instruction results in an SVC that invokes the systems resources manager (SRM) routines. The purpose of SRM is to determine those address spaces that can remain in real storage at any one time and can still maintain the most effective use of system resources or meet user-specified installation objectives.

The initiator/terminator issues these SYSEVENT macro instructions:

- JOBSELCT, indicating to SRM that a job has been selected by JES for initiator processing.
- REQSWAP, indicating that a task is to become non-swappable.
- INITATT, indicating that a task has been attached.
- INITDET, indicating that a task has been detached.
- JOBTERM, indicating that a job has terminated.
- DONTSWAP, indicating that an address space is not to be swapped.

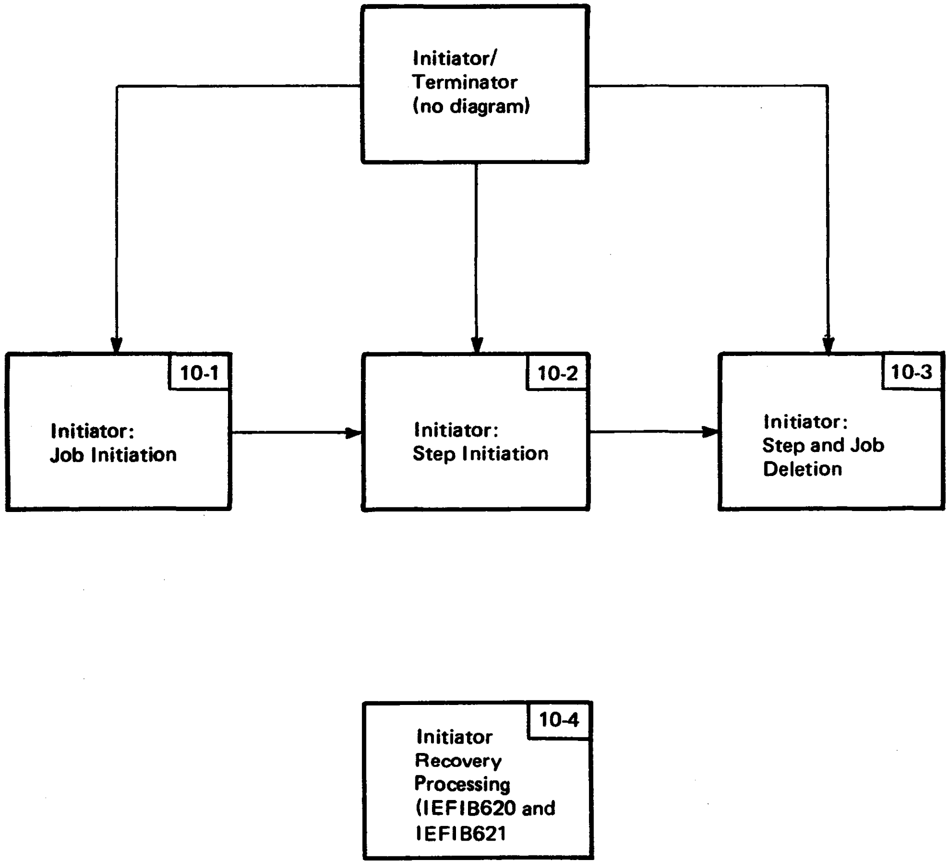
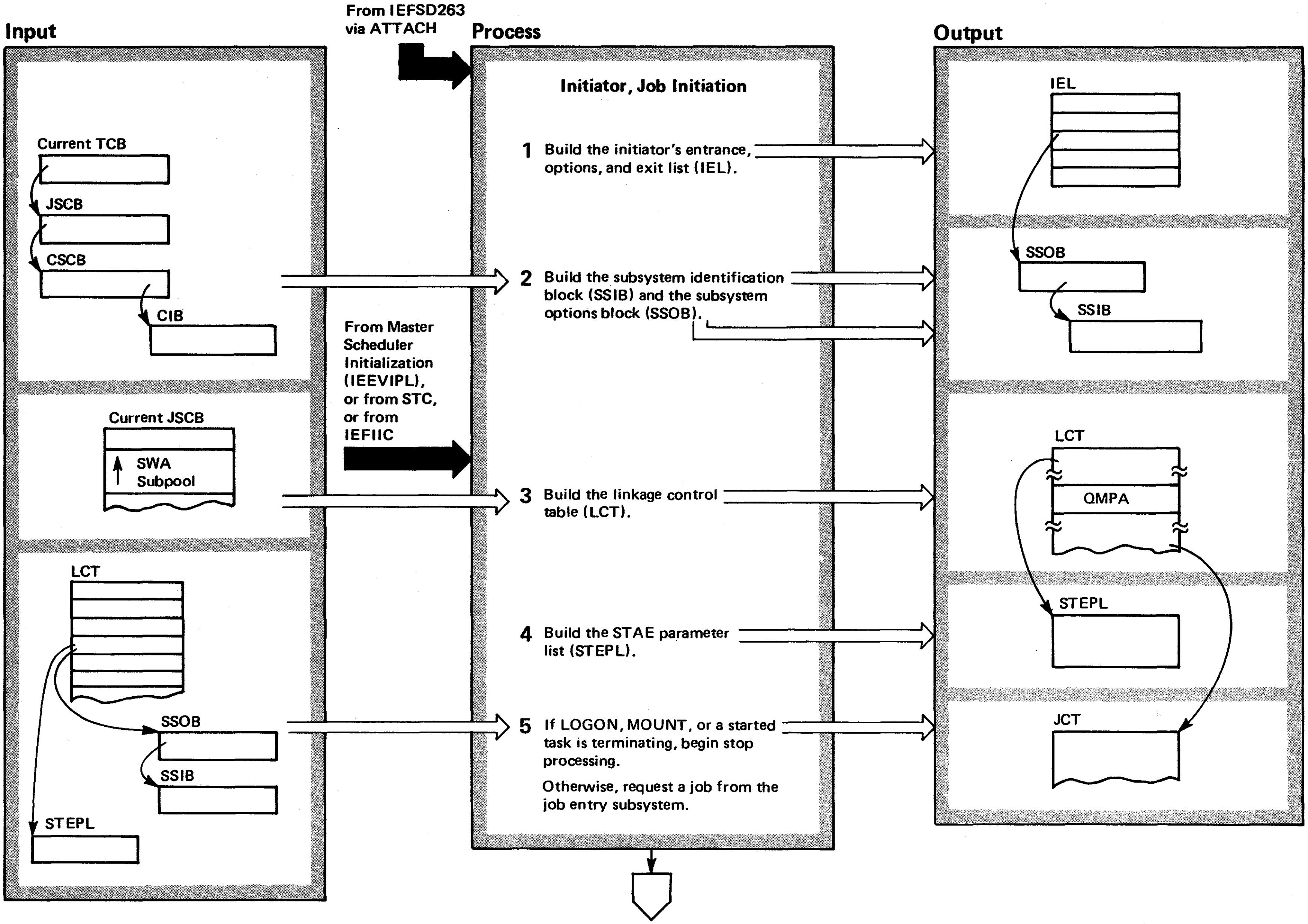


Figure 2-17. Initiator/Terminator Visual Contents

Diagram 10-1. Initiator: Job Initiation (Part 1 of 4)



1 Build the initiator's entrance, options, and exit list (IEL).

2 Build the subsystem identification block (SSIB) and the subsystem options block (SSOB).

3 Build the linkage control table (LCT).

4 Build the STAE parameter list (STEPL).

5 If LOGON, MOUNT, or a started task is terminating, begin stop processing.

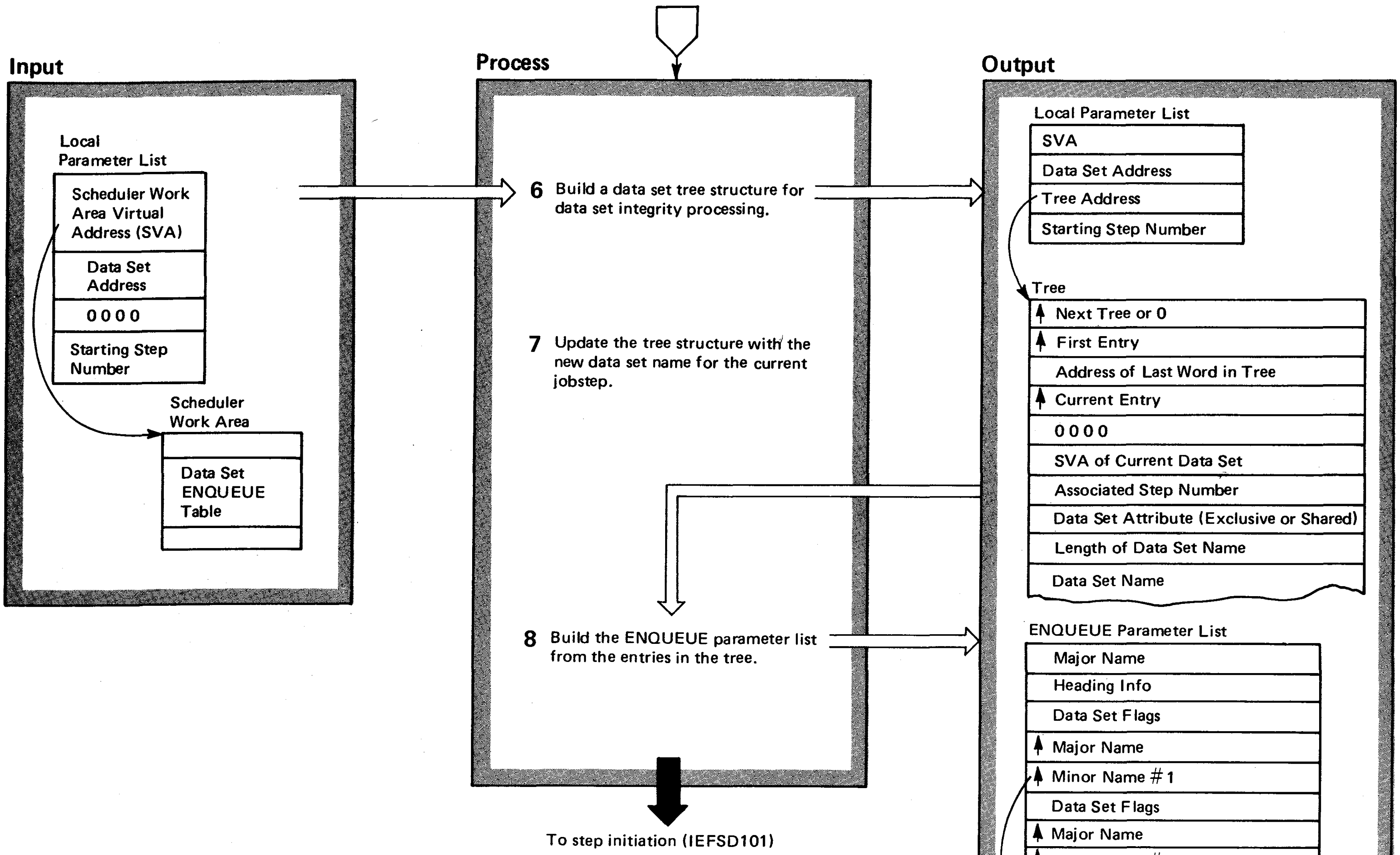
Otherwise, request a job from the job entry subsystem.

Diagram 10-1. Initiator: Job Initiation (Part 2 of 4)

Extended Description	Module	Label
The initiator interface control module (IEFIIC) issues a MODESET macro instruction to put the initiator task into supervisor state; it then begins building the control blocks required to process a jobstep or task.		
1 IEFIIC issues a GETMAIN macro for storage to build the initiator's entrance, options, and exit list (IEL).	IEFIIC	
2 IEFIIC gets storage to construct the subsystem identification block (SSIB) and the subsystem options block (SSOB). It determines the name of the subsystem which will select jobs for this initiator and places it in the SSIB: If the subsystem name was specified on a START command, a command input buffer (CIB) exists for it and the subsystem name is taken from there. If no CIB exists, IEFIIC checks for a subsystem name in the PARM field of the EXEC statement for this step and uses it. If no subsystem has been specified on an EXEC statement, the default value (the primary subsystem name found in the JESCT) is used. IEFIIC deletes the RACF accessor environment if one was obtained for the initiator. IEFIIC sets an initiator indicator in the command scheduling control block (CSCB) and passes control to IEFSD160. IEFSD160, the initiator subroutine receives control from IEFIIC for a normally initiated job or task, from started task control processing for a started task, or from master scheduler initialization.		

Extended Description	Module	Label
3 IEFSD160 gets storage for the linkage control table (LCT) from the SWA subpool pointed to by the current job step control block (JSCB); it then moves information from the IEL into the LCT.	IEFSD160	
4 After initializing the queue management parameter area (QMPA) in the LCT, IEFSD160 builds a 16-byte parameter list for a STAE exit routine, then issues an ESTAE macro instruction. It places a pointer to this private STAE parameter list (STEPL) in the LCT. IEFSD160 passes control to IEFSD161.		
5 IEFSD161, the job select routine, checks an indicator in the LCT to determine if STOP processing is required. If so, it frees the SSOB, SSIB, and the STEPL pointer if one exists, and passes control to a termination routine specified in the initiator's exit list. If STOP processing is not required, IEFSD161 issues the IEFSSREQ macro instruction, a routine that interfaces with the subsystem interface routine. When control is returned to IEFSD161 along with job status information, it checks the return code in the SSOB or register 15 to determine if the initiator should stop at this point. If so, it frees the SSOB, SSIB and the STEPL and passes control to a termination routine specified in the initiator's exit list. IEFSD161 next checks an indicator in the LCT to determine if the selected job is being warmstarted. If it is, control passes to the step delete routine, IEFSD164, to delete the current step.	IEFSD161	

Diagram 10-1. Initiator: Job Initiation (Part 3 of 4)



6 Build a data set tree structure for data set integrity processing.

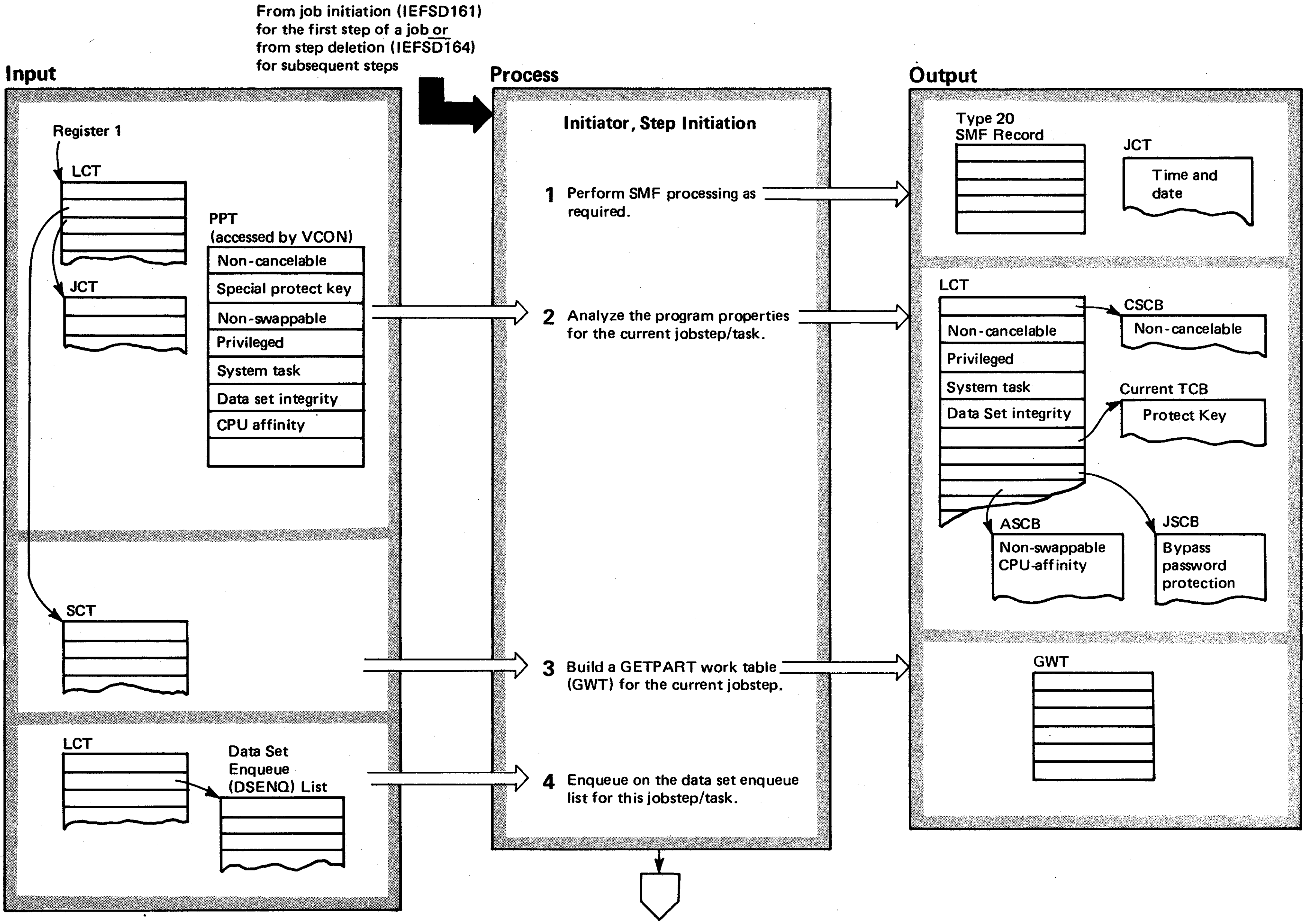
7 Update the tree structure with the new data set name for the current jobstep.

8 Build the ENQUEUE parameter list from the entries in the tree.

Diagram 10-1. Initiator: Job Initiation (Part 4 of 4)

Extended Description	Module	Label
6 IEFSD161 then checks an indicator in the JCT to determine if data set integrity processing is necessary for this job. If it is, IEFSD161 reads each data set name and passes it in a parameter list to IEFDSTBL. To process data set integrity (the assignment of the exclusive or shared attribute to a data set), IEFDSTBL builds a data set tree structure. The purpose of the tree is to eliminate duplicate data set names in the ENQUEUE parameter list which will ultimately be built for a job. The parameter list passed to IEFDSTBL contains the step number at which the job started, as well as a data set name and its current associated step number. The entire procedure ensures that a data set in use for a job will not be freed until after the last step needing it has used it. If this is the first entry into IEFDSTBL for a job, IEFDSTBL issues a GETMAIN for storage for the tree and initializes it with control information.	IEFSD161 IEFDSTBL	
7 IEFDSTBL determines if the job is a restart by comparing the starting step number in the parameter list with the current step number in the data set entry. If the current step number is larger, the job is a restart. No further data set integrity processing is needed since a DSENQ list already exists for a restarted job. IEFDSTBL simply returns control to IEFSD161. For jobs that are not restarts or a first entry, IEFDSTBL compares the data set name in the parameter list with the first data set entry in the tree. If the two data set names match, IEFDSTBL compares the associated step number in the tree to the current step number. If the current step number is higher, IEFDSTBL replaces the step number in the tree with the current step number. It also replaces the associated data set attribute (exclusive or shared) in the tree if the current attribute is more restrictive (exclusive). If the data set names do not match, IEFDSTBL continues searching through the tree until it does find a match; then, if necessary, it updates the step number and data set attribute in the tree. If IEFDSTBL reaches the end of the tree without finding a match, it adds the new data set name , its associated step number, and its attribute to the end of the tree. It returns control to IEFSD161.	IEFDSTBL	
IEFSD161 looks at the CPU-task affinity indicator in the program properties table (PPT). When affinity is required, IEFSD161 calls IEFICPUA to assign CPU-task affinity to the job. If the return code from IEFCPUA is not zero, affinity cannot be assigned and IEFSD161 issues an appropriate message via IEFIMASK which converts the CPU information in the PPT to readable text.	IEFSD161	
8 Once the tree structure contains all the data set entries for a job, IEFSD161 passes control to IEFDSLST to build the ENQUEUE parameter list. IEFDSLST places the system data set name (major name) and the individual data set names (minor names) from the tree into the ENQUEUE parameter list and frees the tree, since it is no longer needed. Control returns to IEFSD161.	IEFSD161 IEFDSLST	

Diagram 10-2. Initiator: Step Initiation (Part 1 of 8)



1 Perform SMF processing as required.

2 Analyze the program properties for the current jobstep/task.

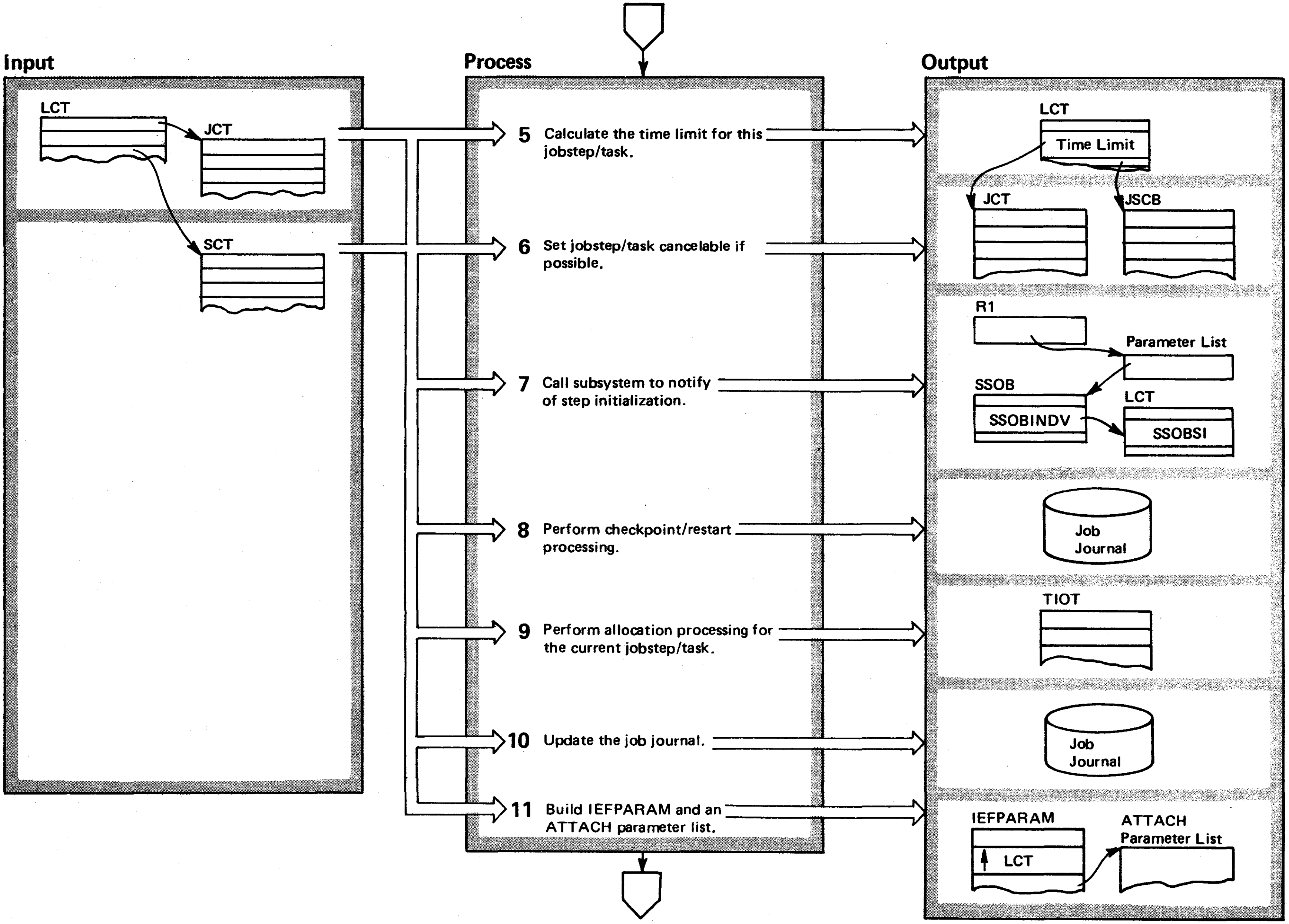
3 Build a GETPART work table (GWT) for the current jobstep.

4 Enqueue on the data set enqueue list for this jobstep/task.

Diagram 10-2. Initiator: Step Initiation (Part 2 of 8)

Extended Description	Module	Label
1 IEFSD161 passes control to the PPT scan routine, IEFSD101, which in turn calls IEFSMFIE for SMF processing. Once IEFSMFIE has determined that SMF options are to be performed, it stores the current time and date in the JCT.	IEFSD101 IEFSMFIE	
For the first step of a job, IEFSMFIE, the SMF initialization exit support routine, constructs a timing control table (TCT). At this point, if a user job initiation routine is provided, it is executed. When control returns to IEFSMFIE, it builds a type 20 SMF record.		
For every step in a job, IEFSMFIE executes a user step initiation routine, if one is provided. When control returns to IEFSMFIE, it passes control to IEFSD101 with an indicator in the JCT if either the user's job or step initiation routine caused job cancellation.		
2 By checking a protect key in the JCT, IEFSD101 determines if the current job step is to run in V=R or V=V. In either case, it moves the protect key into the current TCB. (When a user has specified V=R for a job step, his program is allocated a contiguous area of real storage and of virtual storage, both with identical addresses. His entire program is loaded into real storage at one time and cannot be paged.)	IEFSD101	
Before assigning any other special properties to this program, IEFSD101 sets to zeroes the special properties indicators that were set for a previous step. It then scans the program properties table (PPT) for the following properties:		
Special protect key — If a special protect key is indicated in the PPT, IEFSD101 moves it into the current TCB.		
Non-cancelable job — If the non-cancelable property is indicated in the PPT, IEFSD101 sets an indicator in the LCT and marks the CSCB non-cancelable.		
Non-swapable — If the program is marked non-swapable in the PPT, IEFSD101 sets the appropriate indicator in the ASCB.		
Privileged — If the program is marked privileged in the PPT, IEFSD101 sets an indicator in the LCT. The privileged property ensures that a program will not be swapped unless it is in a long wait.		
System task that is also a one-step started task — IEFSD101 sets an indicator in the LCT that indicates that the task need not be timed.		
System task that is an initiated task and/or consists of more than one step — IEFSD101 sets an indicator in the LCT to assign some of the normal program properties to the task and to issue an appropriate message.		
No data set integrity — For a one step job, IEFSD101 sets an indicator in the LCT to assign this property to a program. For a job consisting of more than one step, an indicator is set in the LCT to assign some of the normal program properties to the program and to issue an appropriate message.		
Bypass password protection — IEFSD101 sets an indicator in the JSCB.		
CPU task affinity — IEFSD101 checks this property for all steps in a job other than the first. When affinity is required, IEFSD101 calls IEFICPUA to assign CPU task affinity to the step via an indicator in the address space control block (ASCB). If the return code from IEFCPUA is not zero, affinity cannot be assigned and IEFSD101 issues an appropriate message by invoking IEFIMASK to convert the CPU information in the PPT to readable text.	IEFICPUA IEFIMASK	
3 IEFSD101 builds a GETPART work table (GWT) for the current job step if the user specified the REGION parameter or V=R mode or if a region beginning at a specific address is required for a checkpoint restart. A pointer to the GWT is placed in the LCT and control passes to IEFSD102.	IEFSD101	
4 If no data set enqueue list exists and the job is successful to this point, IEFSD102 passes control to the device allocation interface routine, IEFSD162.	IEFSD102 IEFSD162	
If a data set enqueue (DSENQ) list exists, but the job is unsuccessful, IEFSD102 frees the DSENQ list before it passes control to IEFSD162.		
If a data set enqueue list exists, this is the first step of a job that requires non-temporary data sets. IEFSD102 marks the CSCB cancelable and issues an ENQUEUE macro instruction for the DSENQ list.	IEFSD102	
If the ENQUEUE is unsuccessful, IEFSD102 issues an error message; otherwise, IEFSD102 waits for the ENQUEUE ECB to be posted (indicating that the specified data sets are now available) or the CANCEL ECB to be posted as a result of an operator CANCEL. In any case, control passes to the device allocation interface routine, IEFSD162.	IEFSD102	

Diagram 10-2. Initiator: Step Initiation (Part 3 of 8)



5 Calculate the time limit for this jobstep/task.

6 Set jobstep/task cancelable if possible.

7 Call subsystem to notify of step initialization.

8 Perform checkpoint/restart processing.

9 Perform allocation processing for the current jobstep/task.

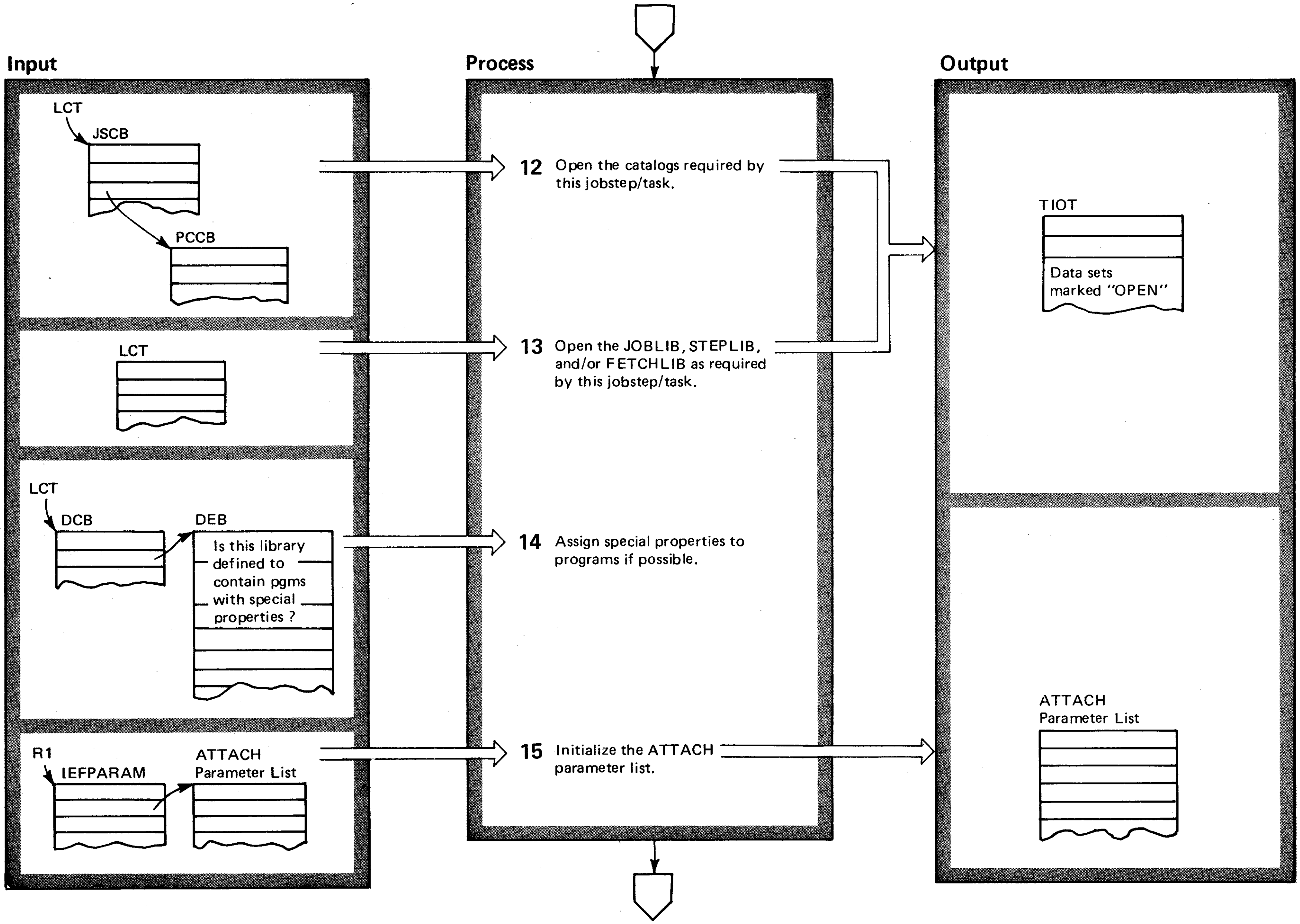
10 Update the job journal.

11 Build IEFPARAM and an ATTACH parameter list.

Diagram 10-2. Initiator: Step Initiation (Part 4 of 8)

Extended Description	Module	Label
5 IEFSD162 first calculates the step time limit using input from the SCT, JCT and LCT; the resultant time limit for the current job step is stored in the LCT.	IEFSD162	
6 If the current jobstep task is a started task (this is indicated in the CSCB), IEFSD162 sets up fields in the command scheduling control block (CSCB) so that the task will have a name that can be specified on a CANCEL command.		
7 IEFSD162 builds the SSOBSI extension in the LCT work area. Then, using the IEFSSREQ macro, it calls the subsystem to notify it of step initialization, providing step names and step number. On return from the interface, if register 15 does not indicate a "successful call" or "function not supported by subsystem", issue a X'0BA' ABEND.	IEFSD162	IEFJSREQ
8 If checkpoint/restart processing is required, IEFSD162 calls IEFXB604 to set appropriate job status bits in the job step control block (JSCB) and JCT to indicate that allocation processing is beginning for the current jobstep/task. IEFXB604 also writes the step's header record in the job journal before returning control to IEFSD162.	IEFXB604	
9 IEFSD162 gets storage for both a save area and parameter list for the allocation routines. At this time, if the current jobstep/task is a system task, IEFSD162 marks the CSCB cancelable for the duration of allocation processing:	IEFSD162	
it then branches to the device allocation load module, IEFW21SD. When IEFSD162 again receives control, if necessary, it restores the non-cancelable status of the task.	IEFW21SD	
If allocation was unsuccessful, IEFSD162 sets an indicator in the initiator exit list (IEL) and passes control to IEFSD164 to delete the jobstep/task.	IEFSD162	
10 After allocation processing, IEFSD162 updates the JSCB and JCT and calls IEFXB500 to write the updated information into the job journal.	IEFSD162 IEFXB500	
11 In preparation for ATTACH processing, IEFSD162 issues a GETMAIN for storage for IEFPARAM, which will serve as the initiator's internal parameter list, and for an ATTACH parameter list. IEFSD162 places a pointer to the LCT and to jobstep/task TIOT (created by the allocation routines) in IEFPARAM. It next places a pointer to IEFPARAM in the STEPL. IEFSD162 then calls SWA manager to write the SCT and JCT into the job journal.	IEFSD162	

Diagram 10-2. Initiator: Step Initiation (Part 5 of 8)



12 Open the catalogs required by this jobstep/task.

13 Open the JOBLIB, STEPLIB, and/or FETCHLIB as required by this jobstep/task.

14 Assign special properties to programs if possible.

15 Initialize the ATTACH parameter list.

Diagram 10-2. Initiator: Step Initiation (Part 6 of 8)

Extended Description	Module	Label
12 Before beginning OPEN processing, IEFSD162 places a pointer to the jobstep/task TIOT in the initiator's own TCB. It then checks the jobstep/task JSCB to see if there are catalogs to be opened. If so, IEFSD162 calls the initiator interface to catalog control, IEFICATL. This routine scans the DSAB (data set association block) chain associated with the jobstep/task to identify the required catalogs. It then invokes IEFAB4F5 to open these catalogs and update the private catalog control blocks (PCCBs), and returns control to IEFSD162. If OPEN processing is unsuccessful, IEFSD162 branches to IEFSD164 to delete the jobstep/task.	IEFSD162	
13 IEFSD162 issues an OPEN macro instruction for the JOBLIB, if one exists, or for the STEPLIB if a STEPLIB exists. It issues another OPEN macro instruction for FETCHLIB if it is also required. When OPEN processing has completed successfully, IEFSD102 restores the TIOT pointer in the initiator's TCB so that it once again points to the initiator's own TIOT.	IEFSD162	
14 IEFSD162 checks the related data event blocks (DEBs) to see if the job library or step library just opened is an authorized library (this is indicated in the DEB). If the library is authorized, complete the assignment of special properties. If the library is not authorized, assign normal properties to the job step and issue an appropriate message. When this is done, IEFSD162 branches to IEFSD103 for ATTACH processing. (Special and normal properties are discussed in *OS/VS2 SPL: Job Management.*)	IEFSD162	
15 IEFSD103, the ATTACH interface routine, places the following information in the ATTACH parameter list passed to it: • The entry point of the problem porgram to be attached in behalf of the jobstep/task. • The address of the ATTACH ECB. • The address of the FETCHLIB DCB. • The address of the STEPLIB or JOBLIB DCB. • The identification of which SWA subpool (236 or 237) cannot be shared. If the DPRTY parameter was specified for the jobstep, IEFSD103 calculates an address space priority for the job. If DPRTY was not specified, the automatic priority group (APG) from the CVT is used. In either case, IEFSD103 puts the memory priority, along with the performance group number, into IEFPARAM. It then branches to the ATTACH routine, IEFSD263.	IEFSD103	

Diagram 10-2. Initiator: Step Initiation (Part 7 of 8)

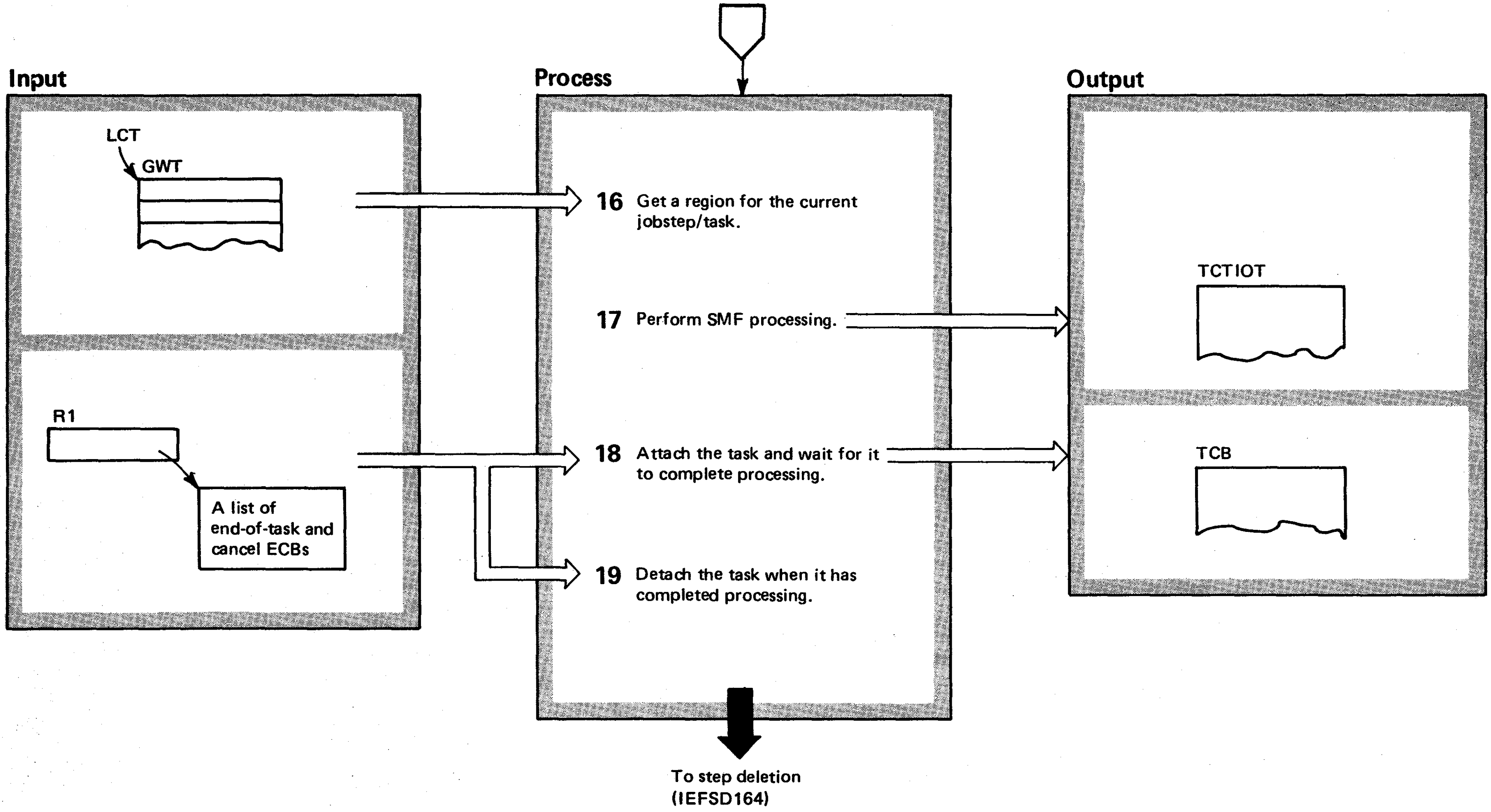
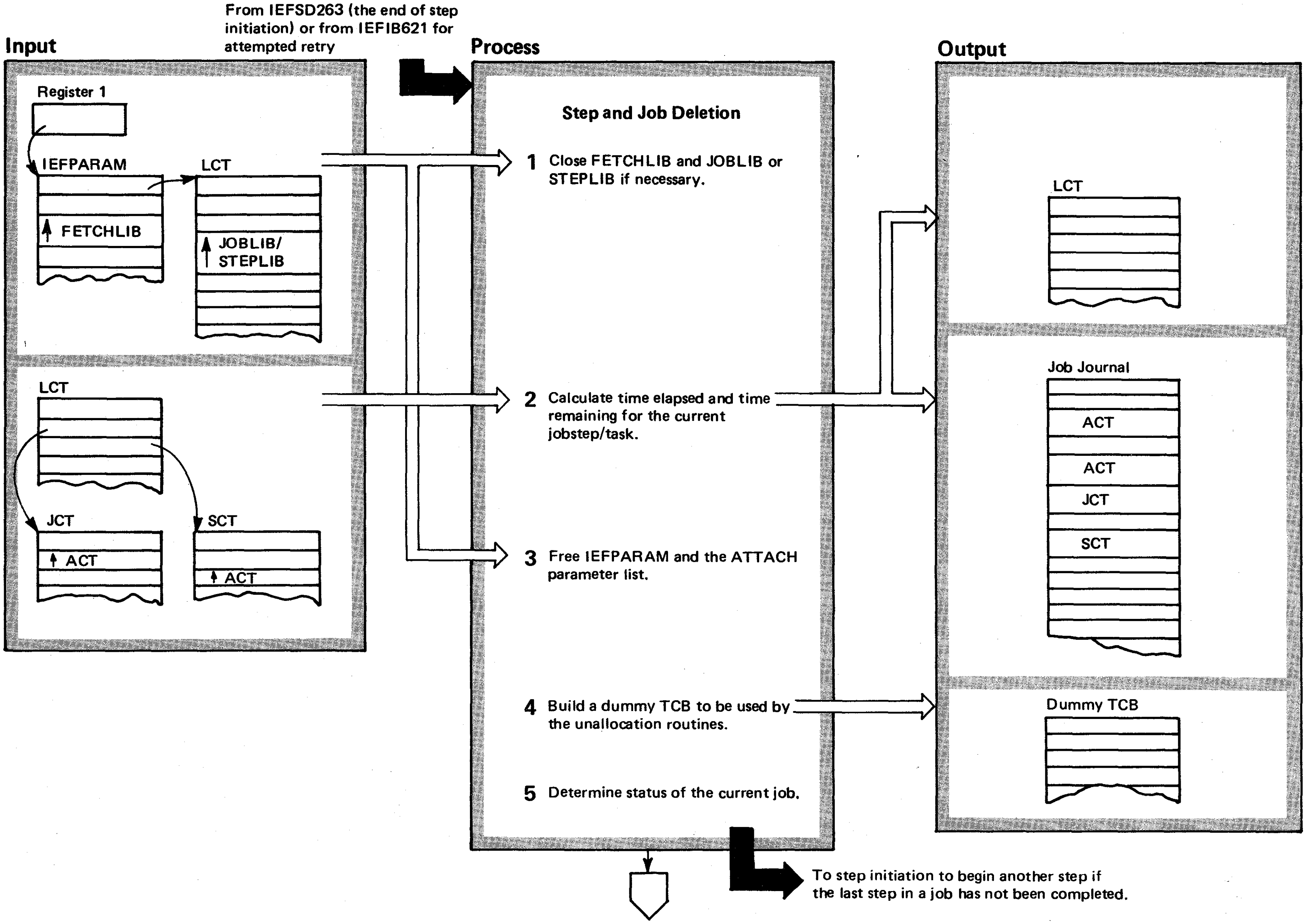


Diagram 10-2. Initiator: Step Initiation (Part 8 of 8)

Extended Description	Module	Label
16 If the jobstep/task is not swapable, IEFSD263 issues a SYSEVENT macro instruction, REQSWAP, that causes the initiator's own address space to be swapped out. It also frees the initiator's region.	IEFSD263	
When no GETPART work table (GWT) exists for a jobstep/task, IEFSD263 obtains a V=V region of default size.		
If there is a GWT, a special type of region is required for the jobstep/task. IEFSD263 issues a GETMAIN for a region. If the request cannot be immediately satisfied, IEFSD263 waits for a GETPART or CANCEL ECB to be posted indicating whether the GETMAIN completed successfully or failed.		
17 IEFSD263 calls IEFAB820 to build a TCTIOT (timing control TIOT), if one is required.	IEFAB820	
18 When IEFSD263 regains control from IEFAB820, it moves the jobstep parameter area from subpool 253 to subpool 0, issues an ATTACH for the jobstep/task, and sets a time limit in the ASCB. It takes the task's ASCB priority from IEFPARAM, issues the STATUS macro instruction to make the newly created TCB dispatchable, and then issues a WAIT macro instruction. It waits for the end-of-task and for the cancel ECBs associated with the attached task to be posted.	IEFSD263	
19 If the cancel ECB is posted, IEFSD263 invokes the abnormal termination routines via SVC 34 and issues another WAIT macro instruction for abnormal termination processing to complete.	IEFSD263	
When the cancel ECB is posted a second time, or when the end-of-task ECB is posted once, IEFSD263 begins DETACH processing.		
If the jobstep was timed, IEFSD263 saves the time allowed for the job and the time used by the job step (both in the LCT) and calculates the time remaining. It builds a parameter list to be used for step deletion processing, frees the jobstep/task region and if one exists, the GWT, and finally branches to the step delete routine, IEFSD164.	IEFSD263	

Diagram 10-3. Initiator: Step and Job Deletion (Part 1 of 4)



Step and Job Deletion

1 Close FETCHLIB and JOBLIB or STEPLIB if necessary.

2 Calculate time elapsed and time remaining for the current jobstep/task.

3 Free IEFPARAM and the ATTACH parameter list.

4 Build a dummy TCB to be used by the unallocation routines.

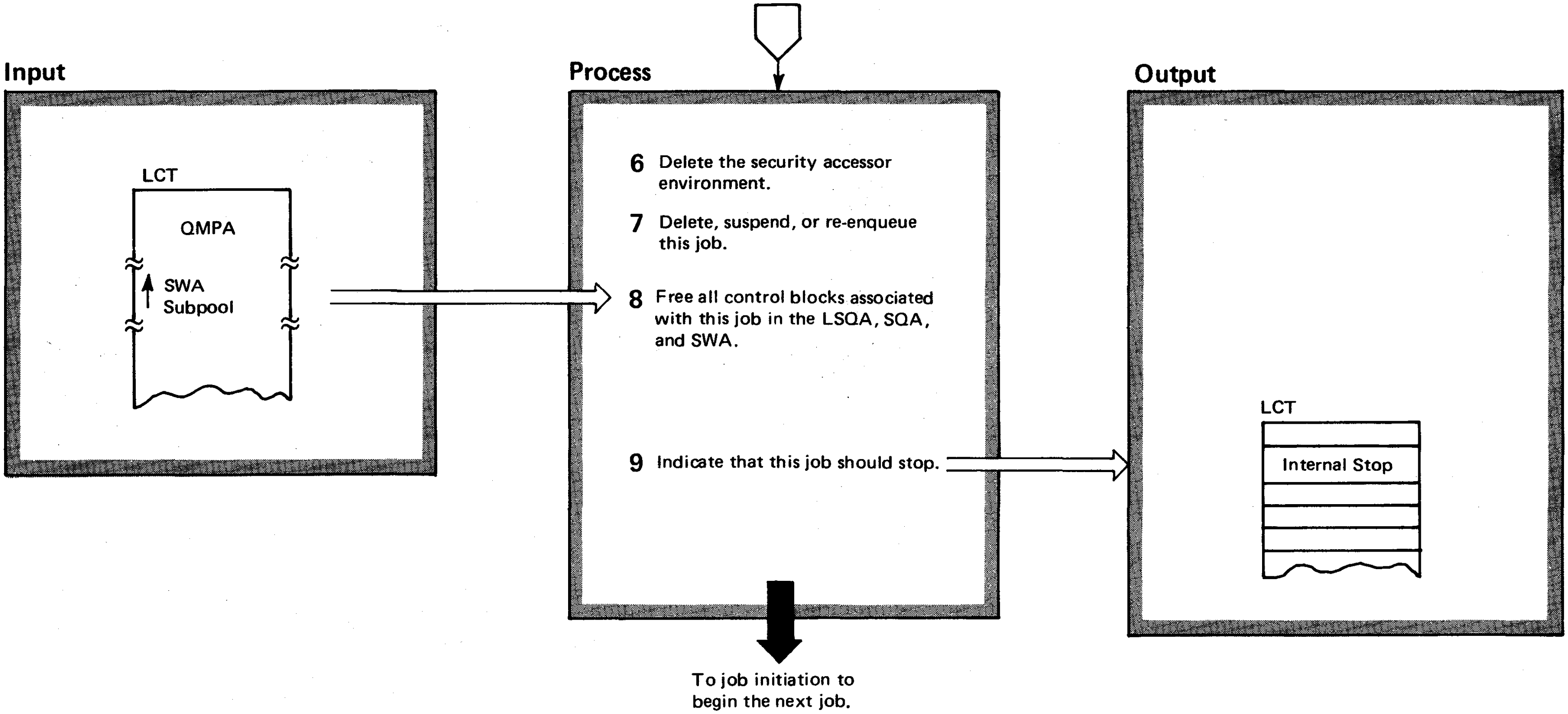
5 Determine status of the current job.

To step initiation to begin another step if the last step in a job has not been completed.

Diagram 10-3. Initiator: Step and Job Deletion (Part 2 of 4)

Extended Description	Module	Label
When the step delete routine IEFSD164 receives control, it checks indicators in the JCT and LCT to determine if the jobstep is being deleted for warmstart processing or because of an error during allocation. If either of these conditions exists, IEFSD164 begins processing at step 3.	IEFSD164	
1 IEFSD164 closes FETCHLIB if it was used by the jobstep/task and frees the storage its DCB occupied; it does the same for a JOBLIB or STEPLIB.	IEFSD164	
2 IEFSD164 calculates the SRB time for the jobstep and writes it the SCT. It calculates the execution time for the jobstep and writes it into the step account table (ACT). It does the same calculations for the job and writes the resultant figures in the JCT and the job account table (ACT) respectively. IEFSD164 then calls SWA manager to write the updated block into the job journal.	IEFSD164	
3 IEFSD164 frees IEFPARAM, sets to zero the pointer to it in STEPL, and also frees the ATTACH parameter list.	IEFSD164	
4 IEFSD164 builds a dummy TCB to be used by the unallocation routines; the dummy TCB contains the jobstep/task status and completion codes. When the dummy TCB is completed, control passes to the unallocation routines to free the data sets and devices used by the jobstep/task.	IEFSD164	
5 When IEFSD164 regains control from unallocation, it frees the dummy TCB and checks the return codes.	IEFSD164	

Diagram 10-3. Initiator: Step and Job Deletion (Part 3 of 4)



6 Delete the security accessor environment.

7 Delete, suspend, or re-enqueue this job.

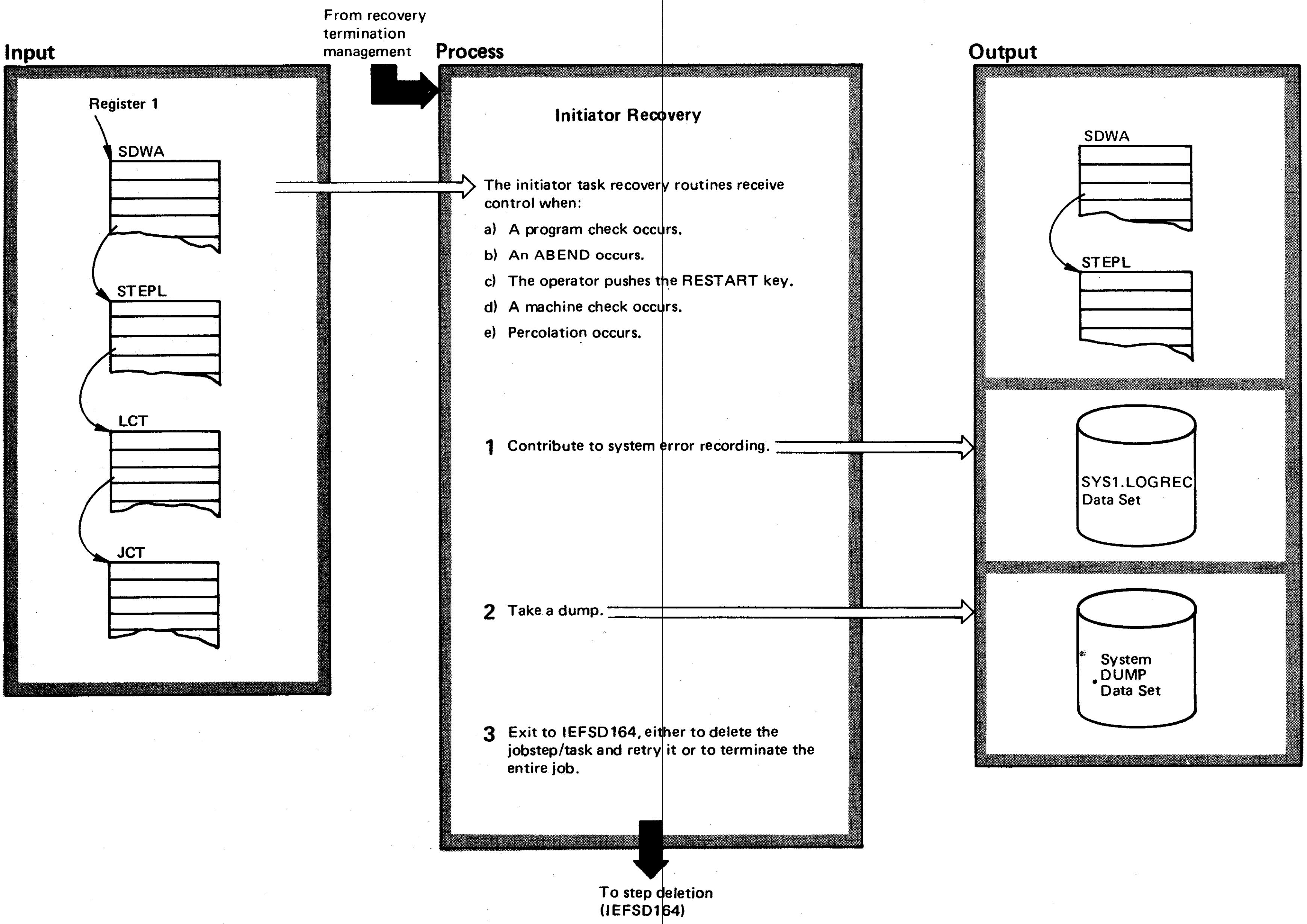
8 Free all control blocks associated with this job in the LSQA, SQA, and SWA.

9 Indicate that this job should stop.

Diagram 10-3. Initiator: Step and Job Deletion (Part 4 of 4)

Extended Description	Module	Label
6 If RACINIT processing was performed for this job step/started task by the SWA create routine (IEFIB600), then delete the RACF environment since the task has completed.		
7 If another step in the job is to be initiated, control passes to IEFSD101, the step initiation routine.	IEFSD101	
If the job step just completed was the last step in a job, control passes to IEFSD166, the job delete routine.	IEFSD166	
If the job associated with the jobstep/task is to be suspended, control passes to IEFSD166 to do this.		
If the job ran in V=R mode, IEFSD166 releases the job's protect key. It gets storage for job delete or job enqueue processing. The decision to delete or re-enqueue a job depends on the function code in a two-word parameter list pointed to by register one. IEFSD166 sets appropriate indicators in the SSOB and issues the IEFSSREQ macro instruction requesting the job entry subsystem to delete or re-enqueue the job.		
If no error occurs in job entry subsystem processing, IEFSD166 puts the return code from the subsystem into the IEL.	IEFSD166	
8 IEFSD166 frees all the control blocks associated with this job in the LSQA and SQA. It passes control to the SWA management routines requesting deletion of job related blocks in SWA. It then calls the auxiliary storage manager routine (ILRJTERM) that frees any ASM control blocks still existing for VIO data sets created by the job being terminated.		
9 If the completed job was a normally initiated task, IEFSD166 removes the job name from the initiator's TIOT.	IEFSD166	
If the completed task was not begun by the initiator, IEFSD166 sets an internal stop indicator in the LCT.		
Control passes to IEFSD161.	IEFSD161	

Diagram 10-4. Initiator: Recovery Processing (Part 1 of 2)



The initiator task recovery routines receive control when:

a) A program check occurs.

b) An ABEND occurs.

c) The operator pushes the RESTART key.

d) A machine check occurs.

e) Percolation occurs.

1 Contribute to system error recording.

2 Take a dump.

3 Exit to IEFSD164, either to delete the jobstep/task and retry it or to terminate the entire job.

Diagram 10-4. Initiator: Recovery Processing (Part 2 of 2)

Extended Description	**Module**	**Label**
The initiator task recovery routine (IEFIB620) receives control when: a) a program check occurs, b) an ABEND occurs, c) the operator pushes the RESTART key, d) a machine check occurs, e) percolation occurs.		
1 IEFIB620 receives control from recovery/termination management (R/TM). If R/TM does not provide a STAE diagnostic work area (SDWA), IEFIB620 simply sets an indicator in register 15 to continue termination processing and returns to R/TM.	IEFIB620	
Unless the error that occurred was an OPEN failure or unless the routine received control as a result of percolation, IEFIB620 records the error in the SDWA.		
2 If entry into this routine is not the result of percolation, recursion, an OPEN failure, or a machine check, IEFIB620 issues an SDUMP macro instruction.		
3 If this is not a recursion or if the LCT does not contain both a JCT SWA address and an SCT SWA address, IEFIB620 sets a retry indicator in the SDWA and places the address of the retry routine in the SDWA. It then returns to its caller, R/TM.		
R/TM, in turn, passes control to IEFIB621, the initiator task recovery retry routine, which will enable the retry and then pass control to IEFSD164, the initiator step delete routine to delete the step currently in progress.	IEFIB621	

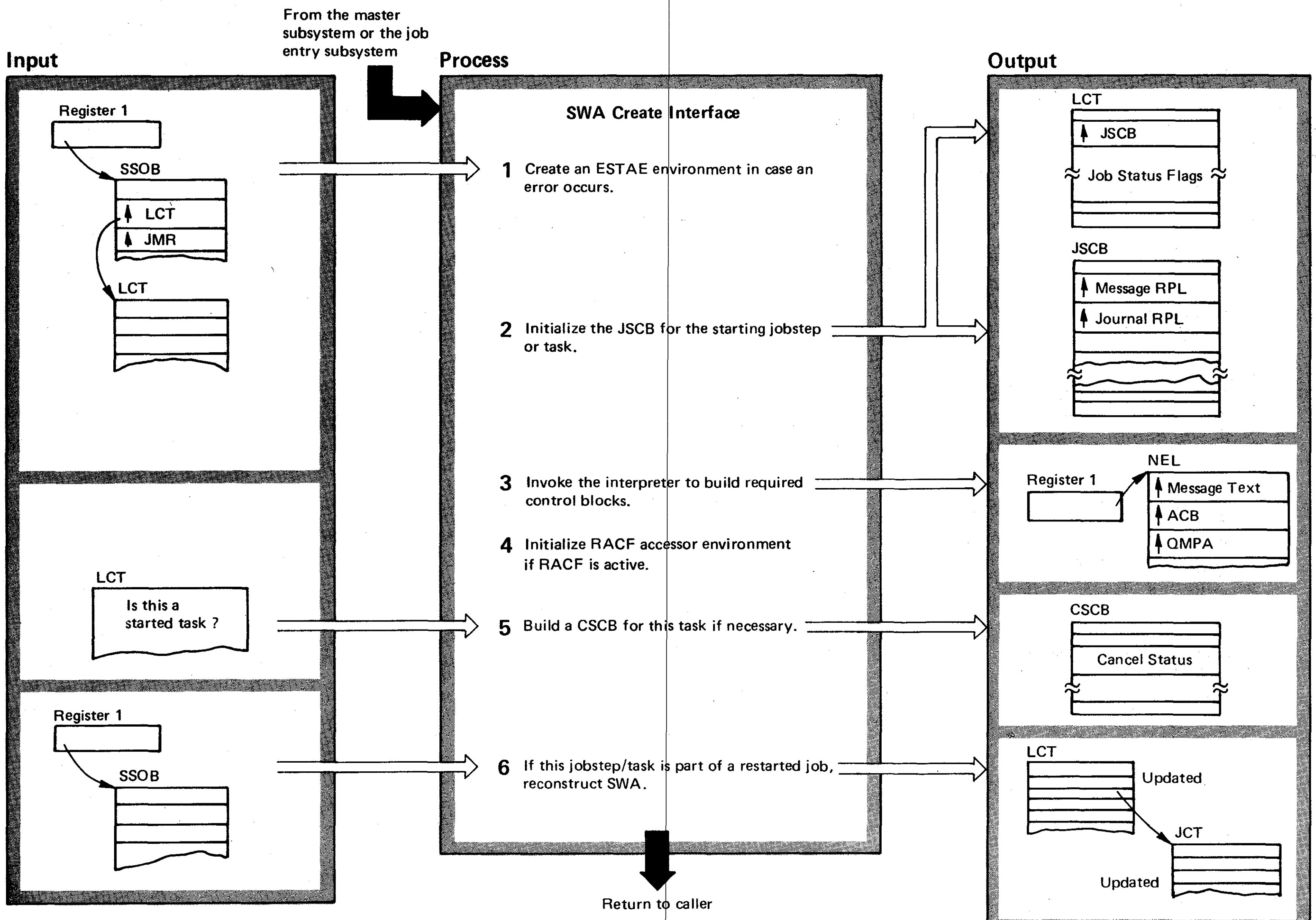
SWA Create Interface

The SWA create interface routines receive control from either the master subsystem or the job entry subsystem. Their main function is to prepare a job for the interpreter by setting up its job step control block (JSCB) chain. One of the SWA create interface routines, IEFIB600, passes control to the interpreter, and when control returns, it places the SWA address of the JCT in the JSCB for a job. It is the interpreter that actually builds the SWA and many of the control blocks that reside in SWA, for example, the JCT.

Whenever the current job is not a started task, the SWA create interface routines build a command scheduling control block (CSCB) to represent the job. (The CSCB for a started task is created by the started task control routines.)

The SWA create interface routines also reconstruct SWA for restarted jobs.

Diagram 11-1. SWA Create Interface (IEFIB600) (Part 1 of 2)



SWA Create Interface

1 Create an ESTAE environment in case an error occurs.

2 Initialize the JSCB for the starting jobstep or task.

3 Invoke the interpreter to build required control blocks.

4 Initialize RACF accessor environment if RACF is active.

5 Build a CSCB for this task if necessary.

6 If this jobstep/task is part of a restarted job, reconstruct SWA.

Diagram 11-1. SWA Create Interface (IEFIB600) (Part 2 of 2)

Extended Description	Module	Label
The SWA create interface routines set up the control blocks for a job before the job enters the interpreter. The SWA in which the control blocks will reside is created during interpreter processing.		
1 IEFIB600 first creates an ESTAE environment by issuing an ESTAE macro instruction. As a result, if an error occurs during SWA create interface processing, control will first pass to a recovery/termination routine, and from there, back to a SWA create exit routine, IEFIB645. The exit routine takes a dump of storage if it is required, and specifies retry. It then returns control to recovery/termination.	IEFIB600 IEFIB645	
Once the ESTAE environment is established, IEFIB600 sets the job status flags in the LCT indicating whether the job is an automatic checkpoint restart, a step restart, or a warmstart.	IEFIB600	
2 IEFIB600 next issues a GETJSCB macro instruction and when that is done, it chains the job's JSCBs and places a pointer to the first JSCB in the LCT. It initializes the JSCB with the following information:	IEFIB600	
• The address of the message request parameter list (RPL). • The address of the journal RPL. • The address of the QMPA. • The address of the CSCB for the job, if one exists. • An indicator that the job is entering the interpreter. • An indicator that no journaling is required. • A restart indicator. • The SWA subpool number. • The ASID for the job.		
3 IEFIB600 then initializes the interpreter entrance list (NEL) and issues a LINK macro instruction to pass control to IEFNB903, the first routine of the interpreter. Register 1 points to the NEL.	IEFIB600	
When control has returned, if an error occurred during interpreter processing, IEFIB600 frees appropriate control blocks, places an error return code in register 15, and returns to the original calling routine.		
When interpreter processing has completed successfully, IEFIB600 invokes a SWA manager routine to read the job control table (JCT) created by the interpreter. It places the SWA address of the JCT in the first JSCB, in the JCT itself, and in the LCT.		
4 If the RACF function is active, check if the userid is valid. If not, fail the job and issue an error message (IEF722I). If successful, check if automatic data set protection was requested. If it was, set the JSCBADSP in the active job step control block for use by allocation.		
5 If the job in processing is not a started task, IEFIB600 builds a command scheduling control block (CSCB) to represent the job. (If the job is a started task, the started task control routines have already built the CSCB.)		
6 Finally, IEFIB600 checks indicators in the SSOB and LCT to determine if the SWA for this job must be reconstructed. For restarted jobs, SWA must be rebuilt to reflect previous processing, as well as newly begun restart processing.	IEFIB600	
Whenever SWA reconstruction is necessary, IEFIB600 passes control to IEFIB605, the SWA reconstruct module, * otherwise, it returns control to the original caller.		
IEFIB605 first determines if the job is an automatic restart, step restart, a warmstart, or deferred restart. For any case but deferred restart, IEFIB605 invokes the SWA merge routine (IEFXB601).	IEFIB605 IEFXB601	
Before calling the merge module, IEFIB605 builds a parameter list, the merge entrance list (MEL) and places a pointer to it in register 1.	IEFIB605	
When merge processing has completed and control is returned, IEFIB605 checks job status again for an automatic or deferred restart. In both cases, it invokes the data set descriptor record processor, IEFXB609, and it passes it a pointer to the LCT.	IEFXB609	
This time, when control returns to IEFIB605, a subroutine checks job status for a warmstart. If the job is a warmstart, IEFIB605 determines whether the error that caused the warmstart occurred in allocation, execution, or termination and sets appropriate indicators.	IEFIB605	
In every case, IEFIB605 returns control to IEFIB600, who then returns to the caller.		

---

*This module is part of Checkpoint/Restart processing.

The Purpose of the Converter

The following is a brief overview of converter functions. For a thorough look at converter processing see the method-of-operation diagrams and extended descriptions.

In MVS, the converter/interpreter performs most of the functions performed by the reader/interpreter in OS. However, the converter/interpreter does not read in-stream JCL statements or any input stream data. The converter executes as a subroutine of the job entry subsystem (JES). JES actually reads JCL statements and input stream data and spools them to appropriate data sets. The converter then takes the records from these data sets and converts them into internal JCL text to be used by the interpreter. It also merges JCL that it reads from the procedure library with the JCL and input stream data spooled by JES.

Identifying JCL Statements

Once initialization is complete, the converter GET routine, IEFVHA, begins processing by obtaining a JCL statement (an 80-byte card image) from the JCL data set and/or from a cataloged or in-stream procedure.

Comments and Continuation

The next converter routine, IEFVHC, continues processing JCL statements by checking for a valid continuation. It branches to IEFVHEB if a continuation was expected and was or was not received, to IEFVHCB if a continuation was not expected, and to IEFVHA if a comment was received.

JOB, EXEC, DD Statements

Once a JOB, EXEC, or DD statement is identified, the converter pre-scan (IEFVHEB) performs some initialization functions and branches to the scan routine, IEFVFA. It is IEFVFA that converts all JCL card images taken from the JCL data set into internal text and then moves them to a text data set that will be used by the interpreter.

NULL Statements

The NULL statement processor, IEFVHL, analyzes the conditions under which it was entered.

If the NULL statement represents the end of an input stream job and more statements must be read from a procedure, control returns to the converter's GET routine, IEFVHA. When IEFVHA encounters a procedure end-of-file, it generates a NULL statement to signify the end of the procedure.

If the NULL statement indicates that there are no more JCL statements to be read and that the JCL data set and all procedures have been processed by the converter, IEFVHL invokes the converter termination routine, IEFVHF.

PROC and PEND Statements

An EXEC PROC statement identifies a procedure that exists in the system's procedure library. A PROC statement marks the beginning of an in-stream procedure. When the converter encounters a PROC statement in the input stream, it converts it to an EXEC PROC statement. For both cases, control passes to an in-stream procedure control routine, IEFVINA, that in turn calls a series of special processors.

The first of these, IEFVINE, is a syntax check routine. If it finds the PROC statement valid, it returns this information to the control routine.

The next module called, IEFVINB, scans the entries in the In-Stream Procedure Directory. If IEFVINB does not find an entry for the procedure name sepcified on the PROC statement, the control routine invokes another module, IEFVINC, to build a new entry.

When the entry is complete, the control routine branches to another module, IEFZNCODE; this one compresses the JCL statement and stores an pointer to the statement next to the procedure name in a local work area.

The control routine, IEFVINA, continues reading and compressing data until it encounters some kind of delimiter. When it reaches a PEND statement signifying the end of a procedure, it returns control to the converter GET routine, IEFVHA, for the next statement.

Symbolic Parameters

A user defines symbolic parameters either in statements within a procedure itself or in statements that override the statements in a referenced procedure (for example, one in the procedure library). Therefore, the Converter may encounter symbolic parameters in three places:

- In an EXEC statement that calls a procedure.

- In input stream statements that override procedure statements.
- In statements within a procedure.

When a symbolic parameter is specified on an EXEC statement, the converter scan routine, IEFVFA, uses the symbolic parameter processing routine, IEFVFB, to place an entry in a table of symbolic parameters and assigned values (SYMBUF).

When a symbolic parameter appears in an input-stream statement or in a statement in a procedure, IEFVFB, substitutes a corresponding value already in a symbolic parameter table entry for the symbolic parameter.

Command Statements

When the converter verb identification routine, IEFVHCB, cannot recognize a verb, it assumes that the verb is a command. The command validation routine, IEFVHM, verifies that the verb is one allowed in the input stream and issues an SVC 34 (the command scheduling supervisor call) to execute the command.

Service Routines

During converter processing, most converter routines use a set of service routines that perform some common functions.

The message module, IEFVGM, puts the converter messages into the message data set and JCL statements into the list data set.

The operator message module, IEFVHR, places messages intended for the operator into the message data set.

The SWA (shceduler work area) manager interface module, IEFVHQ, enables the converter routines to assign control blocks to SWA, to locate blocks there, and to read from them and update them on SWA, as well.

The Purpose of the Interpreter

The interpreter operates as a subroutine of the Initiator but is actually called by SWA create interface. The purpose of the interpreter is to build the scheduler control blocks rquired to execute a job. The interpreter transforms the keywords and parameters specified in the JCL statements to specific table entries. In the interpreter, the JCL statements appear in the form of JCL text, the output of converter processing.

When interpreter initialization is done, the interpreter GET routine, IEFVHE, receives control. It determines whether a statement it is processing is a JOB, EXEC, or DD statement and then routes it to an appropriate processor.

The JOB Statement

The JOB statement processor (IEFVJA), initializes a job control table (JCT) and the job account control table (JACT) for a job. It also checks the validity of the JOB statement keyword values and enters them into the tables.

The EXEC Statement

IEFVEA processes EXEC statements. It creates a step control table (SCT) and a step account table (ACT) for each EXEC statement. IEFVEA also chains the job file control blocks (JFCB) whenever a JOBLIB has been specified, and chains the SCT for data set concatenations.

The DD Statement

The DD statement processor (IEFVDA) creates the step input/output tables (SIOTs) and JFCBs for a step and a data set enqueue table (DSENQ table) entry for all data sets explicity named by the DSNAME parameter. IEFVDA marks each data set entry in the DSENQ table as exclusive or shared according to the user's specifications.

Service Routines

The interpreter initialization routines, perform several common functions by using a set of service routines.

The message module, IEFVGM, puts the interpreter messages into the message data set.

The operator message module, IEFVHR, also puts messages intended for the operator into the message data set.

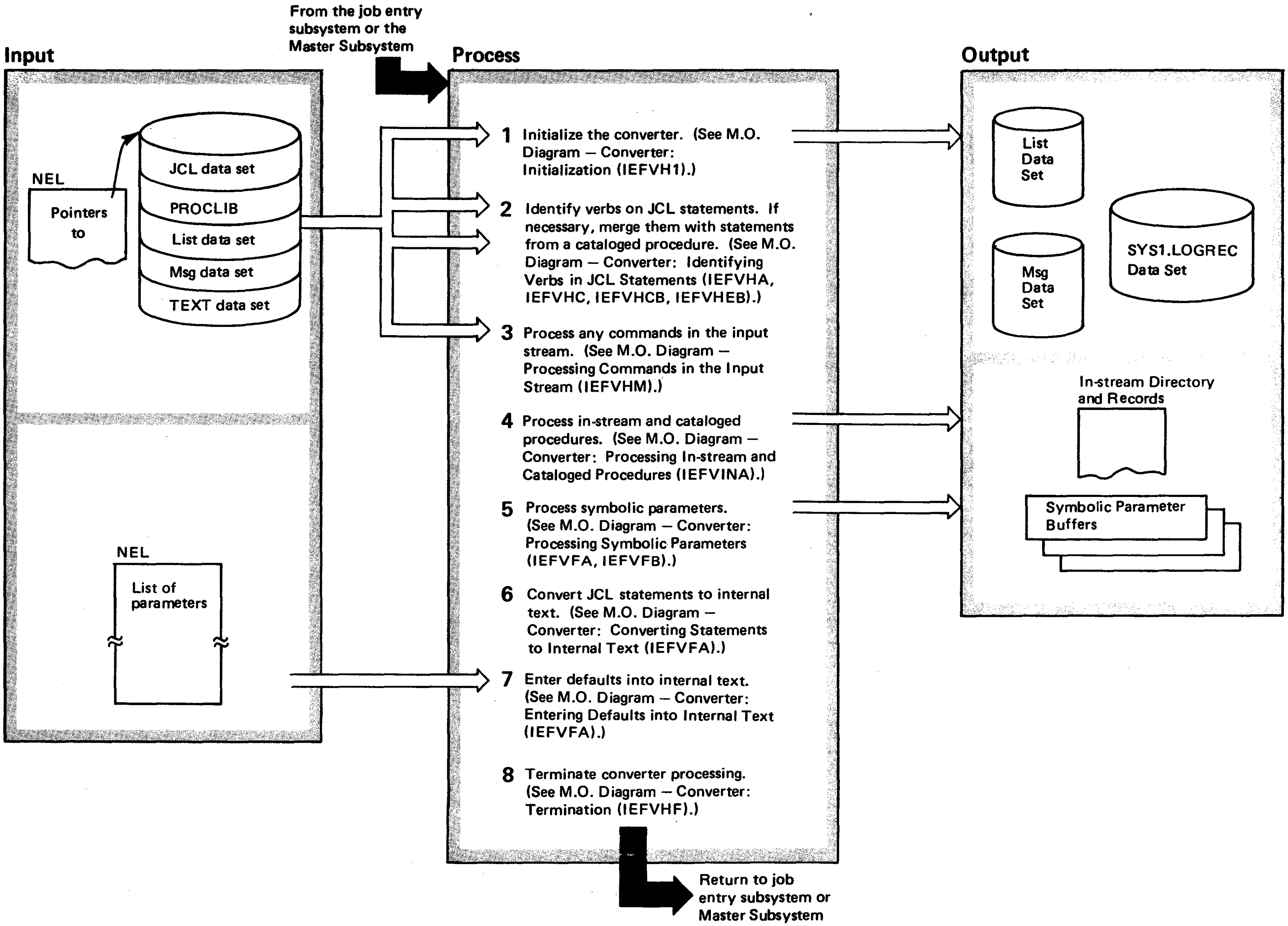
The SWA manager interface module, IEFVHQ, enables the interpreter to assign control blocks to SWA, to locate blocks there, and to read from them and write to them.

The statement processors, IEFVJA, IEFVEA, and IEFVDA, use a special set of service routines for functions common to them.

The interpreter GET parameter routine, IEFVGK, locates each parameter for the command statement processor. It branches to an appropriate keyword subroutine to perform a basic check for errors and then returns to the command statement processor.

The test and store routine, IEFVGT, enables the command statement processor to determine what processing must be done for each parameter. There is a parameter descriptor table (PDT) that lists each keyword, the operation to be performed for it, and the location at which the results must be stored.

The EXEC and DD statement processors, IEFVEA and IEFVDA, respectively, use a dictionary entry routine (IEFVGI) to place an entry in the "refer-back" table. They also use a dictionary search routine, IEFVGS to search the "refer-back" table for the address of an existing SCT, SIOT, or JFCB. Both IEFVGI and IEFVGS return to the calling routines.



1 Initialize the converter. (See M.O. Diagram – Converter: Initialization (IEFVH1).)

2 Identify verbs on JCL statements. If necessary, merge them with statements from a cataloged procedure. (See M.O. Diagram – Converter: Identifying Verbs in JCL Statements (IEFVHA, IEFVHC, IEFVHCB, IEFVHEB).)

3 Process any commands in the input stream. (See M.O. Diagram – Processing Commands in the Input Stream (IEFVHM).)

4 Process in-stream and cataloged procedures. (See M.O. Diagram – Converter: Processing In-stream and Cataloged Procedures (IEFVINA).)

5 Process symbolic parameters. (See M.O. Diagram – Converter: Processing Symbolic Parameters (IEFVFA, IEFVFB).)

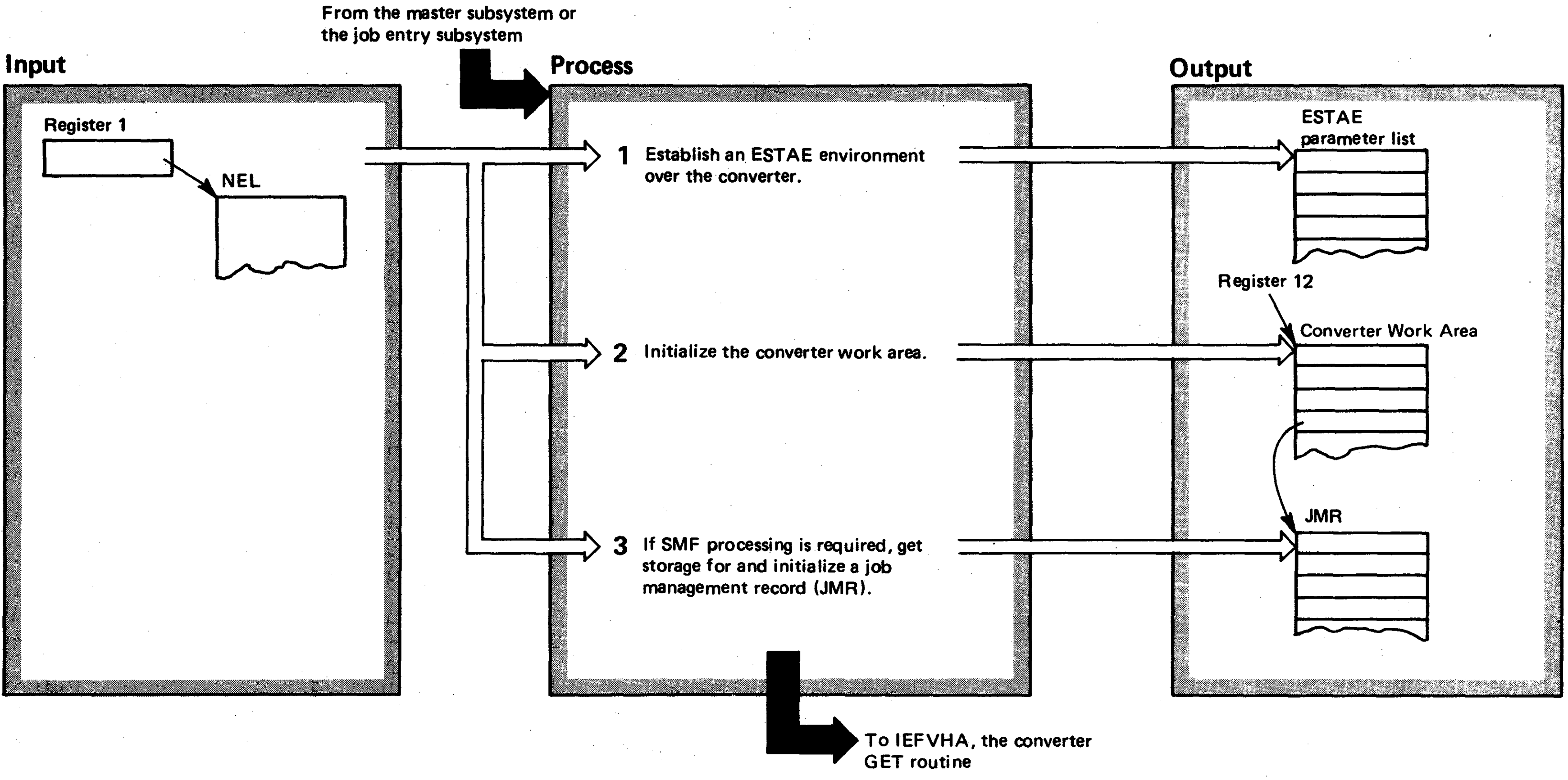
6 Convert JCL statements to internal text. (See M.O. Diagram – Converter: Converting Statements to Internal Text (IEFVFA).)

7 Enter defaults into internal text. (See M.O. Diagram – Converter: Entering Defaults into Internal Text (IEFVFA).)

8 Terminate converter processing. (See M.O. Diagram – Converter: Termination (IEFVHF).)

Figure 2-17a. Converter Visual Contents

Diagram 12-1. Converter: Initialization (IEFVH1) (Part 1 of 2)



1 Establish an ESTAE environment over the converter.

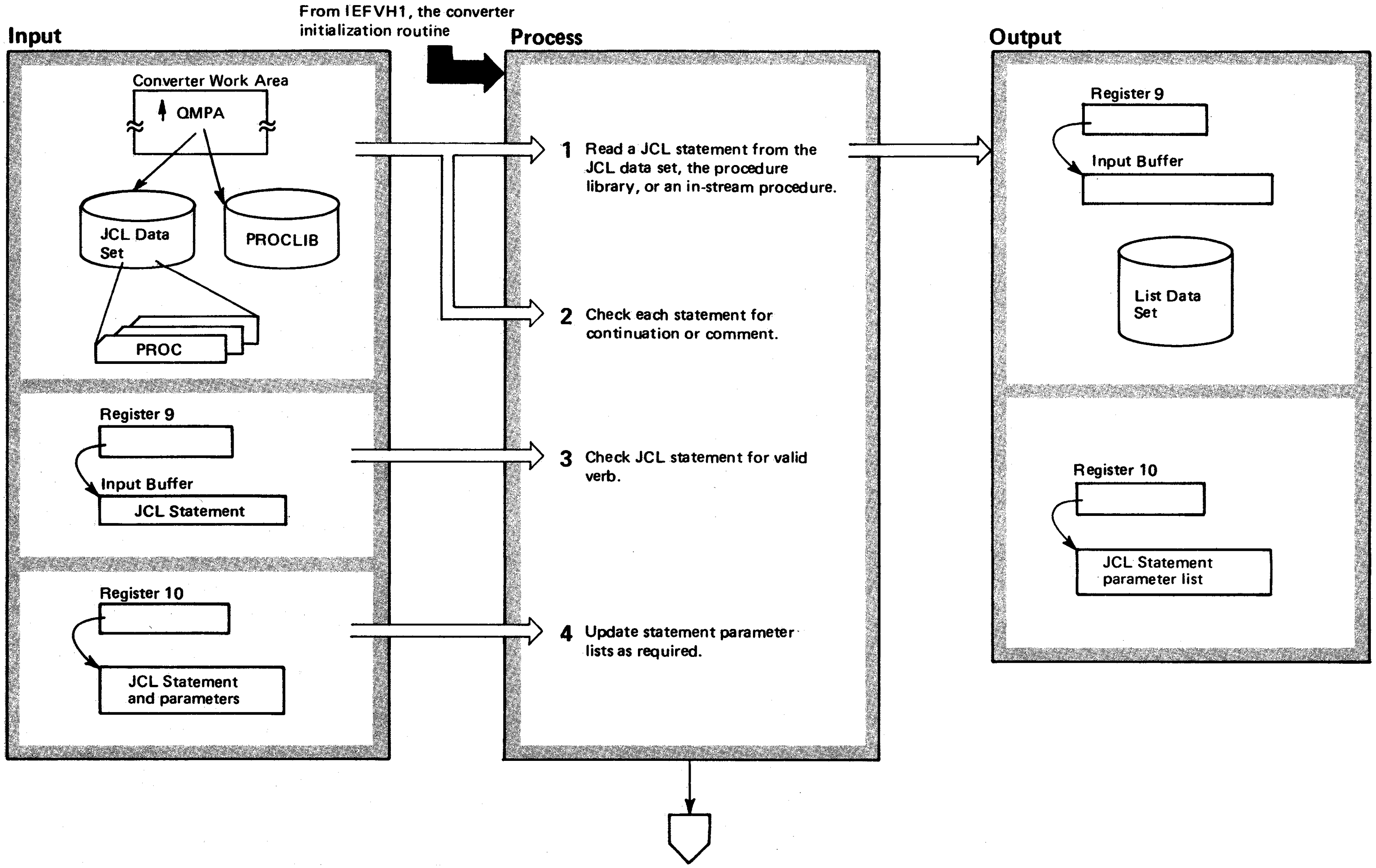
2 Initialize the converter work area.

3 If SMF processing is required, get storage for and initialize a job management record (JMR).

Diagram 12-1. Converter: Initialization (IEFVH1) (Part 2 of 2)

Extended Description	Module	Label
This module, IEFVH1, is the converter initialization routine. It receives control from the job entry subsystem or the master subsystem via a link macro instruction.		
1 IEFVH1 obtains storage for and initializes an ESTAE parameter list for the converter, then issues an ESTAE macro instruction.	IEFVH1	
2 IEFVH1 initializes the converter work area with the following information: ● The address of the interpreter entrance list (NEL). ● The address of the calling routine's save area. ● The address of the input statement buffer. ● The address of the internal text buffer. ● The address of the procedure library's DCB. ● The address of the procedure statement buffer. ● The address of the message buffer. ● The address of a local work area. ● The address of the converter's own register save area. ● The entire queue management parameter area (QMPA) passed by the calling routine.	IEFVH1	
3 IEFVH1 checks an indicator in the NEL to determine if SMF processing is required. If it is, IEFVH1 obtains storage for and initializes the job management record (JMR) for this job. It then passes control to IEFVHA, the converter GET routine.	IEFVH1	

Diagram 12-2. Converter: Identifying Verbs on JCL Statements (Part 1 of 4)



1 Read a JCL statement from the JCL data set, the procedure library, or an in-stream procedure.

2 Check each statement for continuation or comment.

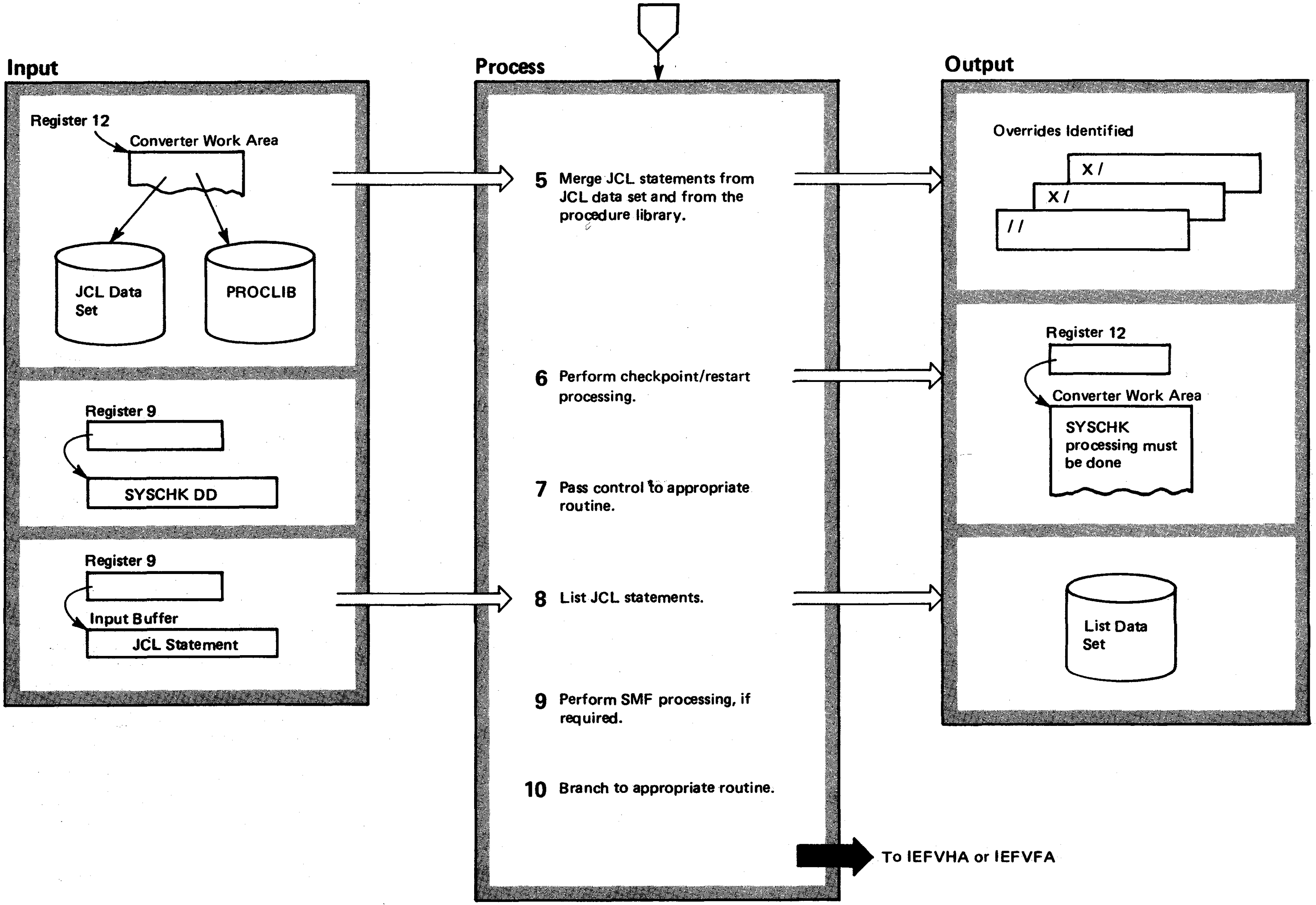
3 Check JCL statement for valid verb.

4 Update statement parameter lists as required.

Diagram 12-2. Converter: Identifying Verbs on JCL Statements (Part 2 of 4)

Extended Description	**Module**	**Label**
IEFVHA is the converter GET routine that reads JCL statements from the JCL data set or in-stream procedures spooled by the job entry subsystem and/or from the procedure library. IEFVHA receives control from the converter initialization routine, IEFVH1.		
1 IEFVHA issues a GET macro instruction for a statement from the JCL data set or the procedure library according to indicators in the converter work area. The statement is placed in an input buffer. Whenever IEFVHA encounters an end-of-file condition, it moves a NULL statement into the input buffer. In any case, it branches to IEFVHC, the comment or continuation validation routine.	IEFVHA	
2 IEFVHC determines whether a valid comment or continuation is indicated on the JCL statement in the input buffer. If IEFVHC expects a continuation of the statement or if it receives a comment, it passes control to a print routine in IEFVHEB. (See Step 8.) If IEFVHC does not expect a continuation, it passes control to IEFVHCB, the verb identification routine.	IEFVHC	
3 IEFVHCB checks the JCL statement for a valid verb.	IEFVHCB	
4 It updates the statement parameter list based on its findings.	IEFVHCB	

Diagram 12-2. Converter: Identifying Verbs on JCL Statements (Part 3 of 4)



5 Merge JCL statements from JCL data set and from the procedure library.

6 Perform checkpoint/restart processing.

7 Pass control to appropriate routine.

8 List JCL statements.

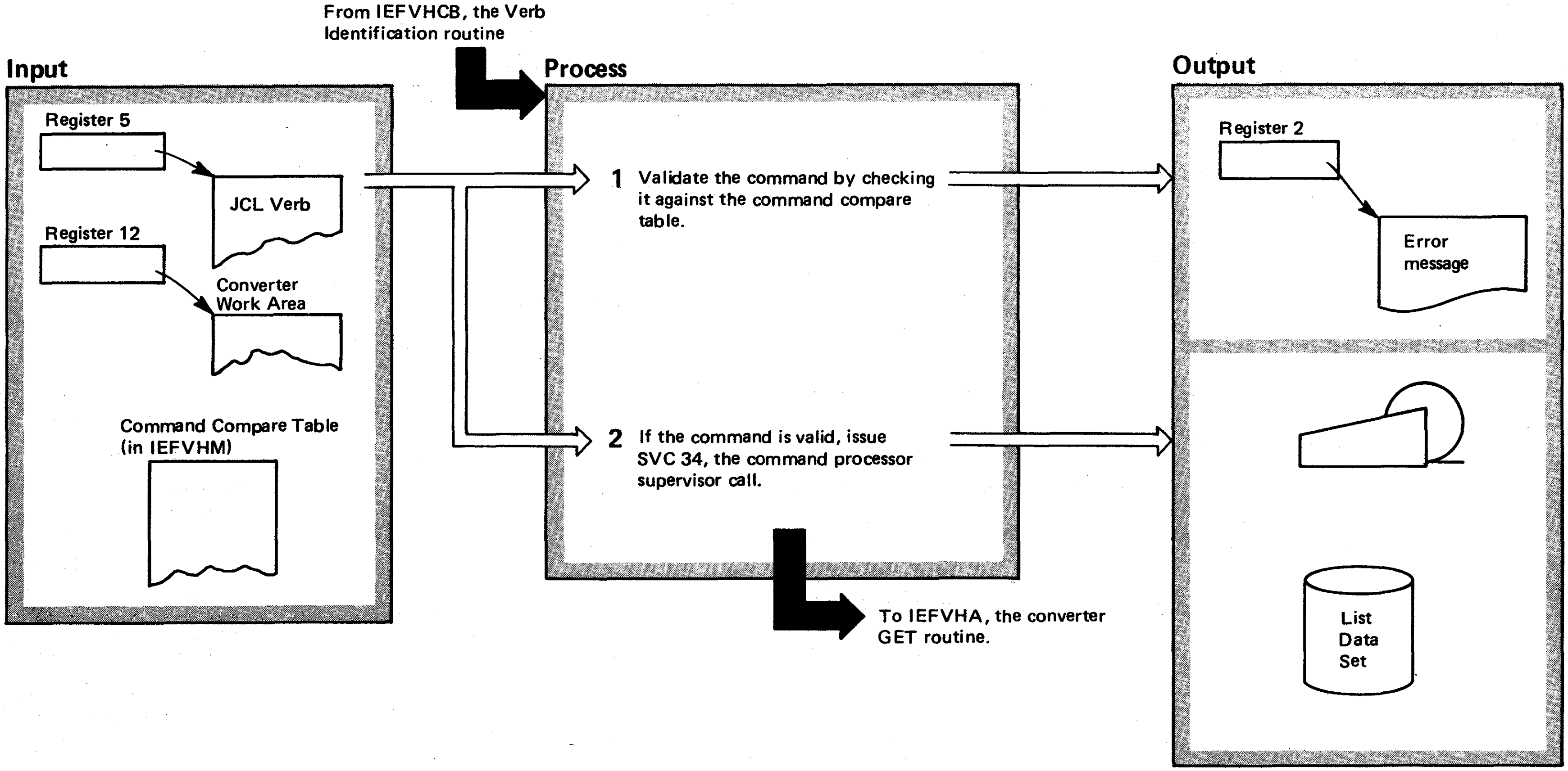
9 Perform SMF processing, if required.

10 Branch to appropriate routine.

Diagram 12-2. Converter: Identifying Verbs on JCL Statements (Part 4 of 4)

Extended Description	Module	Label
5 IEFVHCB merges JCL statements from the JCL data set and the procedure library as follows:	IEFVHCB	
When it encounters a procstepname.ddname in a DD statement from the JCL data set, it sets an indicator in its parameter list and continues normal processing. Each time IEFVHCB again receives control to examine a DD statement from the procedure library, it compares the procstepname from the JCL data set to the one from the library. If it finds a match, it uses the override information from the statement in the JCL data set to process the statement from the procedure library.		
If IEFVHCB does not find a match before it encounters the next EXEC statement from the JCL data set, it simply adds the DD statement with the override information to the other DD statements for the previous step.		
6 Whenever IEFVHCB recognizes a RESTART keyword parameter on a JCL statement and has previously found a SYSCHK DD statement, it sets an indicator in the converter work area before it passes control to IEFVHEB. IEFVHEB (see Step 8) continues processing JOB, EXEC, and DD statements.	IEFVHCB IEFVHEB	
7 If IEFVHCB has identified a NULL verb on a statement, it passes control to IEFVHL.		
If it has identified a PROC statement, it branches to IEFVINA.		
If it has not been able to identify the verb, it assumes the verb is a command and passes control to IEFVHM, the command verb validation routine. IEFVHM uses the print routine in IEFVHEB to print the command statement. (See Step 8.)	IEFVHM	
8 When the print routine in IEFVHEB has received control from IEFVHC or IEFVHM, it moves the JCL statement passed to it into an output buffer and branches to IEFVGM, the converter/interpreter message module. IEFVGM puts the statement into the list data set. The list data set contains all the JCL statements that must be printed on an output listing.	IEFVHEB	
When IEFVHEB has received control from IEFVHCB, it performs pre-scan processing as well printing the JCL statement.		
9 If necessary, IEFVHEB branches to an SMF user exit routine.	IEFVHEB	
10 When a comment has been completely processed, and there are more statements to be processed, IEFVHEB returns control to IEFVHC which returns to the GET routine, IEFVHA, for the next statement.	IEFVHEB	
Otherwise, IEFVHEB branches to IEFVFA, the post-scan routine for further processing.		

Diagram 12-3. Converter: Processing Commands in the Input Stream (IEFVHM) (Part 1 of 2)



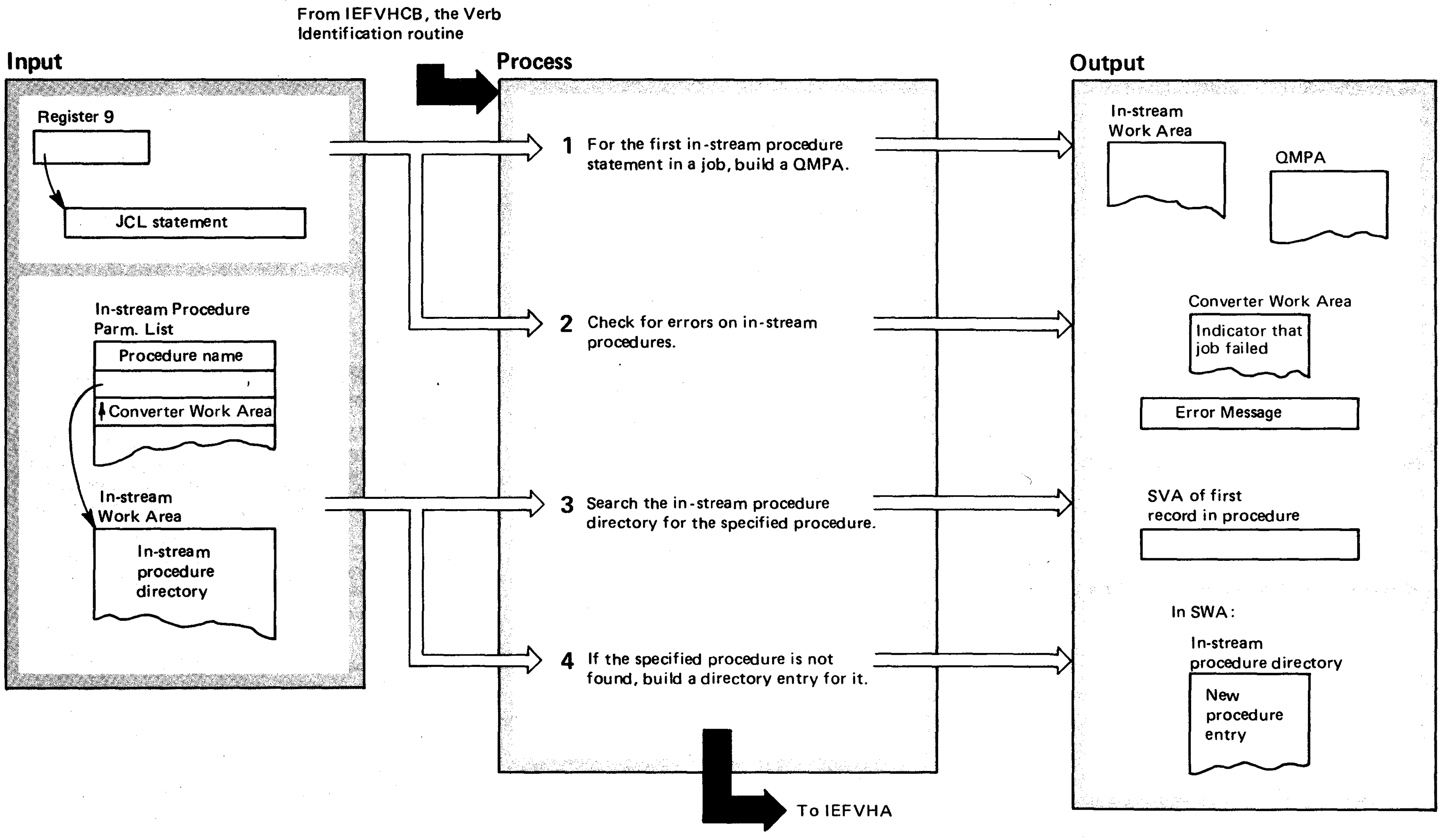
1 Validate the command by checking it against the command compare table.

2 If the command is valid, issue SVC 34, the command processor supervisor call.

Diagram 12-3. Converter: Processing Commands in the Input Stream (IEFVHM) (Part 2 of 2)

Extended Description	Module	Label
When the verb identification routine, IEFVHCB, is unable to recognize a verb on a JCL statement, it assumes the verb is a command and passes control to the command verb validation routine, IEFVHM.		
1 IEFVHM verifies that the command is one that is allowed in the input stream by checking it against a command compare table.	IEFVHM	
2 IEFVHM checks a disposition associated with the command in the interpreter entrance list (NEL). If the disposition is 0, IEFVHM causes the command to be executed by issuing an SVC 34. If the disposition is 1, IEFVHM writes the command into the list data set by branching to IEFVGM, then it displays the command to the operator by branching to the operator message module, IEFVHR; finally, IEFVHM executes the command by issuing SVC 34. If the disposition is 2, IEFVHM displays the command to the operator and requests his authorization to execute the command. When the operator replies in the affirmative, IEFVHM executes the command by issuing SVC 34. If the disposition is 3, IEFVHM ignores the command. In any case, IEFVHM returns control to the GET routine, IEFVHA.	IEFVHM	

Diagram 12-4. Converter: Processing In-stream and Cataloged Procedures (IEFVINA) (Part 1 of 2)



1 For the first in-stream procedure statement in a job, build a QMPA.

2 Check for errors on in-stream procedures.

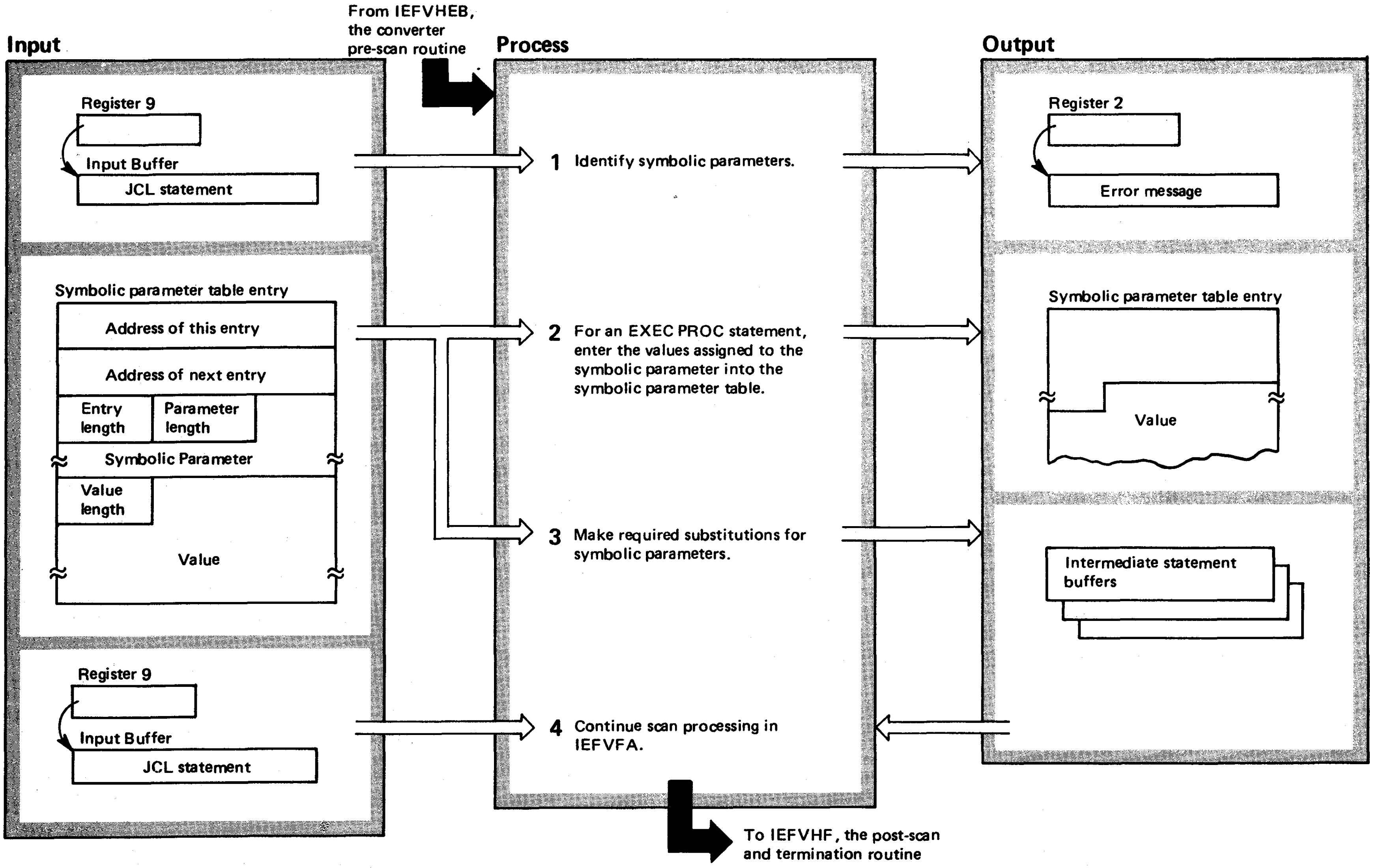
3 Search the in-stream procedure directory for the specified procedure.

4 If the specified procedure is not found, build a directory entry for it.

Diagram 12-4. Converter: Processing In-stream and Cataloged Procedures (IEFVINA) (Part 2 of 2)

Extended Description	Module	Label
IEFVINA is the control and GET routine for in-stream procedures. It receives control from IEFVHCB, the verb identification routine.		
1 When IEFVHCB encounters a PROC statement that is the first statement in an in-stream procedure, it obtains storage for a queue management parameter area (QMPA) and an in-stream procedure work area; then it branches to IEFVINA.	IEFVHCB	
2 When IEFVINA receives control, it in turn branches to IEFVINE, the in-stream procedure syntax check routine. This routine checks the validity of the label and operation fields in the PROC statement and passes a return code to IEFVINA.	IEFVINA IEFVINE	
If the return code is 0, 12, or 16, the PROC statement contains syntax errors and IEFVINA sets a job-fail indicator in the JCT and uses the message module, IEFVGM, to issue an appropriate error message.	IEFVINA IEFVCM	
If the return code is 8, there are no syntax errors in the PROC statement, and IEFVINA initializes the QMPA and checks the converter work area to determine if the procedure being processed is the first in-stream procedure in this job. If it is, control passes to IEFVINC, the in-stream procedure directory build routine. If it is not, the module builds a parameter list and branches to IEFVINB, the in-stream procedure directory search routine, instead.	IEFVINA IEFVINC IEFVINB	
3 IEFVINB scans the entries in the in-stream procedure directory searching for the procedure name specified in the PROC statement. When the procedure name is found, this routine obtains the SWA virtual address of the first record containing the procedure and places it in the return code field of its parameter list. If the procedure is not found, the routine sets a return code of zero in the parameter list before branching to IEFVINC, which will build a procedure directory entry.	IEFVINB IEFVINC	
4 IEFVINC enters the procedure name in the directory and invokes the SWA manager interface routine, IEFVHQ, to assign the entry to SWA. IEFVINC then takes the SWA address of the entry returned from IEFVHQ and places it in the directory next to the procedure name. IEFVINC returns to IEFVINA; IEFVINA then branches to IEFVHA for the next statement in the procedure.	IEFVINC	

Diagram 12-5. Converter: Processing Symbolic Parameters (IEFVFA and IEFVFB) (Part 1 of 2)



1 Identify symbolic parameters.

2 For an EXEC PROC statement, enter the values assigned to the symbolic parameter into the symbolic parameter table.

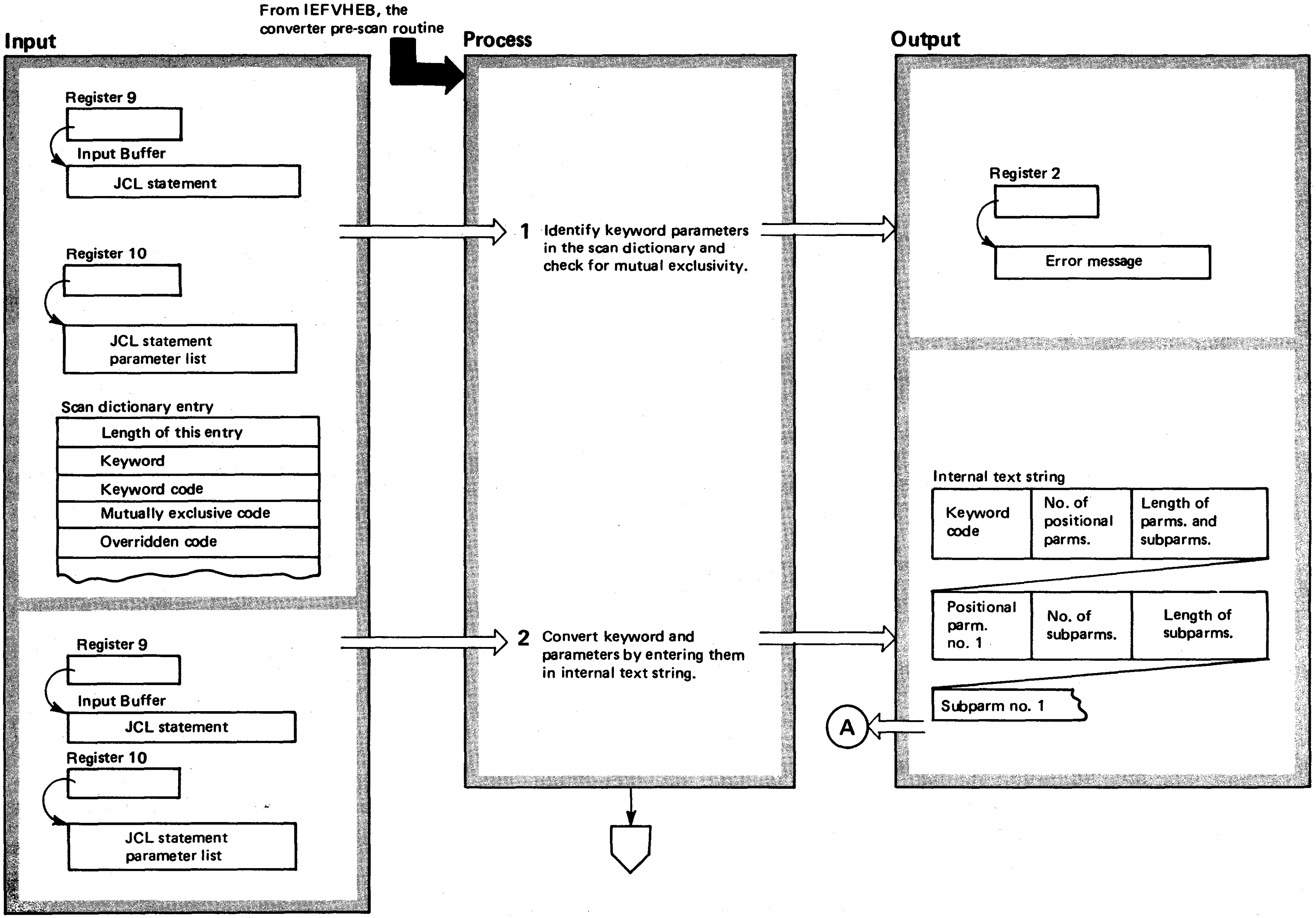
3 Make required substitutions for symbolic parameters.

4 Continue scan processing in IEFVFA.

Diagram 12-5. Converter: Processing Symbolic Parameters (IEFVFA and IEFVFB) (Part 2 of 2)

Extended Description	Module	Label
IEFVFA is the converter scan routine. It scans each JCL statement for syntax errors and, if necessary, uses IEFVGM, the message module, to issue an appropriate message. IEFVFA performs three other major functions: 1. Detecting symbolic parameters. 2. Converting JCL statements to internal text. 3. Default processing. This method-of-operation diagram describes the detection and processing of symbolic parameters. The other two functions are described in the two method-of-operation diagrams following this one.		
1 IEFVFA may encounter symbolic parameters in EXEC statements that call procedures, in input stream statements that override procedure statements, or in statements within a procedure. A symbolic parameter that appears in an EXEC statement that calls a procedure has the format of an EXEC statement keyword parameter. IEFVFA searches a scan dictionary for each JCL statement it processes. If it does not find a match for an EXEC statement keyword, it assumes that the keyword and its associated parameter are a symbolic parameter and its assigned value. A symbolic parameter that appears in an input stream statement that overrides a procedure statement, or in a statement in a procedure, is immediately preceded by an ampersand (&). Whenever IEFVFA detects a symbolic parameter, it branches to the symbolic parameter processor, IEFVFB.	IEFVFA	
2 For an EXEC PROC statement, IEFVFB verifies that the EXEC statement calls a procedure, then determines whether a symbolic parameter table has been initialized for this procedure. If not, the routine initializes one, and creates an entry containing the symbolic parameter and its value. If a symbolic parameter table has been initialized, IEFVFB searches it for an entry corresponding to the current symbolic parameter. If no such entry exists, the routine creates one; if an entry exists, the routine ignores the current assigned value.	IEFVFB	
3 For symbolic parameters in overriding statements or in a procedure, IEFVFB searches the symbolic parameter table for an entry that matches the current symbolic parameter. If it finds it, it substitutes the value assigned to the parameter in the table in an intermediate statement buffer. After making the substitution, IEFVFB invokes the message module, IEFVGM, to issue a substitution message.	IEFVFB	
4 In any case, IEFVFB returns to IEFVFA, which continues scanning the JCL parameters in the intermediate statement buffer. When IEFVFA has completed all processing, it branches to IEFVHF, the post-scan and termination routine.		

Diagram 12-6. Converter: Converting Statements to Internal Text (IEFVFA) (Part 1 of 4)



1 Identify keyword parameters in the scan dictionary and check for mutual exclusivity.

2 Convert keyword and parameters by entering them in internal text string.

Diagram 12-6. Converter: Converting Statements to Internal Text (IEFVFA) (Part 2 of 4)

Extended Description	Module	Label
IEFVFA is the converter scan routine. It scans each JCL statement for syntax errors and, if necessary, uses IEFVGM, the message module, to issue an appropriate message. IEFVFA performs three other major functions: 1. Detecting symbolic parameters. 2. Converting JCL statements to internal text. 3. Default processing. This method-of-operation diagram describes the conversion of JCL statements to internal text. The other two functions are described in two method-of-operation diagrams, the one preceding and the one following this one.		
1 As IEFVFA examines a statement, it looks up each keyword in its own scan dictionary. For each valid keyword, the scan dictionary entry corresponding to it contains a one-byte binary code for that keyword and a code for each keyword mutually exclusive with it. IEFVFA sets flags in the duplicate table in the converter work area for the codes corresponding to the mutually exclusive codes. Every time another keyword is encountered, its flag is checked in the duplicate table. If the flag is set, IEFVFA branches to IEFVGM to issue a mutually exclusive message.	IEFVFA	

Extended Description	Module	Label
2 IEFVFA converts keywords and parameters into internal text. Internal text contains the following information: • The keyword code. • The number of parameters for the keyword. • The length of the first parameter. • The parameter in EBCDIC. • The length of the next parameter, if any. • The next parameter, if any, in EBCDIC. If the keyword is comprised of parameters and subparameters, internal text contains this information: • The keyword code. • The number of parameters for the keyword. • The length of the first parameter. • The parameter in EBCDIC. • The number of subparameters. • The length of the first subparameter. • The subparameter in EBCDIC. • The length of the second subparameter. • The second subparameter in EBCDIC. The information in internal text varies with the number of parameters and subparameters.	IEFVFA	

Diagram 12-6. Converter: Converting Statements to Internal Text (IEFVFA) (Part 3 of 4)

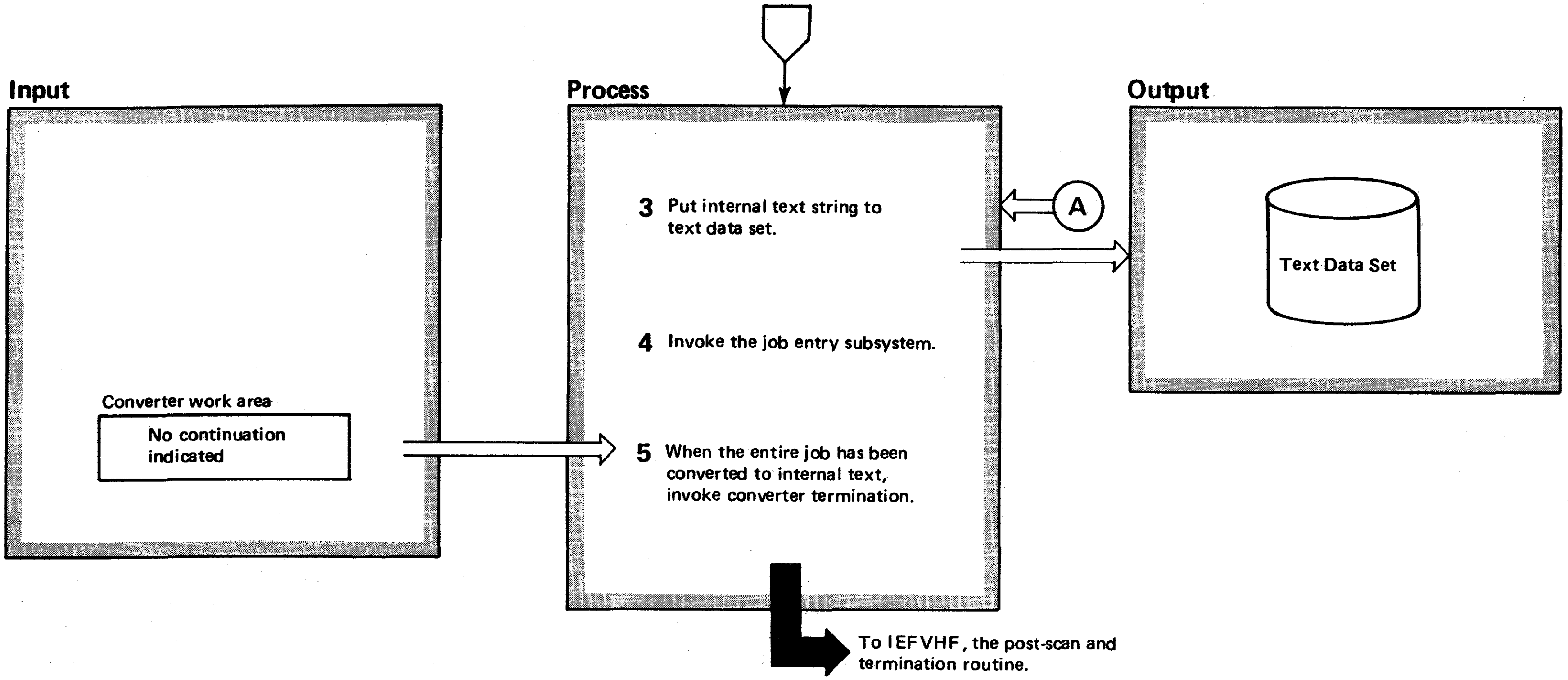
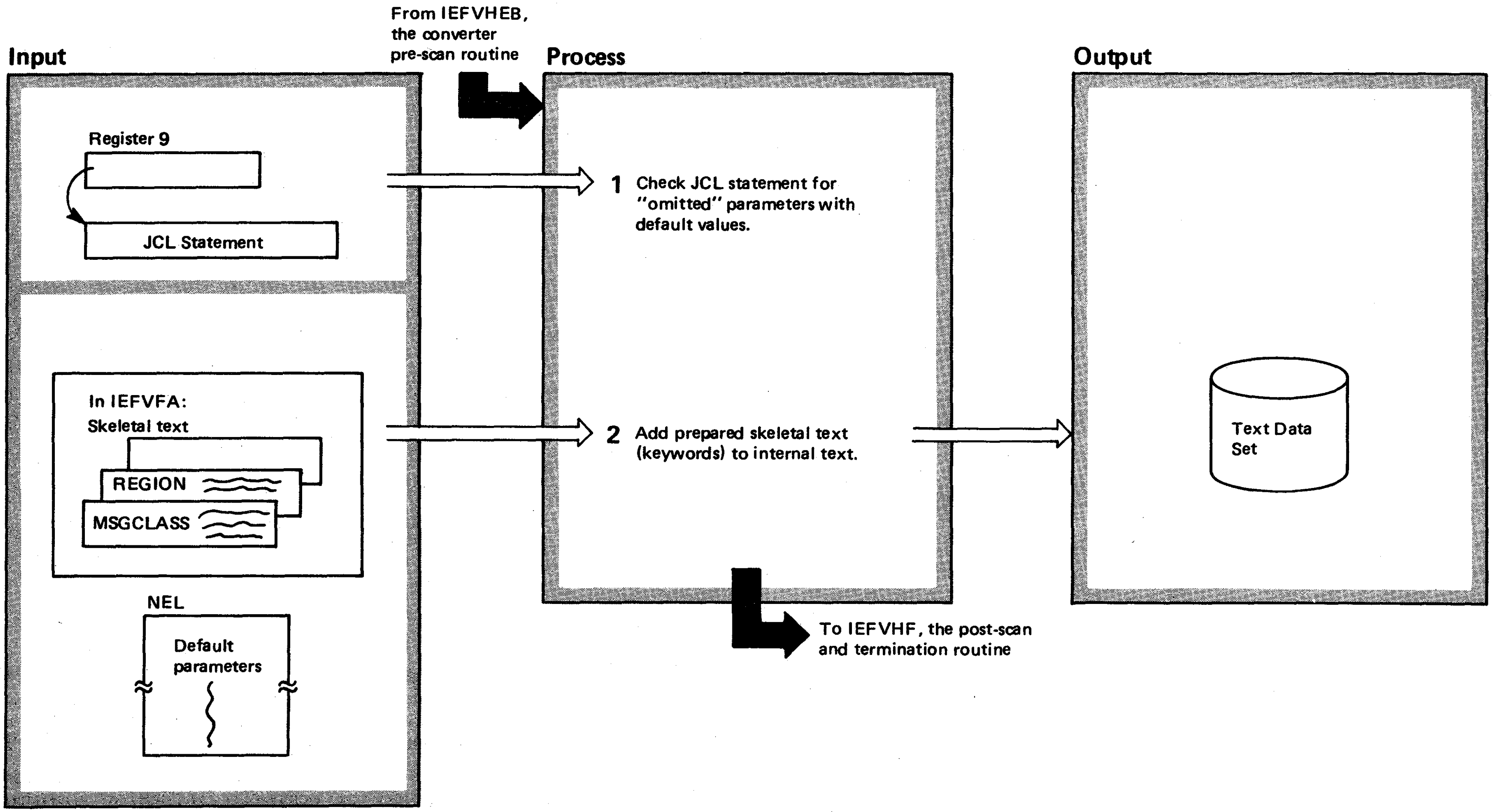


Diagram 12-6. Converter: Converting Statements to Internal Text (IEFVFA) (Part 4 of 4)

Extended Description	Module	Label
3 IEFVFA sets a flag in the converter work area to indicate that the current statement has been converted to internal text. Later on, IEFVHCB will check that flag and write the converted statement into the text data set before beginning pre-scan processing of the next statement.	IEFVFA IEFVHCB	
4 IEFVFA contains an interface to the job entry subsystem (JES). JES makes any required changes in the internal text string for SYSIN and SYSOUT processing and then returns control to IEFVFA.	IEFVFA	
5 When IEFVFA has completed all processing, it branches to IEFVHF, the post-scan and termination routine.	IEFVFA	

Diagram 12-7. Converter: Entering Defaults into Internal Text (IEFVFA) (Part 1 of 2)



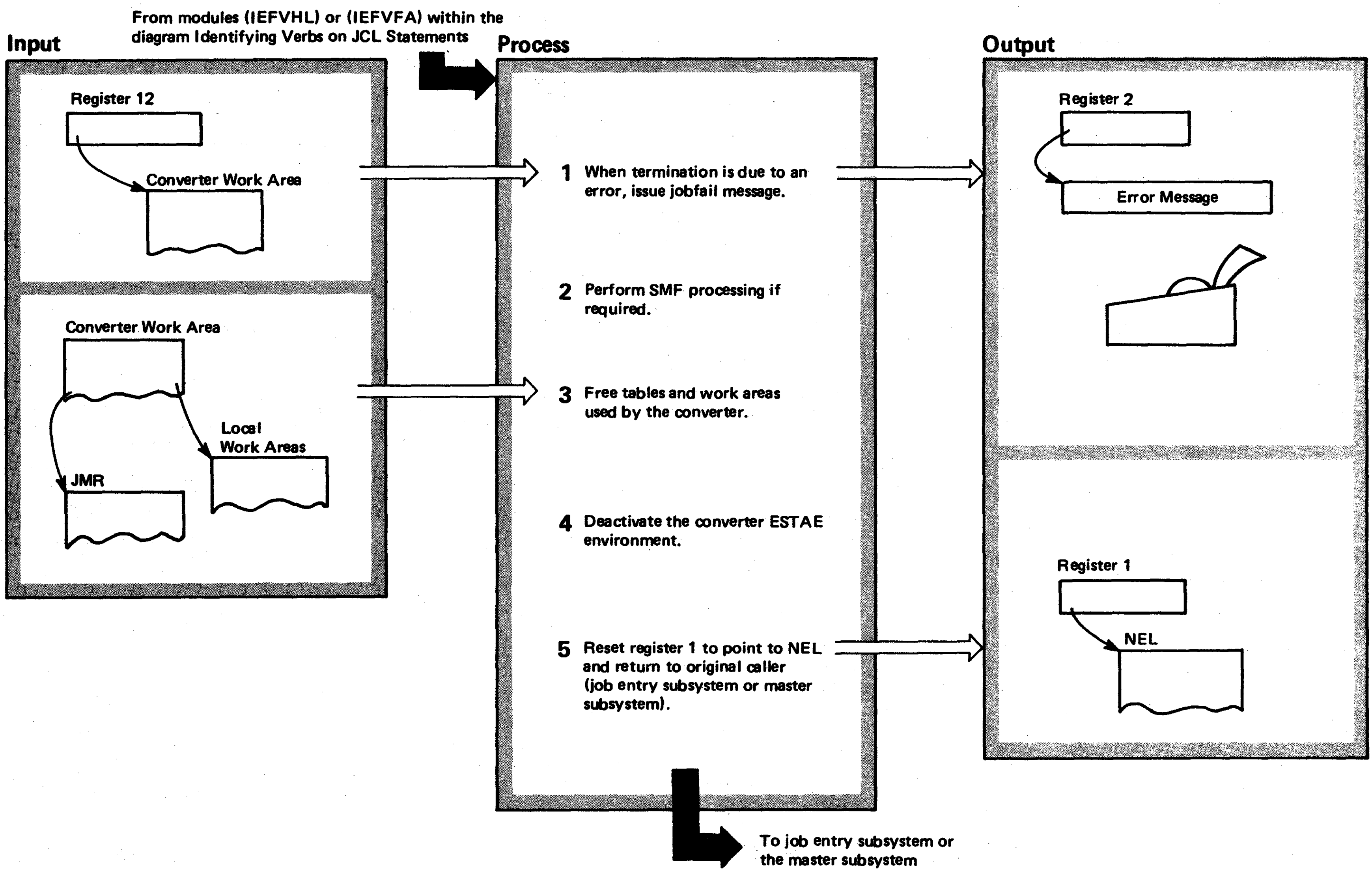
1 Check JCL statement for "omitted" parameters with default values.

2 Add prepared skeletal text (keywords) to internal text.

Diagram 12-7. Converter: Entering Defaults into Internal Text (IEFVFA) (Part 2 of 2)

Extended Description	Module	Label
IEFVFA is the converter scan routine. It scans each JCL statement for syntax errors and, if necessary, uses IEFVGM, the message module, to issue an appropriate message.		
IEFVFA performs three other major functions:		
1. Detecting symbolic parameters. 2. Converting JCL statements to internal text. 3. Default processing.		
This method-of-operation diagram describes default processing. The other two functions are described in the two method-of-operation diagrams immediately preceding this one.		
1 IEFVFA checks each JCL statement in the input buffer for omitted parameters that have default values assigned to them.	IEFVFA	
2 It appends skeletal text that represents the omitted keyword parameters to the JCL statement that has already been converted into internal text.	IEFVFA	
In addition to the keyword parameters, IEFVFA also places their associated default values obtained from a list in the NEL into the JCL statement. This completes default processing.		
When IEFVFA has completed all processing, it branches to IEFVHF, the post-scan and termination routine.		

Diagram 12-8. Converter: Termination (IEFVHF) (Part 1 of 2)



1 When termination is due to an error, issue jobfail message.

2 Perform SMF processing if required.

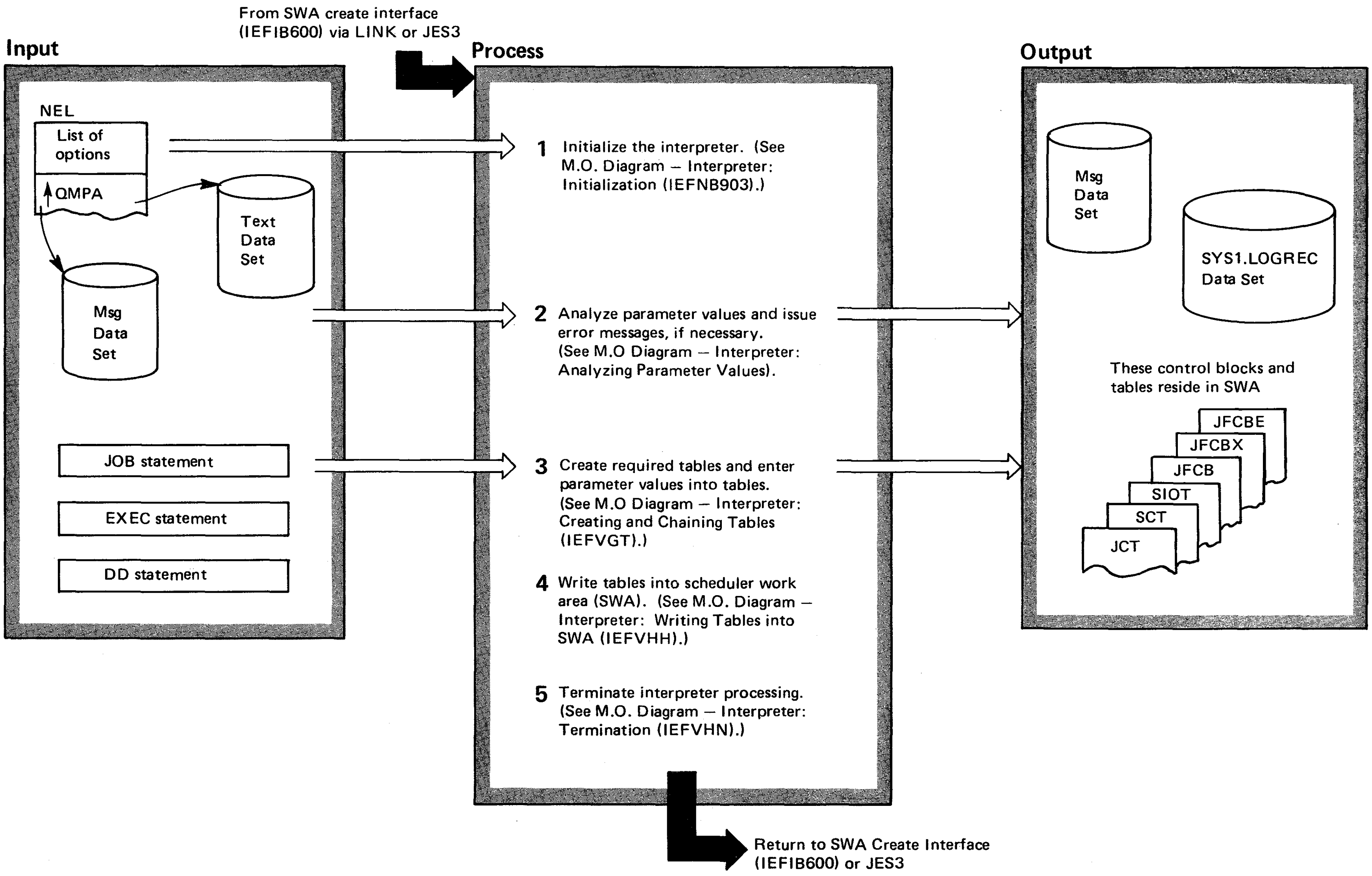
3 Free tables and work areas used by the converter.

4 Deactivate the converter ESTAE environment.

5 Reset register 1 to point to NEL and return to original caller (job entry subsystem or master subsystem).

Diagram 12-8. Converter: Termination (IEFVHF) (Part 2 of 2)

Extended Description	Module	Label
IEFVHF is the converter post-scan and termination routine; it receives control from IEFVHL when an end-of-file condition occurs on a procedure or on the JCL data set; it also receives control from IEFVFA when scan processing of a JCL statement has been completed.		
1 IEFVHF checks an indicator in the converter work area to see if a warning message has been issued during converter processing. If one has, this routine uses the operator message module, IEFVHR, to write a message to the operator.	IEFVHF	IEFVHF
2 If SMF processing is required, IEFVHF branches to an SMF user exit routine.	IEFVHF	IEFVHF
3 When control returns, IEFVHF frees the storage occupied by the JMR, local work areas, and the converter work area.	IEFVHF	IEFVHF
4 IEFVHF deactivates the ESTAE environment over the converter.	IEFVHF	IEFVHF
5 If an end-of-file condition exists and converter processing is to end, IEFVHF restores the pointer to the NEL in register 1 and returns to the job entry subsystem.	IEFVHF	IEFVHF
IEFVHA checks an indicator in the converter work area to see if more JCL statements must be read. If so, it branches to IEFVHA, the GET routine, for the next statement.		



1 Initialize the interpreter. (See M.O. Diagram – Interpreter: Initialization (IEFNB903).)

2 Analyze parameter values and issue error messages, if necessary. (See M.O Diagram – Interpreter: Analyzing Parameter Values).

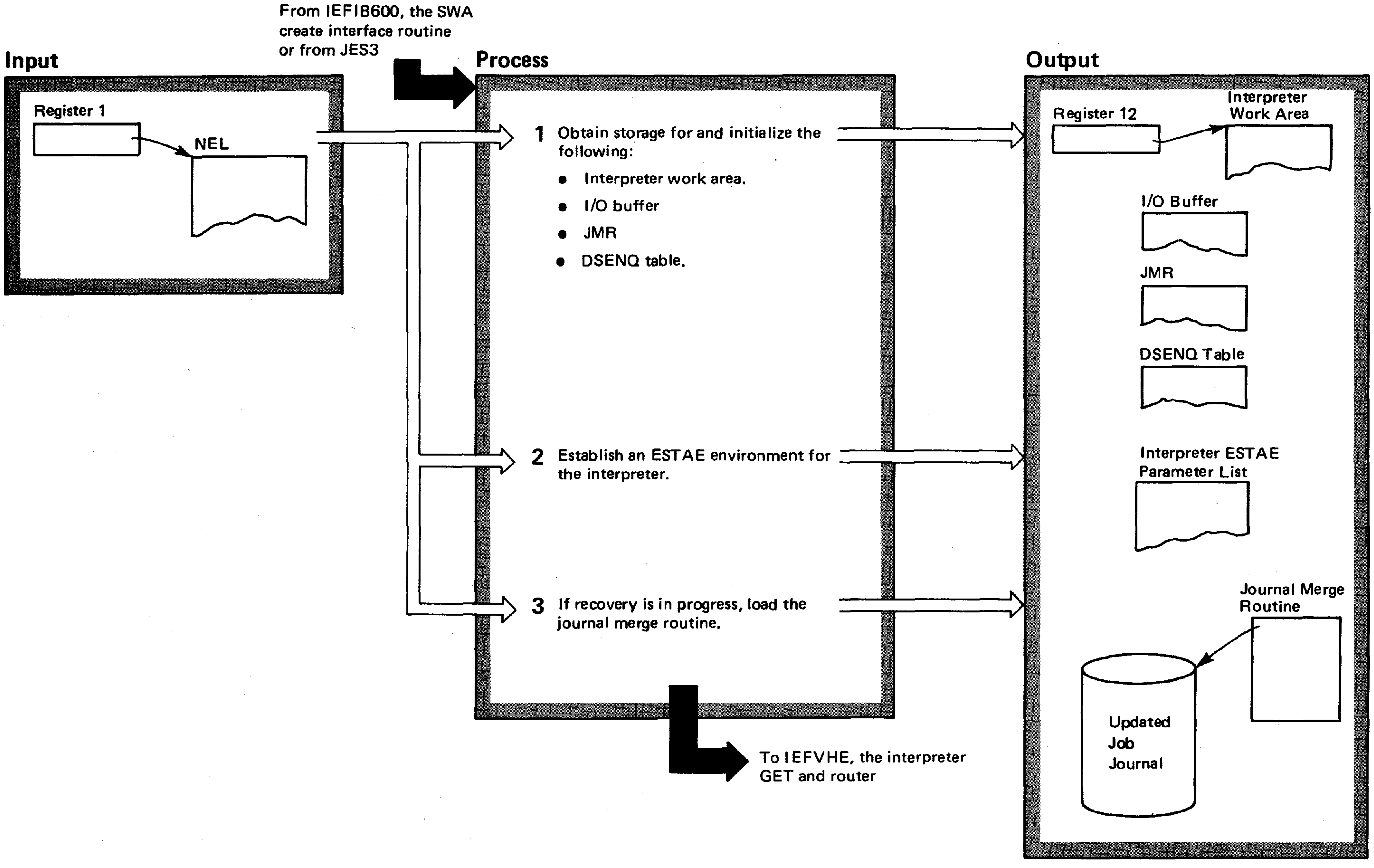
3 Create required tables and enter parameter values into tables. (See M.O Diagram – Interpreter: Creating and Chaining Tables (IEFVGT).)

4 Write tables into scheduler work area (SWA). (See M.O. Diagram – Interpreter: Writing Tables into SWA (IEFVHH).)

5 Terminate interpreter processing. (See M.O. Diagram – Interpreter: Termination (IEFVHN).)

Figure 2-17b. Interpreter Visual Contents

Diagram 12-9. Interpreter: Initialization (IEFNB903) (Part 1 of 2)



1 Obtain storage for and initialize the following:

- Interpreter work area.
- I/O buffer
- JMR
- DSENQ table.

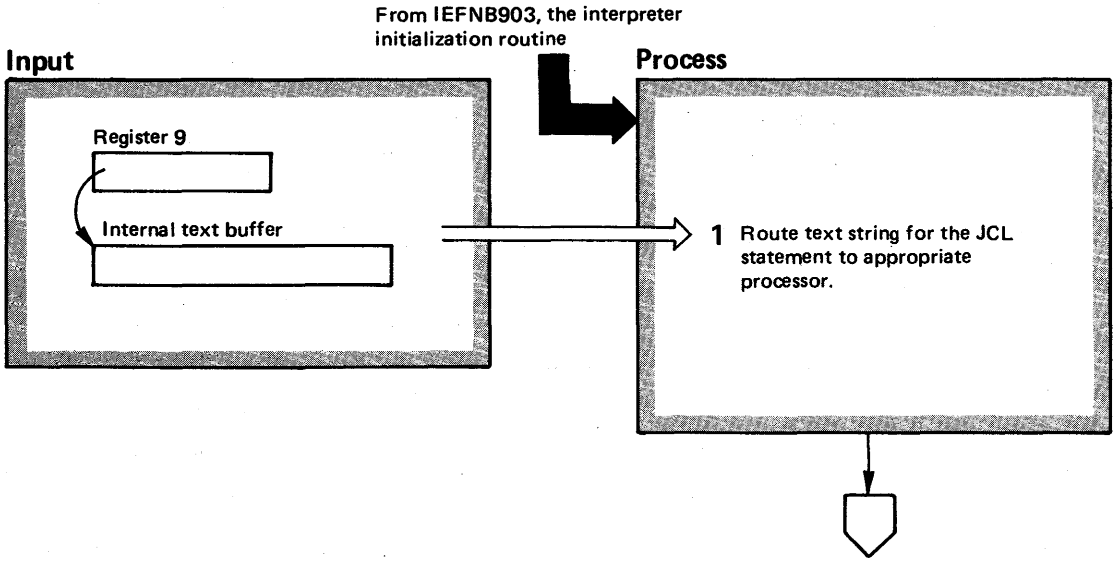
2 Establish an ESTAE environment for the interpreter.

3 If recovery is in progress, load the journal merge routine.

Diagram 12-9. Interpreter: Initialization (IEFNB903) (Part 2 of 2)

Extended Description	Module	Label
IEFNB903 is the interpreter initialization routine; it receives control form IEFIB600, the SWA create interface routine.		
1 IEFNB903 obtains storage for and initializes: • The interpreter work area. • The interpreter's I/O buffer. • The job management record (JMR). • Some local work areas. • A data set ENQUEUE (DSENQ) table. • A message buffer. This module sets a pointer to the interpreter work area in register 12 for the duration of interpreter processing.	IEFNB903	
2 IEFNB903 builds a parameter list for ESTAE processing and then issues an ESTAE macro instruction to enable the interpreter to recover from an error.	IEFNB903	
3 IEFNB903 checks an indicator in the NEL to determine if a recovery attempt is currently in progress. If it is, this module looks for the address of the journal merge interface routine in the NEL exit list. If it finds the address, it loads the routine and branches to it for journal merge processing. The journal merge interface routine returns to IEFNB903. In any case, when initialization processing is complete, IEFNB903 branches to IEFVHE, the interpreter GET and router routine.	IEFNB903	

Diagram 12-10. Interpreter: Analyzing Parameter Values (Part 1 of 4)



1 Route text string for the JCL statement to appropriate processor.

Diagram 12-10. Interpreter: Analyzing Parameter Values (Part 2 of 4)

Extended Description	Module	Label
Each of the JCL statement processors receives control from the interpreter GET and router routine, IEFVHE. The statement processors, IEFVJA, IEFVEA, and IEFVDA, first perform initialization functions, and then branch to IEFVGK, the GET parameter service routine. IEFVGK returns control to a keyword routine in the appropriate statement processor.		
The keyword routine branches to IEFVGT, the test and store service routine for parameter processing. IEFVGT returns to the keyword routine in the statement processor.		
IEFVJA is the JOB statement processor. It initializes a job control table (JCT) and the job account control table (ACT) for a job. It also checks the validity of the JOB statement keywords, and in some cases, their values.	IEFVJA	
IEFVEA processes EXEC statements. It creates a step control table (SCT) for each EXEC statement, SCT extensions for parameter information, and any required override or "refer-back" tables. IEFVEA also chains the step input/output tables (SIOTs) and the job file control blocks (JFCBs) for a JOBLIB or JOBCAT DD when they have been specified, and the SCTs and SIOTs for data set concatenations.	IEFVEA	
IEFVDA is the DD statement processor; it creates a DSENQ table entry for all data sets explicitly named by the DSNAME parameter. It then marks each data set entry as exclusive or shared according to the user's specifications. IEFVDA also creates SIOTs and JFCBs for a DD statement, JFCB extensions (JFCBX) for "VOL=SER" parameters, and JFCB extension (JFCBE) for JCL parameters (CHARS, BURST, MODIFY, FLASH) related to the 3800 printer.	IEFVDA	
IEFVHE is the interpreter GET and router routine; it receives control from IEFNB903, the interpreter initialization routine.		
1 IEFVHE gets each JCL statement from the internal text data set, identifies its verb as a JOB, EXEC, or DD, and branches to the appropriate statement processor: IEFVJA for JOB, IEFVEA for EXEC, or IEFVDA for DD. These three processor routines are similar in construction; each one consists of a single control section containing a header routine, a keyword routine for each keyword in the statement, a branch table of entries to keyword routines, a parameter descriptor table for each keyword, and a clean-up routine.	IEFVHE IEFVJA IEFVEA IEFVDA	
When a statement processor routine is first entered, the header routine performs initializing functions which include clearing the storage area occupied by the tables to be created by the routine (except for fields filled in by previously executed routines) and initializing the local work area. It then uses a BALR instruction to pass control to IEFVGK, the GET parameter routine.		

Diagram 12-10. Interpreter: Analyzing Parameter Values (Part 3 of 4)

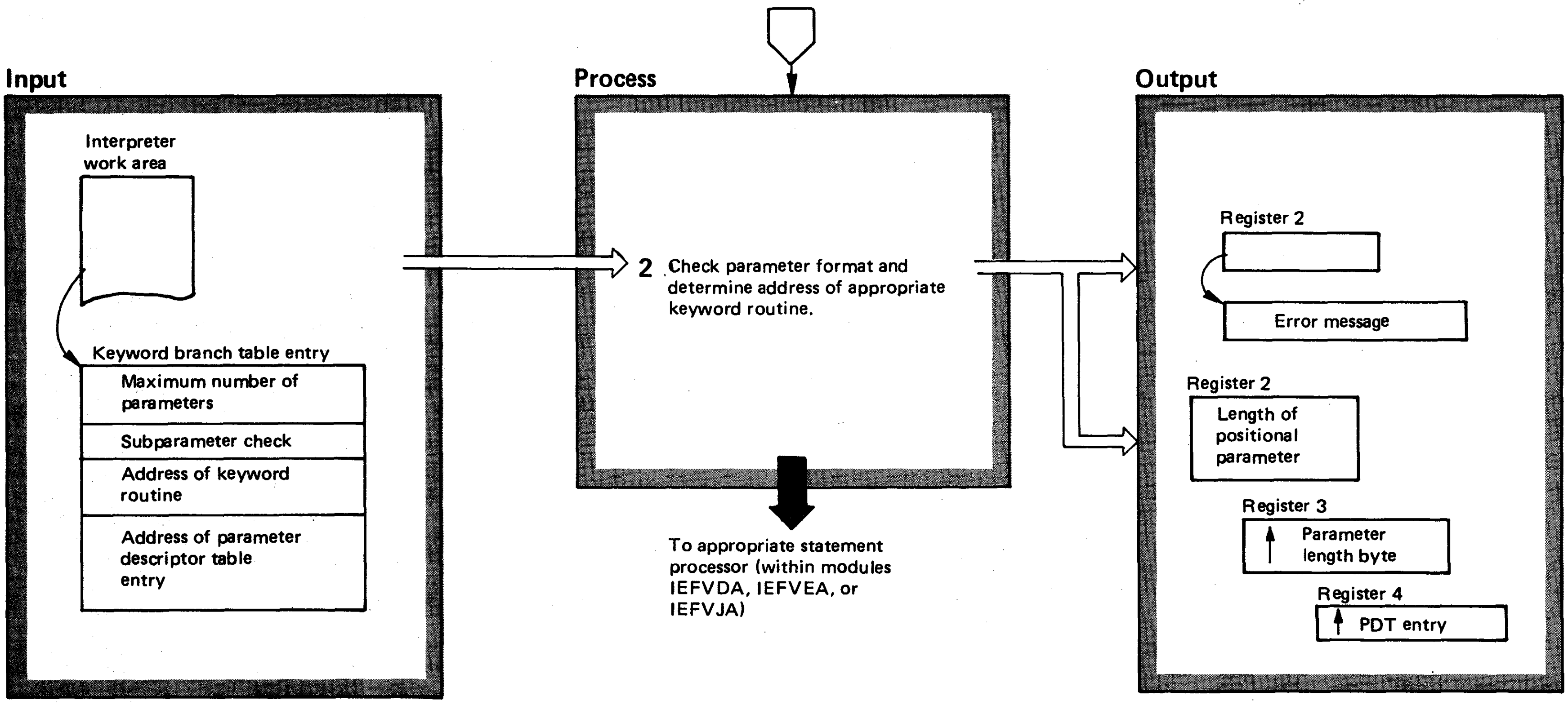
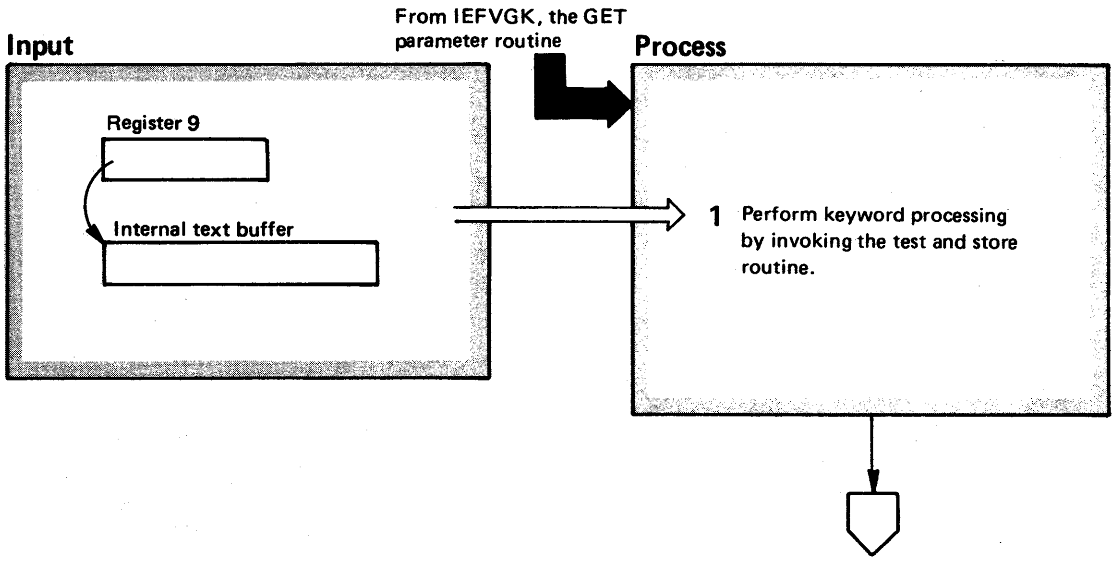


Diagram 12-10. Interpreter: Analyzing Parameter Values (Part 4 of 4)

Extended Description	Module	Label
2 The GET parameter routine is used by the JCL statement processor routines to find the next parameter in a statement, perform basic error checking of that parameter, and find and pass control to the appropriate keyword routine with pointers to the parameter and to the appropriate parameter descriptor table (PDT).	IEFVGK	
When IEFVGK is initially entered, the only non-zero portion of the interpreter work area is the address of the keyword branch table and the address of the processor cleanup routine. The keyword branch table is a table of offsets that allows the GET parameter routine to determine the actual main storage address of the appropriate keyword routine and parameter descriptor table entry. Additional fields in the table allow basic error checking to be done.		
When IEFVGK is entered to find the first parameter in a new statement, it extracts the base key (the key number that represents JOB, EXEC, or DD) from the internal text buffer and stores it. There are three sets of key numbers: one set for the JOB statement, one set for the EXEC statement, and one set for the DD statement. The base key, which corresponds to the verb in the statement, is the highest number in the set. It is the offset of the last entry in the table from the first entry. Whenever the routine is entered, it subtracts the current key from the base key, multiplies the result by 6 (the size of a keyword branch table entry), and adds the product to the machine address of the first entry in the table. The result is the machine address of the keyword branch table entry corresponding to the current keyword.	IEFVGK	
IEFVGK first finds the proper keyword branch table entry, then determines whether the maximum number of parameters for the keyword has been exceeded, and stores the subparameter check byte in its work area. Each bit in the subparameter check byte corresponds to a positional parameter; if the bit is on, it means that the corresponding parameter may have subparameters associated with it. For example, if the first positional parameter associated with a keyword were the only one that could consist of a subparameter list, the high-order bit in the field would be on. If the seventh and eighth positional parameters could have subparameters, the two low-order bits would be on.	IEFVGK	
The two offset fields are used to compute the actual main storage address of the appropriate keyword routine and of the appropriate parameter descriptor table entry; the positional parameter length, the parameter length byte address (in internal text) and the parameter descriptor table entry address are placed in general registers, and control is passed to the keyword routine in the appropriate statement processor.		
On subsequent entries to IEFVGK, the pointers are updated so that they point to the next operand (positional parameter or subparameter), and control is returned to the keyword routine at the instruction after the branch to IEFVGK. When the next keyword is encountered, however, the branch table is again used, and control is passed to a new keyword routine in the appropriate statement processor.		

Diagram 12-11. Interpreter: Creating and Chaining Tables (IEFVGT) (Part 1 of 4)



1 Perform keyword processing by invoking the test and store routine.

Diagram 12-11. Interpreter: Creating and Chaining Tables (IEFVGT) (Part 2 of 4)

Extended Description	Module	Label
1 The test and store routine, IEFVGT, is a service routine invoked by the statement processor keyword routine to determine the processing required for a parameter (as described in the parameter descriptor table), and to perform that processing.	IEFVGT	

The parameter descriptor table included in each JCL statement processor describes the processing to be done for each parameter that may be found in the statement. There is an entry for each keyword, which begins with a field containing the length of the keyword entry. The keyword entry is made up of positional parameter entries describing the processing to be done on the positional parameters associated with the keyword.

Each parameter entry contains two kinds of information: length and error checking information is followed by control information, which describes the functions to be performed on the parameter, and the tables (created above) and displacements in which the result is to be stored.

The first byte in each parameter entry (the parameter descriptor table length field) contains the length of the entry; the first half of the second byte (the control field length field) contains the length of the control information. The format of the remainder of the entry depends on the type of parameter and on the functions to be performed.

There are four types of parameters:

- A **required-format parameter** is a known string of characters. The first positional parameter following the DISP= keyword, for example, must be either "OLD", "NEW", "MOD", or "SHR". In this case, since there are four possibilities, there are four parts to the entry; the test and store routine compares the parameter to the constant in each of the four parts, and performs the function specified in the control information field of the part in which it obtained an equal compare.
- A **variable-format parameter** may be any string of characters up to a known maximum length.
- A **no-action parameter** specifies a default option.
- An **unconditional-action parameter** indicates that the presence of the parameter requires that the same functions be performed regardless of the form or content of the parameter.

Diagram 12-11. Interpreter: Creating and Chaining Tables (IEFVGT) (Part 3 of 4)

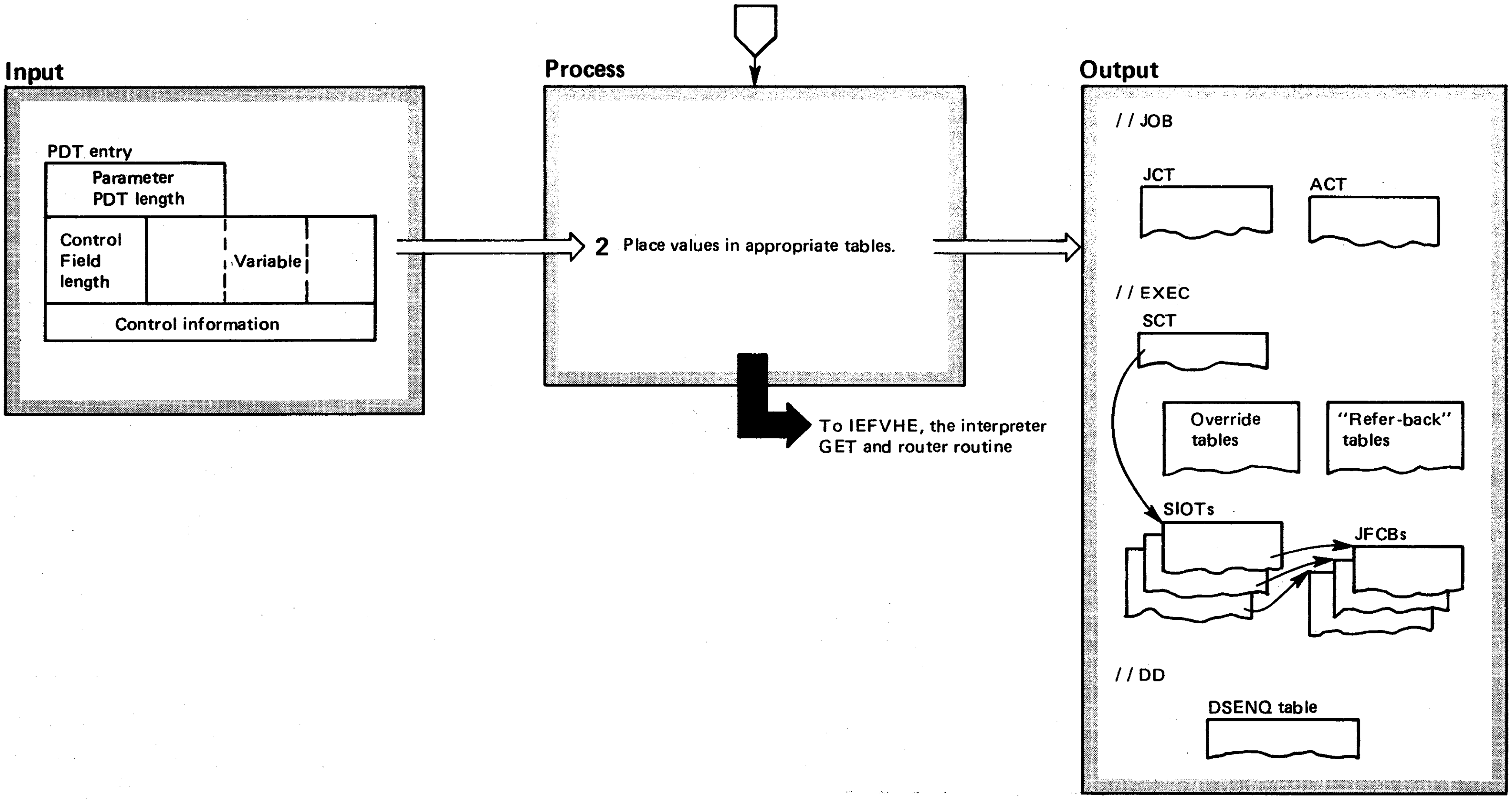


Diagram 12-11. Interpreter: Creating and Chaining Tables (IEFVGT) (Part 4 of 4)

Extended Description **Module** **Label**

2 The control information portion of a parameter PDT entry defines the operations to be performed when the parameter is processed, specifies the location in which the results are to be stored, and may contain data to be used in the operation. The control information portion may be up to 15 bytes in length; it consists of the following fields: IEFVGT

- **Function:** The first four bits of a control information field contain a number from 0 to 7, which specifies one of the following operations:
  - **OR (Code 0):** A logical OR operation is performed, using the bit pattern field in the control information portion of the entry, against the bit pattern at the location specified by the table and offset fields.
  - **CVB1 (Code 1):** A convert to binary operation is performed and a maximum value check is made. The converted information is stored (right justified) in the one-byte field specified by the table and offset fields, and compared against the maximum value, which is right-justified in the third byte of the control information part of this entry.
  - **CVB2 (Code 2):** This operation is similar to CVB1, except that the result is right-justified in a two-byte field, and the maximum value is found right-justified in the third and fourth byte of the control information portion of the entry.
  - **CVB3 (Code 3):** This operation is similar to the CVB1 and CVB2 operations, except that the result is right-justified in a three-byte field, and the maximum value is found in the third, fourth, and fifth byte of the control information portion of the entry.
  - **AND (Code 4):** A logical AND operation is performed, using the bit pattern field in the control information portion of the entry against the bit pattern at the location specified by the table and offset fields.
  - **MVC (Code 5):** A move characters operation is performed, using the parameter length specification in the internal text buffer. The parameter is moved to the location specified in the table and offset fields in the entry.
  - **First Character Alpha Check and MVC (Code 6):** This function is similar to the MVC function, except that before the move is performed the first character of the parameter is inspected to insure that it is alphabetic.

Extended Description **Module** **Label**

  - **Alpha/Numeric Check and MVC (Code 7):** This function is similar to the MVC function, except that before the move is performed a character (a one character parameter) in the text buffer is inspected to determine whether it is alphabetic or numeric. IEFVGT
- **Table:** The second four bits of the control information portion of a parameter PDT entry contains a number between 0 and 15 that specifies the table in which the result of the operation is to be stored.

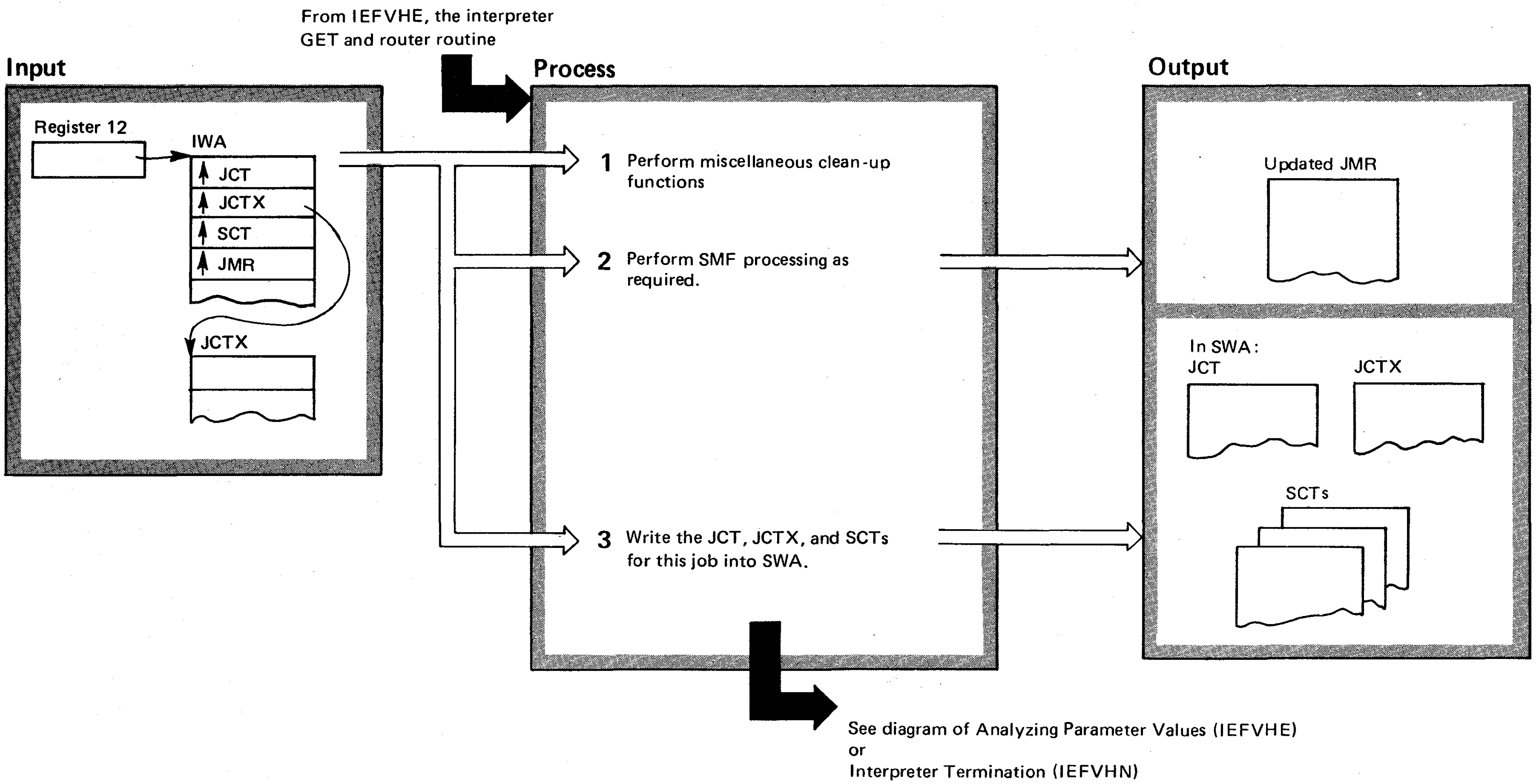
Code Number	Table
0	Local work area
1	Job control table (JCT)
2	Step control table (SCT)
3	Job account control table (ACT)
4	Step input/output table (SIOT)
5	Job file control block (JFCB)
6	JFCB extension (JFCBX)
7	Reserved
8	Data set name table (DSNT)
9	Refer-back Dictionary 1
10	Refer-back Dictionary 2
11	Procedure override table
12	Step account control table (ACT)
13	Reserved
14	Reserved
15	Interpreter work area

- **Offset:** The second byte of the control information of an entry contains the offset, from the beginning of the table, of the field in which the results of the operation are to be stored.
- **Bit Pattern/Maximum Number:** The third through fifth bytes of the control information portion of the entry are used for those operations that require data for logical or comparison functions. If the operation is AND or OR, the third byte contains the bit pattern. If the operation is a CVB operation, the third, fourth and fifth bytes contain the binary representation of the maximum value allowed for that parameter.

When IEFVGT has performed the functions described in the PDT, it returns to the keyword routine in the statement processor from which it received control.

Each statement processor determines that parameter processing for a JCL statement is complete. It then performs miscellaneous clean-up functions before returning to its caller, IEFVHE.

Diagram 12-12. Interpreter: Writing Tables into SWA (IEFVHH) (Part 1 of 2)



1 Perform miscellaneous clean-up functions

2 Perform SMF processing as required.

3 Write the JCT, JCTX, and SCTs for this job into SWA.

See diagram of Analyzing Parameter Values (IEFVHE)
or
Interpreter Termination (IEFVHN)

Diagram 12-11. Interpreter: Creating and Chaining Tables (IEFVGT) (Part 4 of 4)

Extended Description	Module	Label
2 The control information portion of a parameter PDT entry defines the operations to be performed when the parameter is processed, specifies the location in which the results are to be stored, and may contain data to be used in the operation. The control information portion may be up to 15 bytes in length; it consists of the following fields:	IEFVGT	

- **Function:** The first four bits of a control information field contain a number from 0 to 7, which specifies one of the following operations:
  - **OR (Code 0):** A logical OR operation is performed, using the bit pattern field in the control information portion of the entry, against the bit pattern at the location specified by the table and offset fields.
  - **CVB1 (Code 1):** A convert to binary operation is performed and a maximum value check is made. The converted information is stored (right justified) in the one-byte field specified by the table and offset fields, and compared against the maximum value, which is right-justified in the third byte of the control information part of this entry.
  - **CVB2 (Code 2):** This operation is similar to CVB1, except that the result is right-justified in a two-byte field, and the maximum value is found right-justified in the third and fourth byte of the control information portion of the entry.
  - **CVB3 (Code 3):** This operation is similar to the CVB1 and CVB2 operations, except that the result is right-justified in a three-byte field, and the maximum value is found in the third, fourth, and fifth byte of the control information portion of the entry.
  - **AND (Code 4):** A logical AND operation is performed, using the bit pattern field in the control information portion of the entry against the bit pattern at the location specified by the table and offset fields.
  - **MVC (Code 5):** A move characters operation is performed, using the parameter length specification in the internal text buffer. The parameter is moved to the location specified in the table and offset fields in the entry.
  - **First Character Alpha Check and MVC (Code 6):** This function is similar to the MVC function, except that before the move is performed the first character of the parameter is inspected to insure that it is alphabetic.
  - **Alpha/Numeric Check and MVC (Code 7):** This function is similar to the MVC function, except that before the move is performed a character (a one character parameter) in the text buffer is inspected to determine whether it is alphabetic or numeric. (Module: IEFVGT)
- **Table:** The second four bits of the control information portion of a parameter PDT entry contains a number between 0 and 15 that specifies the table in which the result of the operation is to be stored.

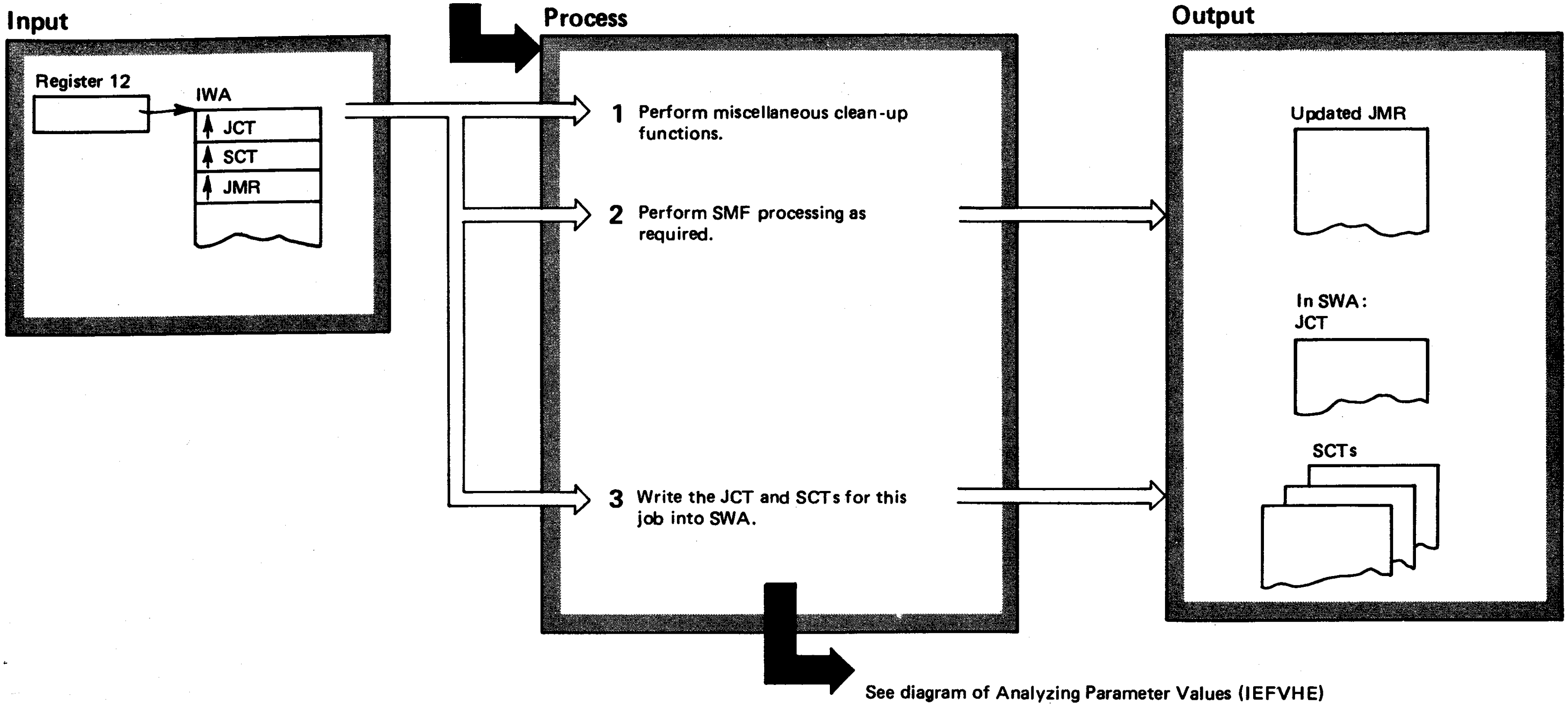
Code Number	Table
0	Local work area
1	Job control table (JCT)
2	Step control table (SCT)
3	Job account control table (ACT)
4	Step input/output table (SIOT)
5	Job file control block (JFCB)
6	JFCB extension (JFCBX)
7	Data set name table (DSNT)
8	Refer-back Dictionary 1
9	Refer-back Dictionary 2
10	Procedure override table
11	Step account control table (ACT)
12	JFCB extension for 3800 (JFCBE)
13	Reserved
14	Reserved
15	Interpreter work area

- **Offset:** The second byte of the control information of an entry contains the offset, from the beginning of the table, of the field in which the results of the operation are to be stored.
- **Bit Pattern/Maximum Number:** The third through fifth bytes of the control information portion of the entry are used for those operations that require data for logical or comparison functions. If the operation is AND or OR, the third byte contains the bit pattern. If the operation is a CVB operation, the third, fourth and fifth bytes contain the binary representation of the maximum value allowed for that parameter.

When IEFVGT has performed the functions described in the PDT, it returns to the keyword routine in the statement processor from which it received control.

Each statement processor determines that parameter processing for a JCL statement is complete. It then performs miscellaneous clean-up functions before returning to its caller, IEFVHE.

Diagram 12-12. Interpreter: Writing Tables into SWA (IEFVHH) (Part 1 of 2)


From IEFVHE, the interpreter GET and router routine

1 Perform miscellaneous clean-up functions.

2 Perform SMF processing as required.

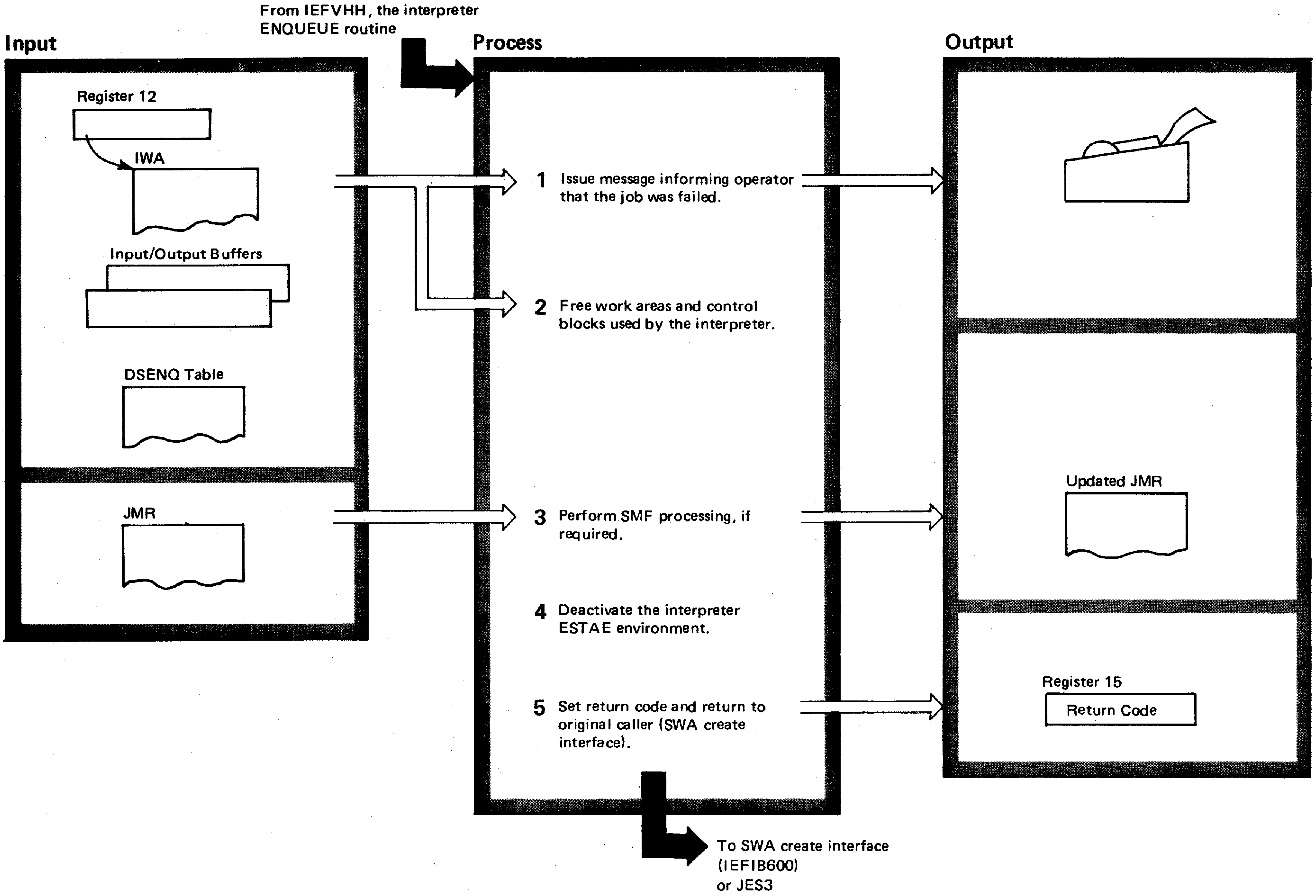
3 Write the JCT and SCTs for this job into SWA.

See diagram of Analyzing Parameter Values (IEFVHE) or Interpreter Termination (IEFVHN)

Diagram 12-12. Interpreter: Writing Tables into SWA (IEFVHH) (Part 2 of 2)

Extended Description	Module	Label
IEFVHH is called the ENQUEUE routine; it receives control from the interpreter GET and router routine, IEFVHE.		
1 IEFVHH ensures that all EXEC statement overrides have been completed at procedure end-of-file.	IEFVHH	
2 If SMF processing is required, IEFVHH branches to a user routine, and when it is returned control, it enters the time at which the interpreter stopped in the JMR.	IEFVHH	
3 Based on indicators it checks in the interpreter work area, IEFVHH branches to the SWA manager interface routine (IEFVHQ) to write the job and step tables into SWA. If the SCT was written, IEFVHH branches to IEFVHE to continue processing.	IEFVHH	
If the JCT and JCTX were written, IEFVHH branches to IEFVHN, the interpreter termination routine.		

Diagram 12-13. Interpreter: Termination (IEFVHN) (Part 1 of 2)



1 Issue message informing operator that the job was failed.

2 Free work areas and control blocks used by the interpreter.

3 Perform SMF processing, if required.

4 Deactivate the interpreter ESTAE environment.

5 Set return code and return to original caller (SWA create interface).

Diagram 12-13. Interpreter: Termination (IEFVHN) (Part 2 of 2)

Extended Description	Module	Label
IEFVHN is the interpreter termination routine; it receives control from IEFVHH.	IEFVHN	
1 If an error occurred during interpreter processing, IEFVHN uses IEFVHR the operator message module to issue an error message.		
2 IEFVHN frees the interpreter's input/output buffers and its local work area. If the SWA manager routines were loaded during interpreter processing, it deletes those.		
3 IEFVHN checks to see if SMF processing was performed by the interpreter. If it was, a user routine was used; IEFVHN deletes the user routine and writes the JMR updated by SMF into the calling routine's storage.		
4 IEFVHN deactivates the ESTAE environment over the interpreter.		
5 Finally, IEFVHN frees the DSENQ table and the interpreter work area. It then sets a return code in register 15 and returns to its caller, SWA create interface.		

SWA Manager

In MVS, to eliminate contention for job queue resources, both the job queue and the queue manager routines have been replaced. The scheduler control blocks for all jobs now reside on a pageable portion of virtual address space called the scheduler work area (SWA). To access SWA, system subcomponents must invoke a small set of routines called the SWA manager.

Figure 2-18 illustrates the general format of a control block in SWA, and an example of a specific control block, the JFCB, as it appears in SWA. All SWA blocks are preceded by prefixes.

The first field in the prefix contains a relative block number (RBN). The RBN enables the system to keep track of each job's SWA control blocks at various points during its execution. In the event that restart processing is necessary, the system can use the RBN to reconstruct the SWA for a restarted job.

The second field in the prefix indicates whether or not the prefix has been initialized with the appropriate control block ID and acronym.

The third field contains the SWA virtual address (SVA), a pointer back to the beginning of the prefix. This is for validity checking.

The fourth field is the SWA manager ID for the specific control block. The following is a list of SWA IDs and the associated control blocks:

00	JCT
01	ACT
02	SCT
03	SIOT
07	DSNT
0A	POT
0C	SCT extension table
0F	DSENQ
1B	JMR
1C	JFCB
1D	JFCX
20	PDID
21	PDIB
22	PDIQ
23	GDGN
25	IWAB
26	VUT
27	DDNT
28	AMPX
29	JFCE
30	JCTX

The fifth field contains the length of the control block.

The sixth field is the control block acronym.

The SWA manager routines, identified as load module, IEFQB550, consist of four object modules: two that actually perform SWA functions, IEFQB550 and IEFQB555, and two that intercept calls to previously existing queue manager routines.

The two function modules each operate in a different mode. IEFQB550 processes "move mode" requests from calling routines; "move mode" requests result in actual movement of data to or from control blocks residing in SWA. IEFQB555 performs "locate mode" operations for calling routines. A "locate mode" operation will return to the calling routine either a SWA virtual address (SVA) or a pointer to a SWA control block; no actual movement of data occurs.

The following is a list of possible move mode requests:

- ASSIGN results in initialization of a SWA control block in a SWA subpool. An ASSIGN request must be made for each control block that is to be initialized by the SWA manager.
- ASSIGN/START results in ASSIGN processing for a job that is just beginning.
- WRITE results in movement of data from the calling routine's buffer into a SWA control block.
- READ results in movement of data from a SWA control block into a calling routine's buffer.
- DELETE results in a FREEMAIN for a SWA subpool.
- WRITE/ASSIGN results in the movement of data into one SWA control block and the initialization of another block in SWA.

This is a list of valid locate mode requests:

- ASSIGN/LOCATE results in initialization of a SWA control block in a SWA subpool.
- WRITE/LOCATE causes a SWA control block to be updated.
- READ/LOCATE returns the address of the beginning of a SWA block, the block ID and the block length, to the calling routine.
- DELETE BLOCK results in a FREEMAIN for a SWA block.

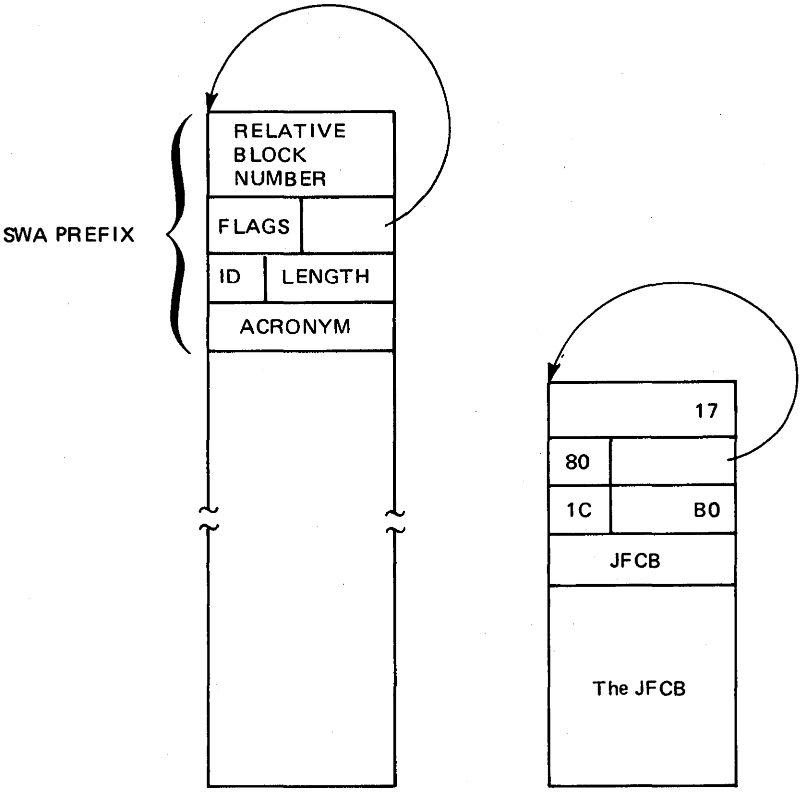


Figure 2-18. General Format of a SWA Control Block and an Example of the JFCB as it Appears in SWA

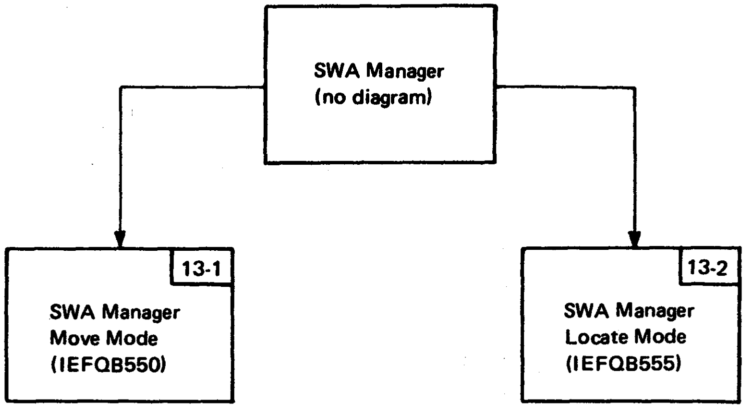
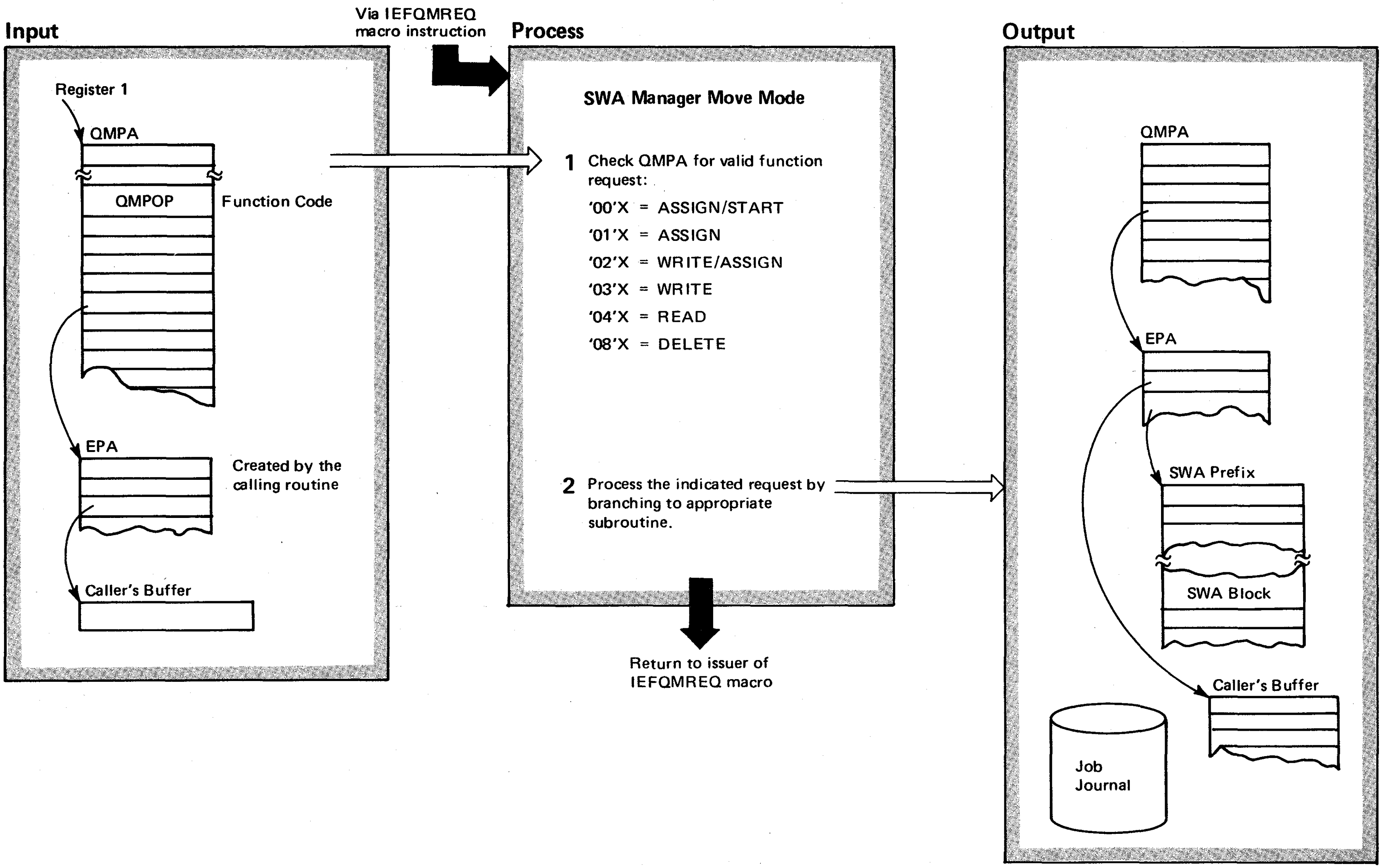


Figure 2-19. SWA Manager Visual Contents

Diagram 13-1. SWA Manager Move Mode (IEFQB550) (Part 1 of 2)



SWA Manager Move Mode

1 Check QMPA for valid function request:

'00'X = ASSIGN/START
'01'X = ASSIGN
'02'X = WRITE/ASSIGN
'03'X = WRITE
'04'X = READ
'08'X = DELETE

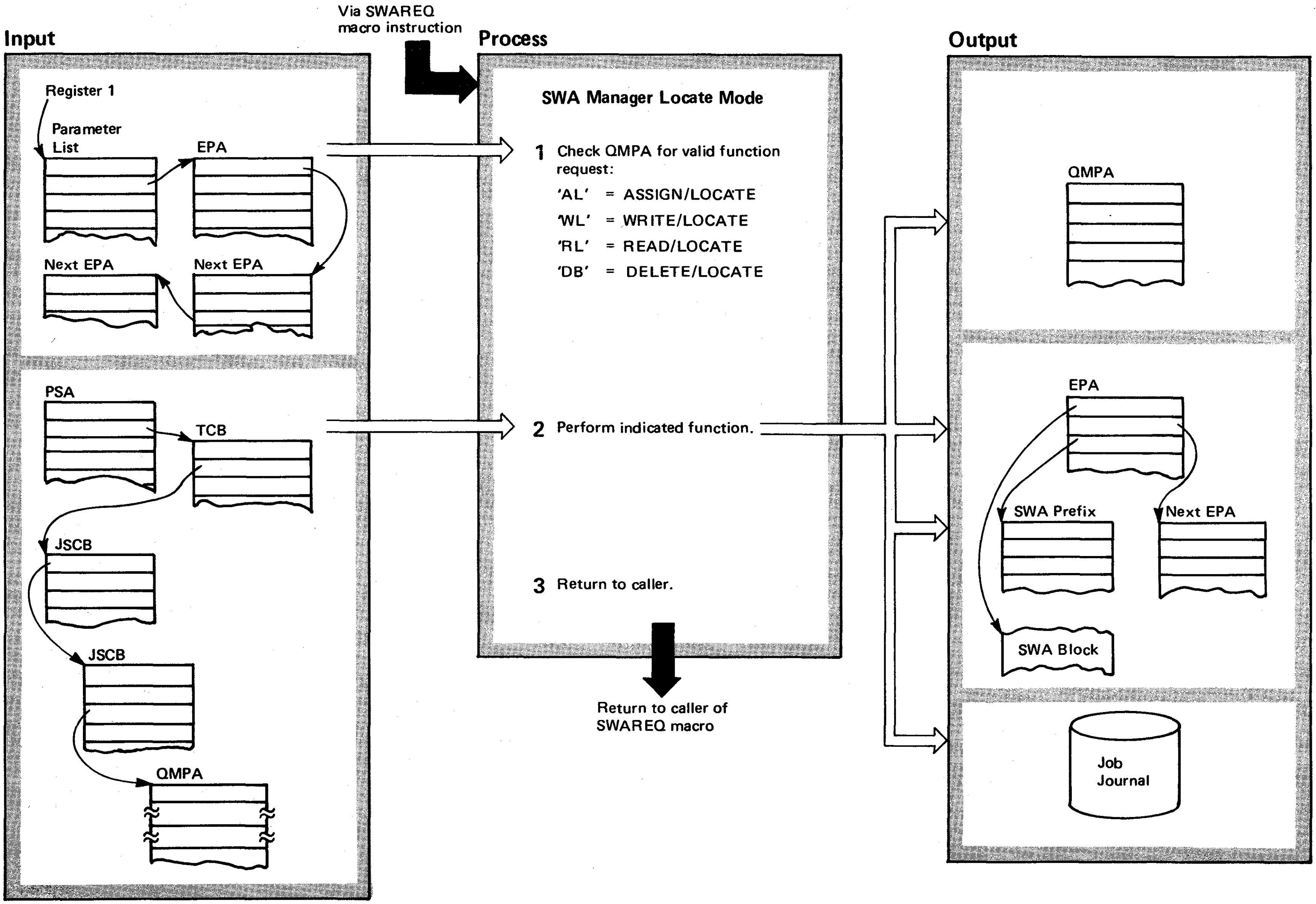
2 Process the indicated request by branching to appropriate subroutine.

Diagram 13-1. SWA Manager Move Mode (IEFQB550) (Part 2 of 2)

Extended Description	Module	Label
Control routine passes to the SWA manager move mode function either directly from a routine requesting move mode processing or from one of two modules that intercept calls to previously existing queue management routines.		
The first intercept module, IEFQB580, is the QMNGRIO macro interface handler. It first checks for valid input parameters in the parameter list pointed to by register 1.	IEFQB580	
The parameter list should contain a request for a READ or WRITE function. If it doesn't, IEFQB580 issues a 0B0 ABEND. When the input parameters are correct, IEFQB580 uses them to build and initialize a queue management parameter area (QMPA) and an external parameter area (EPA); it then invokes the move mode processor, IEFQB550.	IEFQB550	
When IEFQB550 completes processing, it returns control directly to the original calling routine.		
The other module that intercepts references to previous queue management routines is IEFQB585. Depending on the entry point supplied by the calling routine, IEFQB585 inserts an appropriate function code in the QMPA. This list outlines the possible entry points and their related functions.	IEFQB585	
IEFQBVMS IEFQMLK1 } ANY FUNCTION IEFQMSSS IEFQMRAW } READ or WRITE IEFQAGST ASSIGN/START IEFQASGQ } ASSIGN IEFQASGN IEFQDELQ } DELETE IEFQDELE		
IEFQB585 also calls IEFQB550. IEFQB550 returns control directly to the original calling routine.	IEFQB550	
1 If the calling routine has specified an invalid function, IEFQB550 places an appropriate error code in register 15 and issues a 0B0 ABEND.	IEFQB550	
2 For an ASSIGN/START request, IEFQB550 branches to a subroutine that sets the relative block number in the QMPA to 0 and then branches to the ASSIGN subroutine.	IEFQB550	
For one or more ASSIGN requests, the ASSIGN subroutine issues a GETMAIN macro instruction for 192 bytes from the SWA subpool specified in the QMPA (queue management parameter area).* It places the virtual address of the SWA storage in the external parameter area (EPA) and paritially initializes a SWA prefix for the new block in SWA storage.		
The ASSIGN subroutine repeats this entire process for as many ASSIGN requests as there are; when it has finished, it returns control to the calling module.		
For a WRITE request, the WRITE subroutine first determines whether the SWA virtual address (SVA) in the EPA is valid. If it is, it moves 176 bytes of data from the caller's buffer to the specified SWA control block, and then updates the SWA prefix if this is the first time the control block has been written. It repeats this process as many times as necessary and then calls a journal write routine to update these same SWA control blocks in the job journal. When that's done, control returns to the original calling routine.		
When a READ request is made, the READ subroutine checks for a valid SVA and then moves 176 bytes of data from a SWA block into the caller's buffer. It repeats the operation for each READ request and then returns control to the caller.		
The DELETE subroutine simply issues a FREEMAIN macro instruction for the subpool specified in the QMPA and then returns to the caller.		
If a WRITE/ASSIGN code is specified, the calling routine is requesting a WRITE for one SWA block and an ASSIGN for another block. IEFQB550 processes these requests sequentially by branching to the appropriate subroutine.		

* An important consideration in the assignment of SWA storage is the alternation of SWA subpools. During normal execution of problem programs, SWA consists of subpools 236 and 237. (During master scheduler initialization, the SWA subpool is 241.) The SWA blocks for a task attached by started task control (STC) routines reside in subpool 237; STC's own control blocks reside in 236. SWA blocks for a jobstep/task begun by the initiator are in subpool 236, while the initiator's own blocks remain in 237. The alternation of SWA subpools ensures that STC's control blocks, the initiator's control blocks, and the problem control blocks always exist in separate subpools.

Diagram 13-2. SWA Manager Locate Mode (IEFQB555) (Part 1 of 2)



SWA Manager Locate Mode

1 Check QMPA for valid function request:

'AL' = ASSIGN/LOCATE
'WL' = WRITE/LOCATE
'RL' = READ/LOCATE
'DB' = DELETE/LOCATE

2 Perform indicated function.

3 Return to caller.

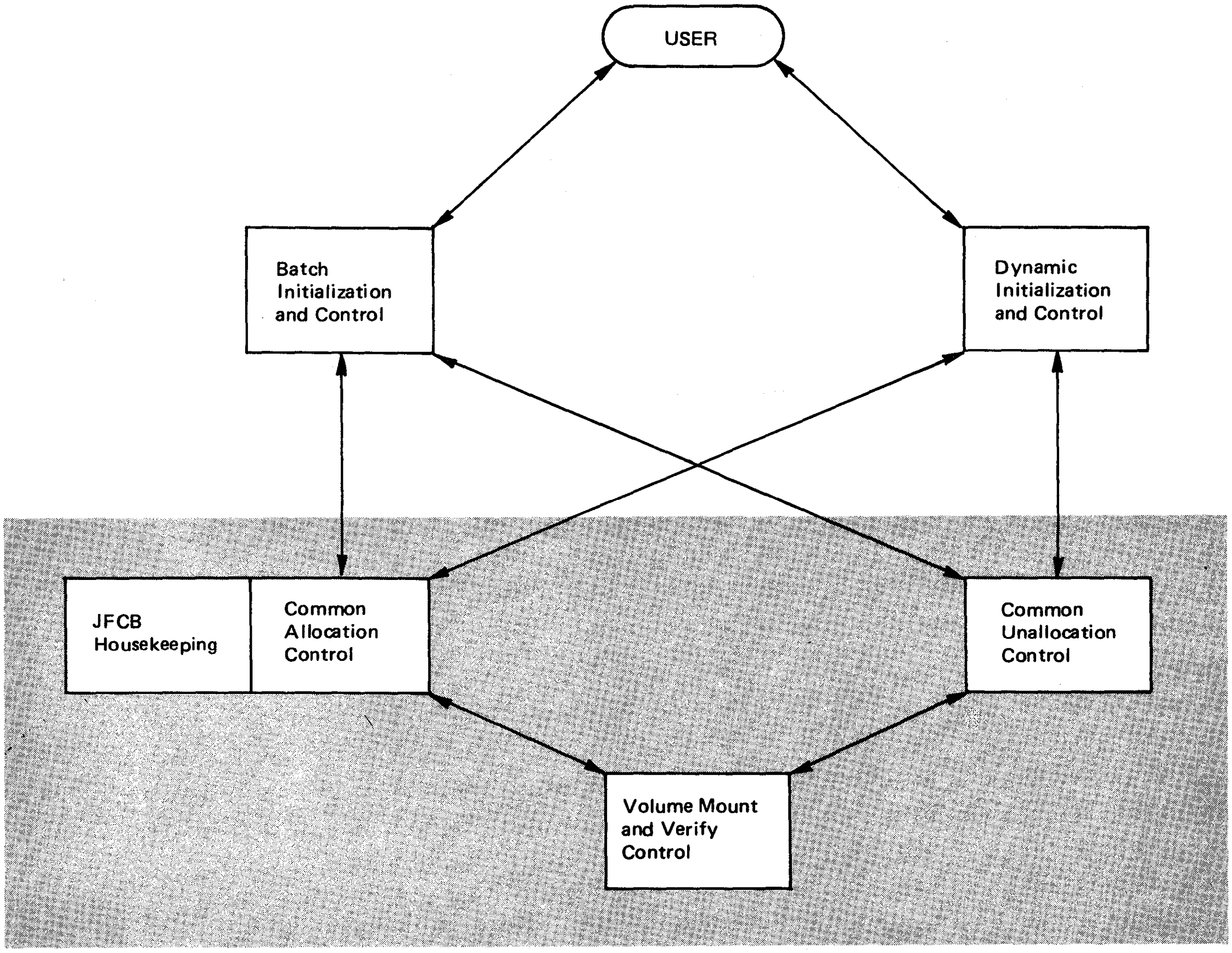
Diagram 13-2. SWA Manager Locate Mode (IEFQB555) (Part 2 of 2)

Extended Description	Module	Label
The SWA manager locate mode function IEFQB555 receives control from routines that issue a SWAREQ macro instruction.	IEFQB555	
1 IEFQB555 begins processing by checking for a valid function code in the second field of the parameter list passed by the calling routine. If the function code is invalid, IEFQB555 places an error code in register 15 and issues a 0B0 ABEND.		
2 If the calling routine requested an ASSIGN/LOCATE, IEFQB555 issues a GETMAIN macro instruction for 192 bytes of storage from the SWA subpool specified in the QMPA.* It places the SWA virtual address (SVA) of those 192 bytes in the EPA (external parameter area), increases the relative block number in the QMPA, and initializes a SWA prefix in the SWA storage it just obtained. IEFQB555 repeats this entire process for each ASSIGN request made by the caller. If the WRITE/LOCATE function was specified by the caller, IEFQB555 updates the SWA prefix as required and repeats the operation for each WRITE request. It also calls the journal write routine to copy the newly updated SWA blocks into the job journal. If READ/LOCATE was requested, IEFQB555 places the specified SWA block address, ID, and block length in the EPA. This enables the calling routine to directly address the SWA block, bypassing the SWA prefix. If DELETE/LOCATE was specified. IEFQB555 simply issues a FREEMAIN macro instruction for the SWA block.	IEFQB555	
3 When IEFQB555 has successfully completed processing, it places a zero return code in register 15 and returns to the calling module.		
*An important consideration in the assignment of SWA storage is the alternation of SWA subpools. During normal execution of problem programs, SWA consists of subpools 236 and 237. (During master scheduler initialization, the SWA subpool is 241.) The SWA blocks for a task attached by started task control (STC) routines reside in subpool 237; STC's own control blocks reside in 236. SWA blocks for a jobstep/task begun by the initiator are in subpool 236, while the initiator's own blocks remain in 237. The alternation of SWA subpools ensures that STC's control blocks, the initiator's control blocks, and the problem program control blocks always exist in separate subpools.		

Allocation/Unallocation can be divided into six major functions:

- Batch Initialization and Control, which is invoked by the initiator to provide allocation and unallocation functions for jobs and logons.
- Dynamic Initialization and Control, which is invoked by SVC 99 or the dynamic allocation interface routine (DAIR) to provide dynamic functions for both the foreground and background user.
- JFCB Housekeeping, which retrieves the information necessary for allocation.
- Common Allocation Control, which processes allocation requests, both batch and dynamic.
- Common Unallocation Control, which processes unallocation requests, both batch and dynamic.
- Volume Mount & Verify (VM&V) Control, which processes requests from Common Allocation and Common Unallocation to unload and/or mount and verify volumes.

The relationship of these functions is illustrated in Figure 2-20. Background information on these functions is presented in the following paragraphs Figure 2-21 lists the method-of-operation (M.O.) diagrams that describe each major function.



Note: Shaded area illustrates functions common to both batch and dynamic functions.

Figure 2-20. Relationship of the Six Major Functions of Allocation/Unallocation

Allocation/Unallocation Function	Related Method-Of-Operation Diagrams
Batch Initialization and Control	IEFBB401 – Initiator/Allocation Interface IEFBB410 – Initiator/Unallocation Interface IEFBB416 – Job Unallocation
Dynamic Initialization and Control	IEFDB4A0 – Dynamic Unallocation Control IEFDB400 – SVC 99 Control IEFDB410 – Dynamic Allocation Control IEFDB450 – Dynamic Concatenation IEFDB460 – Dynamic Deconcatenation IEFDB470 – Information Retrieval IEFDB480 – Remove In-use Attribute IEFDB490 – Ddname Allocation
JFCB Housekeeping	IEFAB451 – JFCB Housekeeping Control IEFAB454 – DD Function Control IEFAB469 – JLOCATE
Common Allocation Control	IEFAB421 – Common Allocation Control IEFAB430 – Fixed Device Control IEFAB433 – Specific Volume Allocation Control IEFAB434 – Allocate Request to Unit IEFAB436 – Nonspecific Volume Allocation Control IEFAB471 – Generic Allocation Control IEFAB476 – Allocation via Algorithm IEFAB479 – Demand Allocation IEFAB485 – Recovery Allocation IEFAB486 – Offline/Allocated Device Allocation IEFAB490 – Common Allocation Cleanup
Common Unallocation Control	IEFAB4A0 – Common Unallocation Control IEFAB4A2 – Disposition Processing IEFAB4A4 – Unit Unallocation
Volume Mount and Verify (VM&V)	IEFAB492 – Allocation/VM & V Interface IEFAB493 – VM&V Control

Figure 2-21. Allocation/Unallocation Functions and Related Method-of-Operation Diagrams

Batch Initialization and Control

Batch Initialization and Control, invoked by the initiator, provides allocation and unallocation functions for job/steps and logons. Common Allocation Control and Common Unallocation Control are called to process the allocation and unallocation requests; the processing performed by Batch Initialization and Control is basically preparation: issuing status messages, testing condition codes, building parameter lists for the common functions.

Dynamic Initialization and Control

Dynamic Initialization and Control, invoked by SVC 99 or the dynamic allocation interface routine (DAIR), provides dynamic functions for both the foreground and background user. Dynamic functions include: dynamic allocation, dynamic unallocation, dynamic concatenation, dynamic deconcatenation, information retrieval, removal of the in-use attribute, and ddname allocation. Common Allocation Control and Common Unallocation Control are called to process dynamic allocation and dynamic unallocation requests.

JFCB Housekeeping

JFCB Housekeeping is a common function, invoked by both Dynamic and Batch Initialization and Control when allocation requests are being processed. JFCB Housekeeping determines what additional data set information is needed to allocate each request, places the information in tables (SIOTs, JFCBs, and JFCBXs), and generates additional tables if necessary.

Common Allocation Control

Common Allocation Control, invoked by both Batch and Dynamic Initialization and Control, processes allocation requests. Common Allocation Control itself is divided into several functions; basically, each function processes a certain type of request or processes requests in a certain way. Each distinct function is presented in a separate method-of-operation diagram (listed in Figure 2-21 and illustrated in Figure 2-26). The basic philosophy of common allocation and background information on the more complex functions are presented in the following paragraphs.

Data Set Requests and Unit Requests

Data set requests are represented by SIOTs; each SIOT (that requires units) is represented by entries in a *volunit table* built by Common Allocation Control. The volunit table contains an entry for every possible unit that each request might need. It is these *volume/unit requests* (each identified by a volunit entry) that Common Allocation Control considers when it allocates requests — not the data set request as a whole.

For example, a data set was requested by means of the following DD statement:

```
//DYD    DD  DSN=DATA,DISP=OLD,
//             VOL=SER=(A,B,C),UNIT=(3330)
```

Three volunit entries are created for this data set request. The three volunit entries indicate unit affinity, which is implied by requesting more volumes than units. To allocate this data set, Common Allocation Control will allocate the three requests represented by the three volunit entries (even though, in total, only one unit is allocated).

Order of Processing Requests

To allow as many allocations as possible to process concurrently, Common Allocation Control is designed to minimize serialization between different allocations (that is, allocations for different users). (Serialization can be defined as sequential processing, as opposed to concurrent processing.) To accomplish this, Common Allocation Control processes requests in the following order:

1. Requests that do not require units and volumes to be allocated: dummy data set requests; VIO requests; subsystem (SYSIN or SYSOUT) data set requests. No serialization is required for this processing.
2. Requests that can be allocated to permanently resident or reserved volumes on direct access devices. Since these units are inherently shareable, serialization is required only with other system functions that might modify UCBs — for example, pending-unload processing and pending-offline processing. Fixed Device Control processes requests in this category; multiple allocations can occur concurrently.
3. Requests for teleprocessing devices. Serialization is required only with other allocations of teleprocessing devices and with other system functions that might modify

UCBs — for example, pending-unload processing and pending-offline processing.

4. Remaining requests. Since the units to be allocated are not inherently shareable, the processing of these requests must be serialized with other allocations and with other system functions that might modify UCBs. Generic Allocation Control, which is invoked by Common Allocation Control to try to allocate all remaining requests, minimizes this serialization by serializing only a subset of units. If all requests cannot be satisfied by Generic Allocation Control, Recovery Allocation is invoked; the units serialized by Generic Allocation that are still needed by unallocated requests remain serialized for Recovery Allocation. Both Generic Allocation Control and Recovery Allocation are described in greater detail in the following paragraphs.

Generic Allocation Control

Generic Allocation Control serializes only a subset of units; it processes only one generic device type at a time and, within that generic, it serializes only those units (**device groups**) needed by unallocated requests. (The order in which Generic Allocation Control selects device types to process is dictated by the installation device precedence list, established during system generation.)

Device Groups

Generic device types are divided into **device groups,** as illustrated in Figure 2-22. The existence of device groups allows an allocation to serialize on a subset of units **within** a generic. For example, using Figure 2-22, if 3330A is requested, the allocation needs to reserve only device group 4, rather than all 3330 devices. As a result, more than one allocation can process the same generic group as long as the allocations require different device groups within that generic.

The guidelines by which the system determines device groups are:

- If a user-assigned name (for example, SYSDA) includes different generic device types, the units in each generic belong to different device groups.
- If a user-assigned name (for example, 3330A) includes only a subset of the units in a generic, that subset is a distinct device group.
- The intersection of any subgroups is a distinct device group.

Note: For specific unit requests (that is, a unit address was specified), all device groups within a generic must be serialized.

Group Masks

Device groups are indicated in group masks, which are simply fields containing bit positions for all the device groups within all the generics in the system. There is a mask in the eligible device table (EDT; a sysgen table) for every possible unit grouping (either generic device type or user-assigned name). For example, the masks representing the unit groupings illustrated in Figure 2-22 would contain five bit positions, one for each device group. The group mask for each unit grouping would be:

2400	10000
2314	01100
3330	00011
SYSDA	00111
3330A	00010
3330B	00001

The masks are used to determine what device groups must be serialized and when serialized device groups can be released. Every data set request (represented by a SIOT) is associated with a list of the device types and devices to which it is eligible (that is, to which it can be allocated). This eligible device list (EDL) points to the mask(s) in the EDT for the unit group(s) to which it is eligible.

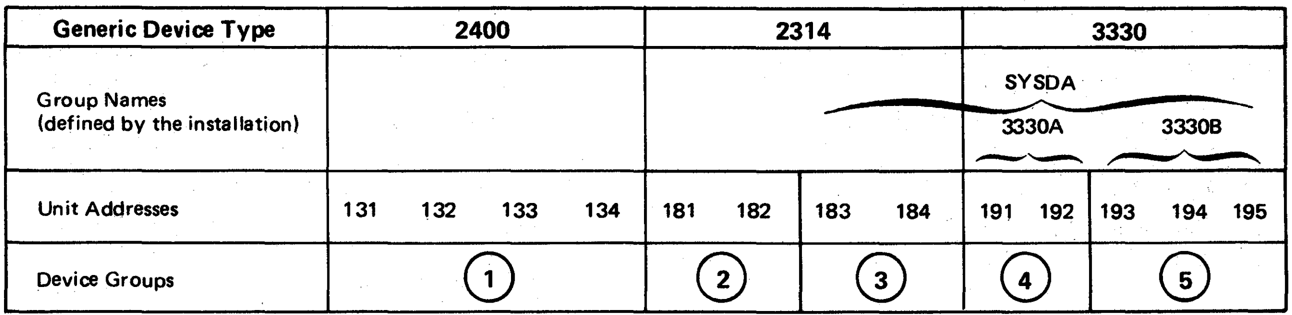


Figure 2-22. The Division of Generic Device Types into Device Groups

Recovery Allocation

Recovery Allocation receives control if all requests were not satisfied by or prior to Generic Allocation Control. (Recovery Allocation, however, will not be called if a retry situation was detected in Generic Allocation Control — see "The Retry Situation.") Recovery Allocation handles the following four situations:

- One or more tape requests could not be allocated because the needed volumes are mounted on a generic different from the generic selected for allocation. (For background information on tape requests, see "Processing Tape Requests.")
- Nonspecific DASD volume requests indicate volume affinity; although at least one request was successfully allocated, a subsequent request could not be allocated because of a DADSM error.
- Nonspecific tape or DASD requests could not be allocated to mounted volumes.
- Needed units are offline or allocated to another job.

Recovery Allocation results in one of the following situations:

- The retry situation is detected. (See the description under "The Retry Situation.")
- The allocation is failed because of operator intervention or an error detected by Recovery Allocation.
- All requests are satisfied; it is unnecessary to wait for units allocated to other users.
- All requests can be satisfied only by waiting for a needed device(s) to become available. If the operator authorizes, the allocation will wait for the needed device(s) to be unallocated, either with or without holding resources already allocated (as directed by the operator). If this allocation will wait **without** holding resources, Common Allocation Cleanup unallocates all requests successfully processed by Generic and Recovery Allocation and then calls Common Allocation Control to reattempt this allocation when the needed units are unallocated.

The Retry Situation

It is possible to encounter a situation called *retry*, in which a subset of the units on which a specifically requested volume may be mounted are serialized. Retry occurs when a request could be allocated if additional device groups were serialized, or if a different eligible generic (tape only) were being processed. For example (using Figure 2-22), a request specified 3330A, causing device group 4 to be serialized. The request, however, requires a volume currently mounted on a unit in device group 5. Because that device group is not serialized, the volume cannot be unloaded. This request is marked for retry.

Retry situations are handled by Common Allocation Cleanup; Common Allocation Cleanup unallocates all requests processed by Generic Allocation Control and by Recovery Allocation (if it was called) and then calls Common Allocation Control. For each request marked for retry, all device groups within the compatible generics will be serialized.

Retry situations are detected by Generic Allocation Control or Recovery Allocation. If a needed tape volume is mounted on a different generic device type, Generic Allocation does not determine if that generic is serialized — the request is marked for retry processing. See the following description of "Processing Tape Requests."

Processing Tape Requests

The dual density feature for tape devices allows a tape device to support more than one density. If tape device types support the same density, they are considered compatible; a tape volume can be mounted on different device types, as long as those device types are compatible. Figure 2-23 lists the tape device types and the densities they support. Using Figure 2-23, device type 2400-4 is compatible to device types 2400, 2400-3, 3400-3, 3400-4, and 3400-6, because they all support a common density; a tape volume that can be mounted on a 2400-4 can also be mounted on any of the compatible device types.

Note: Seven-track and nine-track tape devices are never compatible with each other, even though they might support a common density.

	Generic Device Type	Density
Nine-track tape device types	2400 2400-3 2400-4 3400-3 3400-4 3400-5 3400-6	800 bpi 1600 bpi 800 or 1600 bpi 1600 bpi 800 or 1600 bpi 6250 bpi 1600 or 6250 bpi
Seven-track tape device types	2400-1 2400-2 3400-2	200, 556, or 800 bpi 200, 556, or 800 bpi 200, 556, or 800 bpi

Figure 2-23. Tape Device Types and Supported Densities

Requested Generic Device Type	Generic Device Types Eligible to be Allocated to the Request
2400	2400, 2400-4, 3400-4
2400-1	2400-1, 2400-2, 3400-2
2400-2	2400-2, 3400-2
2400-3	2400-3, 2400-4, 3400-3, 3400-4, 3400-6
2400-4	2400-4, 3400-4
3400-2	3400-2
3400-3	3400-3, 3400-4, 3400-6
3400-4	3400-4
3400-5	3400-5, 3400-6
3400-6	3400-6

Figure 2-24. Tape Device Eligibility

Not all device types that are compatible, however, are eligible to satisfy a single request. Figure 2-24 illustrates the device types to which a request for a generic device type can be allocated. For example, a user requested a 2400-4 tape device for a data set. The data set can be allocated only to a 2400-4 or 3400-4 device, although a volume requested for this data set could be mounted on any of the compatible device types (2400, 2400-3, 2400-4, 3400-3, 3400-4, or 3400-6). An installation can define a user-assigned name that includes one or more tape device types. In this case, the eligible device types are only those included in the user-assigned name. For example, TAPEA includes all 2400-4 devices. A request that specifies TAPEA is only eligible to the 2400-4 devices. It is not eligible to all the devices that can be allocated to a request that specified the generic name 2400-4.

Tape requests are first processed by Generic Allocation Control. If Generic Allocation Control finds a needed tape volume mounted on a device type different from the device type selected for allocation, it determines if the volume is mounted on a compatible device type. If so, the request is marked for retry processing. Retry processing will mark the request for processing by Recovery Allocation. Otherwise, the request is failed.

Recovery Allocation determines if the volume is mounted on a device type eligible to the request. If it is, the request will be allocated where the volume is mounted. Otherwise, the volume must be unloaded. If the device group containing the required volume is not serialized, the request will be marked for retry. (Retry is described under "The Retry Situation.")

Common Unallocation Control

Common Unallocation Control, invoked by both Batch and Dynamic Initialization and Control, processes unallocation requests. Its functions include disposition processing, private catalog unallocation, data set release, unit unallocation, and volume release.

Volume Mount & Verify (VM&V) Control

Volume Mount & Verify (VM&V) Control processes requests from both Common Allocation and Common Unallocation to unload, to rewind, and/or to mount and verify volumes. (Common Unallocation calls VM&V Control only to unload and/or rewind volumes.) Volumes are rewound and/or unloaded as the need arises; volume mounting and verifying, however, is not performed until the end of Common Allocation Control (during Common Allocation Cleanup).

Allocation/Unallocation Module Name Conventions

Each allocation/unallocation module name has the following format:

```
IEF_B4__
```

The IEF indicates the routine is part of the scheduler; the B4 identifies the module as an allocation/unallocation module. (A few allocation modules begin with IEE; these allocation modules are part of the Master Scheduler.) The fourth character indicates the following:

- If A, the module is common to both batch and dynamic processing.
- If B, the module performs batch processing only.
- If D, the module performs dynamic processing only.

The last two characters are a unique module identifier.

Organization of Allocation/Unallocation Method-of-Operation Diagrams

Figure 2-25 illustrates the processing hierarchy of the diagrams for batch and dynamic processing; Figure 2-26, the processing hierarchy of common allocation diagrams. Figures 2-25 and 2-26 do not indicate all calls to each module represented by an M.O. diagram; they are intended only to provide a general structure to the M.O. diagrams.

The method-of-operation (M.O.) diagrams are arranged in alphabetic order according to the module name of the major module described by the diagram. As a result, diagrams for all functions common to both batch and dynamic processing (module titles of the form IEFAB4nn) precede the diagrams for batch only processing (module titles of the form IEFBB4nn), which in turn precede the diagrams for dynamic only processing (module titles of the form IEFDB4nn).

Selected Terms Used in Allocation/Unallocation

Following are definitions of selected terms that are not discussed in the preceding paragraphs but that have special meaning to allocation/unallocation.

demand request: a request that requires a specific unit; that is, a unit address was specified (for example, UNIT=190).

nonshareable request: a direct access request that might require demounting and that therefore must be allocated to a nonshareable device. A direct access request is considered nonshareable if more volumes than units were requested (implicit unit affinity), if DEFER was specified in the UNIT parameter, or if, in the case of explicit unit affinity, more than one volume will use the unit.

private request: (1) for tape requests, a request that specified PRIVATE; or that requires a specific volume; or that does *not* request a temporary data set. (2) for direct access requests, a request that specified PRIVATE. *(Note:* Storage requests can be changed to private requests if sufficient storage volumes are not available.)

public request: for both tape and direct access requests, a request that did not specify PRIVATE, that does not require a specific volume, and that requests a temporary data set.

Segment: functional division of code in a module.

scratch request: identical to public request.

specific volume request: a request that requires a particular volume; for example, a volume serial number was specified; the data set is passed; the data set is cataloged.

storage request: a direct access data set request that did not specify PRIVATE, that does not require a specific volume, and that is not temporary. *(Note:* Storage requests can be changed to private requests if sufficient storage volumes are not available.)

unit affinity: more than one request requires the same unit. Unit affinity can be either explicit (between data set requests when UNIT=AFF is specified) or implicit (within a single data set request when more volumes than units are requested).

volume affinity: more than one request requires the same volume. For example, two requests specified the same volume serial number or a request made volume reference to another request.

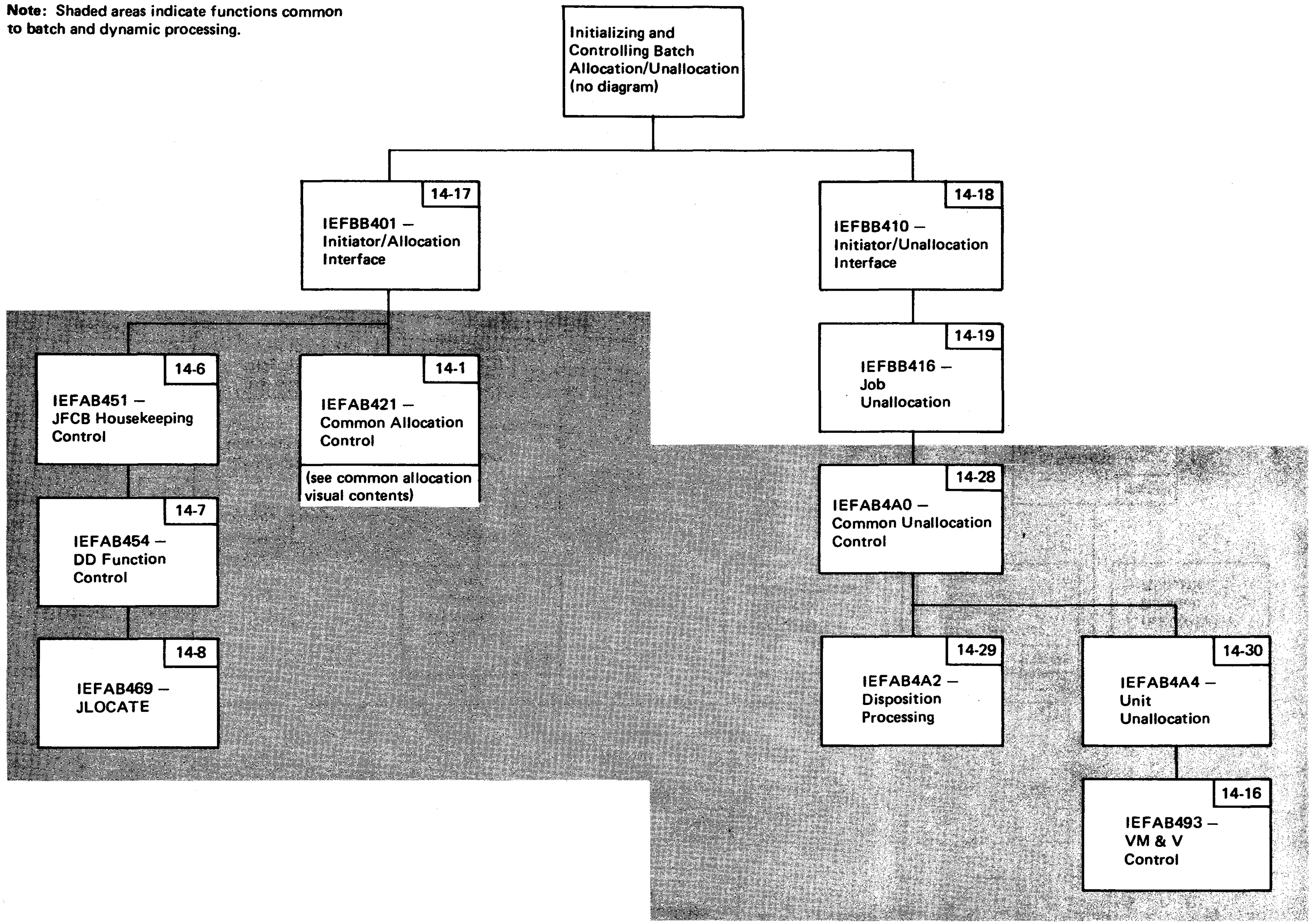
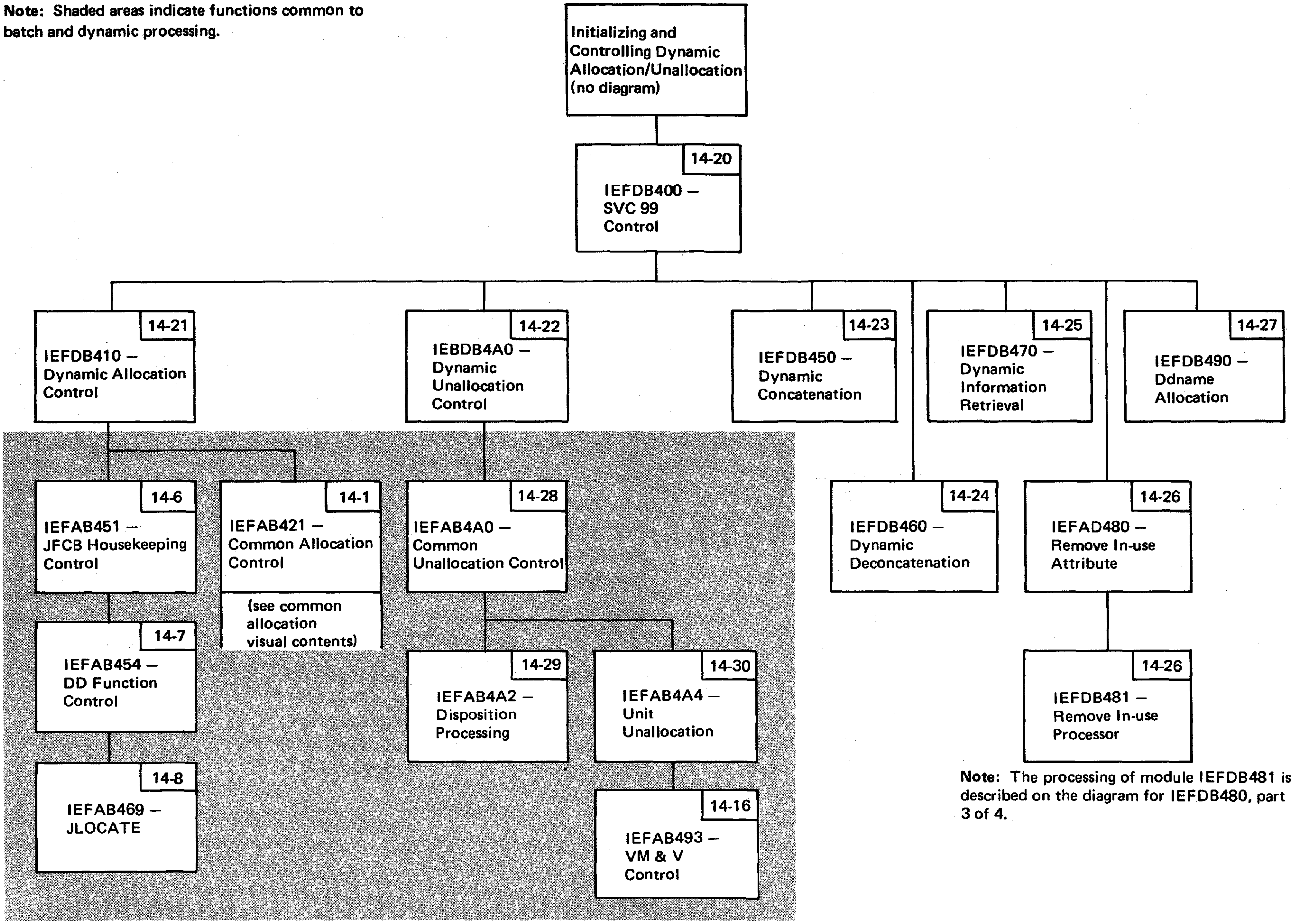


Figure 2-25. Batch and Dynamic Allocation/Unallocation Visual Contents (Part 1 of 2)

Note: Shaded areas indicate functions common to batch and dynamic processing.



Note: The processing of module IEFDB481 is described on the diagram for IEFDB480, part 3 of 4.

Figure 2-25. Batch and Dynamic Allocation/Unallocation Visual Contents (Part 2 of 2)

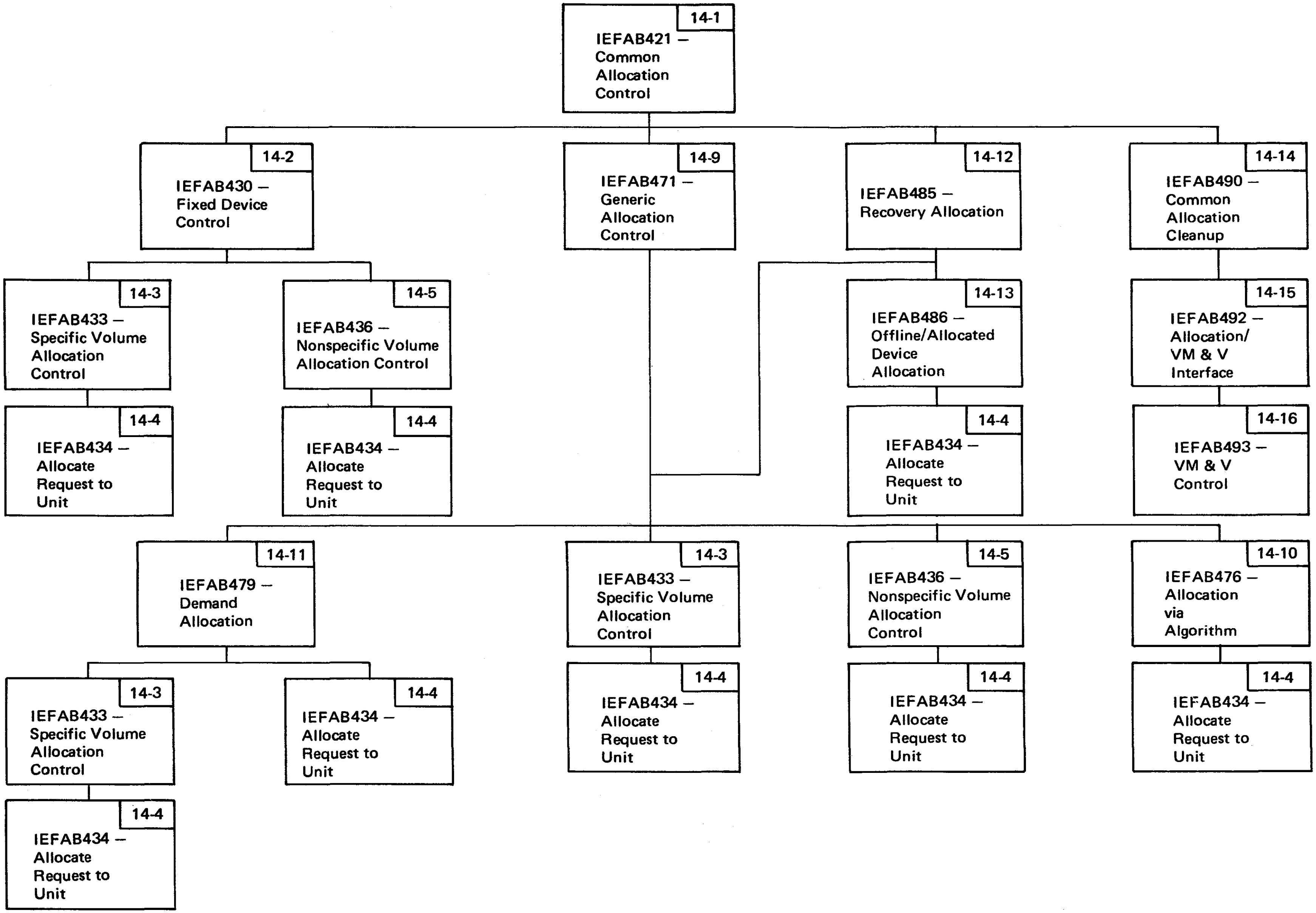
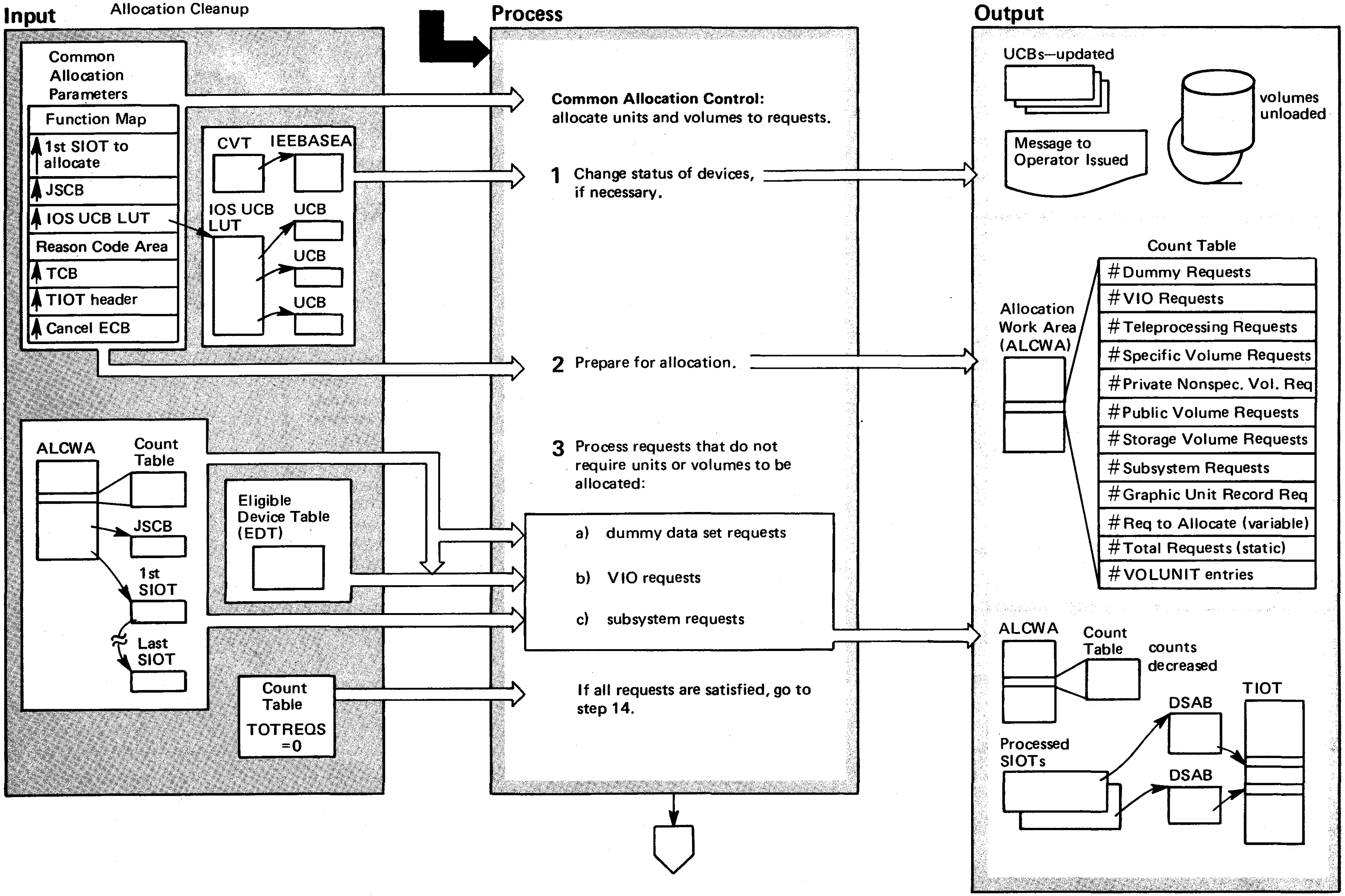


Figure 2-26. Common Allocation Visual Contents

Diagram 14-1. IEFAB421 – Common Allocation Control (Part 1 of 12)

ENTRY from IEFBB404 (see IEFBB401 – Initiator/Allocation Interface) or IEFDB413 (see IEFDB410 – Dynamic Allocation Control) or IEFAB490 – Common Allocation Cleanup



Common Allocation Control: allocate units and volumes to requests.

1 Change status of devices, if necessary.

2 Prepare for allocation.

3 Process requests that do not require units or volumes to be allocated:

a) dummy data set requests

b) VIO requests

c) subsystem requests

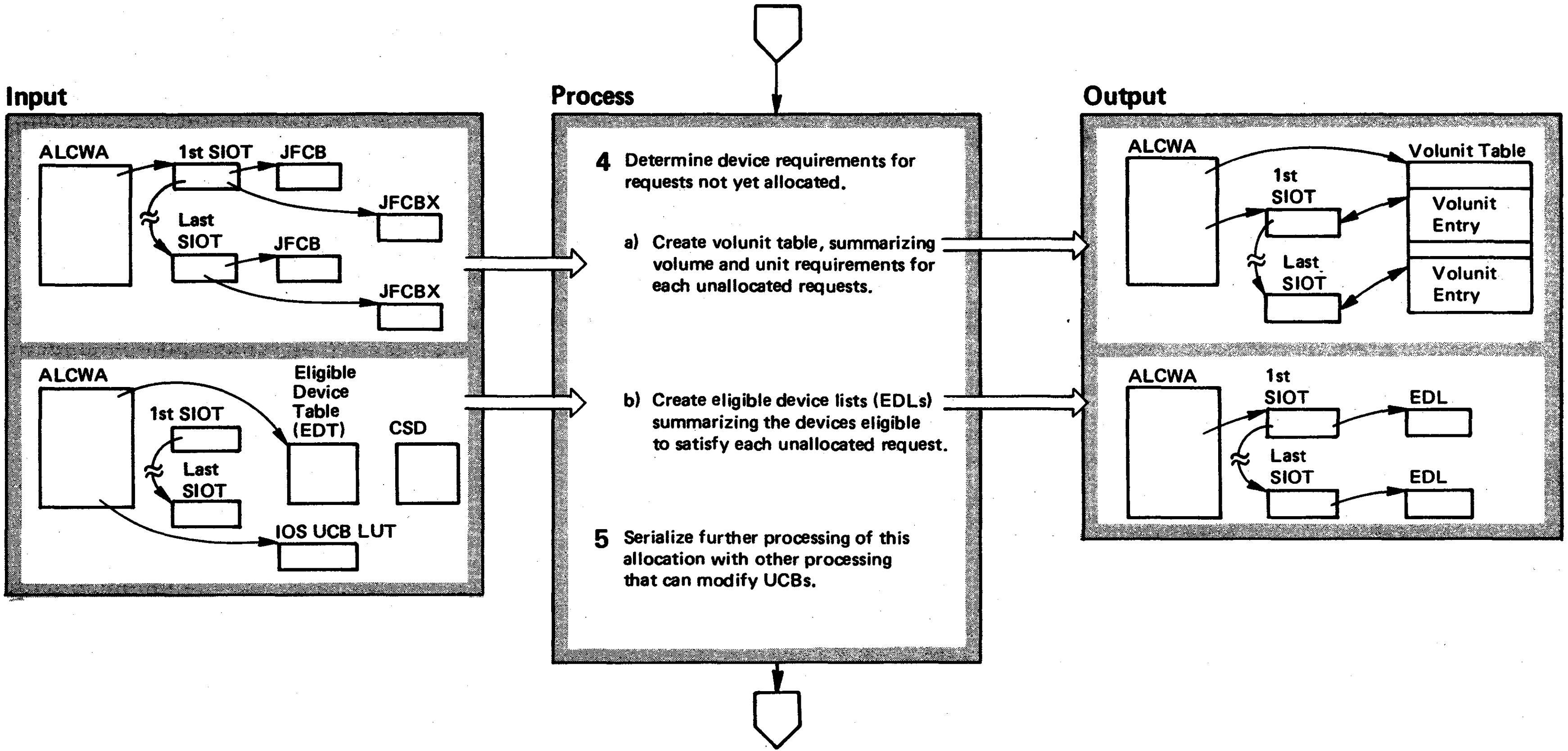
If all requests are satisfied, go to step 14.

Diagram 14-1. IEFAB421 – Common Allocation Control (Part 2 of 12)

Extended Description	Module	Segment
ENTRY Common Allocation Control (IEFAB421) controls the allocation of units and volumes to requests. It is called by:		
• Step Allocation Control (IEFBB404), when the allocation is batch (jobs and logons).		
• Dynamic Allocation Control (IEFDB410), when the allocation is dynamic.		
• Common Allocation Cleanup (IEFAB490), when the allocation is being reattempted because of a retry situation or because the operator authorized the allocation to wait for a device(s) without holding resources.		
1 If IEEBASEA indicates that units are pending a change of status (MSSUM=1), Common Allocation Control searches the IOS UCB LUT to locate UCBs whose status is to be changed. Depending on indicators in the UCB, Common Allocation Control:	IEFAB421	OFFLINES
• Takes a unit offline.		
• Unloads the volume on the unit.		
• Changes a device's status to an active console.		
Common Allocation Control issues a message to the operator informing him of the changed status.		
2 Common Allocation Control issues a GETMAIN macro instruction for the allocation work area (ALCWA) and places information from the common allocation parameter list in ALCWA. The allocation workarea serves as the communications area for all subsequent processing. (For details on ALCWA, see "Section 5: Data Areas.")	IEFAB421	INITWORK
Common Allocation Control also builds a count table in ALCWA, containing both the total number of requests and the different types of requests (see output of step 2) that must be processed. At this time, the counts in the count table reflect the number of SIOTs. The TOTVOLUN field, representing the number of unallocated volunit entries in the volunit table (which is built in step 4), is not initialized at this time. The count table is updated to reflect unallocated volunit entries, rather than SIOTs, in step 6. The counts in the count table are decreased during allocation processing, as each request is satisfied.	IEFAB421	BLDCOUNT

Extended Description	Module	Segment
3 To eliminate unnecessary processing, Common Allocation Control first processes requests that do not require units and volumes to be allocated:	IEFAB421	
a) Dummy data set requests. If the DMYREQS field in the count table does not equal 0, Common Allocation Control searches the SIOTs for the indicator that DUMMY or DSN=NULLFILE (SCTDUMMY=1), QNAME (SIOTQNAM=1), or TERM=TS (SIOTTERM=1) was specified. For each of these requests, Common Allocation Control:	IEFAB421	PROCSDMY
• Creates a DSAB and a TIOT entry.	IEFAB428	
• Marks the SIOT allocated (SIOTALCD=1).	IEFAB421	PROCSDMY
• Decreases the DMYREQS and TOTREQS fields in the count table.	IEFAB421	PROCSDMY
b) VIO requests. If the VAMREQS field in the count table is not equal to 0, VIO Allocation searches the SIOTs for the VIO indicator (SIOTVAM=1). For each VIO request, VIO Allocation:	IEFAB431	
• Places default space information in the JFCB, if space information does not exist. (Default space information is included in the non-executable module IEFAB445.)	IEFAB431	VAMSPACE
• Interfaces with DADSM to obtain a virtual UCB address and places the address in the SIOT.	IEFAB431	VAMDADSM
• Creates a DSAB and a TIOT entry.	IEFAB428	
• Marks the SIOT allocated (SIOTALCD=1).	IEFAB431	
• Decreases the VAMREQS and TOTREQS fields in the count table.	IEFAB431	
c) Subsystem requests; for example, sysin and sysout. If the SUBREQS field in the count table does not equal 0, Subsystem Request Allocation searches the SIOTs for the subsystem data set indicator (SIOTSSDS=1). For each such request, Subsystem Request Allocation:	IEFAB427	
• Interfaces with JES2 to allocate the request.	IEFAB427	BILDSSAL
• Creates a DSAB and a TIOT entry.	IEFAB428	
• Marks the SIOT allocated (SIOTALCD=1).	IEFAB427	
• Decreases the SUBSREQS and TOTREQS fields in the count table.	IEFAB427	
If the TOTREQS field in the count table reaches 0, all requests have been satisfied and processing continues with step 14.	IEFAB421	

Diagram 14-1. IEFAB421 – Common Allocation Control (Part 3 of 12)



4 Determine device requirements for requests not yet allocated.

a) Create volunit table, summarizing volume and unit requirements for each unallocated requests.

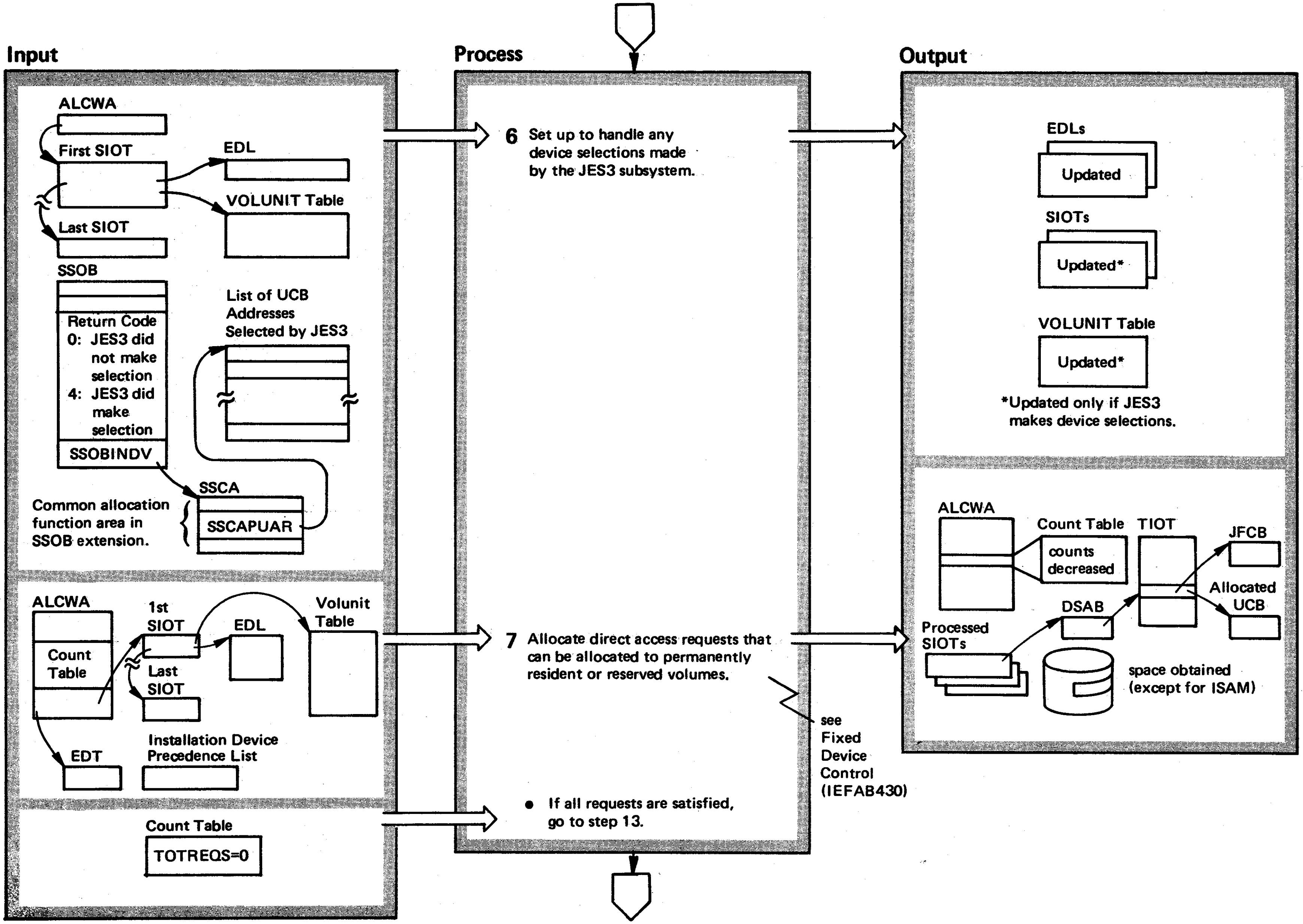
b) Create eligible device lists (EDLs) summarizing the devices eligible to satisfy each unallocated request.

5 Serialize further processing of this allocation with other processing that can modify UCBs.

Diagram 14-1. IEFAB421 – Common Allocation Control (Part 4 of 12)

Extended Description	Module	Segment
4 a) Device Requirements Determination issues a GETMAIN macro instruction to obtain space for a volunit table. The volunit table summarizes the volume and unit requirements of each request not allocated in step 3. For each unallocated SIOT, Device Requirements Determination creates one or more volunit entries in the volunit table; an entry is created for every possible unit the request might need. As the entries are created, Device Requirements Determination assigns the same unitid to the entries for SIOTs that requested unit affinity. (A unitid is an internal number used to identify a unique unit request.) After the volunit table is built, Affinity Resolution receives control to resolve volume affinities. For direct access requests, where duplicate volids are found (whether or not the unitids match), Affinity Resolution creates a new unitid and propagates the new unitid to all duplicate volids. For tape requests, if duplicate volids exist and the unitids are different, Affinity Resolution creates a new unitid and propagates the new unitid to all duplicate volids. The volunit table is then completed by initializing the status field of each volunit entry. The status field includes bits that indicate the type of request (for example, specific volume request, public request, storage request, private request) and the device class (for example, tape, direct access, unit record). For details on the volunit table, see *OS/VS2 Data Areas,* SYB8-0606.	IEFAB423 IEFAB423 IEFAB423 IEFAB423 IEFAB426	 BLDVOLUN INITVOLN UNAFFPRC
b) EDL Build creates an eligible device list (EDL) for each unallocated SIOT; information in the eligible device table (EDT) is used to construct the EDLs. Each EDL summarizes the devices eligible to satisfy a request. EDL Build issues a GETMAIN macro instruction to obtain space for the EDLs. Before the EDLs are created, EDL Build saves the DDR (dynamic device reconfiguration) count from the CSD (common system data area). After the EDLs are created, EDL Build again obtains the DDR count from the CSD and compares it to the saved DDR count. If the count has changed, DDR was invoked during creation of the EDLs and, therefore, the EDLs might be invalid – EDL Build then frees the EDLs and repeats this processing to build new EDLs.	IEFAB424 IEFAB438	
5 Common Allocation Control enqueues shared on SYSIEFSD (minor names Q4, CHNGDEVS, DDRTPUR, DDRDA) in order to serialize the processing of this allocation with other processing that might modify the UCBs (for example, pending-unload processing, pending-offline processing for units that become unallocated during this allocation).	IEFAB421	

Diagram 14-1. IEFAB421 – Common Allocation Control (Part 5 of 12)



6 Set up to handle any device selections made by the JES3 subsystem.

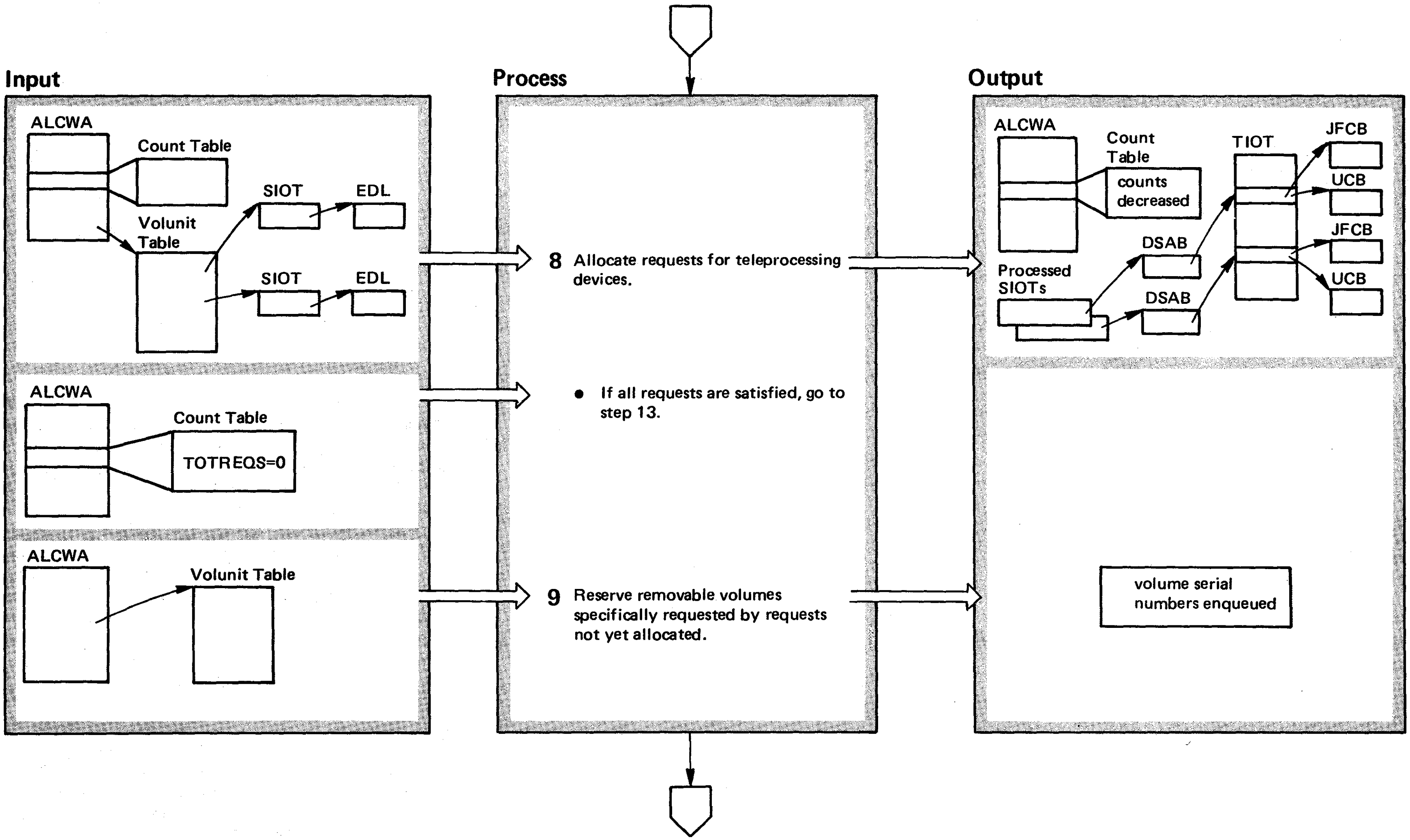
7 Allocate direct access requests that can be allocated to permanently resident or reserved volumes.

- If all requests are satisfied, go to step 13.

Diagram 14-1. IEFAB421 – Common Allocation Control (Part 6 of 12)

Extended Description	Module	Label
6 For each unallocated request, the JES3 interface routine invokes the JES3 subsystem (via the SSOB interface) to determine if JES3 has made device selections. If it has, the EDL is updated so that only those devices selected by JES3 are eligible for allocation. It also stores the selected UCB addresses in the VOLUNIT table and turns on the SIOTJES3 bit to indicate that this is a JES3 request. If JES3 did not make device selections, all units which are managed by JES3 are marked ineligible for allocation in the EDL.	IEFAB422	
7 Fixed Device Control allocates requests eligible to permanently resident and reserved direct access volumes.	IEFAB430	
During Fixed Device Control, the count table is updated to reflect the number of volunit entries still to be allocated, rather than the number of SIOTs. (The TOTREQS field is the only field that continues to represent a number of unallocated SIOTs.) Individual fields (for example, SPECREQS) and the TOTVOLUN field are decreased as unit requests (that is, volunit entries) are allocated; the TOTREQS field is decreased as SIOTs are completely allocated. If the TOTREQS field in the count table reaches 0, all requests have been satisifed and processing continues with step 13. For details on Fixed Device Control, see the M.O. diagram Fixed Device Control (IEFAB430).	IEFAB430	UPDTCNT

Diagram 14-1. IEFAB421 – Common Allocation Control (Part 7 of 12)



8 Allocate requests for teleprocessing devices.

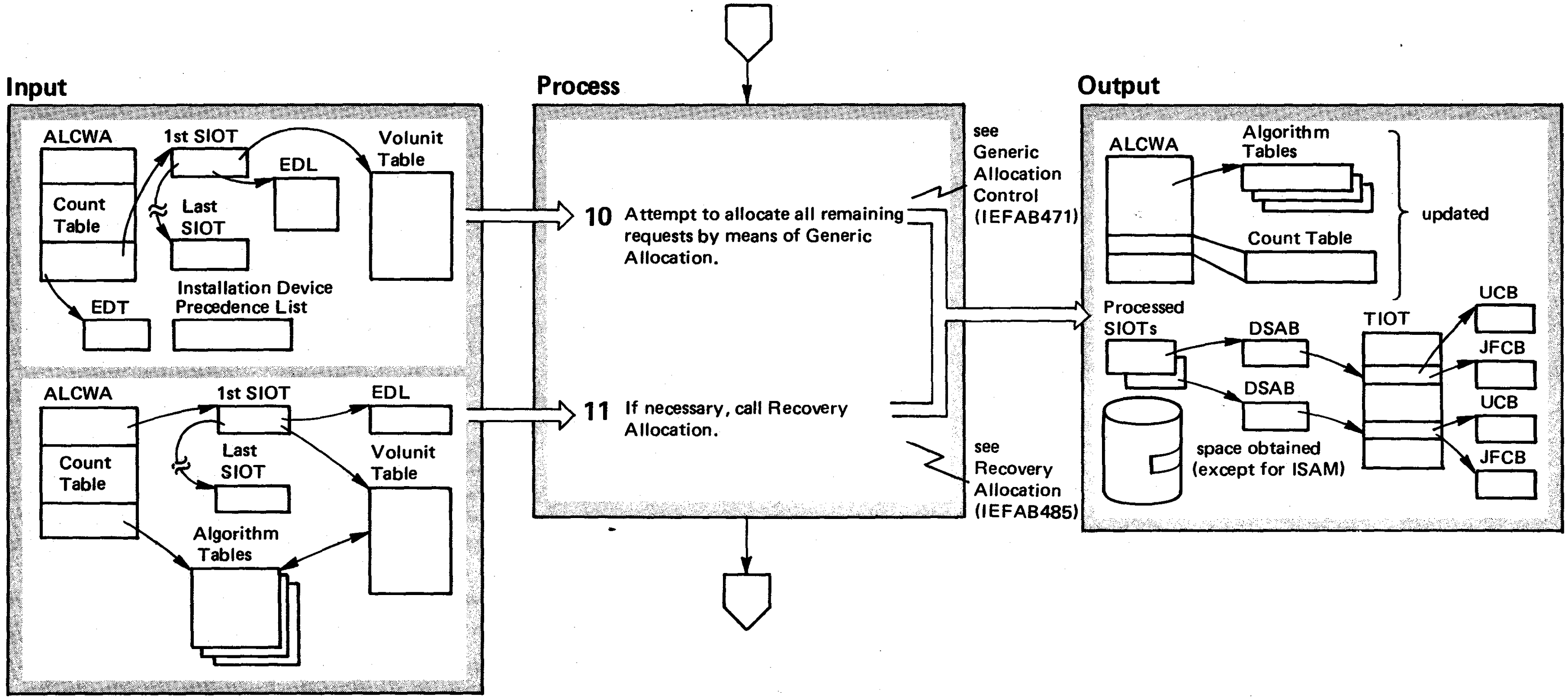
- If all requests are satisfied, go to step 13.

9 Reserve removable volumes specifically requested by requests not yet allocated.

Diagram 14-1. IEFAB421 – Common Allocation Control (Part 8 of 12)

Extended Description	Module	Segment
8 If the number of teleprocessing requests in the count table is not zero (TPREQS≠0), Common Allocation Control calls IEFAB425 (Process TP Requests). IEFAB425 enqueues exclusive on the allocation resource SYSIEFSD (minor name ALLOCTP) to serialize its processing with other allocations of teleprocessing devices. IEFAB425 then searches the status field of each volunit table entry for the indicator that a teleprocessing (communications) device is requested. When it finds the indicator, IEFAB425:	IEFAB421 IEFAB425	
• Selects a teleprocessing device from the EDL for this SIOT.	IEFAB425	TPEDLSEL
• Allocates the device to the request (see the M.O. diagram Allocate Request to Unit (IEFAB434), if the device is unallocated, is not an active console, and is not in use by a system service. Otherwise, this allocation is failed.	IEFAB434	
• Marks the volunit entry as allocated.	IEFAB425	
• Marks the SIOT allocated (SIOTALCD=1) if all volunit entries for the SIOT are allocated.	IEFAB425	
IEFAB425 repeats this processing for each teleprocessing request, decreasing the TPREQS and TOTREQS fields of the count table as each SIOT is allocated. When all teleprocessing requests are completed, IEFAB425 dequeues from the allocation resource SYSIEFSD (minor name ALLOCTP).		
9 Common Allocation Control enqueues on the volume serial numbers of removable volumes specifically requested by unallocated requests. The enqueue is either shared or exclusive, depending on whether other requests can share the volume. (For example, if the volume might be demounted during execution of the problem program or if the volume is tape, the enqueue must be exclusive.) The status field in each volunit entry indicates if a volume is specifically requested and if the volume is shareable.	IEFAB421	DOVOLENQ

Diagram 14-1. IEFAB421 – Common Allocation Control (Part 9 of 12)



10 Attempt to allocate all remaining requests by means of Generic Allocation.

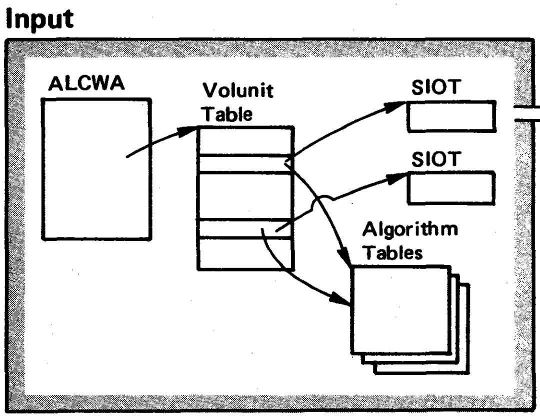
11 If necessary, call Recovery Allocation.

Diagram 14-1. IEFAB421 – Common Allocation Control (Part 10 of 12)

Extended Description	Module	Segment
10 Generic Allocation Control attempts to allocate remaining requests. It chooses the first generic device type in the installation device precedence list that includes devices required by unallocated requests and serializes the needed device group(s) within that generic. (For a description of device groups, see "Device Groups" in the "Introduction to Allocation/Unallocation.") Generic Allocation Control then allocates requests eligible to this device type in the following order:	IEFAB471	
a) Specific unit requests.	IEFAB479	
b) Specific volume requests for tape or DASD, if the volume is mounted.	IEFAB433	
c) Specific volume requests for tape or DASD, if the volume is not mounted; non-DASD and non-tape requests; nonspecific requests for private tape or DASD volumes.	IEFAB476	
d) Nonspecific non-private tape or DASD requests to public mounted volumes.	IEFAB436	
If all requests eligible to this generic device type are allocated, Generic Allocation Control releases the device group(s). Otherwise, the device group(s) remain serialized until the outstanding requests are allocated or until wait-without-holding-resources (part of step 11) is processed. Generic Allocation Control then chooses the next generic device type from the installation device precedence list that includes required devices and repeats this processing until all requests have been considered. For details on Generic Allocation Control, see the M.O. diagram "IEFAB471 – Generic Allocation Control."	IEFAB471	
11 Recovery Allocation receives control if the following conditions are true: • Requests still remain to be allocated (the TOTREQS field in the count table does not equal 0). • Retry is not indicated – INDRETRY=0 in ALCWA. (Retry is indicated if step 10 found a needed DASD or tape volume mounted on a unit not included in the serialized device groups. For retry, all allocated requests are unallocated and the entire allocation is reattempted; therefore, it is unnecessary to perform recovery allocation. For a description of retry, see "The Retry Situation" in the "Introduction to Allocation/Unallocation." The following situations result in recovery allocation: • A tape request(s) could not be allocated because the needed volume(s) is mounted on a generic device type different from (but compatible to) the generic selected for allocation. This request will go through retry processing before it is processed by Recovery Allocation. • Nonspecific DASD volume requests ask for volume affinity; although at least one request was successfully allocated, a subsequent request could not be allocated because of a DADSM error. • Nonspecific non-private tape or DASD requests could not be allocated to public mounted volumes. • Needed units are offline or allocated (and are not shareable) to another job. If Recovery Allocation can satisfy the unallocated requests only by waiting for devices to be unallocated, the operator is queried; he can cancel the job or instruct allocation to wait with or without holding resources: • If the job is cancelled, Recovery Allocation returns to Common Allocation Control, which completes the processing of this allocation. • For wait-without-holding-resources, Recovery Allocation returns to Common Allocation Control. Common Allocation Clean-up will unallocate the requests that have been allocated and the allocation will be reattempted when the awaited device(s) becomes available. • For wait-with-holding-resources, IEFAB491 (Wait Holding Resources) waits until the needed device(s) become available and then allocates it. For details on Recovery Allocation, see the M.O. diagram "IEFAB485 – Recovery Allocation."	IEFAB485	

Diagram 14-1. IEFAB421 – Common Allocation Control (Part 11 of 12)

Input



Process

12 If necessary and if authorized by operator, wait for needed devices, holding resources already allocated, and allocate remaining requests when devices become available.

13 End the serialization of this allocation with other processing that can modify UCBs.

14 Cleanup allocation processing.

See Common Allocation Cleanup (IEFAB490)

Return to IEFBB404 (See Initiator/Allocation Interface (IEFBB401)) or IEFDB413 (see Dynamic Allocation Control (IEFDB410)) or Common Allocation Cleanup (IEFAB490)

Output

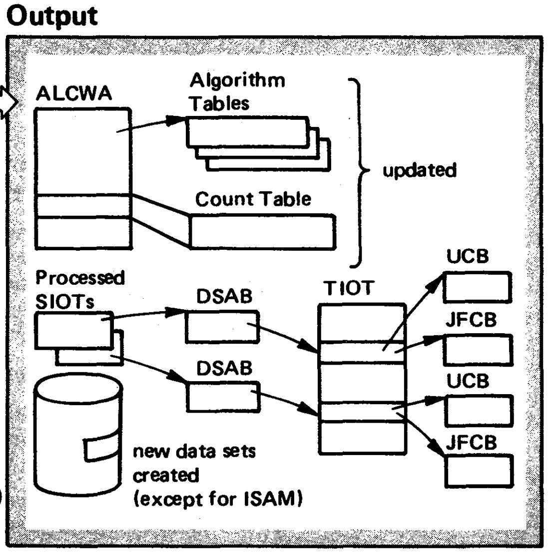
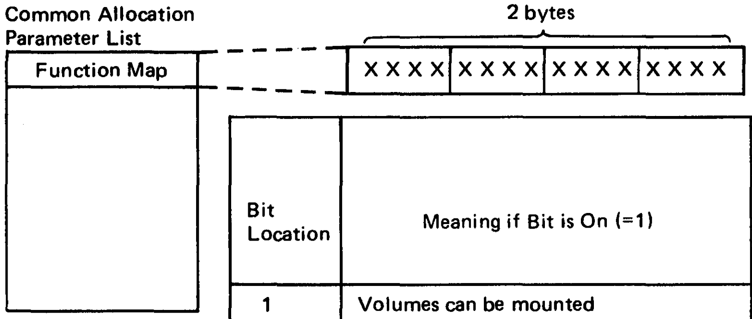


Diagram 14-1. IEFAB421 – Common Allocation Control (Part 12 of 12)

Extended Description	Module	Segment
12 This step is performed only if this allocation must wait for a needed device(s) to be unallocated and only if the operator authorized allocation to wait holding the resources already allocated. Common Allocation Control calls IEFAB491 (Wait Holding Resources) to wait for the needed device(s). The following steps are performed:	IEFAB491	
a) IEFAB491 informs the Allocation Queue Manager of the device groups from which an allocated unit is needed. The Allocation Queue Manager allows other allocations that cannot wait for devices (for example, dynamic allocation requests) to use the device groups serialized by this allocation until a needed device becomes available.	IEFAB4FA	
b) When a needed device becomes available, IEFAB491 first tries to allocate demand requests, if any have not yet been satisfied. Demand Allocation is called to process any outstanding demand requests – see the M.O. diagram "IEFAB479 – Demand Allocation"	IEFAB491 IEFAB479	
c) To determine if non-demand requests cannot be satisfied by using only online and unallocated (or allocated but shareable) devices, IEFAB491 calls the Cover/Reduce Algorithm. If all requests can be satisfied, IEFAB478 (Allocate from Groups the Algorithm Picked) allocates the outstanding requests, interfacing with ICBME for Mass Storage System (MSS) mount equalization and the System Resources Manager, to select the device to be allocated. IEFAB434 actually allocates the device – see the M.O. diagram "IEFAB434 – Allocate Request to Unit."	IEFAB480 IEFAB478 IEFAB434	
If all requests cannot be satisfied by considering only online, unallocated devices, IEFAB491 calls the Cover/Reduce Algorithm to determine if requests can still be satisfied if allocated units are considered. If not, the environment has changed and this allocation cannot be successfully completed – for example, a volume on a needed allocated unit has become reserved and that volume cannot be used. IEFAB491 returns to Common Allocation Control; this allocation is failed. Otherwise, IEFAB491 waits for the needed units – processing is repeated with step 12a.	IEFAB480 IEFAB491	
d) When all requests are satisfied, the Allocation Queue Manager releases the device groups no longer needed. (Only device groups that contain units on which public volumes must be mounted will remain serialized.)	IEFAB4FA	
13 Common Allocation Control dequeues from SYSIEFSD (minor names Q4, CHNGDEVS, DDRTPUR, DDRDA); since this allocation will no longer modify UCBs, further processing need not be serialized with other processing that might change the UCBs.	IEFAB421	
14 Common Allocation Cleanup completes allocation processing. One of three situations exists:	IEFAB490	
a) All requests still are not satisfied *and* either retry is indicated or the operator authorized allocation to wait without holding resources.		
b) All requests are satisfied.		
c) A terminating error occurred during allocation or the operator cancelled the job.		
The processing that occurs in each of these cases is described in the M.O. diagram "IEFAB490 – Common Allocation Cleanup."		
Error Processing		
An error occurring in any routine causes control to be returned to the calling routine. In this diagram, errors in steps 1-12 cause control to be passed to step 13.		
When IEFAB421 receives control, it creates an ESTAE environment so that its exit receives control if the program abnormally terminates.		



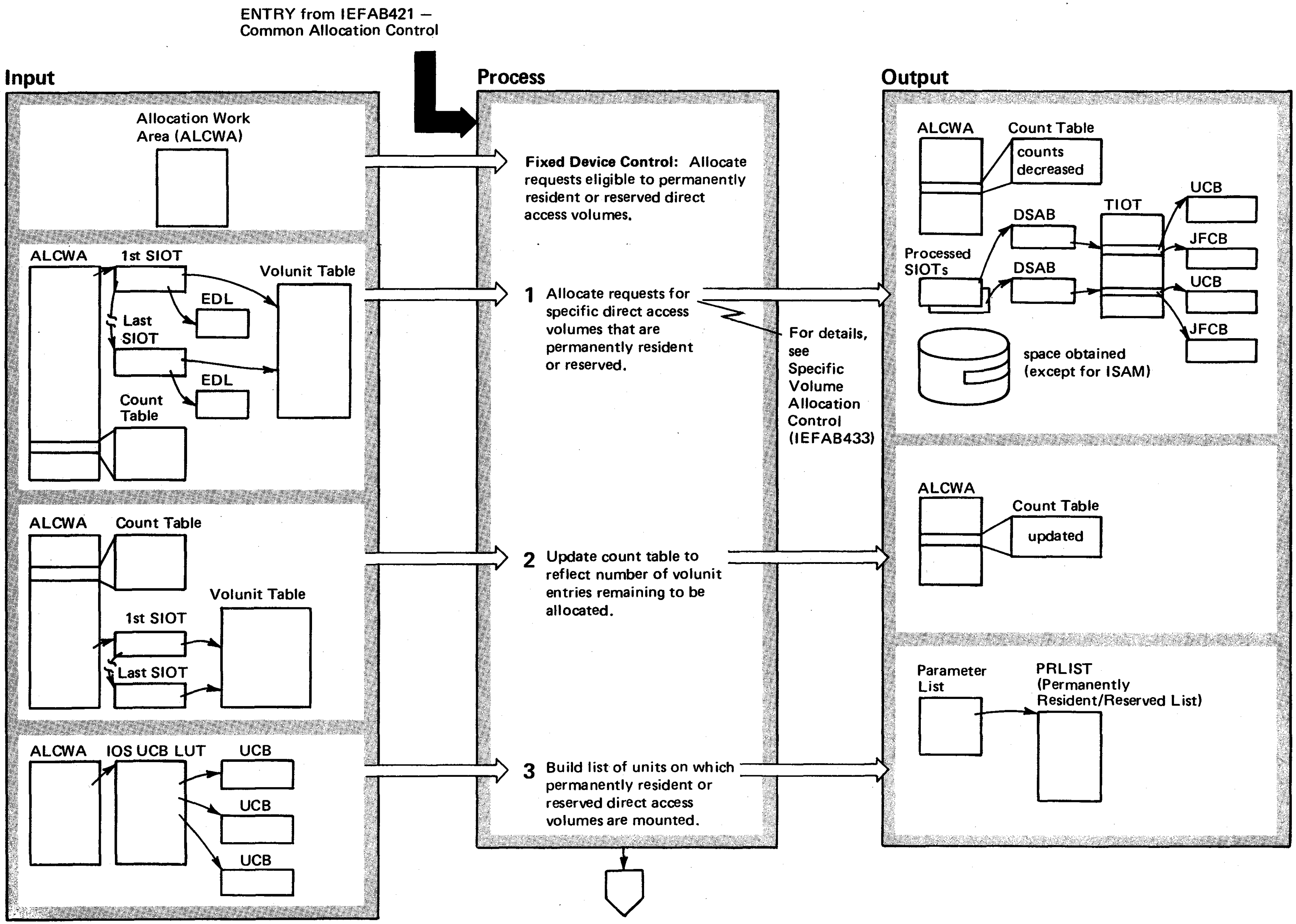
Bit Location	Meaning if Bit is On (=1)	Conditions When Bit is On (=1)	
		Caller is Step Allocation Control or Common Allocation Cleanup for batch allocation	Caller is Dynamic Allocation Control or Common Allocation Cleanup for dynamic allocation
1	Volumes can be mounted	Always	Depends on what user specifies
2	Allocation messages to be written	MSGLeVEL=(,1) was coded	
3	Allocation can wait for units allocated to another user	Only if request is not a logon	Only in two situations described in note 1
4	Allocation can wait for volumes	Only if request is not a logon	Only in two situations described in note 1
5	Reserved	———	———
6	Allocation can consider offline devices	Always	Depends on what user specifies
7	Mount requests to be issued in form of WTOR	Never	If bit number 1 is set on
8	Entire generic to be reserved for some specific volume requests	Retry is being performed	
9	Reserved	———	———
10	Allocation header message to be written	Always	Never
11	Allocation message for unit record devices to be issued to console	Monitor jobnames in effect	
12	Unallocation should indicate that scratch function should not enqueue on TIOT	Never	Always
13-16	Reserved	———	———

Note 1: Bits 3 and 4 are set on (=1) if the caller is Dynamic Allocation and if:

* A checkpoint data set is being allocated for use by the scheduler.
* A private catalog is being allocated for use by JFCB Housekeeping.

Figure 2-27. Function Map of Common Allocation Parameter List

Diagram 14-2. IEFAB430 – Fixed Device Control (Part 1 of 4)



Fixed Device Control: Allocate requests eligible to permanently resident or reserved direct access volumes.

1 Allocate requests for specific direct access volumes that are permanently resident or reserved.

For details, see Specific Volume Allocation Control (IEFAB433)

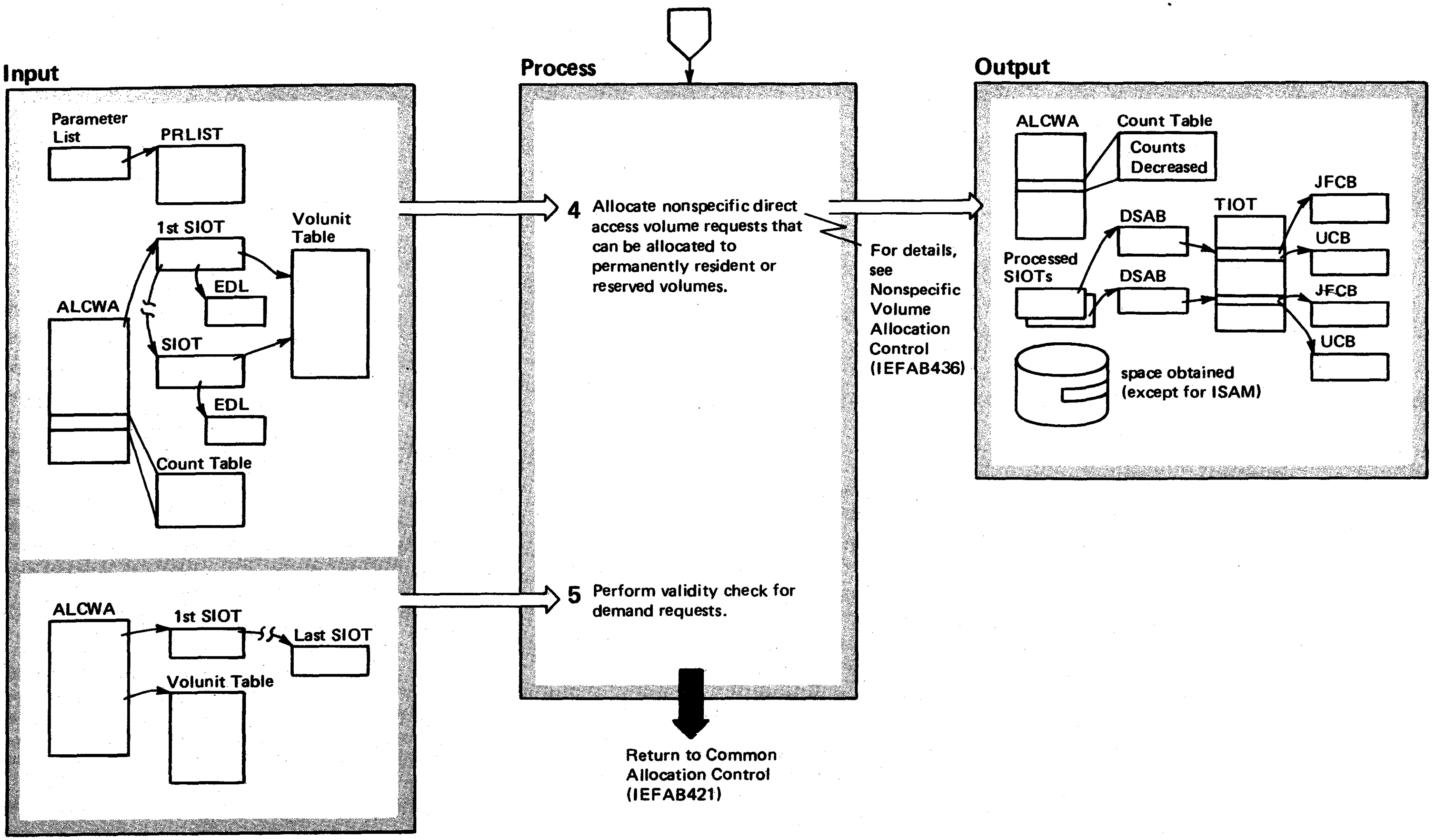
2 Update count table to reflect number of volunit entries remaining to be allocated.

3 Build list of units on which permanently resident or reserved direct access volumes are mounted.

Diagram 14-2. IEFAB430 – Fixed Device Control (Part 2 of 4)

Extended Description	Module	Segment
ENTRY Fixed Device Control (Main Path Control) is called by Common Allocation Control) to allocate direct-access requests that can be assigned to permanently resident or reserved volumes and to update the count table.		
1 This step is performed only if the SPECREQS field in the count table is not zero – that is, only if there are specific volume requests to be processed. Specific Volume Allocation Control allocates specific volume requests if the requested volume is a permanently resident or reserved direct access volume. For details, see the M.O. diagram Specific Volume Allocation Control (IEFAB433).	IEFAB433	
2 For each SIOT not yet allocated (SIOTALCD=0), Fixed Device Control examines the unallocated volunit entries and updates the count table to reflect the number of volunit entries to be allocated. (Up to this point, the count table represented the number of SIOTs to be allocated.) The following fields in the count table are updated: • TOTVOLUN – total number of volunit entries remaining to be allocated. • SPECREQS – number of specific requests. • PVTNREQS – number of private, nonspecific requests. • PUBLREQS – number of public requests. • STRGREQS – number of storage requests.	IEFAB430	UPDTCNT
3 This step is performed only if there are storage or public requests to be processed (in the count table, STRGREQS ≠ 0 or PUBLREQS ≠ 0). Fixed Device Control builds a list of pointers to devices on which permanently resident or reserved direct access volumes are mounted. To do this, Fixed Device Control searches through the IOS UCB LUT for direct access UCBs that meet the following conditions: • The volume mounted on the direct access device is permanently resident (UCBPRES=1) or reserved (UCBRSVD=1). • The unit is not pending offline (UCBCHNGS=0) and not pending unload (UCBUNLD=0). • The unit is online (UCBONLI=1) and ready (UCBNOTRD=0). • The unit is not being used by a system task (UCBNALOC=0). This permanent resident/reserved list (PRLIST) is used by Nonspecific Volume Allocation Control – see step 4.	IEFAB430	BLDPRLST

Diagram 14-2. IEFAB430 – Fixed Device Control (Part 3 of 4)



4 Allocate nonspecific direct access volume requests that can be allocated to permanently resident or reserved volumes.

For details, see Nonspecific Volume Allocation Control (IEFAB436)

5 Perform validity check for demand requests.

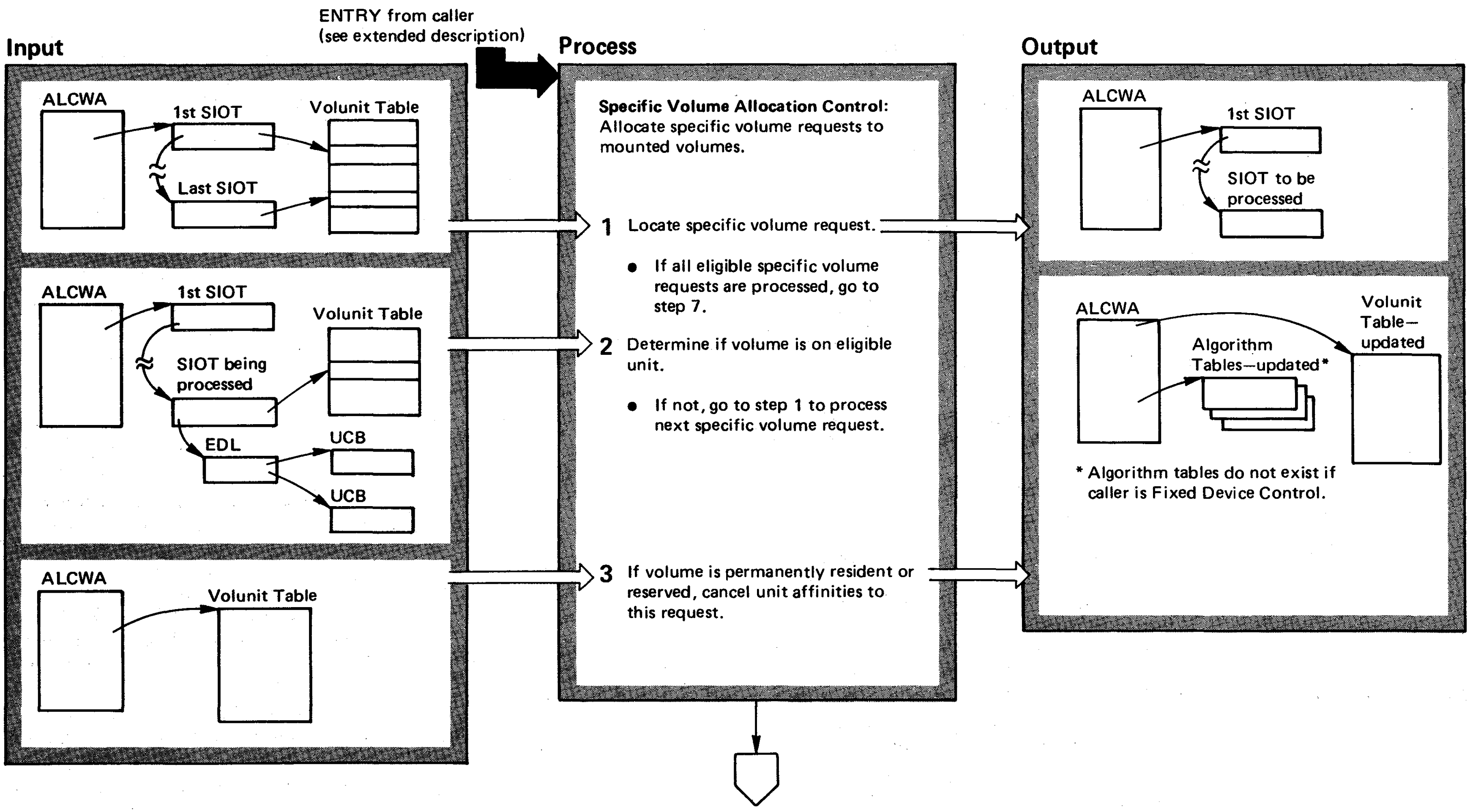
Return to Common Allocation Control (IEFAB421)

Diagram 14-2. IEFAB430 – Fixed Device Control (Part 4 of 4)

Extended Description	Module	Segment
4 Nonspecific Volume Allocation Control processes nonspecific volume requests that can be allocated to permanently resident or reserved volumes. Fixed Device Control calls this routine to allocate:	IEFAB436 IEFAB430	
a) Storage requests to storage volumes if STRGREQS ≠ 0 in the count table.		
b) Public requests to public volumes if PUBLREQS ≠ 0 in the count table.		
c) Public requests to storage volumes if all public requests were not allocated in the preceding call (4b).		
The parameter list includes a function map that indicates the type of request Nonspecific Volume Allocation Control should allocate. The parameter list also contains a pointer to the permanently resident/reserved list (PRLIST) built in step 5 – Nonspecific Volume Allocation Control uses the PRLIST to build a list of units eligible to satisfy individual requests. For details on Nonspecific Volume Allocation Control, see the M.O. diagram Nonspecific Volume Allocation Control (IEFAB436).		
When processing is complete, Fixed Device Control issues a FREEMAIN macro instruction to release the permanently resident/reserved list.	IEFAB430	

Extended Description	Module	Segment
5 Two or more requests can specifically request the same unit (for example, two DD statements specify UNIT=190) only if the following conditions are true:		
a) The unit is a direct access device.		
b) The same volume can be used for both requests:		
• Identical volume serial numbers are coded or volume affinity is indicated; or,		
• For nonspecific volume requests, none of the requests are private or nonshareable.		
To determine if the preceding conditions are satisfied when two or more requests specify the same unit, Fixed Device Control searches the SIOTs for demand requests (SIOTDMND=1) and checks the status field of the volunit entries for those SIOTs that request the same unit. If the preceding conditions are not satisfied, further processing of this allocation is terminated.	IEFAB430	CHEKDMND
Error Processing		
An error in any routine causes control to be returned to the calling routine.		
In the event of an abnormal termination, the ESTAE exit routine established by IEFAB421 performs any necessary cleanup.		

Diagram 14-3. IEFAB433 – Specific Volume Allocation Control (Part 1 of 4)



Specific Volume Allocation Control: Allocate specific volume requests to mounted volumes.

1 Locate specific volume request.

- If all eligible specific volume requests are processed, go to step 7.

2 Determine if volume is on eligible unit.

- If not, go to step 1 to process next specific volume request.

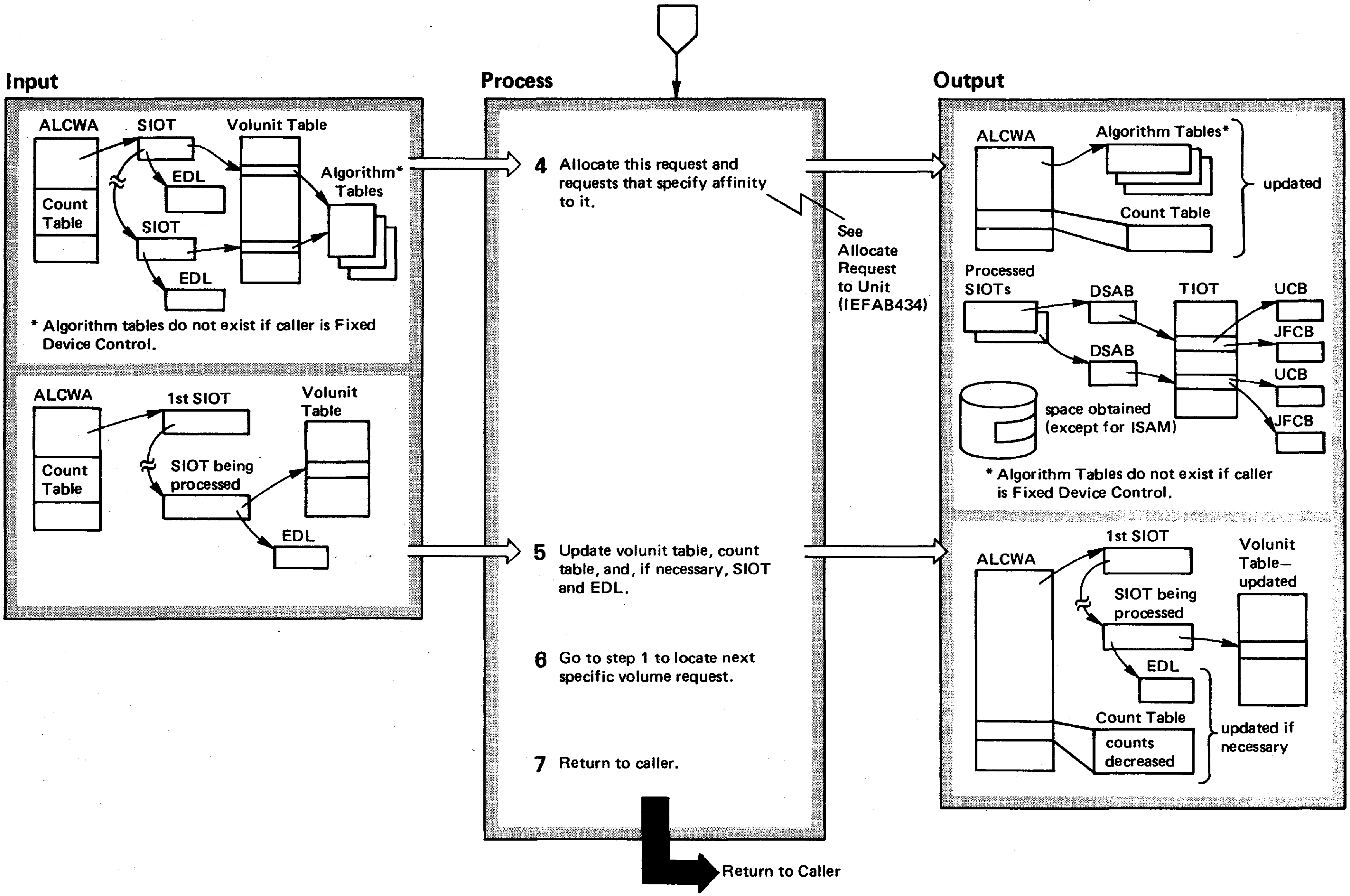
3 If volume is permanently resident or reserved, cancel unit affinities to this request.

* Algorithm tables do not exist if caller is Fixed Device Control.

Diagram 14-3. IEFAB433 – Specific Volume Allocation Control (Part 2 of 4)

Extended Description	Module	Segment
ENTRY Specific Volume Allocation Control (IEFAB433) allocates specific volume requests if the volume is mounted. It is called by: • Fixed Device Control (IEFAB430) to allocate specific volume requests if the volume is a permanently resident or reserved direct access volume. • Demand Allocation (IEFAB479) to allocate specific volume requests (if the volume is mounted) when a specific unit was also requested. • Allocation Within Generic (IEFAB475) to allocate specific volume requests not allocated during Fixed Device Control – if the volume is mounted. • Recovery Allocation (IEFAB485) to allocate specific tape volume requests marked for recovery processing. (This occurs when a needed tape volume is mounted on a generic device type different from the generic selected for allocation, but still eligible to the request – see "Processing Tape Requests" in the "Introduction to Allocation/Unallocation.")		
The processing described in this diagram is a loop performed for every specific volume request.		
Note: Every request that requires a particular volume is considered a specific volume request; for example, a volume serial number was coded, the data set was passed, the data set is cataloged.		
1 Specific Volume Allocation Control (IEFAB433) searches the SIOT chain for a SIOT that is not yet allocated (SIOTALCD=0) and that is not marked ineligible (SIOTGIGN=0). (When the caller is Fixed Device Control or Recovery Allocation, no SIOTs are marked ineligible. Demand Allocation and Allocation Within Generic are part of Generic Allocation Control; when they call IEFAB433, all SIOTs except those eligible to the generic being processed are marked ineligible.)	IEFAB433	
For an eligible SIOT, IEFAB433 checks the status field of volunit entries that are unallocated and are not marked for recovery, to determine if a specific volume was requested. (Recovery Allocation turned off the recovery indicator for volunit entries that should be processed.)	IEFAB433	
Processing of nonspecific volume requests is deferred; if the SIOT does not include specific volume requests, IEFAB433 searches the SIOT chain for the next eligible SIOT. Steps 2-5 are performed for every volunit entry requesting a specific volume (an eligible volunit entry). All eligible volunit entries are processed for a SIOT, one at a time, before the next SIOT is selected. When all eligible SIOTs have been processed, control passes to step 7 – IEFAB433 returns to the caller.		
2 IEFAB433 checks the UCBs pointed to by the EDL for this SIOT to determine if the requested volume is mounted on a unit eligible to this request. If not, the next specific volume request is located (step 1). Otherwise, IEFAB433 determines if the unit meets the following conditions: • The unit is online. • The unit is not being used by a system task. • No mount is pending for this unit, unless mounting is allowed for this allocation. • If the caller is Fixed Device Control, the volume on the unit is a permanently resident or reserved direct access volume. • The unit is not requested specifically by a request requiring a different volume.	IEFAB433	FINDSPC
If these conditions are not met, the unit cannot be allocated at this time. Further processing of this request is deferred and IEFAB433 selects the next eligible request – see step 1.		
If the preceding conditions are met, the unit can be allocated.		
3 This step is performed only if the volume is permanently resident or reserved. If another request indicates unit affinity to this request, IEFAB442 cancels the unit affinity by increasing the number of units required. (Unit affinity can be either implied or explicit – see "Selected Terms Used in Allocation/Unallocation" in the "Introduction to Allocation/Unallocation.") If, as a result of increasing the unit requirements, a SIOT would require more than 59 units, the allocation is failed.	IEFAB442	
Otherwise, the unit requirements are increased and IEFAB4F2 updates the algorithm tables, if necessary, to reflect the changed unit requirements.	IEFAB4F2	

Diagram 14-3. IEFAB433 – Specific Volume Allocation Control (Part 3 of 4)



* Algorithm tables do not exist if caller is Fixed Device Control.

4 Allocate this request and requests that specify affinity to it.

See Allocate Request to Unit (IEFAB434)

5 Update volunit table, count table, and, if necessary, SIOT and EDL.

6 Go to step 1 to locate next specific volume request.

7 Return to caller.

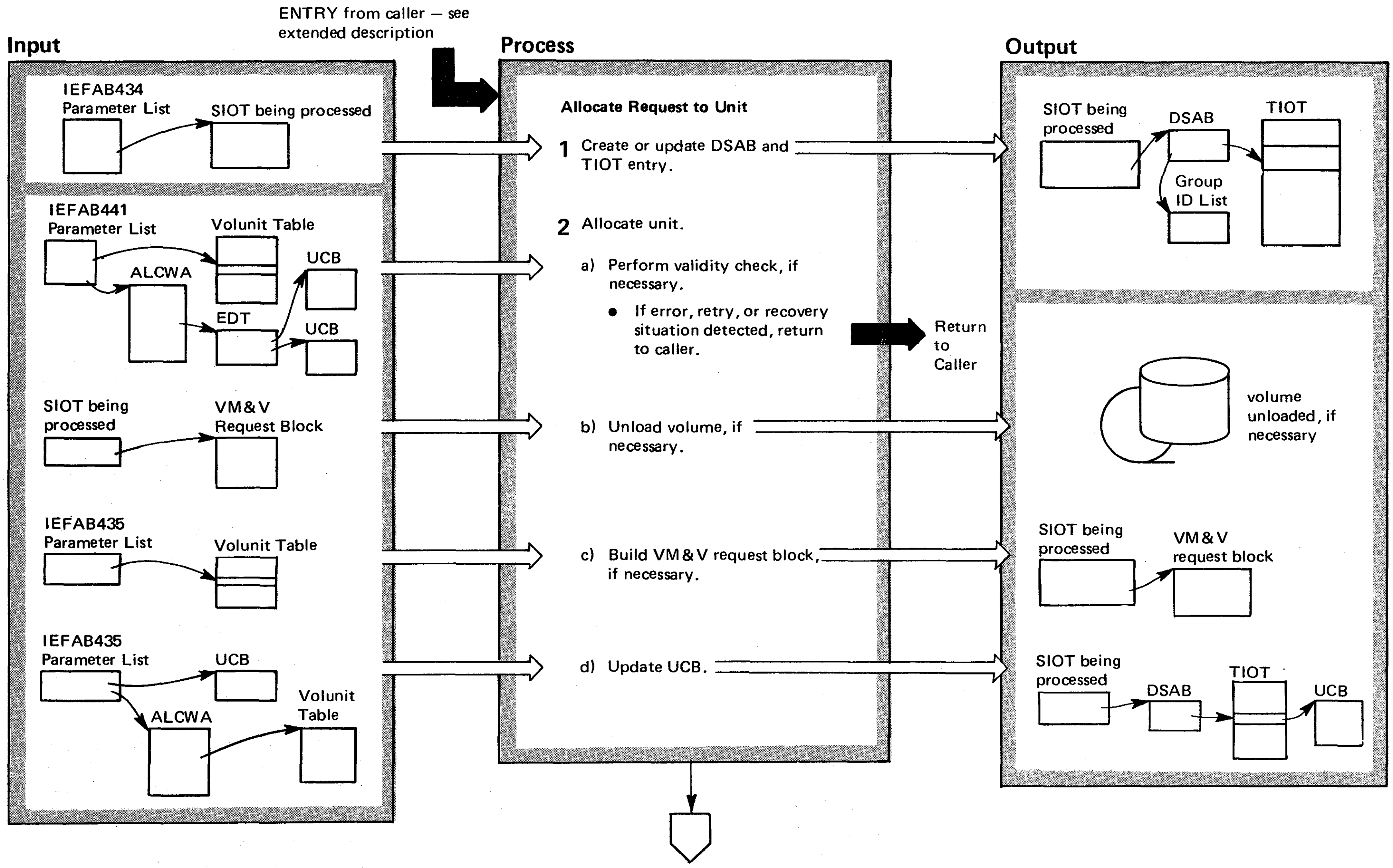
* Algorithm Tables do not exist if caller is Fixed Device Control.

Diagram 14-3. IEFAB433 – Specific Volume Allocation Control (Part 4 of 4)

Extended Description	Module	Segment
4 IEFAB434 (Allocate Request to Unit) allocates the request and any requests that specify affinity to it. For details, see the M.O. diagram Allocate Request to Unit (IEFAB434).	IEFAB434	
5 IEFAB433 marks the volunit entry allocated and decreases the TOTVOLUN field in the count table. The SPECREQS field of the count table is also decreased, unless the caller is Fixed Device Control. (If the caller is Fixed Device Control, the count table reflects the number of unallocated SIOTs and cannot be decreased until all volunit entries for a SIOT are allocated.)	IEFAB433	
IEFAB433 also updates the EDL if the following conditions are true: ● The unit just allocated is the first volunit entry allocated for the SIOT; and, ● The SIOT is a multi-unit request that must be allocated to a single generic.	IEFAB433	VUSCAN
All device types in the EDL are marked ineligible, except for the device type just allocated.	IEFAB433	UPDEDL
If all volunit entries for this SIOT are now allocated, IEFAB433 marks the SIOT allocated (SIOTALCD=1) and decreases the TOTREQS field in the count table. If the caller is Fixed Device Control (and, therefore, the count table represents the number of unallocated SIOTs), the SPECREQS field of the count table is also decreased.	IEFAB433	

Extended Description	Module	Segment
6 IEFAB433 locates the next specific volume request to be processed – go to step 1.	IEFAB433	
7 IEFAB433 returns to the caller. (See the beginning of the Extended Description for a list of callers.)	IEFAB433	
Error Processing An error in any routine causes control to be returned to the calling routine. In the event of an abnormal termination, the ESTAE exit routine established by IEFAB421 performs any necessary cleanup.	●	

Diagram 14-4. IEFAB434 – Allocate Request to Unit (Part 1 of 6)



Allocate Request to Unit

1 Create or update DSAB and TIOT entry.

2 Allocate unit.

a) Perform validity check, if necessary.

- If error, retry, or recovery situation detected, return to caller.

b) Unload volume, if necessary.

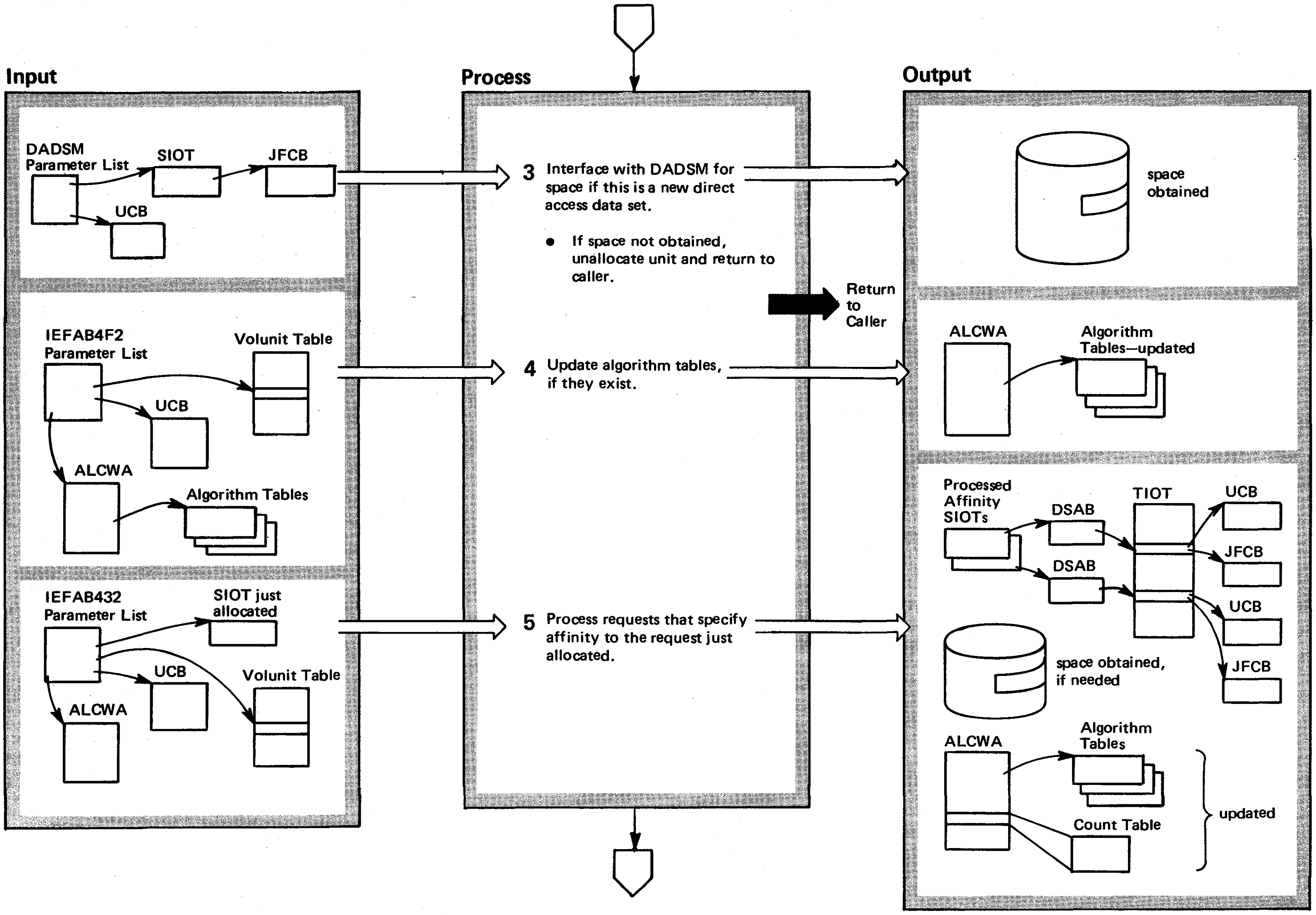
c) Build VM&V request block, if necessary.

d) Update UCB.

Diagram 14-4. IEFAB434 – Allocate Request to Unit (Part 2 of 6)

Extended Description	Module	Segment
ENTRY IEFAB434 (Allocate Request to Unit) is the common service routine that actually allocates a request to a unit. It is called by: • IEFAB425 to allocate teleprocessing requests. • IEFAB432 to allocate requests that specified affinity to an allocated request. • IEFAB433 to allocate specific volume requests if the volume is mounted. • IEFAB436 to allocate nonspecific volume requests if a volume is mounted. • IEFAB441 to allocate requests when the needed volume is found on an eligible unit other than the unit being considered (for example, a unit affinity request is being processed) and the volume is permanently resident or reserved. • IEFAB478 to allocate requests processed by the algorithm. • IEFAB479 to allocate a unit that was specifically requested. • IEFAB489 to allocate online devices during recovery allocation.		
1 IEFAB428 creates or updates a DSAB and TIOT entry for this request, based on the parameter list it receives as input from IEFAB434. If the volunit entry being processed is the first volunit entry to be allocated for this SIOT, the DSAB and TIOT entry must be created; a group ID list is also created, indicating the device group allocated to this request. If this is not the first volunit entry to be allocated for the SIOT, IEFAB428 updates the existing DSAB, TIOT entry, and group ID list.	IEFAB428	
2 IEFAB435 (Update UCB Routine) allocates the unit; the processing of IEFAB435 includes the following steps:	IEFAB435	
a) If indicated by the caller of IEFAB434, IEFAB441 (Volume Validity Checker) receives control to validity check this request. (The validity check is indicated only if a specific volume was requested and that volume is not mounted on the unit to be allocated.) IEFAB441 scans the UCBs pointed to by the EDT group entries to determine if the volume is mounted. If the volume is not mounted, the validity check is unnecessary; processing continues with step 2b. Otherwise, IEFAB441 determines if the request can be allocated. The following are the possible error conditions that can be detected: • The unit is in use by a system task (UCBNALOC=1). • The device type of the unit containing the volume is not compatible with the requested device type. • The volume is permanently resident or reserved and is mounted on a unit that is not eligible to this request. • The volume is mounted on a unit allocated to another user, and this allocation is not allowed to wait for units, as indicated in the common allocation parameter list (see figure 2-27). (If this allocation can wait for units, the request is marked for recovery processing.) If the volume is located on a device group that is not serialized, the request is marked for retry processing (SIOTRTRY=1); ALCWA is also updated to indicate retry is necessary (INDRETRY=1). If no error, retry, or recovery situation is detected, allocation of this request continues.	IEFAB441	
b) IEFAB49C unloads the volume currently mounted on the unit, if that volume cannot be used.	IEFAB49C	
c) IEFAB435 builds a VM&V request block for this SIOT, if a volume must be mounted on the unit. (The volume will be mounted after all requests have been satisfied – see the M.O. diagram Common Allocation Cleanup (IEFAB490).	IEFAB435	VMVSETUP
d) IEFAB435 updates the UCB to indicate it is allocated.	IEFAB435	UPDUCB

Diagram 14-4. IEFAB434 – Allocate Request to Unit (Part 3 of 6)



3 Interface with DADSM for space if this is a new direct access data set.

- If space not obtained, unallocate unit and return to caller.

4 Update algorithm tables, if they exist.

5 Process requests that specify affinity to the request just allocated.

Diagram 14-4. IEFAB434 – Allocate Request to Unit (Part 4 of 6)

Extended Description	Module	Segment
3 IEFAB434 interfaces with DADSM to obtain space for new non-ISAM (see M.O. diagram Common Allocation Cleanup IEFAB490 for a description of the ISAM process) direct access data sets. If DADSM is unable to allocate space, IEFAB434 unallocates the unit, if no other requests were allocated to it, and removes the device entry for this UCB from the TIOT entry. If the use attribute for the volume was changed for this request, the original use attribute is restored. IEFAB434 then returns to the caller. Further processing of this allocation depends on whether the DADSM error is recoverable (for example, if a specific volume was requested, the error is unrecoverable and this allocation is failed).	IEFAB434 IEFAB434	DADSMINT DADSMERR
4 IEFAB4F2 updates the algorithm tables to reflect the request that was just allocated, if the algorithm tables exist. (The algorithm tables are not created until the beginning of Generic Allocation Control.) **Note:** The "permanently ignore" indicator (CVRIGNOR) in the algorithm tables is not set at this time for non-specific volume requests. (When this indicator is set, the allocation of this request is no longer considered by the algorithm.) This is necessary because affinity requests have not yet been processed – if nonspecific requests that specify affinity to each other cannot all be allocated to the current volume, recovery processing will be necessary. If affinity requests are successfully processed, the "permanently ignore" indicator will be set in step 6.	IEFAB4F2	
5 If indicated by the caller, IEFAB432 (Affinity Processor) processes requests that specify affinity to the request just allocated. (Affinity processing is not performed when IEFAB434 is called to allocate an affinity request – see step 5a and 5c). IEFAB432 searches the volunit table for affinity requests. The following steps are performed for each request:	IEFAB432	
a) A validity check is necessary if only unit affinity was requested and if the affinity request needs a specific volume. IEFAB441 (Volume Validity Checker) scans the UCBs pointed to by the EDT group entries to determine if the needed volume is mounted. If the volume is not mounted, further validity processing is unnecessary and processing continues with step 5b. Otherwise, IEFAB441 checks for error, recovery and retry situations:		
• This allocation will be failed if the unit is in use by a system task (UCBNALOC=1) or if the device type of the unit containing the volume is not compatible with the requested device type.		
• This allocation will be failed if the volume is permanently resident or reserved and the unit containing the volume is not eligible to this request.		
• If the volume is mounted on a unit allocated to another user, this allocation will be failed if it is not allowed to wait for units (as indicated in the common allocation parameter list – see figure 2-27). If this allocation can wait for units, the request is marked for recovery processing.		
• If the volume is located on a device group that is not serialized, the request is marked for retry processing (SIOTRTRY=1); ALCWA is also updated to indicate retry is necessary (INDRETRY=1).		
If none of these situations are detected, one of the following situations exists:		
• The volume is not permanently resident or reserved, the device group containing the unit is serialized, and the unit is unallocated. IEFAB49C receives control to unload the volume. Processing of this request then continues with step 5b.	IEFAB49C	
• The volume is permanently resident or reserved and the unit containing the volume is eligible to this request. If another request indicates unit affinity to this request, IEFAB442 cancels the unit affinity by increasing the number of units required. (Unit affinity can be either implied or explicit – see "Selected Terms Used in Allocation/Unallocation" in the "Introduction to Allocation/Unallocation.") If, as a result of increasing the unit requirements, a SIOT would require more than 59 units, the allocation is failed.	IEFAB442	

Step 5 continued on Part 6

Diagram 14-4. IEFAB434 – Allocate Request to Unit (Part 5 of 6)

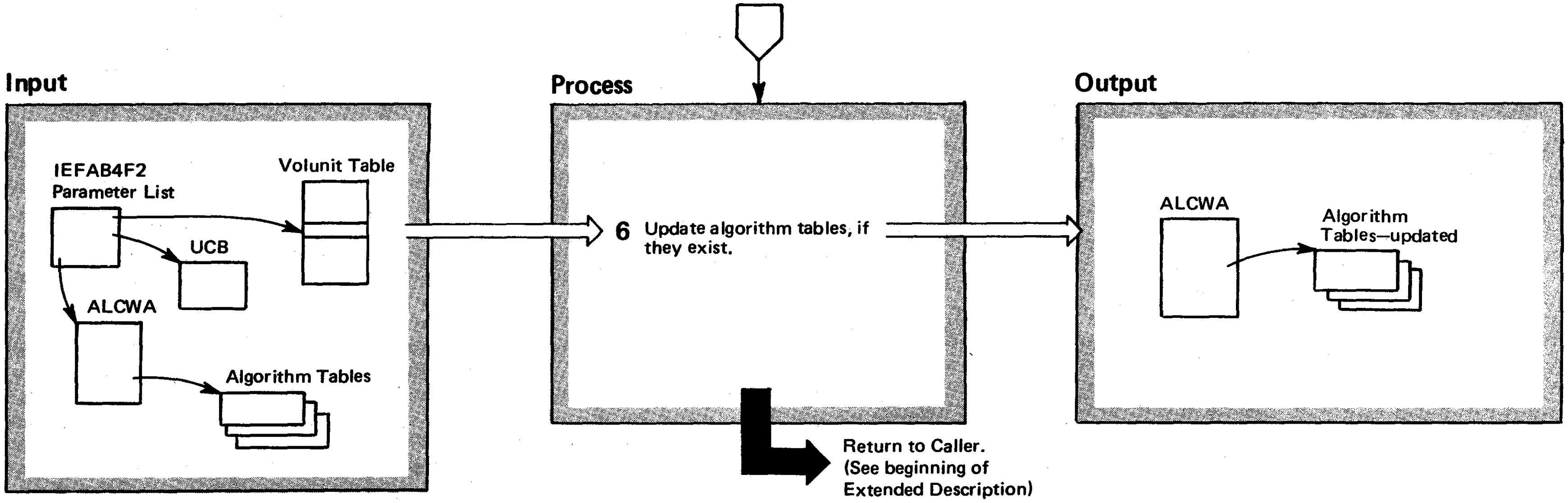


Diagram 14-4. IEFAB434 – Allocate Request to Unit (Part 6 of 6)

Extended Description	Module	Segment
5 (Continued) Otherwise, the unit requirements are increased, IEFAB428 creates a larger DSAB/TIOT entry, and IEFAB4F2 updates the algorithm tables to reflect the changed unit requirements. IEFAB434 allocates the request to the unit containing the needed volume.	IEFAB428 IEFAB4F2 IEFAB434	
IEFAB441 marks the volunit entry as allocated and decreases the appropriate counts in the count table. If the SIOT is now completely allocated, the SIOT is also marked allocated and the TOTREQS field in the count table is decreased.	IEFAB441	
IEFAB432 then selects the next affinity request to be processed (that is, steps 5b through 5d are skipped for this request)	IEFAB432	
b) IEFAB432 ensures that the device type just allocated is eligible to the affinity request being processed. If not, this allocation is failed.	IEFAB432	FINDEDL2
Note: The affinity request need not be eligible to the particular unit allocated, only to the generic device type. For example, 3330 is divided into two separate unit groups, 3330A and 3330B. The request just allocated had specified 3330A; the affinity request specified 3330B. The affinity request is considered eligible to the unit allocated from 3330A.		

Extended Description	Module	Segment
c) IEFAB434 is called to allocate the affinity request.	IEFAB434	
If the request (volunit entry) is successfully allocated and, as a result, all volunit entries for this SIOT are allocated, IEFAB432 marks the SIOT allocated and decreases the TOTREQS field in the count table.	IEFAB432	AFFPROC
d) IEFAB432 updates the EDL if the following conditions are true: • The unit just allocated is the first volunit entry allocated for this SIOT; and, • The SIOT is a multi-unit request that must be allocated to a single generic. All device types in the EDL are marked ineligible except for the device type just allocated.	IEFAB432	UPDEDL
6 IEFAB4F2 is called to update the algorithm tables; if affinity requests were successfully processed, the "permanently ignore" indicator (CVRIGNOR) – which was not updated in step 4 – can now be set on.	IEFAB4F2	

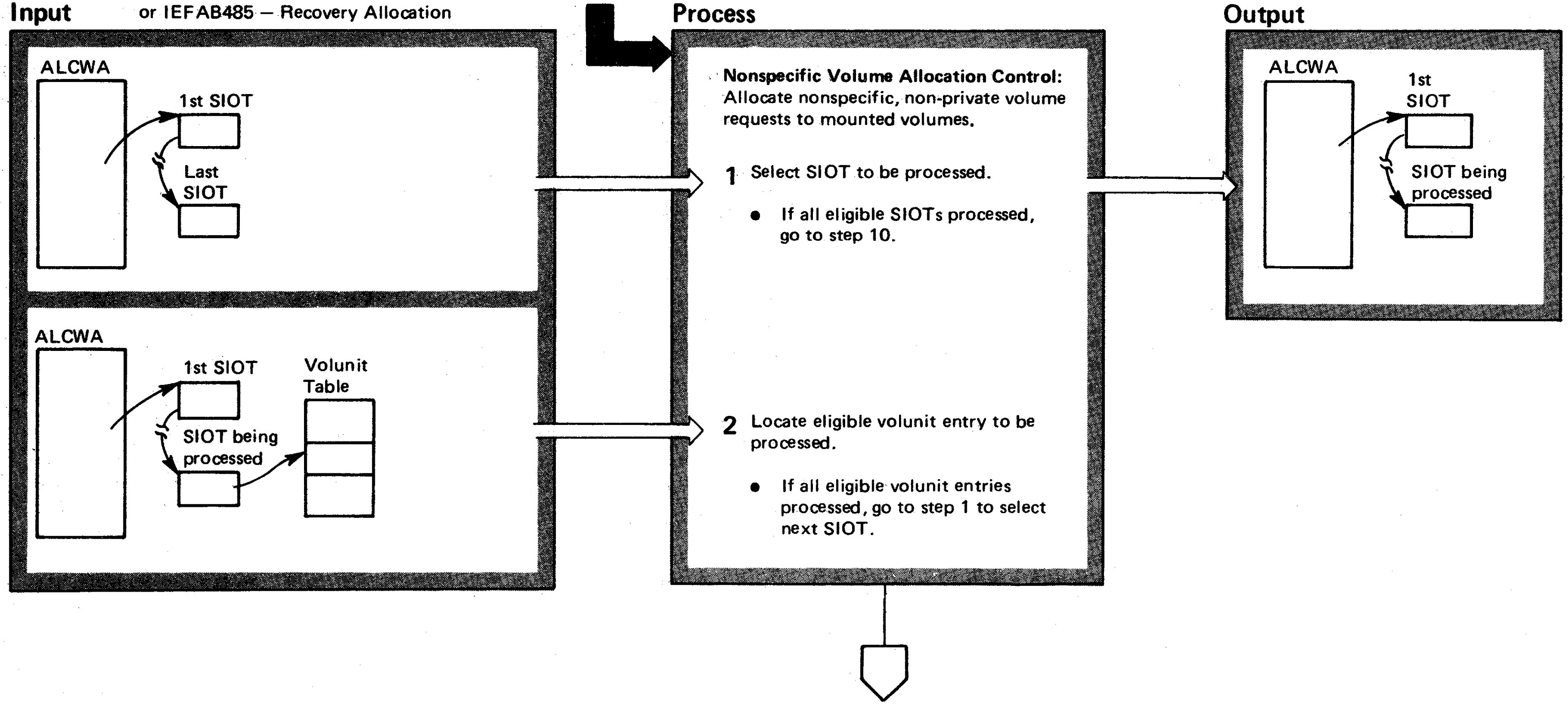
Error Processing

An error in any routine causes control to be returned to the calling routine.

In the event of an abnormal termination, the ESTAE exit routine established by IEFAB421 performs any necessary cleanup.

Diagram 14-5. IEFAB436 – Nonspecific Volume Allocation Control (Part 1 of 6)

ENTRY from IEFAB430 – Fixed Device Control;
IEFAB475 (see IEFAB471 – Generic Allocation Control);
or IEFAB485 – Recovery Allocation



Nonspecific Volume Allocation Control: Allocate nonspecific, non-private volume requests to mounted volumes.

1 Select SIOT to be processed.

- If all eligible SIOTs processed, go to step 10.

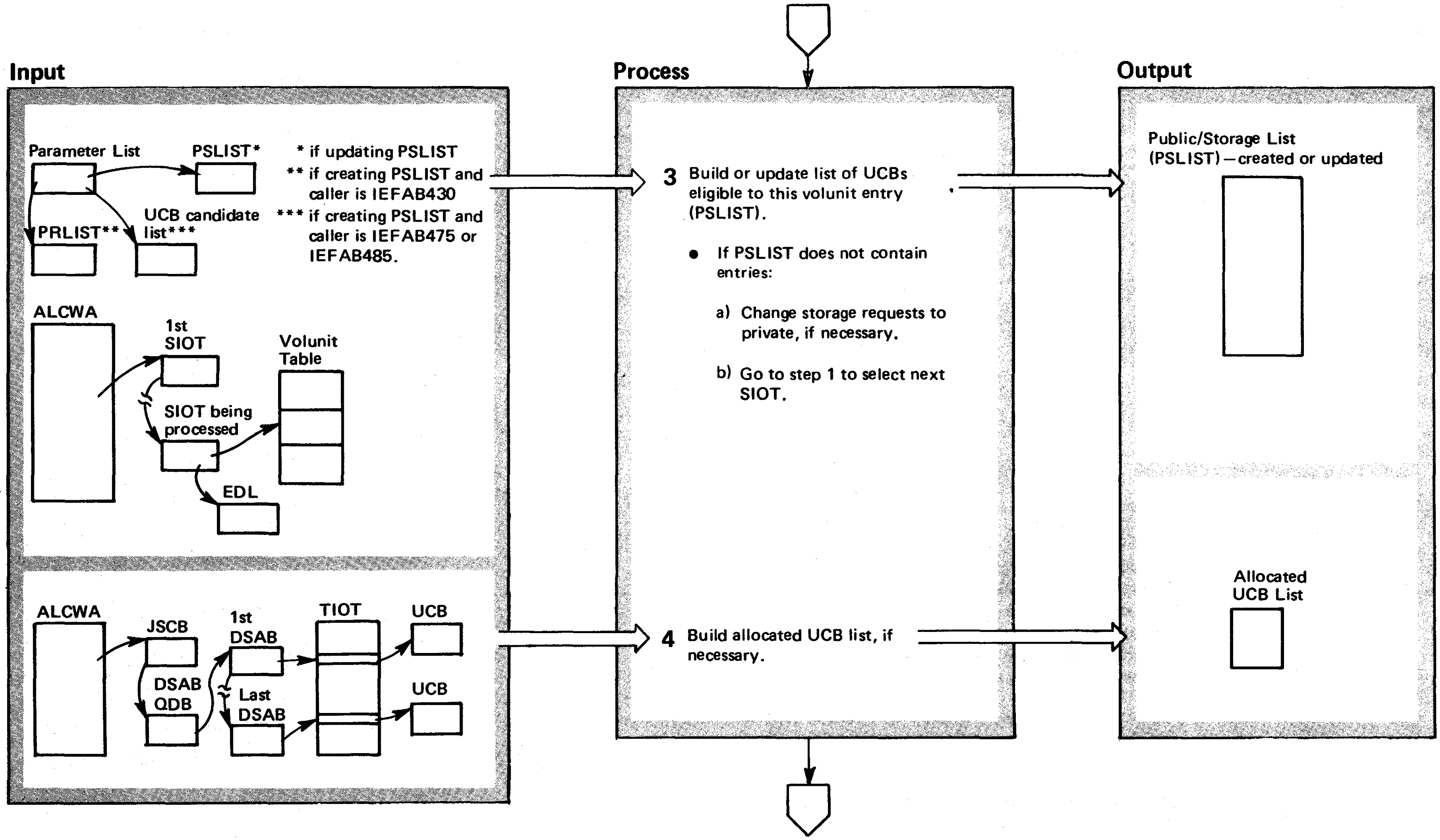
2 Locate eligible volunit entry to be processed.

- If all eligible volunit entries processed, go to step 1 to select next SIOT.

Diagram 14-5. IEFAB436 – Nonspecific Volume Allocation Control (Part 2 of 6)

Extended Description	Module	Segment
ENTRY Nonspecific Volume Allocation Control (IEFAB436) is called by Fixed Device Control (IEFAB430), Allocation Within Generic (IEFAB475), and Recovery Allocation (IEFAB485) to allocate nonspecific volume requests to mounted volumes. IEFAB436 allocates one of the following types of requests each time it is called: • Storage requests to storage volumes. • Public requests to public volumes. • Public requests to storage volumes. The type of request to be allocated is indicated in the function map of the parameter list passed to IEFAB436. **Note:** The processing of IEFAB436 is a series of loops. Step 1 locates a SIOT to be processed; steps 2-8 are performed to locate and process each eligible volunit entry for a selected SIOT. The processing of a single volunit entry can involve loops through steps 3-6 or through steps 3-7, if the volume mounted on a unit selected for the volunit entry cannot be used. The extended description of each step describes the circumstances under which it is performed.		
1 IEFAB436 scans the SIOT chain to locate a SIOT that is not allocated (SIOTALCD=0) and that is not marked ineligible (SIOTGIGN=0). (When the caller is Fixed Device Control, no SIOTs are marked ineligible; Allocation Within Generic is part of Generic Allocation Control, which processes only one generic device type at a time – all SIOTs except those eligible to the device type being processed are marked ineligible; when the caller is Recovery Allocation, all SIOTs except those to be processed are marked ineligible.) If all eligible SIOTs have been processed, step 10 receives control.	IEFAB436	
2 IEFAB436 checks the status field of unallocated volunit entries for this SIOT to locate a unit request to be processed: a request for a public volume if public requests are being processed; or a request for a storage volume if storage requests are being processed. If no eligible volunit entries are located, IEFAB436 selects another SIOT – see step 1.	IEFAB436	

Diagram 14-5. IEFAB436 – Nonspecific Volume Allocation Control (Part 3 of 6)



3 Build or update list of UCBs eligible to this volunit entry (PSLIST).

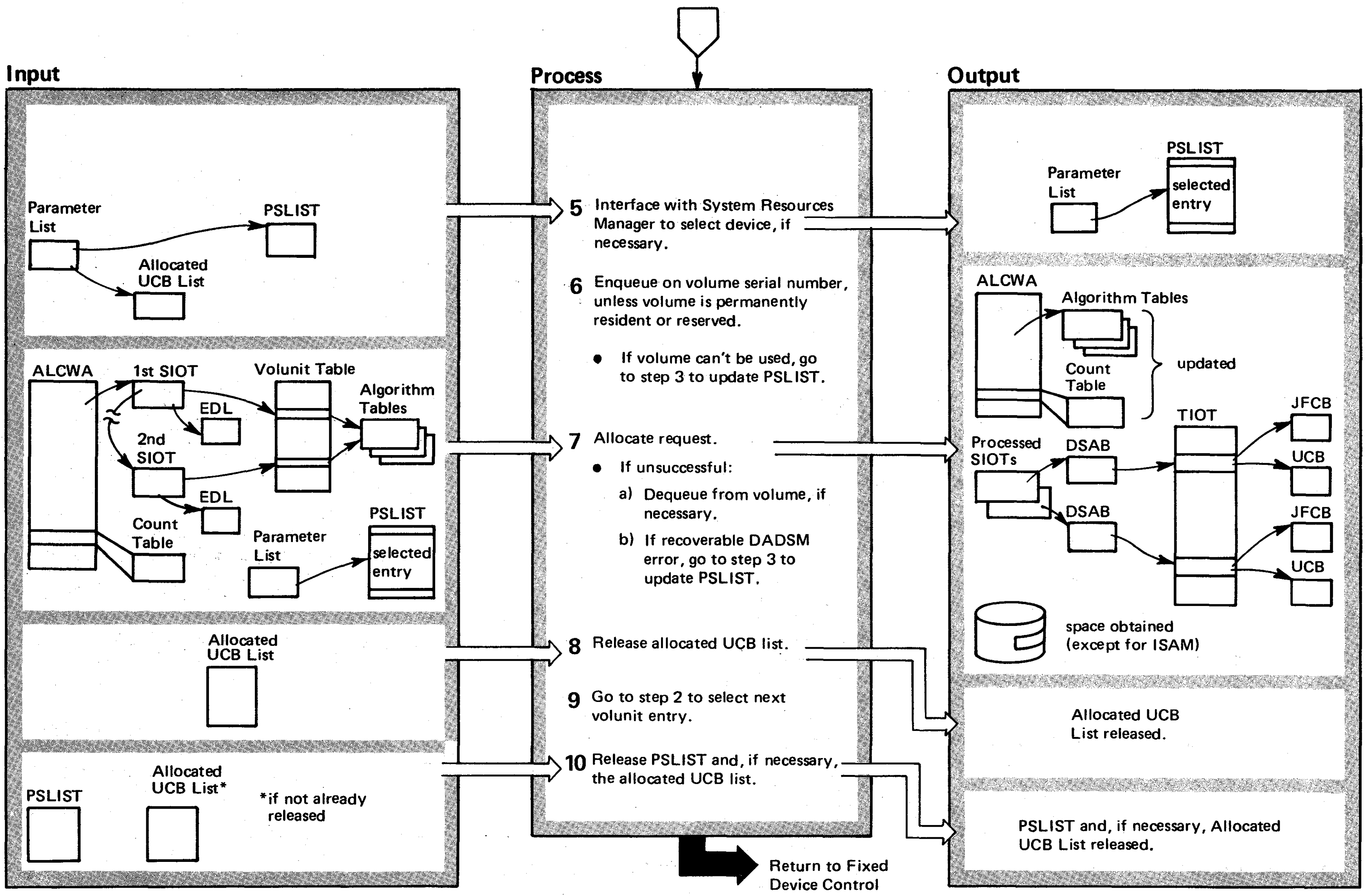
- If PSLIST does not contain entries:
  - a) Change storage requests to private, if necessary.
  - b) Go to step 1 to select next SIOT.

4 Build allocated UCB list, if necessary.

Diagram 14-5. IEFAB436 – Nonspecific Volume Allocation Control (Part 4 of 6)

Extended Description	Module	Segment
3 The purpose of this step is to build a list of units eligible to the volunit entry being processed – a *public/storage list (PSLIST)*. The unit allocated to the volunit entry is chosen from this PSLIST.		
Because the processing of IEFAB436 is a series of loops, this step can be performed after step 2 – the first eligible volunit entry for a SIOT is initially being processed; after steps 6 or 7 – the processing of a volunit entry is being repeated because of a recoverable error; or after step 9 – an additional volunit entry is being processed for this SIOT. Depending on when this step is being performed, the PSLIST is either created or updated:		
a) The PSLIST is created if this volunit entry is the first to be processed for this SIOT – that is, this step receives control from step 2. In this case, input to this step is a UCB candidate list of available units that do not contain private volumes (if the caller is Allocation Within Generic or Recovery Allocation) or a list of UCBs containing permanently resident or reserved direct access volumes (PRLIST, if the caller is Fixed Device Control).	IEFAB436	PSLSTBLD
After the list is created, IEFAB436 determines if it contains sufficient units to allocate all the units required by this SIOT. If it does not, the entries in the PSLIST are deleted and no requests are allocated unless one of the following conditions is true:	IEFAB436	PSLSTMNT
• The caller is Recovery Allocation.		
• The request is eligible to more than one generic and can be allocated across generics.		
b) The PSLIST is updated in the following cases:		
• A volunit entry was just allocated for this SIOT and another volunit entry is being processed. The PSLIST is updated to eliminate the unit just allocated. In addition, if the PSLIST includes units from different generic device types and allocation across generics is not allowed, IEFAB436 eliminates all entries in the PSLIST that represent a device type different from that just allocated. The EDL is also updated to eliminate those generics that are no longer eligible.	IEFAB436 IEFAB436 IEFAB436	PSLSTMNT PSLSTBLD PSLSTMNT
• A unit was selected from the PSLIST to be allocated to this volunit entry, but the volume on the unit could not be used because of a volume enqueue error – see step 6. IEFAB436 eliminates the entry for this unit from the PSLIST.	IEFAB436	PSLSTMNT
• A unit was selected from the PSLIST to be allocated to this volunit entry, but the volume on the unit did not contain sufficient space for the data set. IEFAB436 eliminates the entry for this unit from the PSLIST. (This unit, however, can be considered for other volunit entries for this SIOT, once a unit has been allocated to the current volunit entry.)	IEFAB436	PSLSTMNT
It is possible, after the PSLIST is created or updated, that it does not contain entries. If storage requests are being processed, IEFAB436 changes all storage volunit entries remaining to be allocated to private requests. Any other allocated volunit entries for this SIOT that require the same volume and unit are also changed to private. IEFAB436 also updates the PUBLREQS and PVTNREQS fields in the count table.	IEFAB436	PVTUPDTE
If there are no entries in the PSLIST, no further processing for this SIOT can be performed at this time. IEFAB436 selects the next SIOT – see step 1.		
4 The purpose of this step is to build a list of allocated units. The System Resources Manager uses the allocated UCB list and the PSLIST to determine which unit should be allocated to a request. This step is not performed in either of the following situations:	IEFAB436	PSVOLUN
• The PSLIST (built or updated in step 3) contains only one entry. There is no choice of units and, therefore, no need to interface with the System Resources Manager.		
• An allocated UCB list already exists and can be reused; this is true if the processing of a volunit entry is being repeated due to a volume enqueue or DADSM error.		
IEFAB440 (Build Allocated UCB List) builds the allocated UCB list by obtaining the UCB addresses from TIOT entries.	IEFAB440	

Diagram 14-5. IEFAB436 – Nonspecific Volume Allocation Control (Part 5 of 6)



5 Interface with System Resources Manager to select device, if necessary.

6 Enqueue on volume serial number, unless volume is permanently resident or reserved.

- If volume can't be used, go to step 3 to update PSLIST.

7 Allocate request.

- If unsuccessful:
  - a) Dequeue from volume, if necessary.
  - b) If recoverable DADSM error, go to step 3 to update PSLIST.

8 Release allocated UCB list.

9 Go to step 2 to select next volunit entry.

10 Release PSLIST and, if necessary, the allocated UCB list.

Return to Fixed Device Control (IEFAB430); IEFAB475 (See Generic Allocation Control (IEFAB471)); or Recovery Termination (IEFAB485)

Diagram 14-5. IEFAB436 – Nonspecific Volume Allocation Control (Part 6 of 6)

Extended Description	Module	Segment
5 IEFAB436 interfaces with the System Resources Manager, which selects a device to be allocated to this request. This step is not performed if the PSLIST contains only one entry.	IEFAB436	PSALLOC
6 If the volume on the selected unit is not permanently resident or reserved, IEFAB4F0 (Conditional ENQ/DEQ Routine) enqueues on the volume. The enqueue can result in one of the following situations: ● The enqueue is unsuccessful because the volume is already owned by this job. The volume can be used if the enqueue is share and no unallocated specific volume requests need this volume. ● The enqueue is unsuccessful because another user owns the volume; the volume cannot be used. ● The enqueue is successful; the volume can be used. If the volume cannot be used, the PSLIST must be rebuilt to exclude the entry for this unit and a new unit must then be selected – go to step 3.	IEFAB4F0	
7 IEFAB434 (Allocate Request to Unit) allocates this request and any requests that specified affinity to it. For details, see the M.O. diagram Allocate Request to Unit (IEFAB434).	IEFAB434	
If the allocation is unsuccessful and the volume was enqueued, IEFAB4F0 dequeues from the volume. If the allocation is unsuccessful because of a DADSM error, the PSLIST is rebuilt to exclude the entry for this unit and a new unit is selected – go to step 3.	IEFAB4F0	
8 IEFAB436 issues a FREEMAIN macro instruction to release the allocated UCB list. **Note:** This list is not released if the request could not be allocated due to a volume enqueue or DADSM error; the list can be reused.	IEFAB436	PSALLOC
9 IEFAB436 selects the next volunit entry to be processed – see step 2.	IEFAB436	
10 After all eligible SIOTs have been processed, IEFAB436 issues a FREEMAIN macro instruction to release the storage obtained for the PSLIST. If the allocated UCB list has not been released, IEFAB436 also releases it. (The Allocated UCB list will not have been released if a request was not allocated because of an enqueue or DADSM error and, when the PSLIST was rebuilt, it contained no entries.)	IEFAB436	

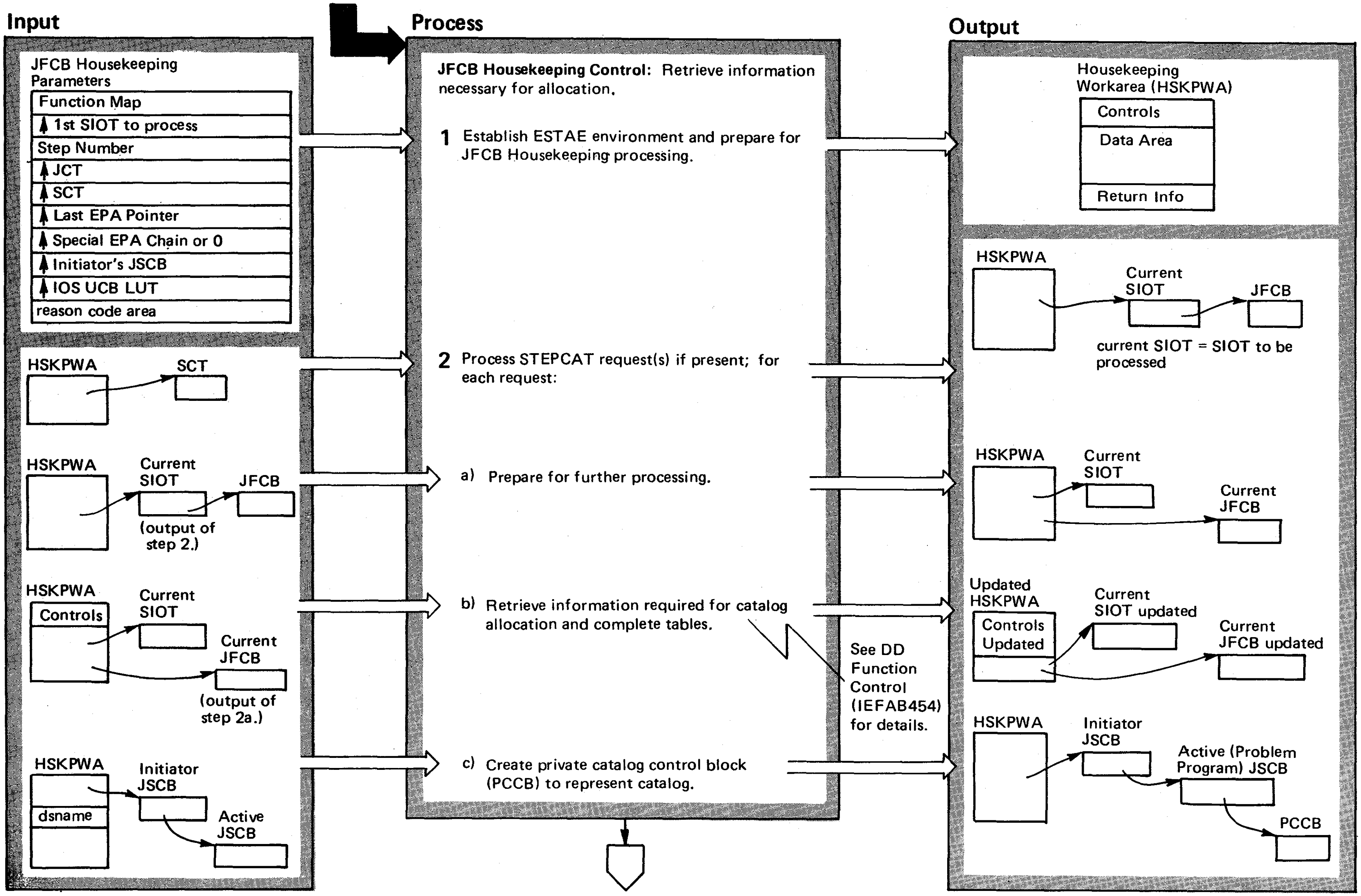
Error Processing

An unrecoverable error in any routine causes control to be returned to the calling routine.

In the event of an abnormal termination, the ESTAE exit routine established by IEFAB421 performs any necessary cleanup.

Diagram 14-6. IEFAB451 – JFCB Housekeeping Control (Part 1 of 6)

ENTRY from IEFBB404 (see IEFBB401 – Initiator/Allocation Interface)
or
IEFDB413 (see IEFDB410 – Dynamic Allocation Control)



JFCB Housekeeping Control: Retrieve information necessary for allocation.

1 Establish ESTAE environment and prepare for JFCB Housekeeping processing.

2 Process STEPCAT request(s) if present; for each request:

a) Prepare for further processing.

b) Retrieve information required for catalog allocation and complete tables.

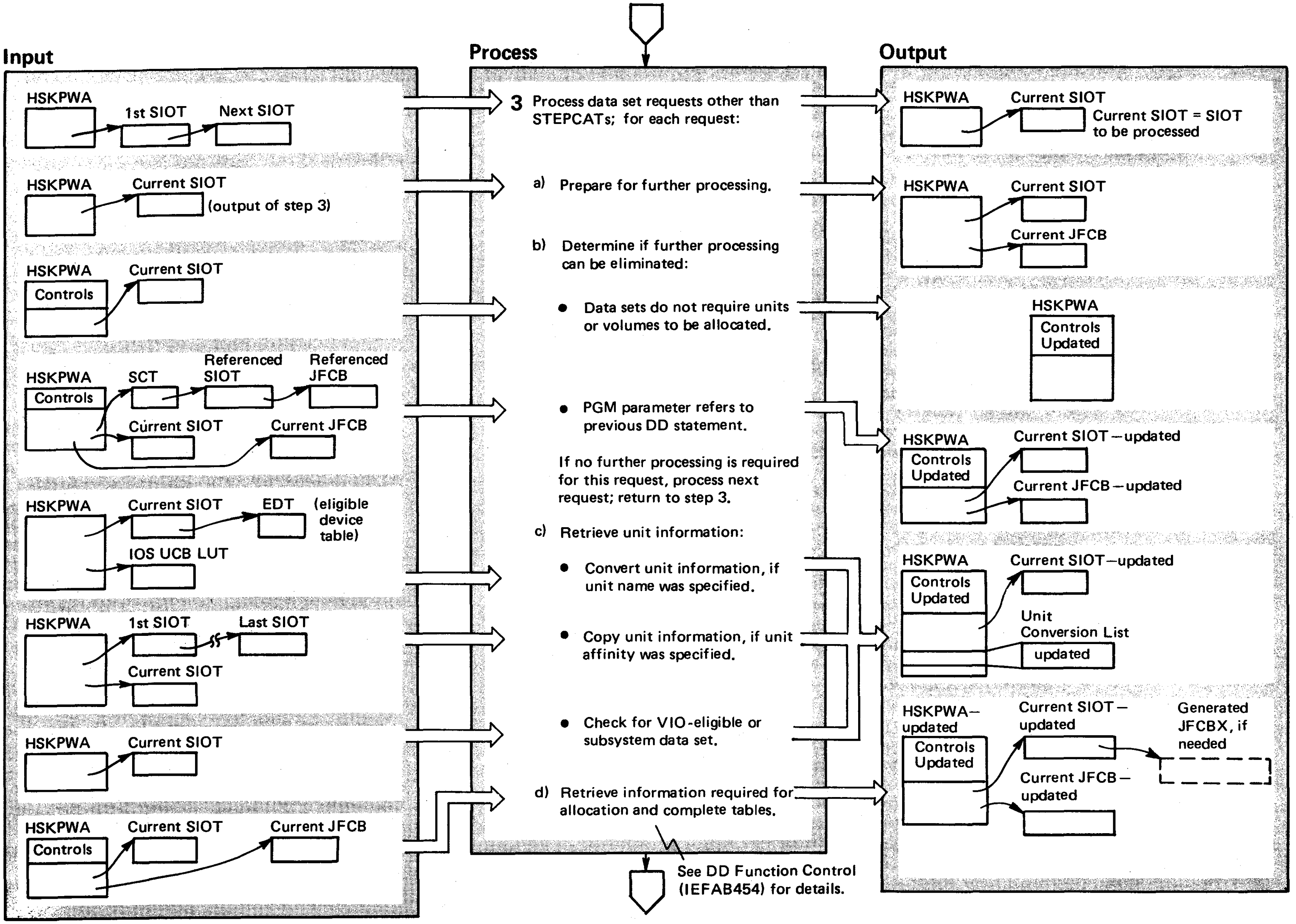
See DD Function Control (IEFAB454) for details.

c) Create private catalog control block (PCCB) to represent catalog.

Diagram 14-6. IEFAB451 – JFCB Housekeeping Control (Part 2 of 6)

Extended Description	Module	Segment
ENTRY JFCB Housekeeping Control, called by either Step Allocation Control (IEFBB404) or Normal Dynamic Allocation (IEFDB413), is responsible for retrieving the information necessary to allocate each request. The only functions actually performed by JFCB Housekeeping Control are initialization and clean-up. To process the requests, JFCB Housekeeping Control calls other routines. DD Function Control, which retrieves necessary information and completes tables (SIOTs, JFCBs, and JFCBXs), is described in detail in the M.O. diagram DD Function Control (IEFAB454). Input to JFCB Housekeeping Control is the parameter list created by Dynamic Allocation Control or Step Allocation Control. In the parameter list, the pointer to the special EPA chain is passed only from Dynamic Allocation Control; it is used for SIOTs, JFCBs, and JFCBXs created by JFCB Housekeeping. The pointer to the last EPA is used for updated tables (such as the JCT), when housekeeping is called by dynamic allocation; and for generated SIOTs, JFCBs, and JFCBXs, as well as the JCT, when housekeeping is called by step allocation. The function map is illustrated in figure 18.		
1 After establishing an ESTAE environment, JFCB Housekeeping Control issues a GETMAIN macro instruction to obtain space for the housekeeping workarea (HSKPWA) and places the JFCB Housekeeping parameter list into HSKPWA. The HSKPWA includes a control area that indicates what processing should be performed; it consists of global controls, local controls, and counters. Global controls are set according to the input function map and pertain to all data set requests to be processed during this invocation of JFCB Housekeeping. Local controls are set by the individual routines and pertain only to the current SIOT (the specific SIOT being processed; SIOTs are processed one at a time); they are turned off as the functions they indicate are performed. Global controls always override local controls if indicators in each conflict. The counters are used to monitor the processing of generated SIOTs in the case of DSN recursion or volume/unit recursion. For details on HSKPWA, see *OS/VS2 Data Areas*, SY38-0606.	IEFAB451	HSKPINIT
2 DD Processing Control is responsible for selecting SIOTs to be processed; one SIOT is completely processed before the next SIOT is selected. DD Processing Control first selects STEPCAT requests, if present. (Note: JOBCAT requests are treated as STEPCAT requests; each JOBCAT DD statement is propagated to every step in the job that does not include a STEPCAT DD statement.) In the SCT, the SCTPCAT field contains a pointer to the first STEPCAT request and the SCTCATCT field contains the number of STEPCAT requests. (STEPCAT requests are chained together within the SIOT chain.) For each STEPCAT request:	IEFAB452	
a) DD Preparation places the address of the JFCB for the current DD request (SIOT) into the HSKPWA.	IEFAB453	
b) DD Function Control controls the retrieval of required volume and unit information. For details, see the M.O. diagram DD Function Control (IEFAB454).	IEFAB454	
c) The PCCB Routine creates a private catalog control block (PCCB) for the STEPCAT request and adds it to the chain of PCCBs for this step.	IEFAB4EF	FINDPCCB

Diagram 14-6. IEFAB451 – JFCB Housekeeping Control (Part 3 of 6)



3 Process data set requests other than STEPCATs; for each request:

a) Prepare for further processing.

b) Determine if further processing can be eliminated:

- Data sets do not require units or volumes to be allocated.
- PGM parameter refers to previous DD statement.

If no further processing is required for this request, process next request; return to step 3.

c) Retrieve unit information:

- Convert unit information, if unit name was specified.
- Copy unit information, if unit affinity was specified.
- Check for VIO-eligible or subsystem data set.

d) Retrieve information required for allocation and complete tables.

See DD Function Control (IEFAB454) for details.

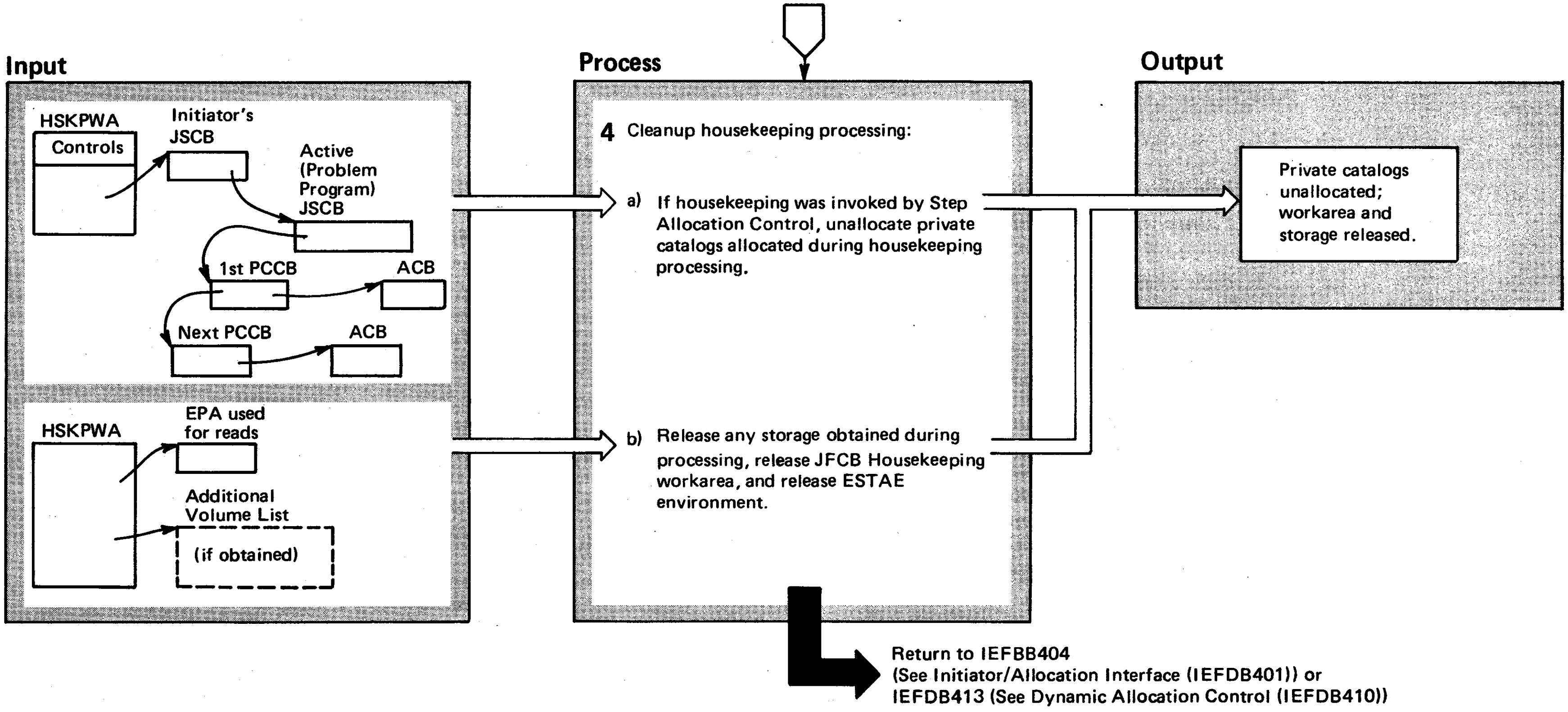
Diagram 14-6. IEFAB451 – JFCB Housekeeping Control (Part 4 of 6)

Extended Description	Module	Segment
3 After STEPCAT requests are processed, DD Processing Control selects remaining requests, one at a time, for processing; each SIOT is completely processed before the next is selected. When all SIOTs have been processed, control is returned to JFCB Housekeeping Control for clean-up processing (step 4). For each SIOT:	IEFAB452	
a) DD Preparation places the address of the JFCB for the current SIOT into the HSKPWA.	IEFAB453	
b) DD Preparation determines if any further processing can be eliminated:	IEFAB453	
• If QNAME (SIOTQNAM=1 in the SIOT) or TERM=TS (SIOTTERM=1 in the SIOT) was specified for this request, DD Preparation sets the local controls to indicate that no further processing is required (HWDDDONE=1). If the request is a dummy data set (DUMMY or DSN=NULLFILE was specified; SCTDUMMY=1 in the SIOT), a subsystem data set (for example, sysin or sysout; SIOTSSDS=1 in the SIOT), or a VIO data set (SIOVAMDS=1 in the SIOT; for checkpoint restart only), DD Preparation indicates in the local controls that Dsname Resolution is not required (HWDSNRQD=0).	IEFAB453	FASTPATH
• When PGM=*.stepname.ddname or PGM=*.procstepname.ddname was specified, DD Preparation calls the SWA Manager Interface to read the SIOT and JFCB of the referenced DD statement; the SCTGOTTR field in the SCT contains the SWA virtual address (SVA) of the referenced SIOT. DD Preparation copies unit, volume, and data set information from the referenced SIOT and JFCB to the current SIOT and JFCB and sets the local controls to indicate no further processing is required (HWDDDONE=1). If the referenced SIOT was not allocated, processing is terminated.	IEFAB453 IEFAB4F7 IEFAB453	FETCHLIB FETCHLIB
If no further processing is indicated (HWDDDONE=1), DD Preparation returns to DD Processing Control, which selects the next SIOT (step 3).	IEFAB453	

Extended Description	Module	Segment
c) DD Preparation is responsible for retrieving unit information, if unit information was not previously converted (SIOUCNVT=0, in the event of check point restart):	IEFAB453	
• If the first subparameter of the UNIT parameter was coded (*i.e.,* a unit address, device type or group name was specified), Unit Name Conversion searches:	IEFAB470	
– The eligible device table (EDT) for a matching unitname. If a match is found, Unit Name Conversion places the EDT look-up value (LUV) in the unit conversion list in HSKPWA and sets local controls to indicate: the unit was converted from the EDT (HWEDT=1); the unit is VIO eligible (HWVAME=1) if the EDT LUV is VIO eligible; the unit is an override candidate (HWOVCAND=1), if the matching unitname in the EDT consists of only one generic device type. The generic device type is also placed in the unit conversion list. (Note: The unit information is placed in the SIOT by IEFAB464 – see the M.O. diagram DD Function Control (IEFAB454).		
– The UCBs (by means of the IOS UCB LUT), if a matching unitname was not found in the EDT. Unit Name Conversion searches the UCBs for a unit address that matches the specified unit information. If a match is found, Unit Name Conversion places the device type and UCB address in the unit conversion list in HSKPWA and sets local controls to indicate: the unit was converted from a UCB (HWUCB=1); the unit is an override candidate (HWOVCAND=1).	IEFAB470	
If the specified unit information is not found in the EDT or in a UCB, processing is terminated.		

Step 3 continued on Part 6

Diagram 14-6. IEFAB451 – JFCB Housekeeping Control (Part 5 of 6)



4 Cleanup housekeeping processing:

a) If housekeeping was invoked by Step Allocation Control, unallocate private catalogs allocated during housekeeping processing.

b) Release any storage obtained during processing, release JFCB Housekeeping workarea, and release ESTAE environment.

Return to IEFBB404
(See Initiator/Allocation Interface (IEFDB401)) or
IEFDB413 (See Dynamic Allocation Control (IEFDB410))

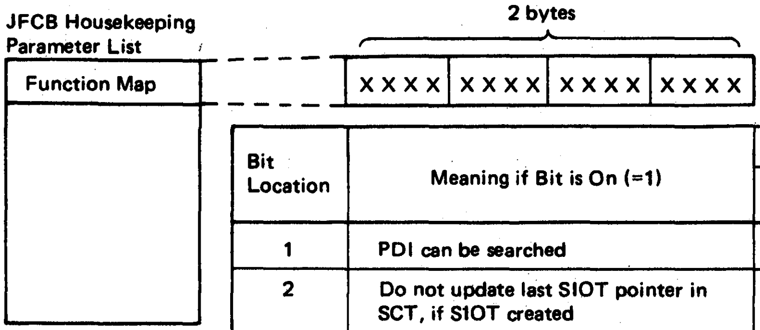
Diagram 14-6. IEFAB451 – JFCB Housekeeping Control (Part 6 of 6)

Extended Description	Module	Segment
3 c) continued		
• If unit affinity was specified, DD Preparation locates the referenced SIOT by comparing the affinity-DD number in the current SIOT (SIOTUNAF field) to the DD numbers of the SIOTs in the SIOT chain (SCTDDINO field). The following processing occurs: – If the referenced SIOT contains converted unit information, DD Preparation copies it into the unit conversion list in HSKPWA. – If the referenced SIOT indicates a VIO data set (SIOVAMDS=1), DD Preparation indicates in the local controls that the unit is VIO-eligible (HWVAME=1).	IEFAB453	SRCHSIOT
– If the referenced SIOT indicates a subsystem data set (SIOTSSDS=1), Unit Name Conversion converts the unitname SYSALLDA and places the converted information into the unit conversion list in HSKPWA.	IEFAB470	
d) DD Function Control controls the retrieval of volume and unit information required for allocation; if necessary, DD Function Control also generates a JFCBX(s). For details, see the M.O. diagram "IEFAB454 – DD Function Control."	IEFAB454	
4	IEFAB451	HSKPCLUP
a) If JFCB Housekeeping was called by Step Allocation Control and private catalogs were allocated during housekeeping processing (see the M.O. diagram JLOCATE (IEFAB469)):		
• Close Private Catalog (a data management routine) closes the catalogs.	IDACAT12	
• Unallocate Private Catalog Routine issues SVC 99 to unallocate the catalogs.	IEFAB4F4	
• The PCCB Routine releases the private catalog control blocks (PCCBs).	IEFAB4EF	FINDPCCB
The active (problem program's) JSCB is used to determine if any private catalogs have been allocated (the pointer to the PCCB chain does not equal 0).		
b) JFCB Housekeeping Control issues FREEMAIN macro instructions to release any storage obtained during housekeeping processing (for example, storage for a volume list if the CRI is too small), to release the housekeeping workarea (HSKPWA), and to release the ESTAE environment.	IEFAB451	HSKPCLUP

Error Processing

In general, an error occurring in any routine causes control to be returned to the calling routine with appropriate return and reason codes. Return and reason codes are listed in Section 6, Diagnostic Aids. Errors occurring in steps 1-3 cause control to be passed to step 4.

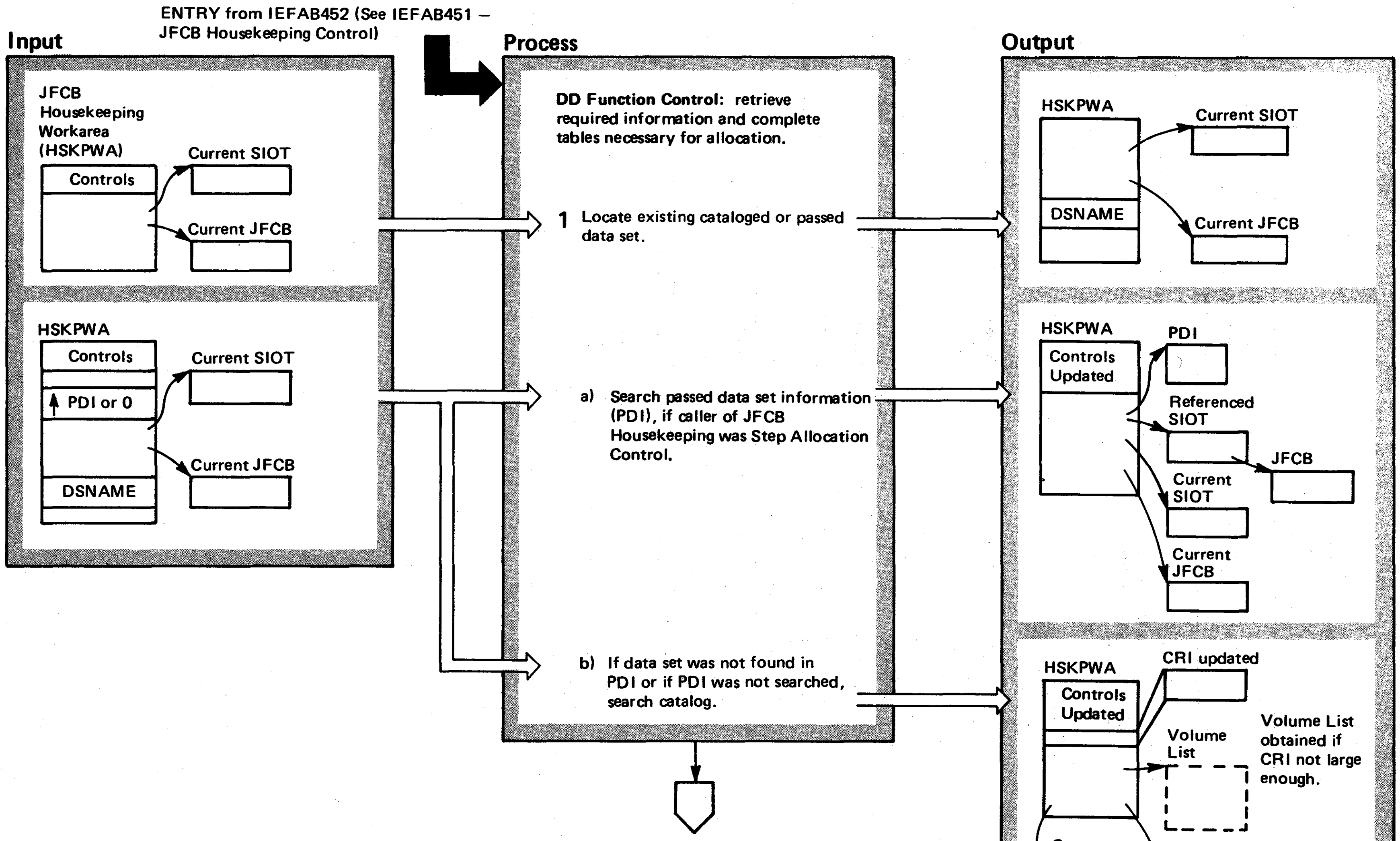
When IEFAB451 receives control, it creates an ESTAE environment so that its exit routine receives control if an abnormal termination occurs.



Bit Location	Meaning if Bit is On (=1)	Conditions when Bit is On (=1)	
		Caller is Step Allocation Control	Caller is Dynamic Allocation Control
1	PDI can be searched	Always	Never
2	Do not update last SIOT pointer in SCT, if SIOT created	Never	Always
3	Catalogs may be mounted	Always	Depends on what user specified
4	Wait for units during catalog allocation	Always	Depends on what user specified
5	Perform catalog recovery	Always	Never
6	Do not create SIOT and JFCB for catalogs	Never	Always
7	Wait for volumes during catalog allocation	Always	Depends on what user specified
8	Do not process JOBCATs/STEPCATs	Never	Always
9	Consider offline devices during catalog allocation	Always	Depends on what user specified
10	Do not enqueue on TIOT	Never	Always
11	Change active JSCB to allocate catalog to initiator	Always	Never
12	Add EPA to chain if JCT is updated	Never	Always
13	Bypass data set integrity ENQ	Never	Depends on what user specified
14	Program authorized to bypass data set integrity if no JOBLIB or STEPLIB	If program is authorized	Never
15-16	Reserved	———	———

Figure 2-28. Function Map of JFCB Housekeeping Parameter List

Diagram 14-7. IEFAB454 – DD Function Control (Part 1 of 12)



DD Function Control: retrieve required information and complete tables necessary for allocation.

1 Locate existing cataloged or passed data set.

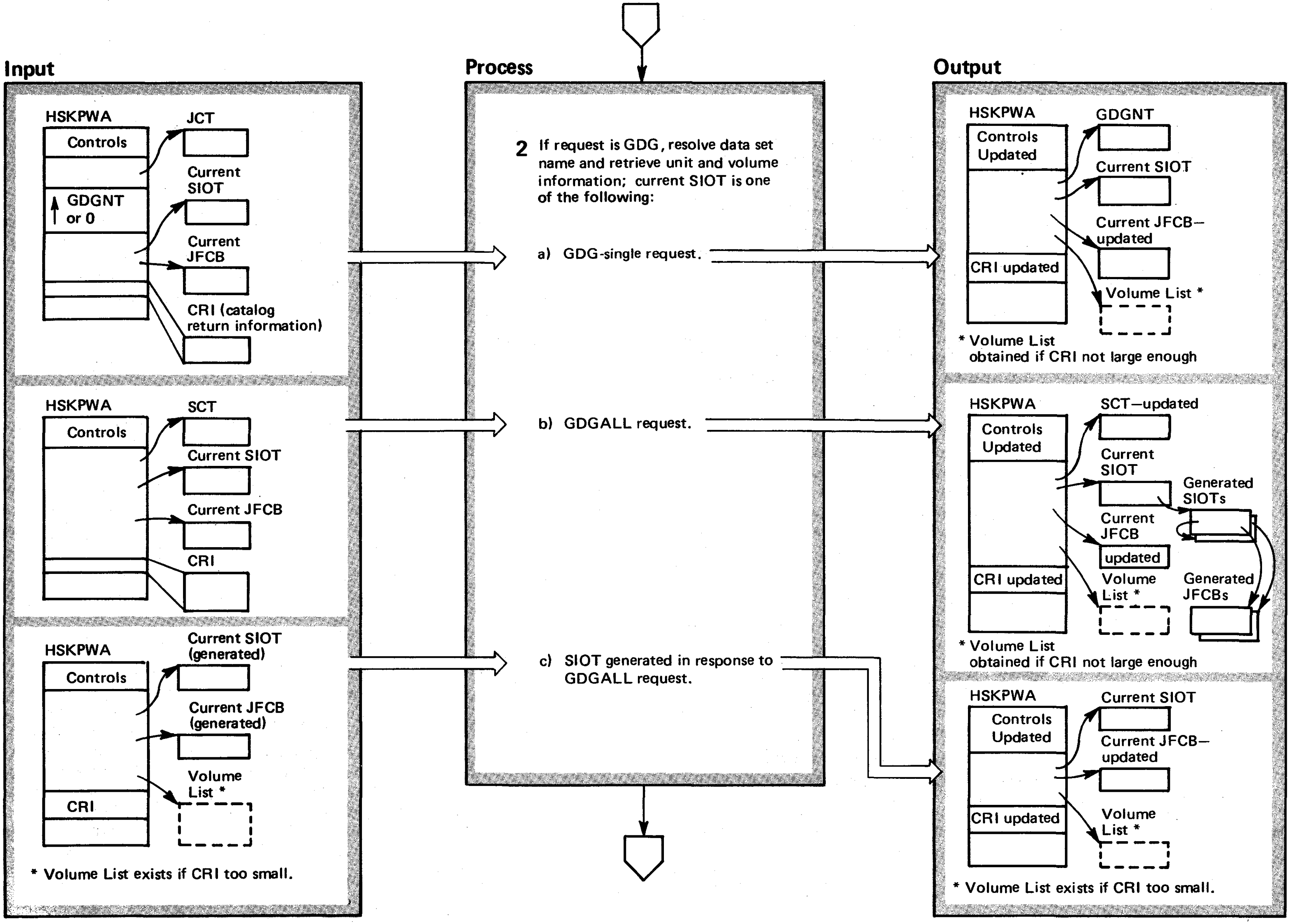
a) Search passed data set information (PDI), if caller of JFCB Housekeeping was Step Allocation Control.

b) If data set was not found in PDI or if PDI was not searched, search catalog.

Diagram 14-7. IEFAB454 – DD Function Control (Part 2 of 12)

Extended Description	Module	Segment
ENTRY DD Function Control, called by DD Processing Control (IEFAB452; see the M.O. diagram JFCB Housekeeping Control (IEFAB451)), determines what additional information is needed to allocate a request, obtains that information, and places it in tables to be used by allocation. Every SIOT that does not complete processing during DD Preparation (HWDDDONE=1 in the local controls if no further processing is required) is processed by DD Function Control. However, not all the steps in DD Function Control are performed for every SIOT – the type of request (for example, GDGALL request, existing cataloged data set) determines what DD Function Control will do to retrieve needed information. The extended description for each step describes when that step is performed.		
In general, steps 1-4 are concerned with retrieving unit and volume information; step 5 copies unit and volume information into the SIOT, JFCB, and, if needed, JFCB extension (JFCBX); steps 6 and 7 complete DCB and DISP information in the JFCB and SIOT.		
1 When this step is processed, two functions are performed:	IEFAB454	ESTABDSN
• This step determines if a request is GDGALL (all levels of a generation data group are requested).		
• If the request is not GDGALL, this step retrieves data set name, volume, and unit information from the PDI (passed data set information) or from a catalog. (Dsname, volume, and unit information for a GDGALL request is retrieved in step 2b.)		
This step is processed only if the following conditions are true:		
• The unit requested is tape or direct access (HWTAPE=1 or HWDA=1 in the local controls).		
• The data set is not a single generation data set (SCTSGDGS=0 in the SIOT), or it is a single generation data set at restart. (Dsname, volume, and unit information for a single generation data set is retrieved in step 2a.)		
• The data set disposition is not new (SCTSNEW=0 in the SIOT).		
• Volume information is not specified: by explicit volume serial numbers, or by a volume reference (SCTVOLAF=0 and SCTVOLCT=0 in the SIOT), or by volume serials which were retrieved from the catalog.		
• The SIOT being processed is not a SIOT generated in response to a GDGALL request (DSN recursion; HWDSNREC=0 in the local controls) or a SIOT generated in response to a request for a data set residing on more than one device type (volume/unit recursion; HWVUREC=0 in the local controls).		
• The data set is not a subsystem, or dummy data set (HWDSNRQD=1 in the local controls).		
DD Function Control calls JLOCATE to search the PDI and/or catalog(s) for the required information, or to update the PDI and allocate a VSAM private catalog or CVOL. For details on the processing of JLOCATE, see the M.O. diagram JLOCATE (IEFAB469).	IEFAB469	

Diagram 14-7. IEFAB454 – DD Function Control (Part 3 of 12)



2 If request is GDG, resolve data set name and retrieve unit and volume information; current SIOT is one of the following:

a) GDG-single request.

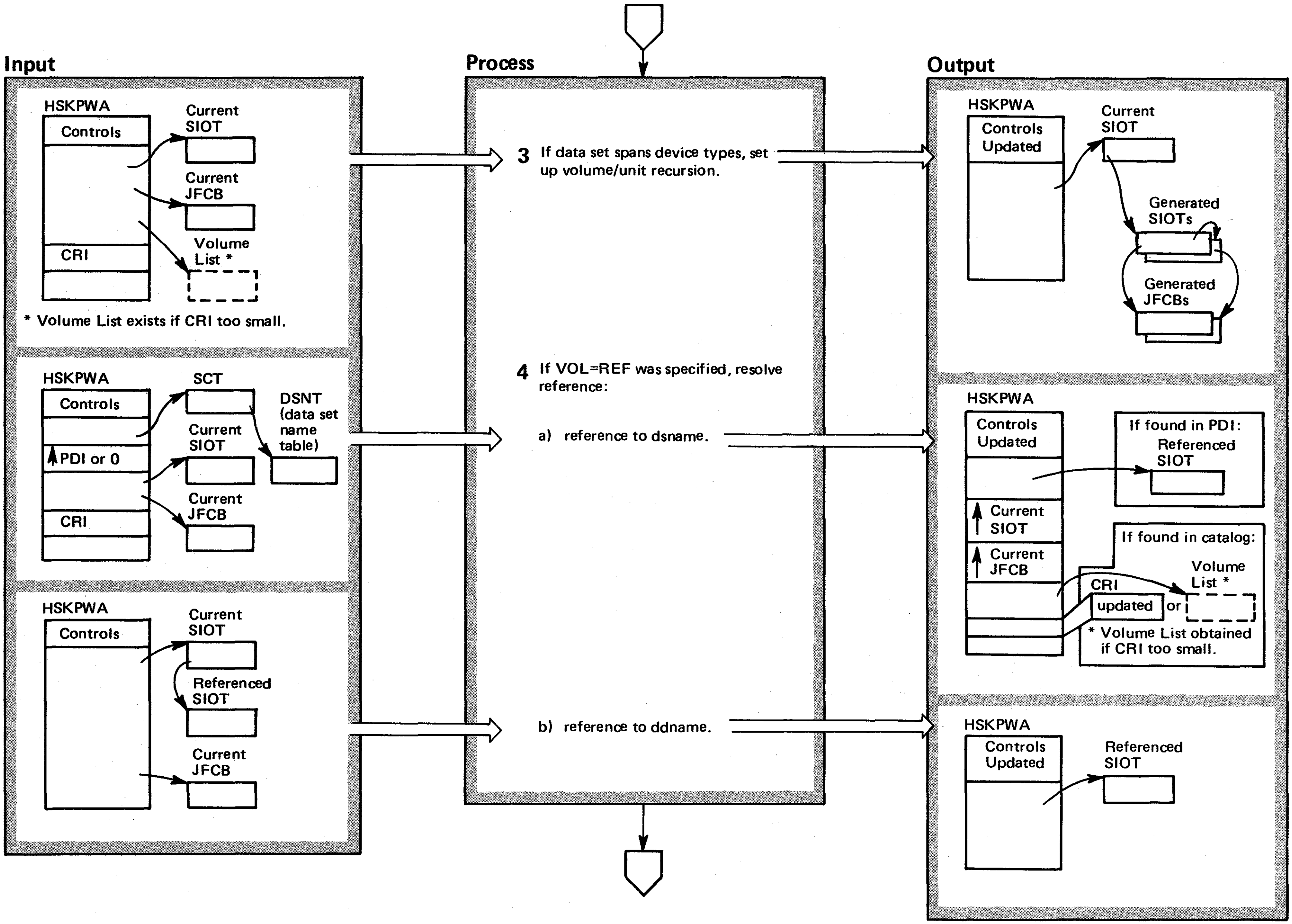
b) GDGALL request.

c) SIOT generated in response to GDGALL request.

Diagram 14-7. IEFAB454 – DD Function Control (Part 4 of 12)

Extended Description	Module	Segment
2 The purpose of this step is to obtain the fully-qualified dsname of a generation data group (GDG) and to locate volume and unit information for the GDG request. A SIOT processed by this step is: a) a SIOT representing a GDG-single request (SCTSGDGS=1 in the SIOT); or, b) a SIOT representing a GDGALL request (HWGDGALL=1 in the local controls); or, c) a SIOT generated in response to a GDGALL request (DSN recursion; HWDSNREC=1 in the local controls). The following processing is performed:	IEFAB456	
a) For a SIOT representing a GDG-single request, GDG Single Processing:	IEFAB461	
• Checks the data set name for correct syntax. If the base name (not including the level number) is greater than 35 characters, control is returned to the caller and processing is terminated.	IEFAB461	
• Obtains the base level of the GDG:	IEFAB461	GNTLCUPD
– If any GDG name tables (GDGNTs) exist for the job (JCTGDGNT≠0), GDG Single Processing searches the GDGNTs for the dsname. (If	IEFAB461	GNTSCAN
HSKPWA does not include a pointer to the GDGNT(s), the SWA manager is called to read the GDGNT(s) for the job and a pointer is placed in HSKPWA.)	IEFAB461 IEFAB4F7	GNTRDLOC
– If the dsname is not found in a GDGNT or if no GDGNTs exist for the job, GDG Single Processing calls JLOCATE to obtain the base level. The processing of JLOCATE is described in the M.O. diagram "IEFAB469 – JLOCATE." If JLOCATE returns with the base level, an entry is created in	IEFAB461 IEFAB469	
a GDGNT, which itself is created if necessary. If JLOCATE is unable to locate the base level, processing terminates; IEFAB454 returns to the caller.	IEFAB4F7	
• Calls JLOCATE to obtain the fully-qualified data set name and unit and volume information for the data set. JLOCATE is described in the M.O. diagram JLOCATE (IEFAB469).	IEFAB469	
b) For a SIOT representing a GDGALL request, two functions are performed:	IEFAB456	
• Tables are created for all the levels of the GDG except the zero level. DSN Resolution turns on the DSN recursion indicator in the local controls (HWDSNREC=1) and updates the counter in the control area with the number of SIOT/JFCB pairs to be created (HWDDSTEP=n).	IEFAB456	GDGACODE
Table Creation generates the required number of SIOTs and JFCBs and chains them to the zero level SIOT.	IEFAB466	
DD Processing Control (see the M.O. diagram JFCB Housekeeping Control (IEFAB451)) will select the generated SIOTs, one at a time, for processing immediately after the SIOT representing the zero-level of the GDG is completely processed.	IEFAB452	
• JLOCATE obtains the fully-qualified dsname and unit and volume information for the zero-level of the GDG. For details on JLOCATE, see the M.O. diagram JLOCATE (IEFAB469).	IEFAB469	
c) For a SIOT generated in response to a GDGALL request, DSN Resolution:	IEFAB456	
• Increases the generated-DD counter (HWDDCTR) in the control area of HSKPWA by 1.		
• Modifies the dsname in the JFCB to appear as a request for the desired level of the GDG; the relative generation number is the negative of the generated-DD counter.		
• Turns off the DSN recursion indicator (HWDSNREC=0) and sets the counters in the control area to 0 if this is the last generated SIOT – that is, if the generated-DD counter (HWDDCTR) equals the number of DDs generated (HWDDSTEP).		
• Calls JLOCATE, which obtains the fully-qualified data set name and unit and volume information for the data set. For details on JLOCATE, see the M.O. diagram JLOCATE (IEFAB469).	IEFAB469	

Diagram 14-7. IEFAB454 – DD Function Control (Part 5 of 12)



3 If data set spans device types, set up volume/unit recursion.

4 If VOL=REF was specified, resolve reference:

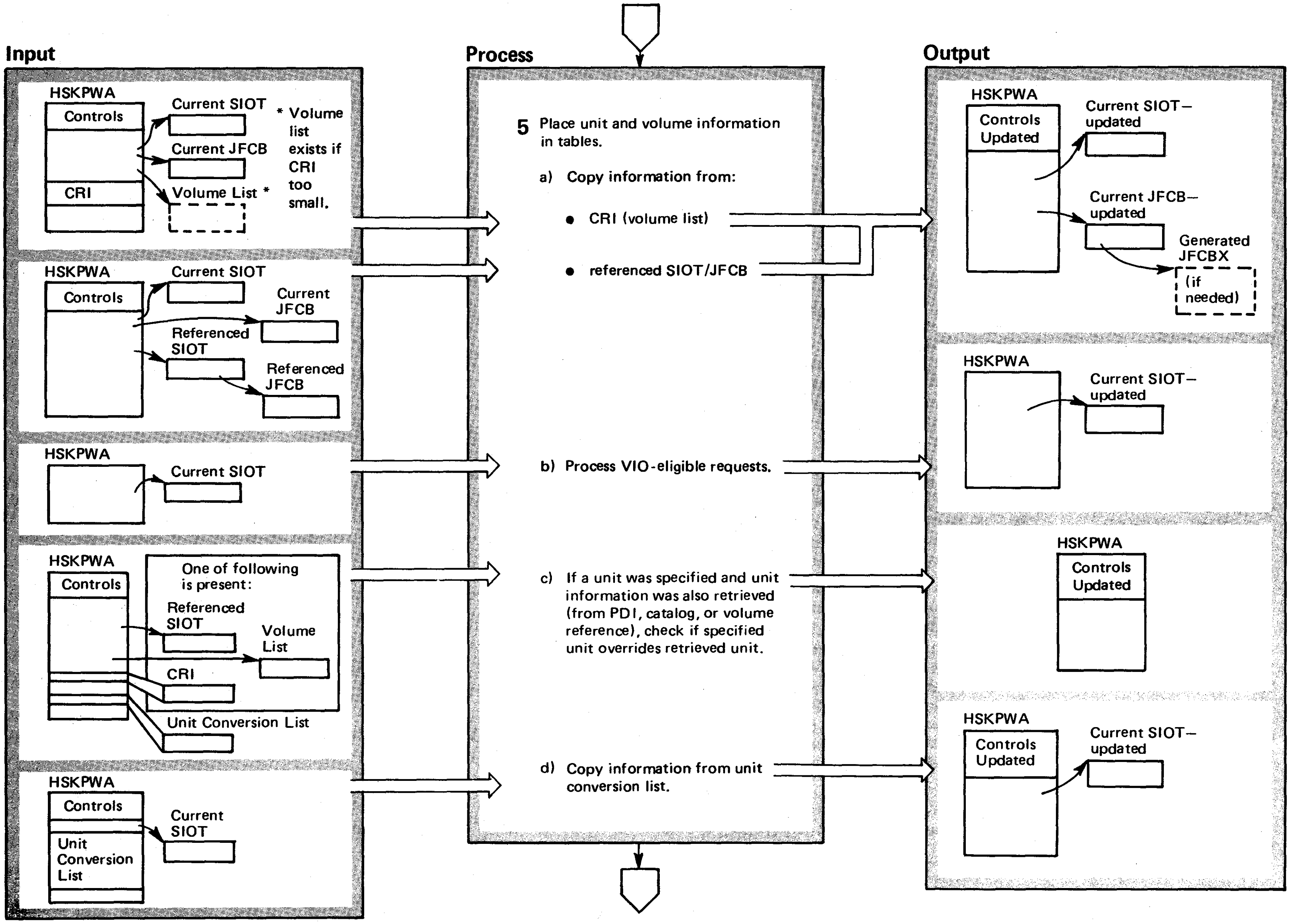
a) reference to dsname.

b) reference to ddname.

Diagram 14-7. IEFAB454 – DD Function Control (Part 6 of 12)

Extended Description	Module	Segment
3 The purpose of this step is to determine if volume/unit recursion is necessary. If the data set was located in a catalog (HWDSNCAT=1 in the local controls, set by JLOCATE), the data set resides on more than one volume, and the current SIOT was not generated in response to a GDGALL request (HWDSNREC=0), Multiple Device Type Determination searches the CRI (volume list) for a change in device type. If more than one device type is found, Volume/Unit Resolution sets the generated-DD counter in the control area to the total number of different device types minus 1 and sets the volume/unit recursion indicator in the local controls (HWVUREC=1). Table Creation creates SIOT/JFCB pairs for the total number of different device types minus 1 and chains them to the current SIOT. DD Processing Control (see the M.O. diagram JFCB Housekeeping Control (IEFAB451)) selects the generated SIOTs, one at a time, for processing immediately after this SIOT is completely processed.	IEFAB463 IEFAB466 IEFAB452	
4 If VOL=REF was specified, Volume/Unit Resolution locates the source of volume and unit information:	IEFAB457	VOLREF
a) For a reference to a dsname (SCTDSNRF=1 in the SIOT), Volume/ Unit Resolution reads the data set name table (DSNT) to obtain the dsname. The SCTADSTB field of the SCT contains the SWA virtual address (SVA) of the DSNT for this step. JLOCATE searches the PDI and/or the catalog for the dsname. (For details on JLOCATE, see the M.O. diagram JLOCATE (IEFAB469)). If JLOCATE determines that the dsname is GDGALL, processing is terminated.	IEFAB469	
b) For a reference to a previous DD statement, either in this step (intra-step) or in a previous step (inter-step), Volume/Unit Resolution reads the SIOT of the referenced DD statement and places a pointer to it in HSKPWA. (The SWA virtual address (SVA) of the referenced SIOT is in the SIOTVRSB field of the current SIOT.) If the reference is inter-step, the JFCB and JFCBXs are read. If the reference is inter-step (SCTVREF=0 in the current SIOT) and the referenced data set was not allocated (SIOTALCD=0 in the referenced SIOT), processing is terminated. Volume/Unit Resolution also updates the local controls to indicate the referenced SIOT is present (HWRESIOT=1).	IEFAB457	VOLREF

Diagram 14-7. IEFAB454 – DD Function Control (Part 7 of 12)



5 Place unit and volume information in tables.

a) Copy information from:

- CRI (volume list)
- referenced SIOT/JFCB

b) Process VIO-eligible requests.

c) If a unit was specified and unit information was also retrieved (from PDI, catalog, or volume reference), check if specified unit overrides retrieved unit.

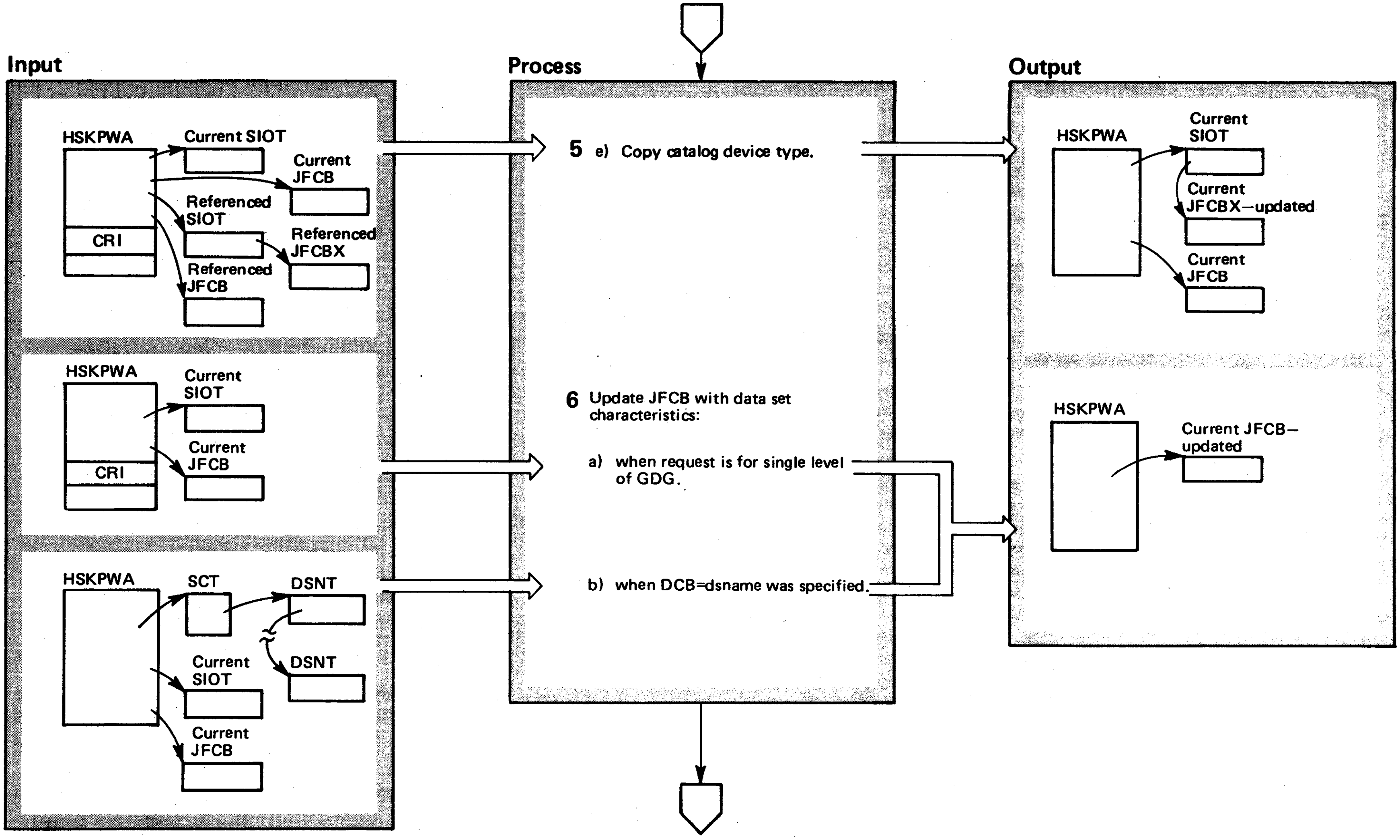
d) Copy information from unit conversion list.

Diagram 14-7. IEFAB454 – DD Function Control (Part 8 of 12)

Extended Description	Module	Segment
5 The purpose of this step is to: a) Place volume information and certain unit information retrieved in the previous steps into the SIOT and JFCB. b) Process VIO-eligible requests. c) Determine if specified unit information overrides retrieved unit information. d) Copy unit information into the SIOT. Step 5 is performed for every SIOT that enters DD Function Control and whose request hasn't already failed.		
a) Volume/Unit Table Completion copies retrieved volume information and certain unit information from a referenced SIOT or from the CRI (volume list) into the current SIOT and JFCB. The retrieved source of volume and unit information (CRI or referenced SIOT) was determined in a previous step:	IEFAB464	
• Unit and volume information exists in the CRI (or volume list) in three cases: – Volume/unit recursion is indicated (HWVUREC=1 in the local controls, set in step 3). – VOL=REF=dsname was specified and the dsname was found in the catalog (HWDSNCAT=1 in the local controls, set in step 4a). – A cataloged data set was located by searching the catalog (HWDSNCAT=1 in the local controls, set in step 1b or step 2).	IEFAB464	CRIRFCMP
• Unit and volume information is copied from a referenced SIOT in two cases: – Volume reference was coded to a ddname (HWRESIOT=1, set in step 4b). – The referenced data set was located in the PDI if VOL=REF=DSN (HWRESIOT=1, set in step 1a or 4a).	IEFAB464	DDREFCMP
Unit information is copied into a local field; for volume/unit recursion, Multiple Device Type Determination uses the generated-DD counter to reference the correct device type in the CRI. The only unit information placed in the SIOT at this time is unit count (SCTNMBUT).	IEFAB464 IEFAB463	
Volume information is copied into the JFCB. Volume/Unit Table Completion copies all the volume serial numbers from the CRI or referenced SIOT, except in two situations:	IEFAB464	
• If information is copied from the CRI and more than one device type exists (volume/unit recursion), volume serial numbers are copied only for one device type. (The generated-DD counter is used to reference the correct device type in the CRI.)	IEFAB464	CRIRFCMP
• If the data set resides on tape and VOL=REF was coded, only the last volume serial number is copied (if more than one exists), because a volume reference implies that the volume can be shared and tape volumes cannot be shared. (This precaution, however, does not guarantee that the data set can be successfully opened.) The volume count in the current SIOT is set to 1 for tape volume references.	IEFAB464	DDVLCOPY
Before copying the volume serial numbers, Volume/Unit Table Completion determines if a JFCB extension (JFCBX) is needed; up to five volume serial numbers can be placed in a JFCB and up to fifteen in a JFCBX. If more space is needed and sufficient JFCBXs were not generated by the Interpreter or by Dynamic Allocation, Volume/Unit Table Completion generates the required JFCBXs.	IEFAB464	JFCBXGEN
Volume/Unit Table Completion also updates the volume counts in the SIOT (SCTVOLCT) and the JFCB (JFCBNVOL).		
For volume reference to tape volumes, the following additional processing is performed:		
– If the referenced SIOT contains volume information that was itself copied from a previous SIOT, Volume/Unit Table Completion reads that SIOT and copies information from it.	IEFAB464	TAPEVREF
– If the device type is tape, Volume/Unit Table Completion updates the unit type to reflect the greatest device range that can satisfy this request. For example, the user specified 2400-4 for a data set with a density of 1600 bpi. Volume/Unit Table Completion updates the device type to 2400-3, which includes dual-density devices (which would have been included in 2400-4) *and* tape devices that can read only in 1600 bpi.	IEFAB464	TAPEONLY

Step 5 continued on Part 10

Diagram 14-7. IEFAB454 – DD Function Control (Part 9 of 12)



5 e) Copy catalog device type.

6 Update JFCB with data set characteristics:

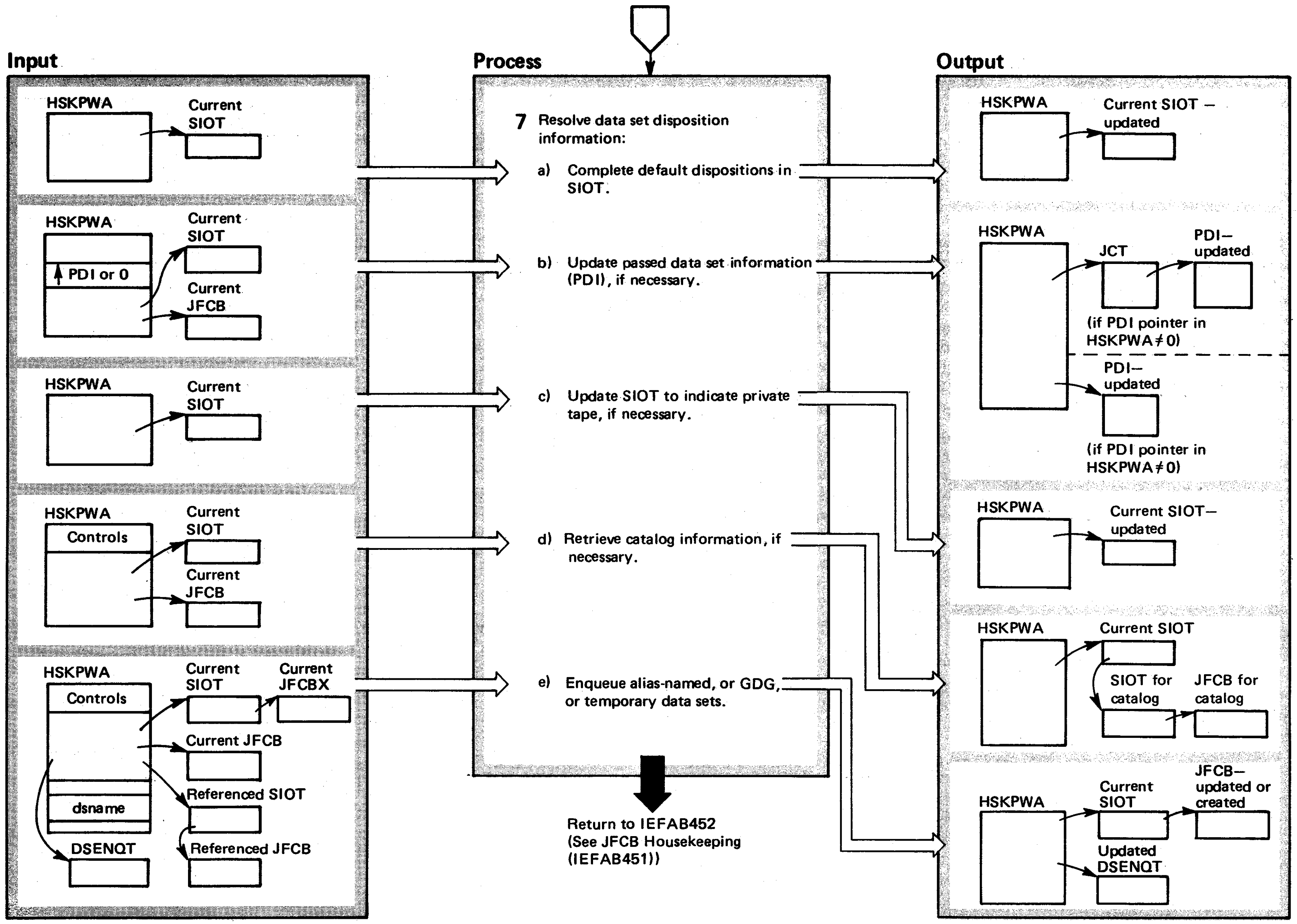
a) when request is for single level of GDG.

b) when DCB=dsname was specified.

Diagram 14-7. IEFAB454 – DD Function Control (Part 10 of 12)

Extended Description	Module	Segment
5 (continued)		
b) Volume/Unit Table Completion marks the current SIOT as a VIO data set (SIOVAMDS=1) if the data set has a system-generated dsname, is not ISAM, is not dummy, and: ● The referenced SIOT (either from the PDI or a volume reference) is VIO; or ● No information was retrieved from a catalog or referenced SIOT, the disposition is NEW, no volumes were specified, and the unit is VIO-eligible.	IEFAB464	VIOCOMP
c) Volume/Unit Table Completion determines if specified unit information overrides retrieved unit information. This step is performed when: ● A unit was coded and is an override candidate (HWOVCAND=1 in the local controls, set by IEFAB470 (Unit Conversion Routine) – see the M.O. diagram JFCB Housekeeping Control (IEFAB451)). ● Unit information was retrieved from the PDI or an inter-step volume reference (HWRESIOT=1 in the local controls and SCTVREF=0 in the current SIOT) or from the catalog (HWDSNCAT=1 in the local controls). Volume/Unit Resolution compares the unit information in the unit conversion list to the unit information in the CRI or in the referenced SIOT. If the unit information in the unit conversion list is the same device type group as the retrieved unit information, the "unit overridden" indicator is set (HWOVRDN=1) in the local controls. This indicator is not set, however, if the referenced SIOT is a dummy, VIO, or subsystem data set. (A specified unit cannot override a retrieved unit in these cases.)	IEFAB464	UNOVERID
d) Volume/Unit Table Completion copies unit information into the SCTUTYPE field of the current SIOT. If the unit was converted from a UCB (that is, a unit address was specified; HWUCB=1 in the local controls), the SIOT is also marked as a demand request (SIOTDMND=1).	IEFAB464	NOREFCMP
e) Volume/Unit Table Completion creates a JFCBX (if none exists) and updates the JFCBX with the device type retrieved from the catalog. This allows the unallocation routine IEFAB4A2 (Disposition Processing) to recatalog the data set using the device type from the catalog.	IEFAB464	CATDEVT
6 DCB Resolution updates the JFCB with data set characteristics from the DSCB (for example, data set organization, record format, logical record length, expiration date). DCB Resolution is performed in two cases: DCB=dsname was specified (SIOTDCBR field in the SIOT); the request is a single level of a GDG (SCTSGDGS=1 in the SIOT). The method of obtaining the DSCB differs in the two cases.	IEFAB458	
a) If DCB=dsname was specified, DCB Resolution obtains the dsname from the data set name table (DSNT).	IEFAB458	
JLOCATE issues a system locate using the dsname to determine the volume serial number of the volume containing the DSCB. For details on JLOCATE, see the M.O. diagram JLOCATE (IEFAB469).	IEFAB469	
b) For a new single-GDG request, DCB Resolution obtains the base level data set name from the JFCB and the volume serial number of the pattern DSCB from the CRI. (The volume serial number of the volume on which the GDG is cataloged was placed in the CRI when the GDG-single request was located – see step 2. The pattern DSCB exists on this volume.)	IEFAB458	
DCB Resolution issues SVC 27 to obtain the Format 1 DSCB and transfers the data set characteristics from it to the JFCB.	IEFAB458 IEFAB458	OBTNDSCB UPDTJFCB

Diagram 14-7. IEFAB454 – DD Function Control (Part 11 of 12)



7 Resolve data set disposition information:

a) Complete default dispositions in SIOT.

b) Update passed data set information (PDI), if necessary.

c) Update SIOT to indicate private tape, if necessary.

d) Retrieve catalog information, if necessary.

e) Enqueue alias-named, or GDG, or temporary data sets.

Return to IEFAB452 (See JFCB Housekeeping (IEFAB451))

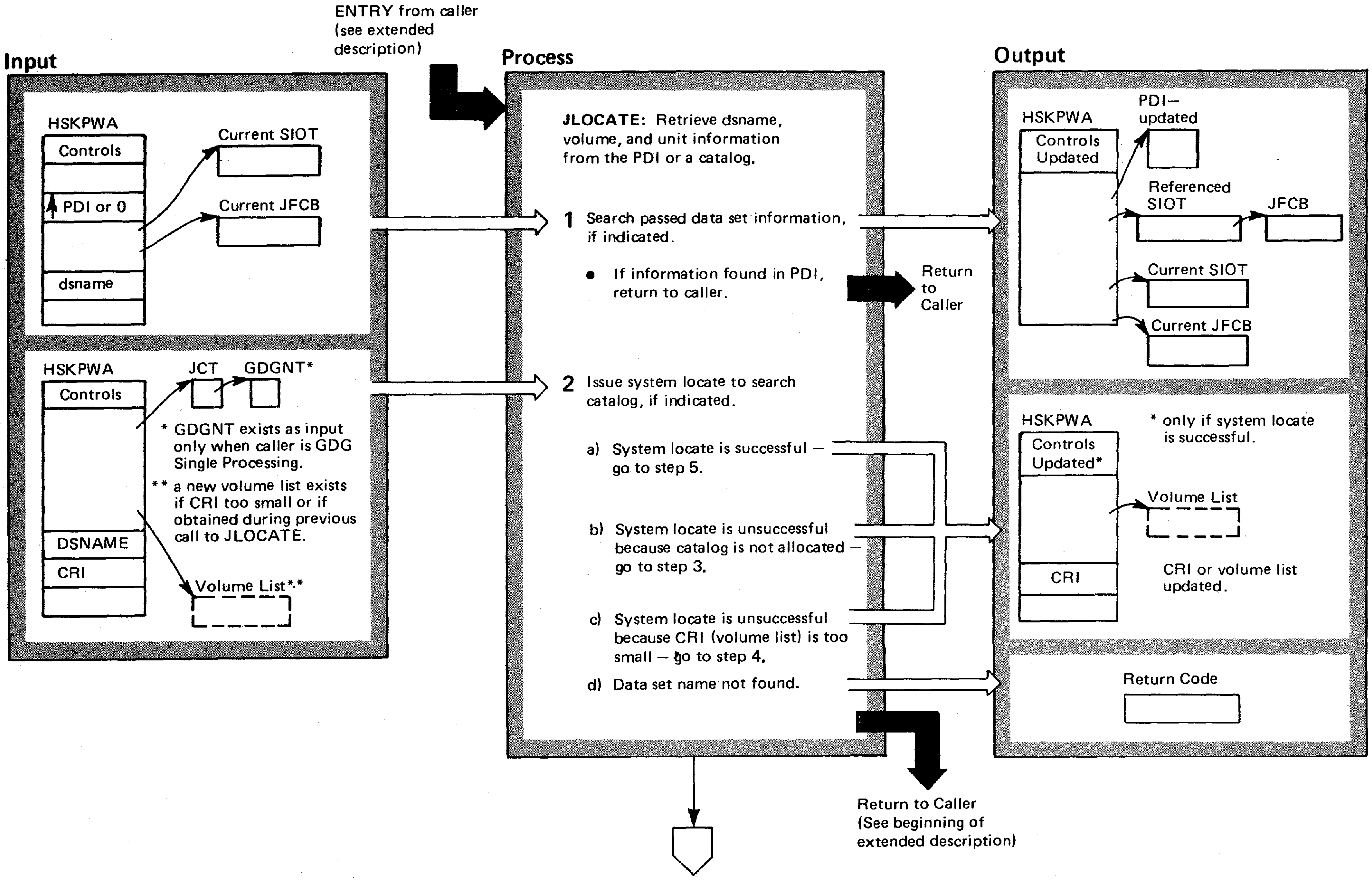
Diagram 14-7. IEFAB454 – DD Function Control (Part 12 of 12)

Extended Description	Module	Segment
7 The purpose of this step is to resolve data set disposition information.		
a) DISP Resolution completes the data set disposition information in the SIOT: ● An existing JOBLIB request is marked PASS (SIOTPASS=1). ● If no disposition is specified, data sets that existed at the beginning of the job are marked KEEP (SIOTKEEP=1); and data sets that specified MOD but for which no unit or volume information could be located are marked DELETE (SIOTDLET=1). (The Interpreter marked as DELETE data sets that specified NEW but that did not specify a disposition.)	IEFAB459	
b) If the request specified PASS (SIOTPASS=1) and PDI processing is allowed (HWDOPDI=1 in the global controls), DISP Resolution updates the PDI with an entry for this data set. If the data set was received by this step (SCTRECVD=1), the original PDI entry (PDIE) is updated; otherwise, a PDIE is created.	IEFAB459	PDIBUILD
c) This step marks a tape data set as private (SIOTPRIV=1), so that it will not be deleted, if attributes of the data set indicate it should be kept.	IEFAB459	
d) DISP Resolution calls JLOCATE to retrieve catalog information for a data set which has a qualified name (JFCBDSNM field in the JOCB), if the following conditions are true: ● A disposition of CATLG was specified (SIOTCTLG=1 in the SIOT), and no STEPCATs exist for this step (SCTPCAT=0 in the SCT). ● A disposition of UNCATLG (SIOTUNCT=1 in the SIOT) was specified, and no STEPCATs exist. ● A disposition of DELETE (SIOTDLET=1 in the SIOT) was specified, and volume information was retrieved from the catalog. For details on JLOCATE, see the M.O. diagram JLOCATE (IEFAB469).	IEFAB459 IEFAB469	
e) If an alias name or GDG was specified for a passed or cataloged data set, DISP Resolution creates a JFCBX (if none exists), updates the JFCBX with the alias data set name, and enqueues the major data set name.	IEFAB459 IEFAB469	REALDSN
Non-VIO temporary data set names are enqueued.	IEFAB459	ENQDSN

Error Processing

An error in any routine causes control to be returned to the calling routine. In the event of an abnormal termination, the ESTAE exit routine established by IEFAB451 performs any necessary cleanup.

Diagram 14-8. IEFAB469 – JLOCATE (Part 1 of 4)



* GDGNT exists as input only when caller is GDG Single Processing.

** a new volume list exists if CRI too small or if obtained during previous call to JLOCATE.

JLOCATE: Retrieve dsname, volume, and unit information from the PDI or a catalog.

1 Search passed data set information, if indicated.

- If information found in PDI, return to caller.

2 Issue system locate to search catalog, if indicated.

a) System locate is successful – go to step 5.

b) System locate is unsuccessful because catalog is not allocated – go to step 3.

c) System locate is unsuccessful because CRI (volume list) is too small – go to step 4.

d) Data set name not found.

* only if system locate is successful.

CRI or volume list updated.

Diagram 14-8. IEFAB469 – JLOCATE (Part 2 of 4)

Extended Description	Module	Segment
ENTRY JLOCATE is a service routine used by several housekeeping routines to retrieve dsname, volume, and unit information from the passed data set information (PDI) or from a catalog. It is called by the following modules (all these modules are described in the M.O. diagram DD Function Control (IEFAB454)):		
• DD Function Control (IEFAB454) to determine if a request is GDGALL (all levels of a GDG are requested). If the request is not GDGALL, retrieves dsname, volume, and unit information for the data set, or updates PDI for restarting steps, or allocates private catalogs for restarting steps.		
• GDG Single Processing (IEFAB461) to obtain the base level of a GDG and to obtain the fully-qualified dsname and volume and unit information for the data set.		
• DSN Resolution (IEFAB456) to obtain volume and unit information for a specific level of a GDGALL request.		
• Volume/Unit Resolution (IEFAB457) to locate volume and unit information when VOL=REF=dsname was coded.		
• DCB Resolution (IEFAB458) to obtain the volume serial number of the volume containing the DSCB for a data set, when DCB=dsname was coded.		
• DISP Resolution (IEFAB459) to make a private catalog available for unallocation.		
Most of these routines use JLOCATE only to search catalogs; the PDI is searched only when:		
• JLOCATE is called by DD Function Control, GDG Single Processing, or Volume/Unit Resolution, and,		
• the SIOT does not represent a STEPCAT request, and,		
• JFCB Housekeeping Control was called by Step Allocation Control (that is, this is a batch request).		

Extended Description	Module	Segment
1 PDI Scan searches the PDI if allowed, as indicated in the local and global controls (HWPDISCN=1; HWDOPDI=1).	IEFAB455	
If the PDI pointer in the housekeeping workarea is nonzero, PDI Read and Chain updates the HSKPWA with pointers to the first and last PDI entries. If the dsname is found in the PDI, PDI Scan:	IEFAB4EB	
• Reads in the SIOT and JFCB of the request that passed the data set.	IEFAB455	PDIDDRD
• Marks the PDI entry as received.	IEFAB455	
• Sets the local controls to indicate that the referenced SIOT is present (HWRESIOT=1).	IEFAB455	
2 This step is performed only if both of the following conditions are true:		
• The dsname was not found in the PDI or the PDI was not searched.		
• Local controls indicate a system locate should be issued (HWSYSLOC=1).		
Input to this step is the data set name or, if JLOCATE was invoked by GDG Single Processing, the data set name and the GDG base level.		
JLOCATE issues a system locate (SVC 26) to search: 1) private catalogs defined for this step by means of JOBCAT or STEPCAT DD statements; 2) the master catalog; 3) catalogs implied by the data set name. The system locate results in one of the following situations:	IEFAB469	PARMINIT
a) The system locate is successful; the dsname is found. Catalog management places unit and volume information in the CRI (volume list), if the request is not GDGALL. If the request is GDGALL, catalog management places the number of levels of the GDG in the CRI (volume list).	IEFAB469	LOCATECT
b) The system locate is unsuccessful because the catalog to be searched is unallocated. Catalog management returns the name of the catalog to be allocated and the catalog connector (alias), if any, in the CRI (volume list). See step 3.	IEFAB469	LOCATECT
Note: A catalog will already be allocated only if it was previously allocated during JLOCATE processing and if it was not subsequently unallocated to release		

Step 2 continued on Part 4

Diagram 14-8. IEFAB469 – JLOCATE (Part 3 of 4)

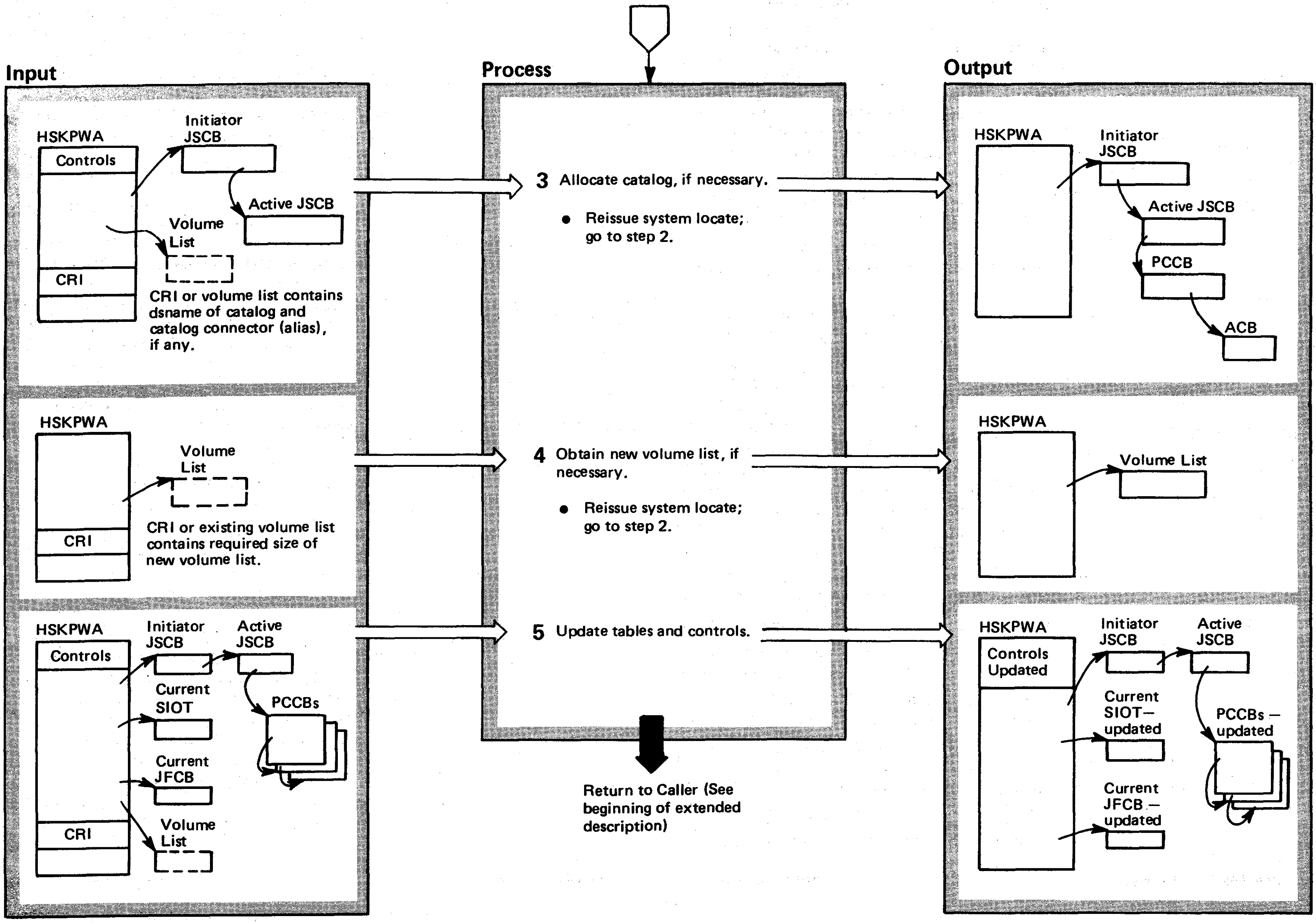
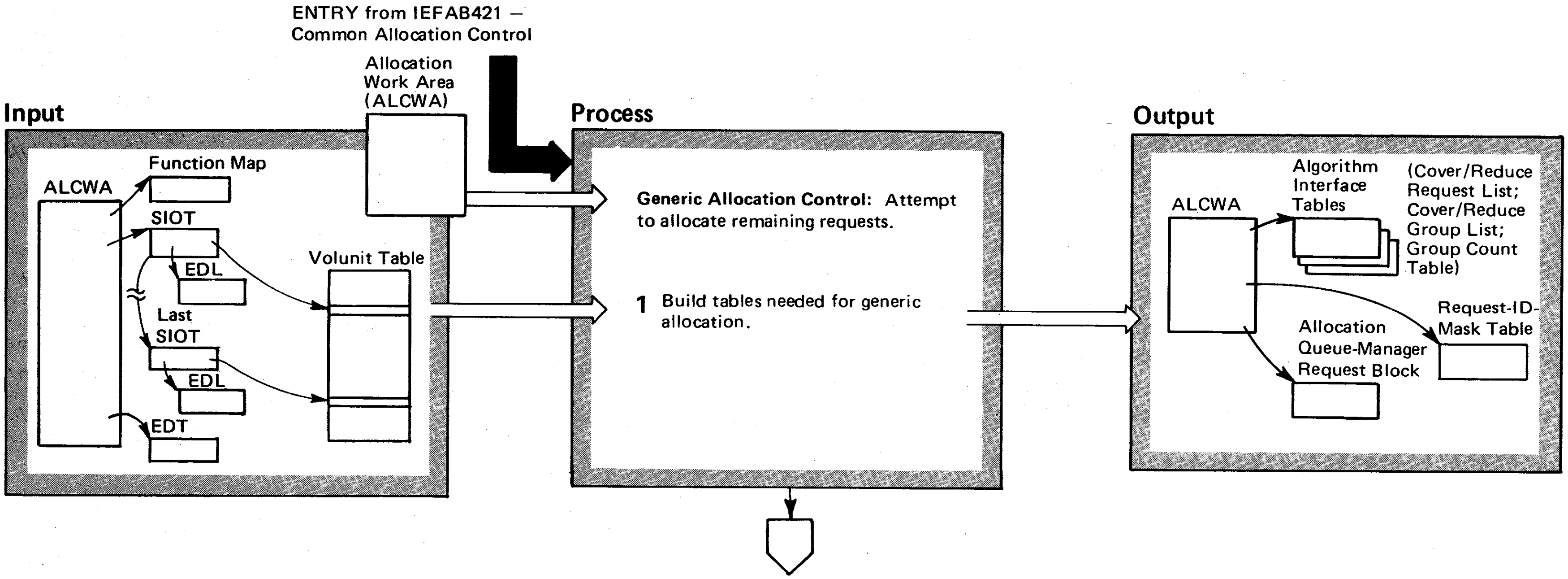


Diagram 14-8. IEFAB469 – JLOCATE (Part 4 of 4)

Extended Description	Module	Segment
2 (Continued) needed resources. Catalogs are allocated during JLOCATE to retrieve information needed for allocation, if JFCB Housekeeping has not been called by Dynamic Allocation. All catalogs will be unallocated during housekeeping clean-up processing, before JFCB Housekeeping Control returns to its caller, if the request is batch (that is, Step Allocation Control called JFCB Housekeeping Control) (see the M.O. diagram JFCB Housekeeping Control (IEFAB451)).		
c) The system locate is unsuccessful because the CRI (or volume list, if one was obtained during a previous call to JLOCATE) is too small. Catalog management returns the required size of the volume list in the CRI or existing volume list. See step 4.	IEFAB469	REDOPREP
d) The system locate is unsuccessful because the dsname could not be found. If this occurs, control is returned to the caller and processing terminates.		
3 If the catalog to be searched is not allocated, the system locate is unsuccessful. JLOCATE will attempt to have the catalog allocated. Allocate Catalog Control issues SVC 99 to have the catalog dynamically allocated; Open Catalog Routine (a data management routine) opens a private catalog and catalog management opens a CVOL (control volume); the PCCB Routine builds or updates a private catalog control block (PCCB) for the catalog. If Housekeeping was called by Step Allocation Control, Table Creation also creates a SIOT and JFCB to represent the catalog (HWMAKTAB=1 and HWDNCCDD=0 in the controls).	IEFAB469 IEFAB4F5 IDACAT11 IEFAB4EF IEFAB466	REDOPREP ALCATLG FINDPCCB
If allocation of the catalog is unsuccessful, processing is terminated unless the failure is due to insufficient resources and Step Allocation Control invoked JFCB Housekeeping. In this case, Unallocate Private Catalog issues SVC 99 to dynamically unallocate all private catalogs previously allocated during housekeeping processing and marks the PCCBs of the unallocated catalogs as inactive (PCCACTIV=0) (so that they will not be searched when the system locate is re-issued). Allocate Private Catalog then reattempts the allocation; if the catalog still cannot be allocated, processing is terminated.	IEFAB4F5 IEFAB4F4 IEFAB4F5	RECOVERY ALCATLG
JLOCATE reissues the system locate if the required catalog is successfully allocated.		
4 The system locate is unsuccessful if the CRI (or volume list, if one was obtained during a previous call to JLOCATE) is too small. JLOCATE: • Issues a FREEMAIN macro instruction to release a previous volume list if one existed (HWNEWVL=1). • Issues a GETMAIN macro instruction for the required amount of storage. • Places a pointer to the volume list in the HSKPWA. • Sets local controls to indicate an additional volume list is in use (HWNEWVL=1).	IEFAB469	REDOPREP
This volume list is used for all subsequent JLOCATE processing unless it is too small; in this case, it is released and JLOCATE obtains a new volume list. If the required volume list is successfully obtained, JLOCATE reissues the system locate.		
5 If information is successfully retrieved from a catalog, JLOCATE: • Copies DSORG information from the CRI (or volume list) into the current JFCB. • Ensures that the disposition is KEEP (SIOTKEEP=1) if the data set is a VSAM data set. • Sets the GDGALL indicator in the local controls (HWGDGALL=1) if the system locate determined this request was GDGALL. • Sets the local controls to indicate the data set was found in the catalog (HWDSNCAT=1). • Updates the PCCBs with the catalog connector (alias), if any, and ensures that all the PCCBs are marked active (PCCACTIV=1) so that the associated catalogs can be searched on subsequent calls to JLOCATE.	IEFAB469	CLEANUP
Error Processing An error in any routine causes control to be returned to the calling routine. In the event of an abnormal termination, the ESTAE exit routine established by IEFAB451 performs any necessary cleanup.		

Diagram 14-9. IEFAB471 – Generic Allocation Control (Part 1 of 10)



Extended Description	Module	Segment
ENTRY Common Allocation Control calls Generic Allocation Control (IEFAB471) to attempt to allocate all remaining requests. The processing of IEFAB471 must be serialized with all other allocations. (For a description of serialization and when it is required see "Common Allocation Control" in the "Introduction to Allocation/Unallocation.") To minimize serialization, IEFAB471 processes one generic device type at a time; within a generic, it serializes only those device groups needed by unallocated requests. (Device groups and their representation in group masks are described under "Generic Allocation Control" in the "Introduction to Allocation/Unallocation.")		
To avoid deadlock situations, all allocations must choose generics in the same order. The installation device precedence list (defined during system generation) dictates the order in which generics are chosen. The generic to be processed is selected in step 2 of this diagram; steps 3-12 are a loop performed for every generic selected. Within this loop, there are four basic allocation processes: • Demand (specific unit) allocation (step 6). • Specific volume allocation (step 8). • Allocation via algorithm (step 9). • Nonspecific volume allocation (step 10).		
Not all of these processes are necessarily performed for each generic group selected in step 2 – the processes performed depend on the types of unallocated requests that are eligible to the generic. See the individual steps for details.		
1 Generic Table Build issues a GETMAIN macro instruction to obtain storage for the following tables required by generic allocation: a) algorithm tables; b) a request-id-mask table; c) an allocation queue manager request block; d) three work masks.	IEFAB472	
a) The purpose of the *algorithm tables* is to summarize the unit requirements of each request, the device groups to which a request is eligible, and information about the units in each device group. Allocation via		

Diagram 14-9. IEFAB471 – Generic Allocation Control (Part 2 of 10)

Extended Description	Module	Segment

1 a) (Continued) Algorithm (IEFAB476 – step 9) selects device groups from which requests should be allocated when a choice of units exists; it uses the algorithm tables to determine from which device groups units should be allocated so that all requests can be satisfied. There are three main sections in the algorithm tables:

- The cover/reduce request list (CVRRQLST) contains an entry for every unallocated volunit entry – that is, for every unallocated possible request for a unit. The entries are updated as they are allocated; each entry points to a corresponding entry in the cover/reduce group list.
- The cover/reduce group list (CVRGPLST) contains an entry for each request in the cover/reduce request list; the entry lists the device groups to which the request is eligible. Each device group listed points to a corresponding entry in the group count table.
- The group count table (CRPCOUNT) includes an entry for every device group. Each entry summarizes the number of units available, the number allocated, the number offline, the total number of units, and the number of units not used by the allocation.

For details on these tables, see Section 5, Data Areas.

Extended Description	Module	Segment
To initialize the tables, Generic Table Build (IEFAB472) scans the SIOT chain for SIOTs that have not been allocated; for every unallocated volunit entry of each unallocated SIOT, it creates an entry in the cover/reduce request list. The EDL for each SIOT contains the generic groups eligible to this request and, within each generic group, the eligible device groups. This information is used to initialize the cover/reduce group list. The group count table is initialized by means of the EDT, which contains all the device groups; an entry is created for each device group but is not further initialized until step 3.	IEFAB472	DOALGTAB
b) The request-id-mask table is used to determine when device groups must be serialized and when serialized device groups can be released. Each entry in the request-id-mask table contains a request identifier (id), the number of unallocated SIOTs associated with the request id, and a group mask indicating the device groups required by the associated SIOTs. (For a description of group masks, see "Group Masks" in the "Introduction to Allocation/Unallocation.")	IEFAB472	DORIMTAB

To build the request-id-mask table, IEFAB472 does multiple scans of the SIOT chain to find unallocated SIOTs whose group masks intersect. (The EDT contains group masks of the unit groups to which each SIOT is eligible; the EDL for each SIOT points to the group masks in the EDT.) All SIOTs whose group masks intersect are assigned the same request id (SIOTGIID in the SIOT) and are represented by the same entry in the request-id-mask table. The mask placed in the entry is a composite mask of the individual masks, showing all the device groups required by the associated SIOTs.

For example, there are four unallocated SIOTs, each with the following group mask:

	device groups				
SIOT	1	2	3	4	5
SIOT1	0	0	0	0	1
SIOT2	0	0	0	1	0
SIOT3	0	0	1	1	0
SIOT4	1	0	1	0	0

The mask of SIOT1 does not intersect with any of the other masks. It is assigned a distinct request id and has a single entry in the request id mask table; the mask in the entry is its group mask, 0 0 0 0 1. The masks of SIOT2, SIOT3, and SIOT4 intersect.

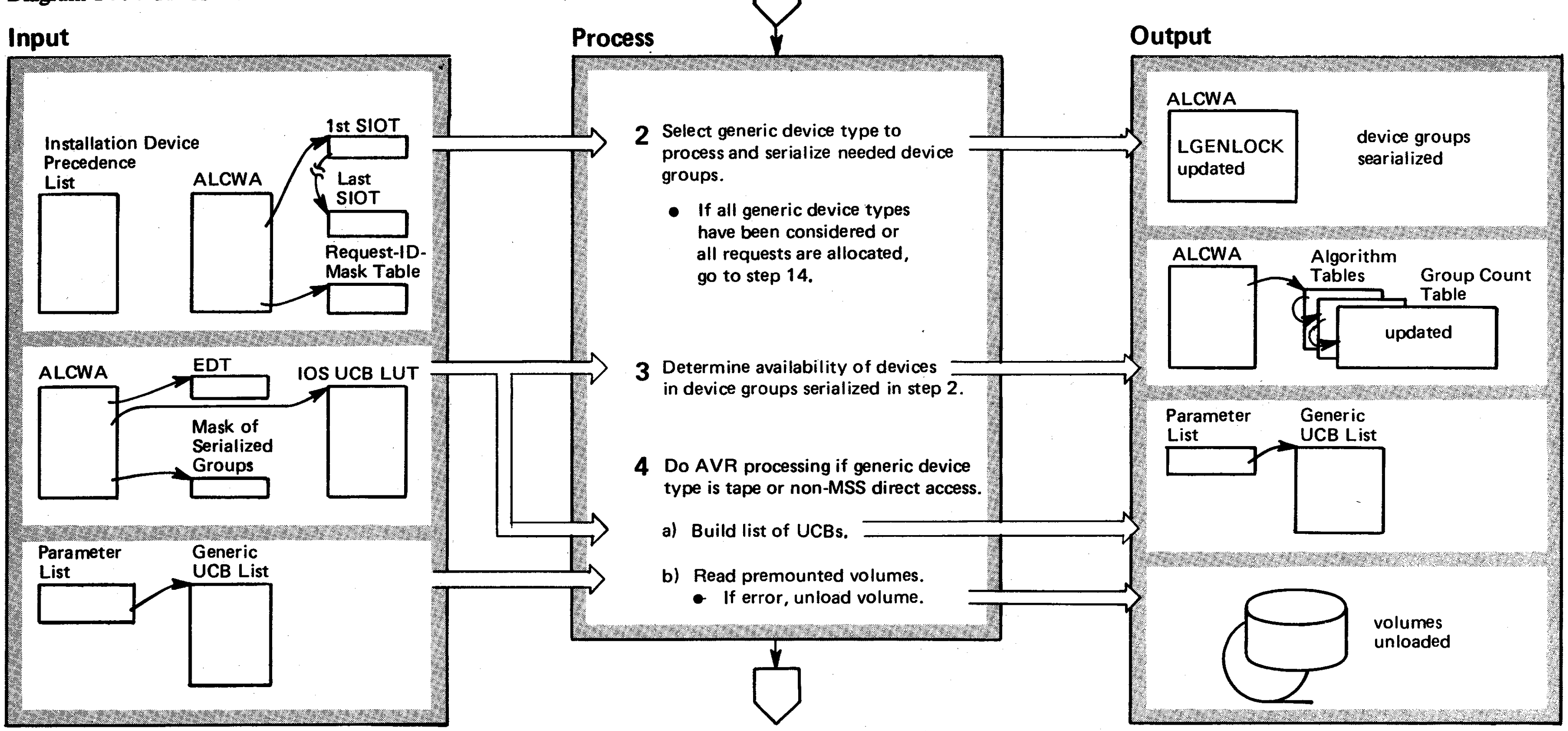
Note: Although the mask of SIOT2 does not intersect with the mask of SIOT4, both intersect with the mask of SIOT3.

These three SIOTs are assigned the same request id and are represented by the same entry in the request-id-mask table; the mask in the entry is the combination of the three masks, 1 0 1 1 0.

The purpose of associating SIOTs is to allow allocations to be rearranged; none of the device groups for the associated SIOTs are released until all the SIOTs are allocated. (For more information on rearranging allocations, see the M.O. diagram Allocation via Algorithm (IEFAB476)).

Step 1 continued on Part 3

Diagram 14-9. IEFAB471 – Generic Allocation Control (Part 3 of 10)



2 Select generic device type to process and serialize needed device groups.

- If all generic device types have been considered or all requests are allocated, go to step 14.

3 Determine availability of devices in device groups serialized in step 2.

4 Do AVR processing if generic device type is tape or non-MSS direct access.

a) Build list of UCBs.

b) Read premounted volumes.

- If error, unload volume.

Extended Description	Module	Segment
1 (Continued)		
c) The allocation queue manager request block (AQMRB) serves as the interface to the allocation queue manager, which actually serializes and releases device groups. In the AQMRB, Allocation Within Generic indicates if this allocation can wait for units; the Allocation Queue Manager then builds the necessary data structures for handling requests from this allocation.	IEFAB472	
d) The work masks are workareas, pointed to by ALCWA, that are used in determining the group mask of device groups that must be serialized and the group mask of device groups that can be released.	IEFAB472	
2 The purpose of this step is to:		
• Select the first generic group from the installation device precedence list that has not yet been selected and that contains devices required by one or more unallocated requests.	IEFAB471	
• Serialize the needed device groups within the selected generic.	IEFAB4FA	
To determine what generic group and which device groups are needed, the following processing is performed:		
a) To obtain a mask of all needed device groups, Generic Allocation Control combines (by means of an "or" function) all the group masks in the request-id-mask table associated with entries that include unallocated SIOTs.	IEFAB471	

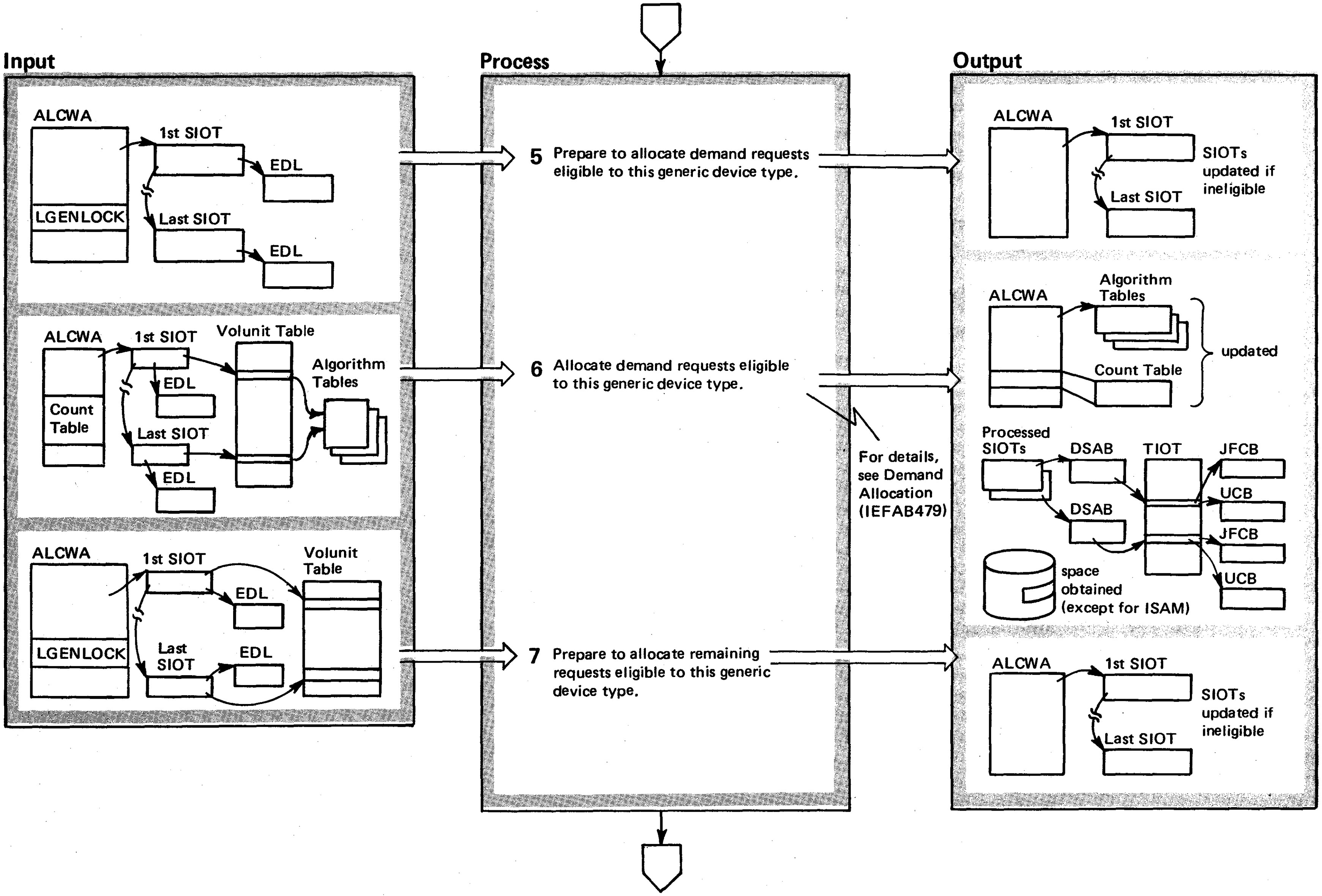
Step 2 continued on Part 4

Diagram 14-9. IEFAB471 – Generic Allocation Control (Part 4 of 10)

Extended Description	Module	Label
2 (Continued)		
b) Generic Allocation Control selects the first generic listed in the installation device precedence list that has not yet been selected.		
c) Generic Allocation Control combines (by means of an "and" function) the mask of the selected generic group (from the EDT) with the mask obtained in step 2a to determine if the generic group contains needed device groups. If no common device groups are found (the resulting mask contains only zeroes), step 2b is repeated to select the next generic. If common device groups are found, the LGENLOCK field in ALCWA is updated with the id of this generic. (The generic id is included in the EDT.)		
d) If a retry is being performed (GENLOKSW=1 in the function map), Generic Allocation Control determines if the entire generic device type must be serialized for direct access, or for additional compatible generic device types, or for tape. (Retry is described under "The Retry Situation" in the "Introduction to Allocation/Unallocation.") Generic Allocation Control searches the SIOT chain for a SIOT marked for retry (SIOTRTRY=1). If the SIOT is eligible to the selected generic, the mask of device groups to be set to the mask of 1) the generic device type for direct access, or 2) all compatible device types for tape. The entire generic must also be serialized if a specific unit was requested. Otherwise, the mask of device groups to be serialized is the mask obtained in step 2c.	IEFAB471	DETLOCK
e) The Allocation Queue Manager serializes the device groups indicated in the mask.	IEFAB4FA	
3 Generic Allocation Control determines the status of the UCBs in the serialized device groups and updates the unit information in the group count table. The EDT contains the device groups and indexes into the IOS UCB LUT for the units in each device group. The group count table is updated to indicate the number of units that are: • Offline (UCBONLI=0). • Allocated (UCBALOC=1). • Available.	IEFAB471	DESTATUS

Extended Description	Module	Segment
4 Automatic volume recognition (AVR) allows the operator to premount tape and direct access volumes prior to the initiation of the job step that requires the volumes. The purpose of this step is to recognize that these volumes have been mounted.		
a) If the generic group is tape or direct access, Generic Allocation Control builds a list of the UCBs in the serialized device groups (generic UCB list). The EDT contains the device groups and indexes into the IOS UCB LUT for the units in each device group.	IEFAB471	CALLAVR
b) AVR Control checks the generic UCB list to locate UCBs that are unallocated, online, ready, and do not contain a volume serial number. For each such unit, Direct Access Label Read (if the device is direct access) or Tape Label Read (if the device is tape) reads the label of the volume. If no error is encountered in reading the label, AVR Control places the volume serial number in the UCB and, for tape, sets the label-type indicator in the UCB. If an error is encountered, AVR Control issues an appropriate error message to the operator and the Unload Interface has the volume unloaded. Errors are encountered in the following situations:	IEFAB473 IEFAB4F8 IEFAB4F9 IEFAB49C	
• A tape volume does not have labels. (Unlabeled volumes cannot be premounted.)		
• A tape volume has non-standard labels and a user routine to read non-standard labels was not included in the system or the user routine did not return a volume serial number.		
• A tape volume has ANSI labels and the ANSI converter routine was not included in the system at system generation.		
• Duplicate volume serial numbers were found. (AVR Control uses the IOS UCB LUT to check if the volume serial number it has read is a duplicate.)	IEFAB473	B473DPCK
After AVR Control completes processing, Allocation Within Generic issues a FREEMAIN macro instruction to release the generic UCB list.	IEFAB471	CALLAVR

Diagram 14-9. IEFAB471 – Generic Allocation Control (Part 5 of 10)



5 Prepare to allocate demand requests eligible to this generic device type.

6 Allocate demand requests eligible to this generic device type.

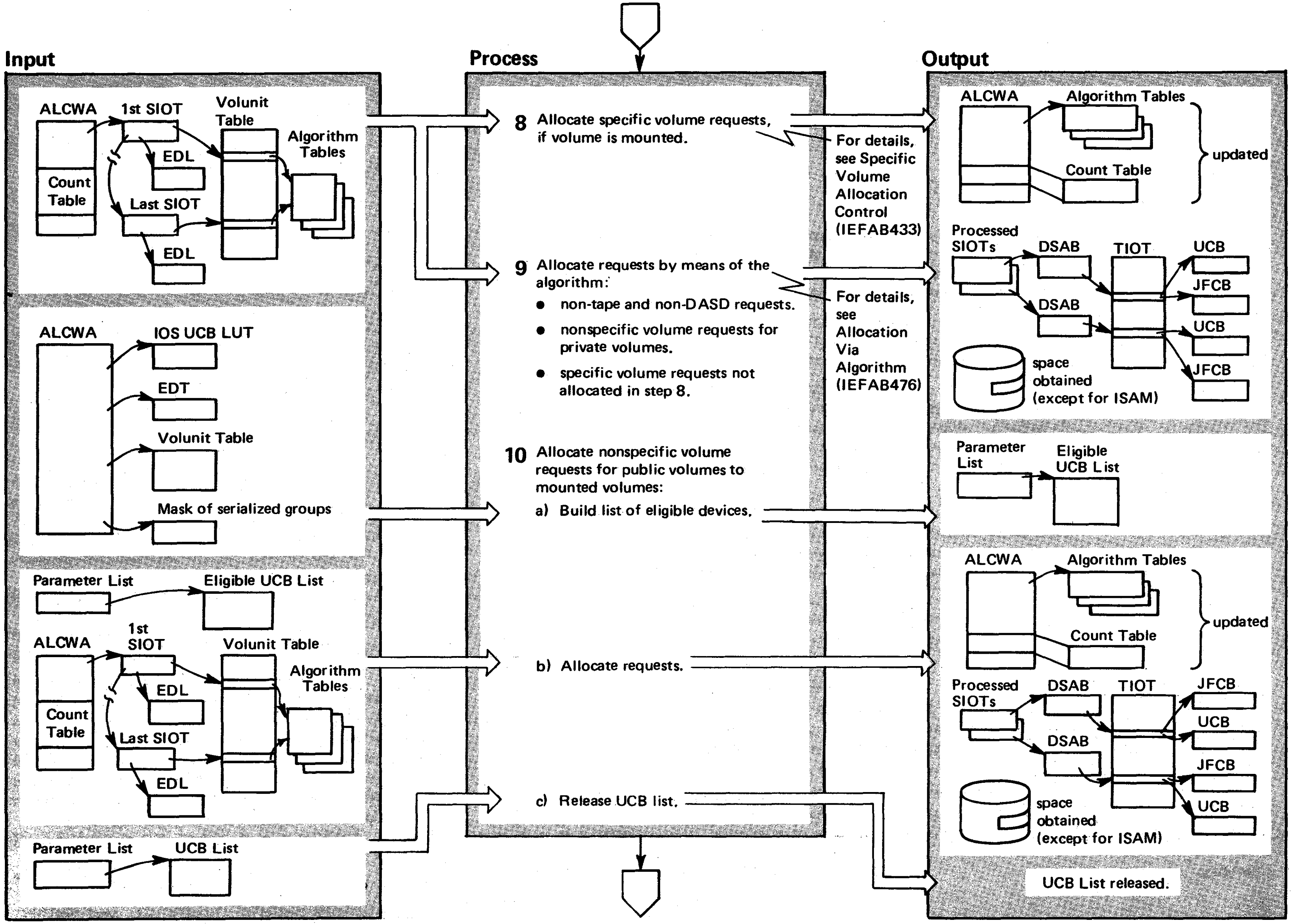
For details, see Demand Allocation (IEFAB479)

7 Prepare to allocate remaining requests eligible to this generic device type.

Diagram 14-9. IEFAB471 – Generic Allocation Control (Part 6 of 10)

Extended Description	Module	Segment
5 Allocation Within Generic marks as ineligible (SIOTGIGN=1) all unallocated SIOTs except those representing demand requests (a unit address was specified; for example, UNIT=190) that are eligible to the generic device type selected in step 2. The LGENLOCK field in ALCWA contains the id of the selected generic; the EDL for each SIOT contains the ids of the generic device types to which the SIOT is eligible; the SIOT itself indicates if it represents a demand request (SIOTDMND=1). If no demand requests eligible to this generic are found, processing continues with step 7.	IEFAB475	FLAGDMAN
6 Demand Allocation processes those SIOTs that were not marked ineligible in step 5 – that is, all SIOTs representing demand requests that are eligible to the generic device type being processed. Demand Allocation is not called if step 5 determined that there are no demand requests eligible to this generic. For details, see the M.O. diagram Demand Allocation (IEFAB479).	IEFAB479	
7 This step has two functions: a) to determine which SIOTs are to be processed; b) to determine what processing is required for those SIOTs. a) Allocation Within Generic marks as ineligible (SIOTGIGN=1) all unallocated SIOTs except those that are eligible to this generic and that do not represent demand requests. (Demand requests eligible to this generic were processed in step 6.) b) To determine what processing is required for the unallocated SIOTs eligible to this generic, Allocation Within Generic categorizes the requests by examining the volunit entries: ● Specific volume requests. These will be processed in step 8 by Specific Volume Allocation Control if the volume is mounted. (If the volume is not mounted, step 8 will indicate that Allocation via Algorithm must process the request.) ● Non-tape and non-DASD requests; nonspecific volume requests for private volumes. These will be processed in step 9 by Allocation via Algorithm. (Allocation via Algorithm will also process specific volume requests if indicated by step 8.) ● Nonspecific requests for public volumes. These requests will be processed in step 10 by Nonspecific Volume Allocation Control. If one of the preceding types of requests is not found, the corresponding allocation processing is not performed; for example, if there are no specific volume requests, Specific Volume Allocation Control (step 8) is not called.	IEFAB475	FLAGIGEN

Diagram 14-9. IEFAB471 – Generic Allocation Control (Part 7 of 10)



8 Allocate specific volume requests, if volume is mounted.

For details, see Specific Volume Allocation Control (IEFAB433)

9 Allocate requests by means of the algorithm:

- non-tape and non-DASD requests.
- nonspecific volume requests for private volumes.
- specific volume requests not allocated in step 8.

For details, see Allocation Via Algorithm (IEFAB476)

10 Allocate nonspecific volume requests for public volumes to mounted volumes:

a) Build list of eligible devices.

b) Allocate requests.

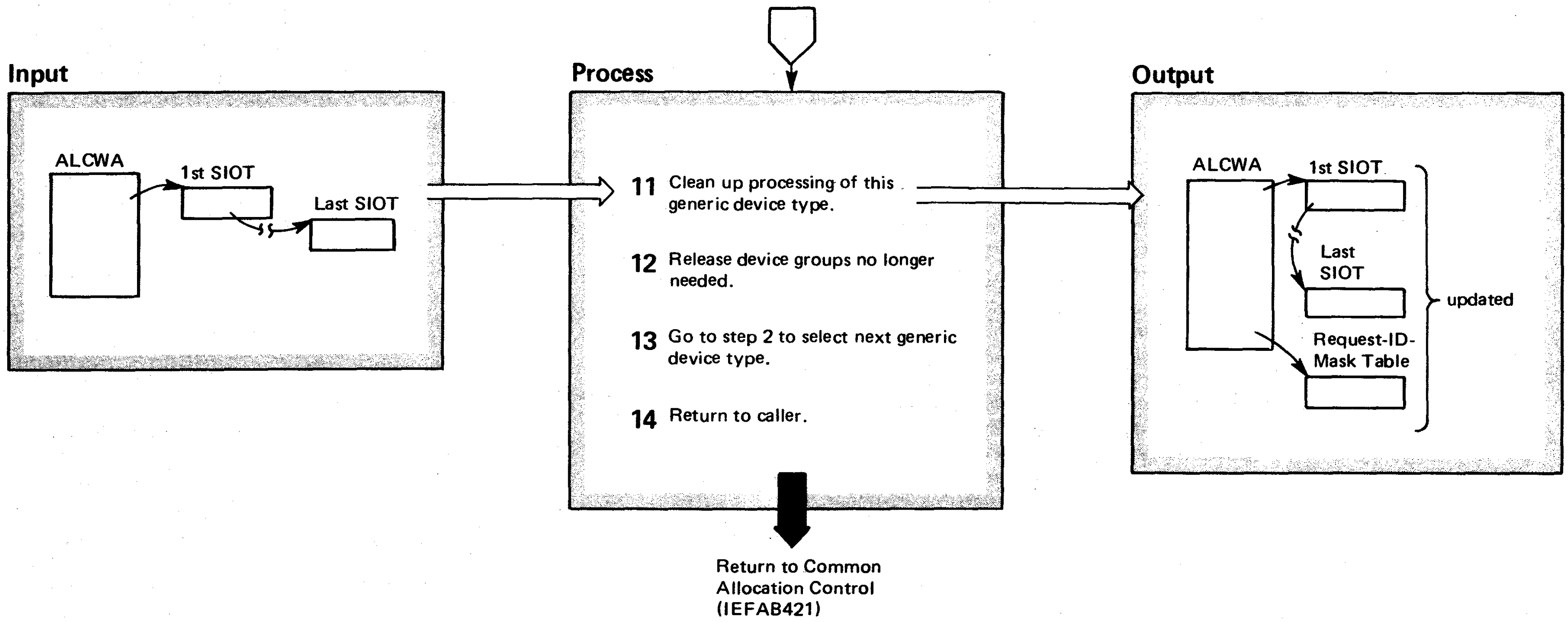
c) Release UCB list.

Diagram 14-9. IEFAB471 – Generic Allocation Control (Part 8 of 10)

Extended Description	Module	Segment
8 This step is performed only if step 7 determined that unallocated specific volume requests are eligible to this generic device type. Specific Volume Allocation Control allocates specific volume requests if the volumes are mounted. If a volume is not mounted, the request will be processed by Allocation via Algorithm (step 9). For details on Specific Volume Allocation Control, see the M.O. diagram Specific Volume Allocation Control (IEFAB433).	IEFAB433	
9 This step is performed if any of the following conditions are true: • Step 7 determined that unallocated requests for non-DASD or non-tape devices are eligible to this generic device type. • Step 7 determined that unallocated nonspecific volume requests for private volumes are eligible to this generic device type *and* volume mounting is allowed (as indicated in the common allocation function map – see figure 17). • Specific volume requests were not allocated in step 8 because the volumes were not mounted *and* volume mounting is allowed (as indicated in the common allocation function map – see figure 2-27).	IEFAB476	
Allocation via Algorithm processes the requests; for details, see the M.O. diagram Allocation via Algorithm (IEFAB476).	IEFAB476	

Extended Description	Module	Segment
10 This step is performed only if step 7 determined that unallocated nonspecific volume requests for public volumes are eligible to this generic device type.		
a) Allocation Within Generic (IEFAB475) builds a list of all the UCBs in the device groups serialized for this generic that meet the following conditions: • The unit is online, ready, and is not pending offline. • The unit is not pending an unload or a mount. • The unit is not in use by a system task (UCBNALOC=0). • The unit is not allocated as nonshareable. • The unit contains a non-private volume. • Another request within this allocation is not waiting for this unit. The EDT contains indexes into the IOS UCB LUT for all the units in each device group. The mask of the serialized groups indicates which device groups should be searched.	IEFAB475	ASCRMONT
b) Nonspecific Volume Allocation Control (IEFAB436) allocates nonspecific volume requests for public volumes to mounted volumes. If a request cannot be allocated to a mounted volume, it will be processed during the processing of another generic device type (if the request is eligible to more than one generic) or by Recovery Allocation (see the M.O. diagram Recovery Allocation (IEFAB485)). For details on Nonspecific Volume Allocation Control, see the M.O. diagram Nonspecific Volume Allocation Control (IEFAB436).	IEFAB436	
c) Allocation within Generic (IEFAB475) issues a FREEMAIN macro instruction to release the UCB list.	IEFAB475	ASCRMONT

Diagram 14-9. IEFAB471 – Generic Allocation Control (Part 9 of 10)



11 Clean up processing of this generic device type.

12 Release device groups no longer needed.

13 Go to step 2 to select next generic device type.

14 Return to caller.

Diagram 14-9. IEFAB471 – Generic Allocation Control (Part 10 of 10)

Extended Description **Module** **Segment**

11 Allocation Within Generic (IEFAB475) scans all the SIOTs and does the following:

IEFAB475 UNFLGGEN

a) If a SIOT was eligible to this generic device type (SIOTGIGN=0) and it is fully allocated (SIOTALCD=1), Allocation Within Generic decreases the number of unallocated SIOTs in the request-id-mask table entry associated with this SIOT.

b) If a SIOT was not eligible to this generic device type (SIOTGIGN=1), Allocation Within Generic turns off the ineligible indicator.

12 In determining what device groups can be released, Generic Allocation Control considers *all* device groups, not merely the device groups within the generic being processed. This is done because device groups serialized for a previous generic might still be serialized but no longer be needed. For example, SIOTA is eligible to two generic device types, 3330 and 2314. SIOTA was not allocated when generic 3330 was processed, so the device groups in 3330 were not released. When generic 2314 is processed, the request is satisfied; it is no longer necessary to serialize the device groups in either generic 3330 or generic 2314 (provided that SIOTA was the only unallocated SIOT eligible to the generics).

IEFAB471 UNLKDEV

To determine what device groups can be released, the following processing is performed:

a) Generic Allocation Control initializes a mask to represent all the device groups still needed (RIMNEEDM). This mask, pointed to by WMASK2P in ALCWA, is originally set to zeroes.

b) If an entry in the request-id-mask table indicates that any of the associated SIOTs are not yet allocated, the device groups required by those SIOTs cannot be released. Generic Allocation Control combines (by means of an "or" function) the group mask for every entry that includes unallocated SIOTs with RIMNEEDM.

c) The group masks in the request-id-mask table do not indicate if all devices in a generic device type were serialized because a retry allocation is being processed (for a description of retry, see "The Retry Situation" in the "Introduction to Allocation/Unallocation"). If this allocation is a retry (GENLOKSW=1 in the function map of the Common Allocation Parameter List), Generic Allocation Control searches the SIOT chain to locate SIOTs marked for retry (SIOTRTRY=1) that are not allocated (SIOTALCD=0). For such SIOTs, all device groups within the eligible generics must remain serialized. The EDL for the SIOT contains the eligible generic device types; the EDT contains a group mask for each generic. The group mask(s) for the eligible generic(s) are combined (by means of an "or" function) with RIMNEEDM.

d) Generic Allocation Control inverts RIMNEEDM to obtain the mask of device groups that can be released (UNLKMASK). WMASKPTR in ALCWA points to the UNLKMASK.

e) If all requests eligible to the generic being processed are satisfied, all device groups within the generic can be released. Generic Allocation Control combines (by means of an "or" function) the group mask of the generic with the mask of device groups that can be released (UNLKMASK), thereby indicating that all device groups in the generic can be released.

f) Volumes are not mounted on allocated units until all requests have been satisfied; therefore, a device group containing a unit on which a nonspecific or private direct access volume will be mounted cannot be released until the volume is mounted. WORK3MP in ALCWA points to a group mask of the device groups that must remain serialized for volume mounting. Generic Allocation Control subtracts this mask from the current UNLKMASK to obtain an updated UNLKMASK.

The Allocation Queue Manager (IEFAB4FA) releases the device groups indicated in the updated UNLKMASK.

IEFAB4FA

Note: Because Generic Allocation Control considered all device groups in building the UNLKMASK, UNLKMASK can indicate that device groups that were never serialized should be released. The Allocation Queue Manager ignores such contradictions.)

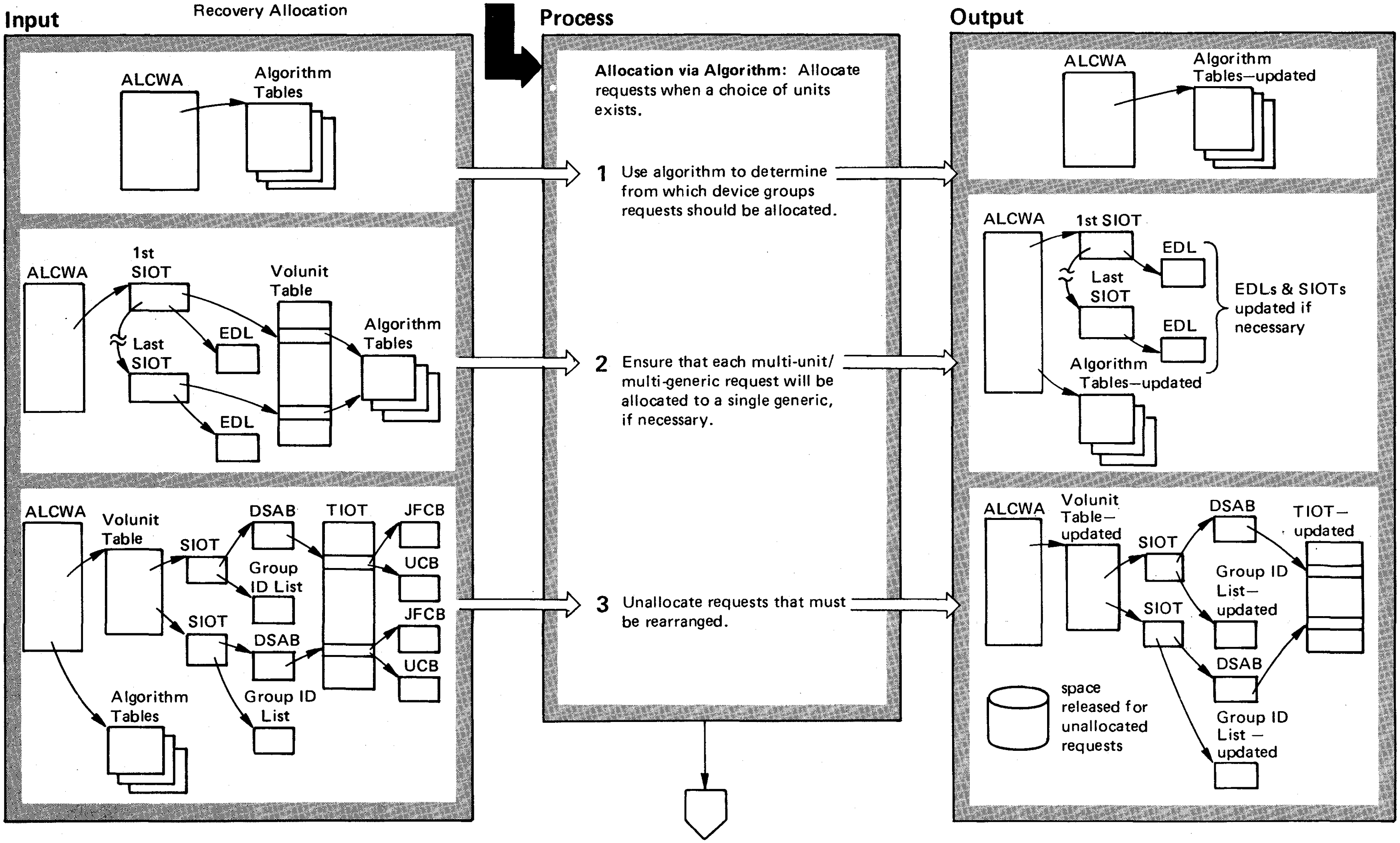
Error Processing

An error in any routine causes control to be returned to the calling routine.

In the event of an abnormal termination, the ESTAE exit routine established by IEFAB421 performs any necessary cleanup.

Diagram 14-10. IEFAB476 – Allocation via Algorithm (Part 1 of 6)

ENTRY from IEFAB475 (see IEFAB471 – Generic Allocation Control) or IEFAB485 – Recovery Allocation



Allocation via Algorithm: Allocate requests when a choice of units exists.

1 Use algorithm to determine from which device groups requests should be allocated.

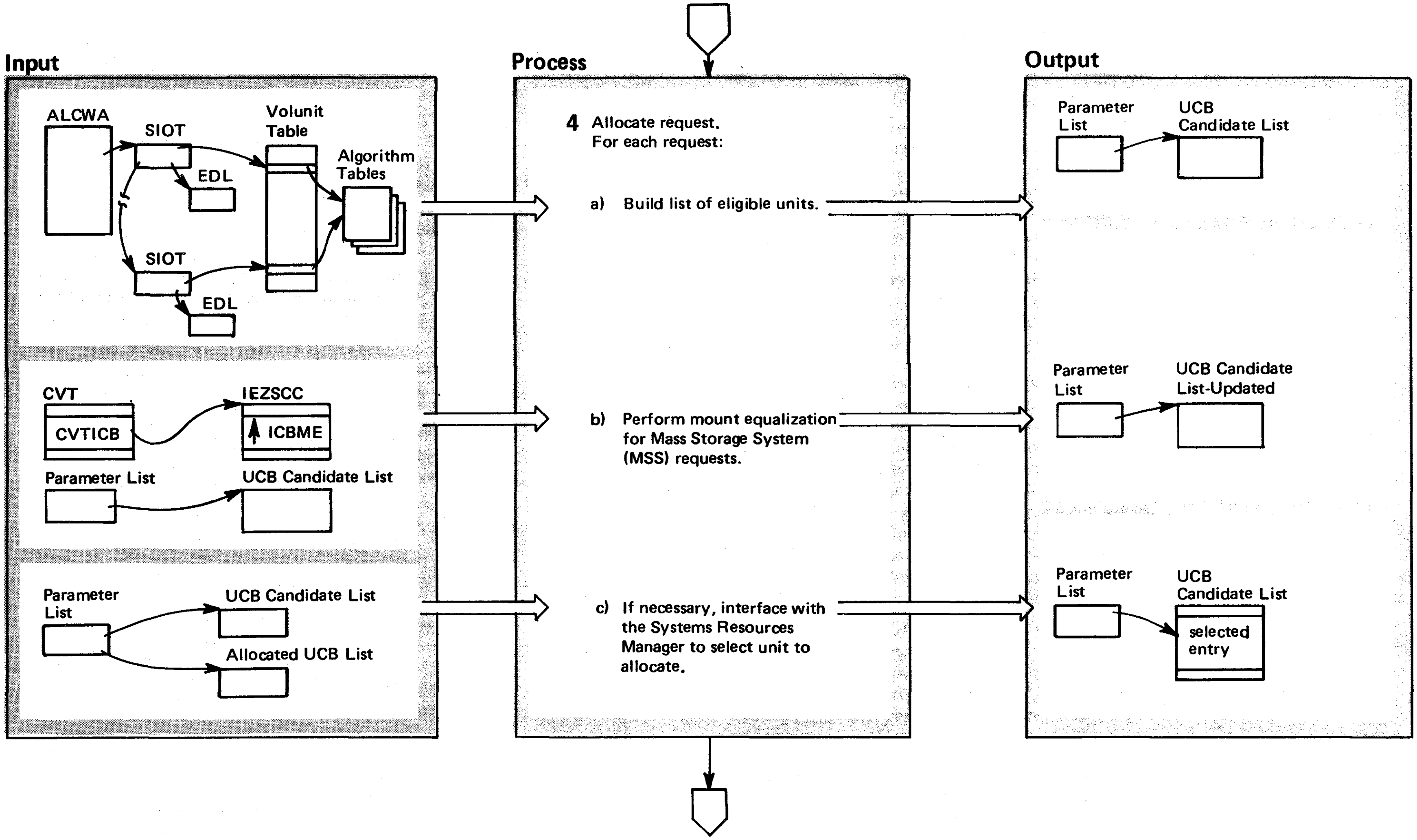
2 Ensure that each multi-unit/multi-generic request will be allocated to a single generic, if necessary.

3 Unallocate requests that must be rearranged.

Diagram 14-10. IEFAB476 – Allocation via Algorithm (Part 2 of 6)

Extended Description	Module	Segment
ENTRY Allocation via Algorithm is called by Allocation within Generic (IEFAB475) and Recovery Allocation (IEFAB485) to process requests when a choice of units exists: a specific unit was not requested; the request cannot be satisfied by allocating it to a mounted volume; the request cannot be allocated to a permanently resident or reserved volume.		
1 The Cover/Reduce Algorithm selects device groups from which unallocated requests should be allocated; the caller indicates in the cover/reduce request list of the algorithm tables the requests that should be processed by the algorithm (CVRSKFLG=0). The Cover/Reduce Algorithm updates the algorithm tables to indicate its selections; a return code indicates if all or only part of the requests to be considered were processed. Further processing of requests not processed by the algorithm (that is, for which the algorithm did not select a device group) is deferred.	IEFAB480	
2 When the algorithm chooses device groups from which requests should be allocated, it does not ensure that a multi-unit request eligible to more than one generic is allocated to a single generic. Only certain multi-unit tape requests can be allocated to more than one generic because of the dual density feature. Each multi-unit request that must be allocated to a single generic – each *MUG request* – was assigned an id (SIOTAFID≠0 in the SIOT) by IEFAB472 during generic allocation. Requests that specify affinity to a MUG request were assigned the same SIOTAFID. The purpose of this step is to ensure that each MUG request will be allocated to a single generic.		
IEFAB474 (Process Multi-Unit/Generic Requests) locates MUG requests with the same SIOTAFID that were assigned to more than one generic by the algorithm, and that were completely processed by the algorithm. (If requests with the same SIOTAFID were not completely processed by the algorithm, further processing of them is deferred.)	IEFAB474	HOWALGC
IEFAB474 tries to satisfy each such MUG request by considering the units that the algorithm selected for the request and:	IEFAB474	FORCEGEN
• The excess (unused) units in the last generic acquired if the caller is Allocation Within Generic.		
• The excess (unused) units in each acquired generic to which the request is eligible if the caller is Recovery Allocation.		
If the MUG request can be satisfied with a single generic, IEFAB474 indicates its selection in the algorithm tables; IEFAB481 (Eliminate Ineligible Groups) updates the EDL and the algorithm tables to mark every other generic ineligible (thereby preventing the algorithm from later rearranging this request to another generic) and sets to zero the SIOTAFID field (thereby indicating that the MUG request is successfully processed). IEFAB481 is also called to update the algorithm tables and the EDL and to set to zero the SIOTAFID field for MUG requests that the algorithm did assign to a single generic.	IEFAB481	
If IEFAB474 cannot successfully process the MUG request, it updates the algorithm tables to cancel the algorithm's selection of device groups for this request. The request is not further processed at this time. (When the caller is Allocation Within Generic, the MUG request might be satisfied with a single generic when a subsequent generic is serialized; otherwise, the request will be processed during recovery allocation. When the caller is Recovery Allocation, MUG requests that cannot be successfully processed at this time will be handled by Offline/Allocated Device Allocation – IEFAB486.)	IEFAB474	GIVEBACK
3 In order to find sufficient units to allocate a request, the algorithm might have had to rearrange requests already allocated – that is, indicate that an allocated request must be assigned to a different device group in order to free units needed to satisfy an unallocated request. Requests to be rearranged are indicated in the group list entry of the algorithm tables – the number of units already allocated (CVRGRPAL) is greater than the number of units the algorithm can select for allocation (CVRGALL). The purpose of this step is to unallocate requests that must be rearranged.		
IEFAB477 (Unallocate Requests to be Rearranged) updates the TIOT, volunit table, SIOT, group id list, and the UCB for each request to be unallocated; Common Unallocation Control releases any space obtained for the request.	IEFAB477 IEFAB4A0	

Diagram 14-10. IEFAB476 – Allocation via Algorithm (Part 3 of 6)



4 Allocate request. For each request:

a) Build list of eligible units.

b) Perform mount equalization for Mass Storage System (MSS) requests.

c) If necessary, interface with the Systems Resources Manager to select unit to allocate.

Diagram 14-10. IEFAB476 – Allocation via Algorithm (Part 4 of 6)

Extended Description	**Module**	**Segment**
4 IEFAB478 (Allocate from Groups the Algorithm Picked) allocates requests to units in the device groups selected by the algorithm. IEFAB478 searches the SIOT chain for unallocated requests that are not part of a MUG group (SIOTAFID=0). For each unallocated volunit entry processed by the algorithm (the algorithm tables indicate that the algorithm chose a device group from which to allocate the entry), the following steps are performed:	IEFAB478	
a) IEFAB478 builds a list of UCBs (UCB candidate list) that are included in the device group selected by the algorithm and that meet the following conditions: • The unit is online and unallocated. • The unit is not in use by a system task. • The volume on the unit is not permanently resident or reserved. The EDL for the SIOT contains the addresses of UCBs in the device groups eligible to the request.	IEFAB478	GENAPREP
b) If the request is for an MSS device, IEFAB478 interfaces with ICBME, which will return a preferred list of UCBs. This list will then be used to merge into and update the UCB candidate list. (See *OS/VS2 Mass Storage System Communicator (MSSC) Logic* for information on module ICBME.)	IEFAB478 ICBME	MONTEQAL
c) If the UCB candidate list contains more than one entry, IEFAB478 interfaces with the System Resources Manager to select the device to be allocated. IEFAB440 builds a list of allocated UCBs. The System Resources Manager uses this list and the UCB candidate list to determine which unit should be allocated.	IEFAB478 IEFAB440	GALGALOC

Diagram 14-10. IEFAB476 – Allocation via Algorithm (Part 5 of 6)

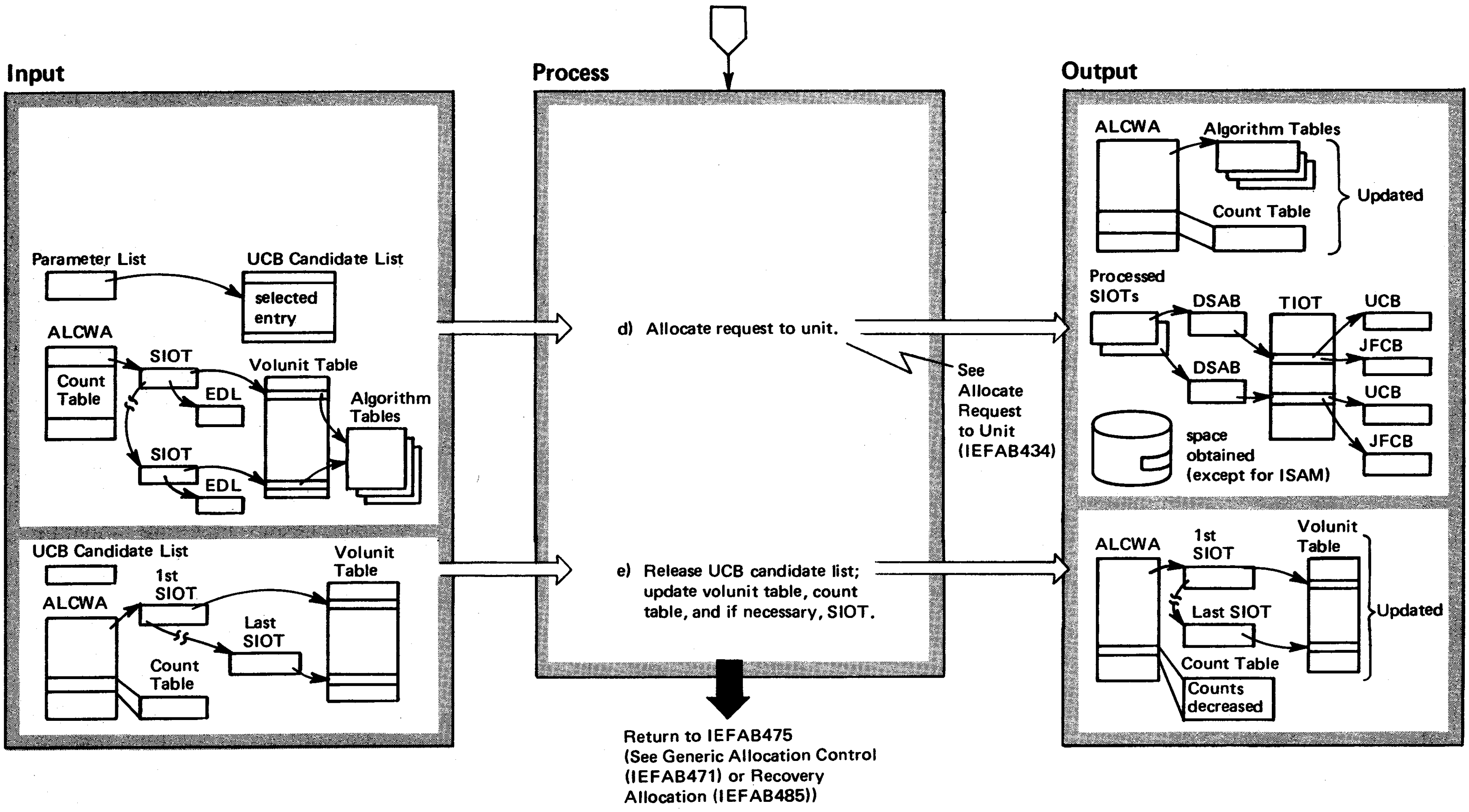


Diagram 14-10. IEFAB476 – Allocation via Algorithm (Part 6 of 6)

Extended Description	Module	Segment
d) IEFAB434 (Allocate Request to Unit) allocates the unit. For details, see the M.O. diagram Allocate Request to Unit (IEFAB434).	IEFAB434	
e) IEFAB478 releases the UCB candidate list, marks the volunit entry as allocated, and decreases the TOTVOLUN field in the count table. If the SIOT is completely allocated, IEFAB478 also marks the SIOT as allocated and decreases the TOTREQS field in the count table.	IEFAB478 IEFAB478 IEFAB478	GENAPREP GARURTN

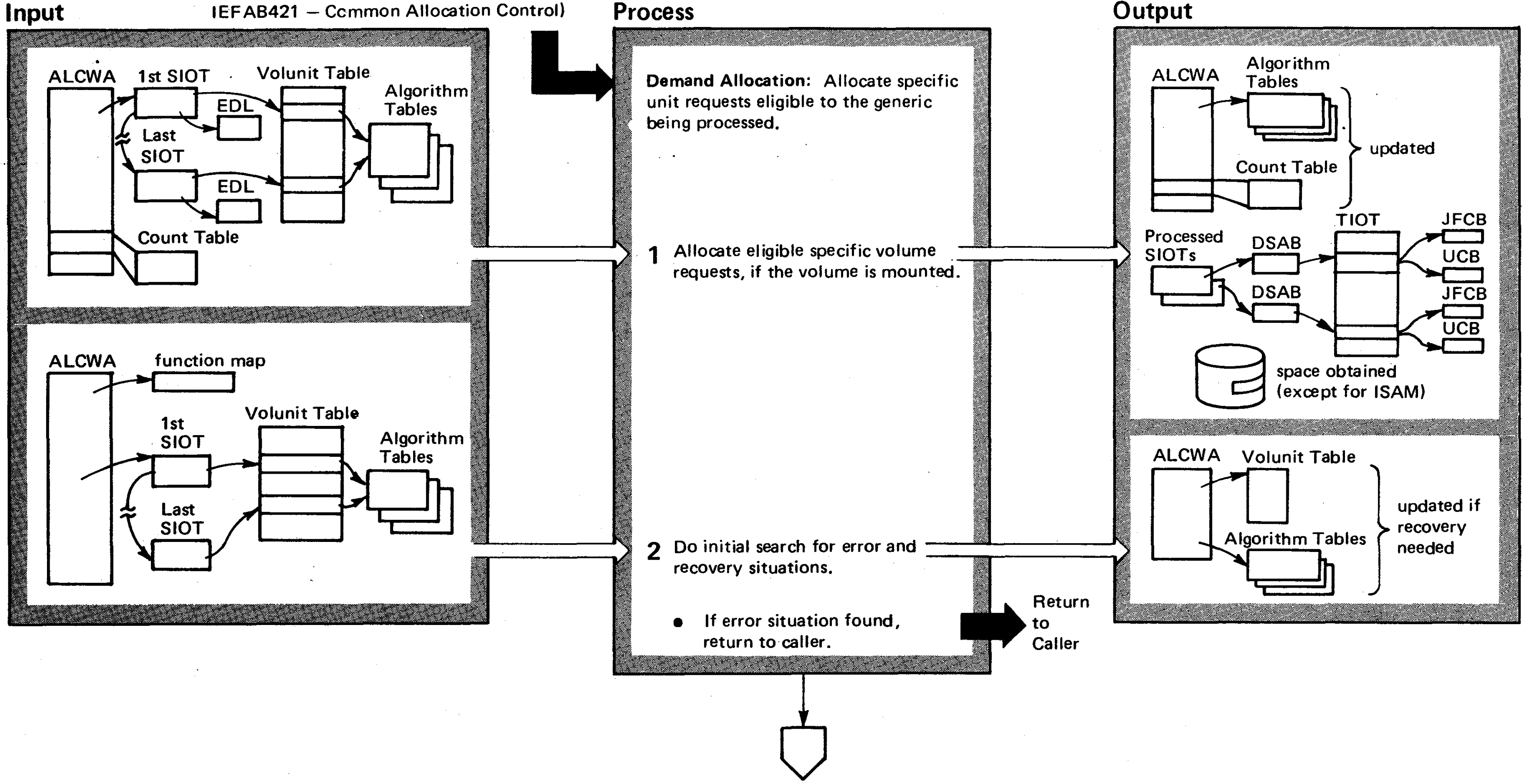
Error Processing

An error in any routine causes control to be returned to the calling routine.

In the event of an abnormal termination, the ESTAE exit routine established by IEFAB421 performs any necessary cleanup.

Diagram 14-11. IEFAB479 – Demand Allocation (Part 1 of 4)

ENTRY from IEFAB475 (see IEFAB471 – Generic Allocation Control); IEFAB485 – Recovery Allocation; or IEFAB491 (see IEFAB421 – Common Allocation Control)



Demand Allocation: Allocate specific unit requests eligible to the generic being processed.

1 Allocate eligible specific volume requests, if the volume is mounted.

2 Do initial search for error and recovery situations.

- If error situation found, return to caller.

Diagram 14-11. IEFAB479 – Demand Allocation (Part 2 of 4)

Extended Description | **Module** | **Segment**

ENTRY A demand request is a specific unit request; that is, a unit address was specified, such as UNIT=190. Demand Allocation allocates demand requests that were not allocated by Fixed Device Control. (Fixed Device Control allocated demand requests for direct access devices, if the volumes on the units were permanently resident or reserved.) Demand Allocation is called by:

- IEFAB475 (Allocation Within Generic) to allocate demand requests within the generic being processed by IEFAB475.
- IEFAB485 (Recovery Allocation) to allocate demand requests for tape devices when the needed volume is mounted on a tape device type different from the requested device type. (Recovery Allocation turns off the recovery indicator in requests that are to be processed before it calls Demand Allocation.)
- IEFAB491 (Wait Holding Resources) when a needed unit becomes unallocated and demand requests remain to be allocated.

Note: When a unit address is specified, all device groups within the generic are serialized.

1 This step is performed only if the caller indicated that specific volume requests were included in the demand requests to be processed. Specific Volume Allocation Control allocates these requests if the volume is mounted. If the volume is not mounted, the request is processed in step 3. For details on Specific Volume Allocation Control, see the M.O. diagram Specific Volume Allocation Control (IEFAB433).

IEFAB479

IEFAB433

2 Demand Allocation checks for error and, if the caller is IEFAB475, recovery situations in unallocated demand requests eligible to the generic being processed. The difference between error and recovery situations depends to a great extent on what is allowed for this allocation, as indicated in the function map: whether this allocation can consider offline devices; whether volumes can be mounted; whether this allocation can wait for units. (The function map is part of the Common Allocation Parameter List and is illustrated in figure 17.) To check for error and recovery situations, Demand Allocation examines the UCB for the requested unit.

IEFAB479 FIRSTSCN

Extended Description | **Module** | **Segment**

If an error situation is detected, Demand Allocation returns to the caller; further processing of requests is terminated. The possible error situations detected at this time are:

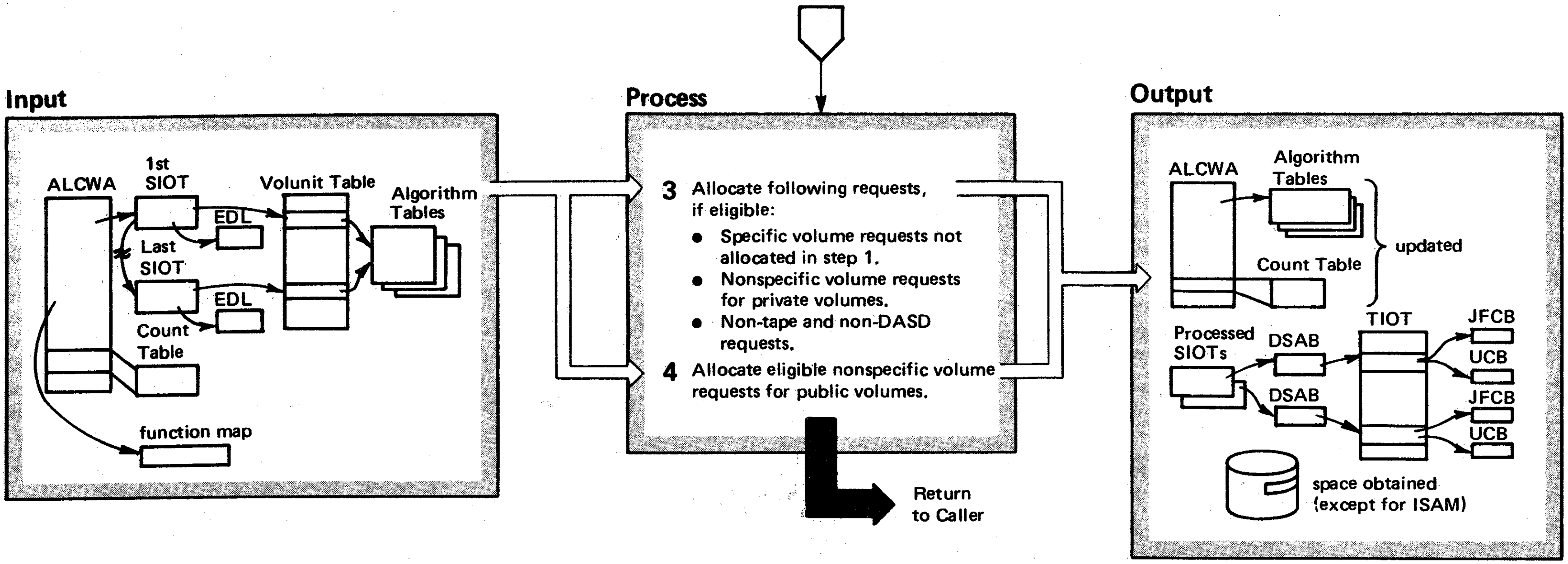
- The unit is in use by a system task (for example, OLTEP) or is an active console.
- More than one unit is needed. (When processing affinities to a prior request, the Affinity Processor (IEFAB432; see the M.O. diagram Allocate Request to Unit (IEFAB434) increased the unit count for a request if the needed volume was mounted on a different unit and was reserved.)
- The unit is offline and either: 1) this allocation cannot consider offline devices; or 2) this allocation can consider offline devices but volume mounting is not allowed. (The second condition applies only to tape and direct access devices.)
- The unit is tape or direct access, is allocated to another user (and is either nonshareable or is being requested nonshareable); this allocation does not allow waiting.
- The unit is tape or direct access and is not allocated, but either no volume or the wrong volume is mounted and volume mounting is not allowed.
- The unit is neither tape nor direct access, is allocated to another user, and waiting is not allowed.
- The unit is direct access, contains a permanently resident or reserved volume, and a different volume is requested, or the request is non-specific and private but the mounted volume is public.
- The unit is tape, contains a reserved volume, and a different volume is requested, or the request is non-specific and private but the mounted volume is public.

If a recovery situation is detected (possible only when the caller is IEFAB475), Demand Allocation updates the volunit entry and the algorithm tables to indicate that the request must be processed by retry. Requests that specify affinity are also marked for recovery. The only recovery situation detected at this time is:

IEFAB479 MARKVUS

- The unit is offline, but this allocation can consider offline devices and, for tape or direct access, volume mounting is allowed.

Diagram 14-11. IEFAB479 – Demand Allocation (Part 3 of 4)



3 Allocate following requests, if eligible:

- Specific volume requests not allocated in step 1.
- Nonspecific volume requests for private volumes.
- Non-tape and non-DASD requests.

4 Allocate eligible nonspecific volume requests for public volumes.

Extended Description	Module	Segment
3 Demand Allocation processes the following demand requests if they are eligible to this generic group (SIOTGIGN=0) and are not marked for recovery (VUDNALOC=0 in the volunit entry):	IEFAB479	ALOCSPFC
• Specific volume requests when the volume is not mounted on the requested unit. (Specific volume requests to mounted volumes were allocated in step 1.)		
• Private non-specific volume requests.		
• Non-tape and non-DASD requests.		
IEFAB434 allocates each request. (For details, see the M.O. diagram Allocate Request to Unit (IEFAB434)). When allocating specific volume requests and when the caller of Demand Allocation is IEFAB475, Allocate Request to Unit can encounter the following special situations, resulting in recovery or termination:	IEFAB434	
• The needed volume is mounted on a device that is allocated to another user. If this allocation can wait for allocated units (as indicated in the function map of the common allocation parameter list; see figure 17), this request and any requests specifying affinity to it are marked for recovery (VUDNALOC=1 in the volunit entry). If waiting is not allowed, Demand Allocation returns to Allocation within Generic and further processing of requests is terminated.		
• The request is tape and the needed volume is mounted on a different generic device type. The request is marked for retry processing.		
As requests are successfully allocated, Demand Allocation marks the volunit entry and SIOT as allocated and decreases the TOTREQS and TOTVOLUN fields in the count table.	IEFAB479	ALOCSPFC

Diagram 14-11. IEFAB479 – Demand Allocation (Part 4 of 4)

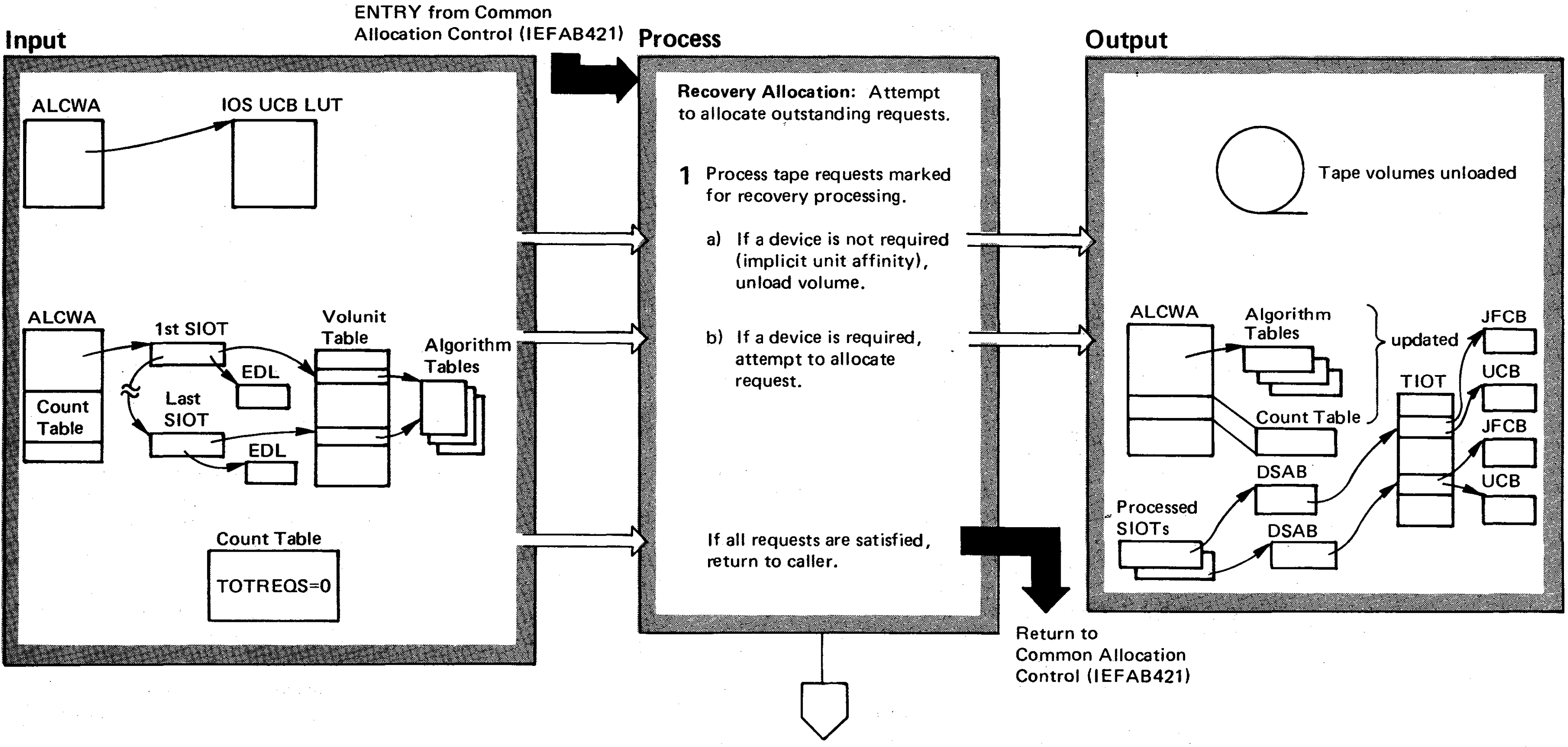
Extended Description	Module	Segment
4 Demand Allocation processes demand requests for nonspecific public volumes if the request is eligible to this generic (SIOTGIGN=0) and is not marked for recovery (VUDNALOC=0 in the volunit entry). Processing depends on the caller. If the caller is IEFAB485, this step is never performed. If the caller is IEFAB491, step 4b is performed. If the caller is IEFAB475, processing depends on whether a volume is mounted on the unit: if a volume is mounted, step 4a is performed; otherwise, step 4b is performed.	IEFAB479	SECNDSCN
a) If a volume is mounted on the requested unit and IEFAB475 called Demand Allocation, the Conditional ENQ/DEQ Routine enqueues on the volume. The enqueue results in one of the following situations:	IEFAB4F0	
● The enqueue is unsuccessful because another request *within* this allocation already owns the volume. The volume can be used.		
● The enqueue is successful; the volume can be used.		
● The enqueue is unsuccessful because another user (that is, a different allocation) owns the volume and the volume cannot be shared with this allocation. If this allocation can wait for units, this request and requests that specify affinity to it are marked for recovery processing. If this allocation cannot wait for units, Demand Allocation returns to Allocation Within Generic and further processing is terminated.	IEFAB479	MARKVUS
If the volume can be used, IEFAB434 allocates the unit. For a new non-ISAM data set, IEFAB434 also interfaces with DADSM for space. If sufficient space is not available, processing depends on whether this allocation already owned the volume:	IEFAB434	
● The volume is already owned by this allocation. If the request previously allocated to the volume is a specific volume request, Demand Allocation returns to Allocation Within Generic and further processing is terminated.	IEFAB479	
Otherwise, the request just allocated is unallocated; this request and the previously-allocated request are marked for processing by Recovery Allocation.	IEFAB479	MARKVUS
● The volume is not already owned by this allocation. The Conditional ENQ/DEQ Routine dequeues from the volume and Demand Allocation recalls IEFAB434 indicating that the volume is unacceptable. IEFAB434 unloads that volume and builds a volume mount and verify request block for this request. (The volume is not mounted until all requests are successfully allocated; see the M.O. diagram Common Allocation Cleanup (IEFAB490).)	IEFAB4F0 IEFAB434	
For details on IEFAB434, see the M.O. diagram Allocate Request to Unit (IEFAB434).		
b) If the caller is IEFAB491, or if the caller is IEFAB475 and a volume is not mounted on the unit, IEFAB434 allocates the unit and builds a volume mount and verify request block for this request. For details on IEFAB434, see the M.O. diagram Allocate Request to Unit (IEFAB434).	IEFAB434	
If the unit is successfully allocated, Demand Allocation marks the volunit entry and SIOT as allocated and decreases the TOTREQS and TOTVOLUN fields in the count table.	IEFAB479	

Error Processing

An error in any routine causes control to be returned to the calling routine.

In the event of an abnormal termination, the ESTAE exit routine established by IEFAB421 performs any necessary cleanup.

Diagram 14-12. IEFAB485 – Recovery Allocation (Part 1 of 8)



Recovery Allocation: Attempt to allocate outstanding requests.

1 Process tape requests marked for recovery processing.

a) If a device is not required (implicit unit affinity), unload volume.

b) If a device is required, attempt to allocate request.

If all requests are satisfied, return to caller.

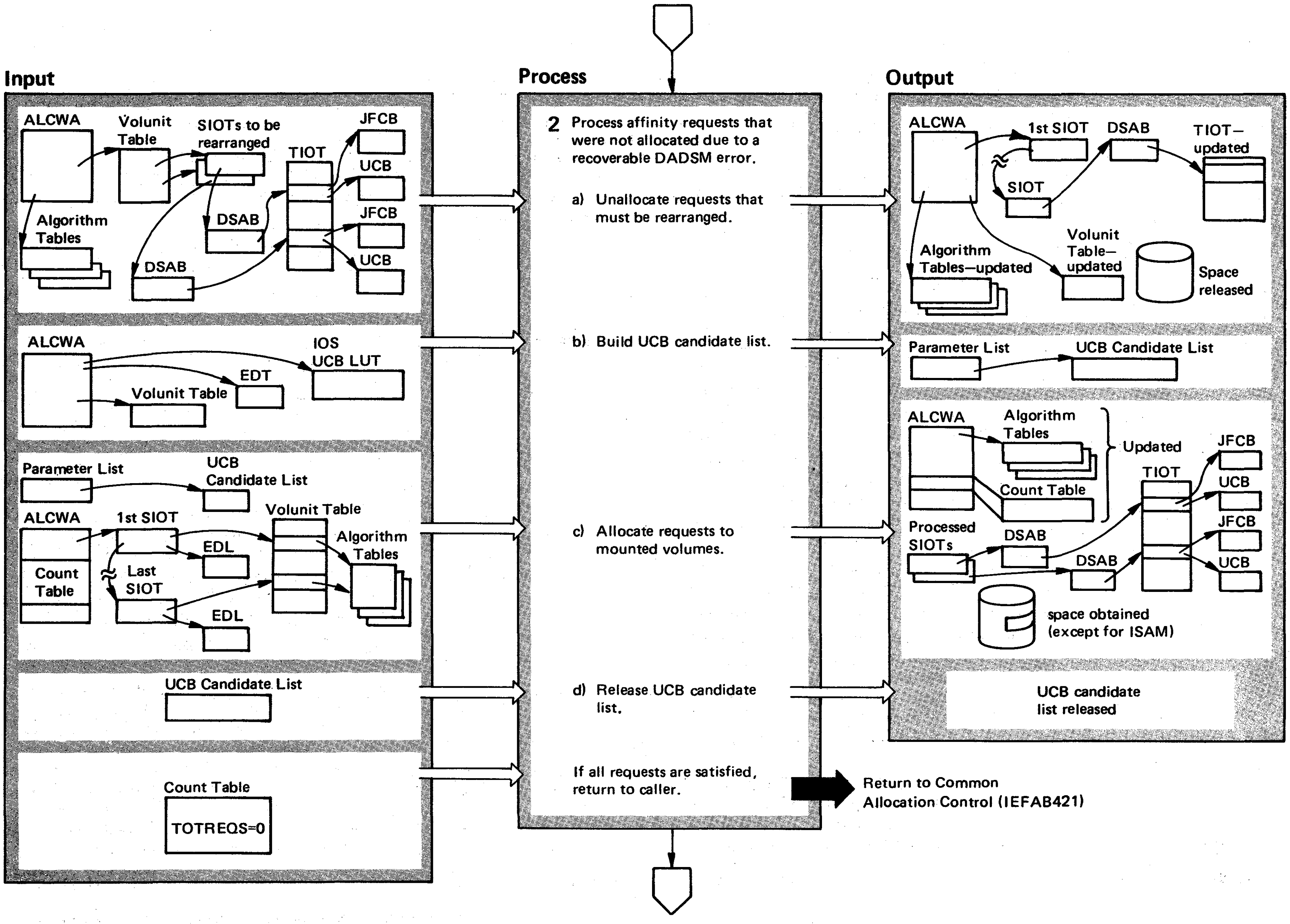
Diagram 14-12. IEFAB485 – Recovery Allocation (Part 2 of 8)

Extended Description	Module	Segment
ENTRY Recovery Allocation is called by Common Allocation Control if both of the following conditions are true:		
• Requests still remain to be allocated. (The TOTREQS field in the count table does not equal 0.)		
• Retry is not indicated (INDRETRY=0 in ALCWA). (Retry is indicated 1) when a needed DASD volume was mounted on a unit not included in the serialized device groups, or 2) a needed tape volume was found on a unit of a different device type, but the device type is compatible with the one being processed by Generic Allocation. For retry, allocated requests are unallocated and the allocation is reattempted; therefore, it is unnecessary to perform recovery allocation.)		
Recovery Allocation attempts to allocate all remaining requests; the four steps in this diagram reflect the four different situations that result in Recovery Allocation. In the first three steps, only unallocated, allocated but shareable, and online devices are considered for allocation; in the fourth step, Recovery Allocation will consider offline and/or allocated (nonshareable) devices, if allowed for this allocation (as indicated in the function map of the common allocation parameter list).		
In the first three steps, unsuccessful attempts to allocate requests do not necessarily result in failure of the allocation, because of the possibility that the requests can be satisfied in step 4 if offline and/or allocated devices can be considered.		
1 During Generic Allocation Control, tape requests were marked for retry processing. Then in retry processing such tape requests were also marked for recovery processing, if a needed volume was located on a generic different from the generic selected for allocation. This step processes these requests. (For background information, see "Processing Tape Requests" in the "Introduction to Allocation/Unallocation.")		
Recovery Allocation scans the volunit table for tape requests (VOLTAREQ=1) that indicate recovery processing is necessary and that indicate the needed volume was mounted on a device type other than the one selected by Generic Allocation. Processing of these requests depends on if 1) a device is not required for this volunit entry; or 2) a device is required for this volunit entry.	IEFAB485	TAPEVALI
a) A device is not required if more volumes than units were requested (implicit unit affinity) – for example, the user coded:	IEFAB485	TVALUNLD

```
//DD1 DD DSN=ALLOC,DISP=OLD,UNIT=2400, *
//           VOL=SER=(A,B)
```

Extended Description	Module	Segment
During generic allocation, volume A was successfully allocated to a 2400 device but volume B was mounted on a device of a different generic type. Recovery Allocation searches the UCBs (by means of the IOS UCB LUT) to locate the needed volume. If the volume is found, the Unload Interface unloads the volume. (If the volume is not found, it has already been unloaded; for example, another request needed the unit on which it was mounted and had the volume unloaded.)	IEFAB49C	
b) If a device is required, Specific Volume Allocation Control tries to allocate the volume where it is mounted. (For details on Specific Volume Allocation Control, see the M.O. diagram Specific Volume Allocation Control (IEFAB433). If the request cannot be allocated where it is mounted (for example, although the volume is mounted on a compatible device type, the request is not eligible to that generic – see "Processing Tape Requests" in the "Introduction to Allocation/Unallocation") and if volume mounting is allowed (as indicated in the function map of the common allocation parameter list):	IEFAB433	
• Demand Allocation receives control, if there are demand requests. For details on Demand Allocation, see the M.O. diagram Demand Allocation (IEFAB479).	IEFAB479	
• Allocation via Algorithm receives control if there are requests other than demand requests. For details on Allocation via Algorithm, see the M.O. diagram Allocation via Algorithm (IEFAB476).	IEFAB476	
If volume mounting is not allowed, this allocation is failed. Recovery Allocation returns to Common Allocation Control.		
Note: Unsuccessful attempts to allocate requests by Demand Allocation or Allocation via Algorithm do not result in failure, because of the possibility of allocating these requests in step 4 if offline and/or allocated devices can be considered.		

Diagram 14-12. IEFAB485 – Recovery Allocation (Part 3 of 8)



2 Process affinity requests that were not allocated due to a recoverable DADSM error.

a) Unallocate requests that must be rearranged.

b) Build UCB candidate list.

c) Allocate requests to mounted volumes.

d) Release UCB candidate list.

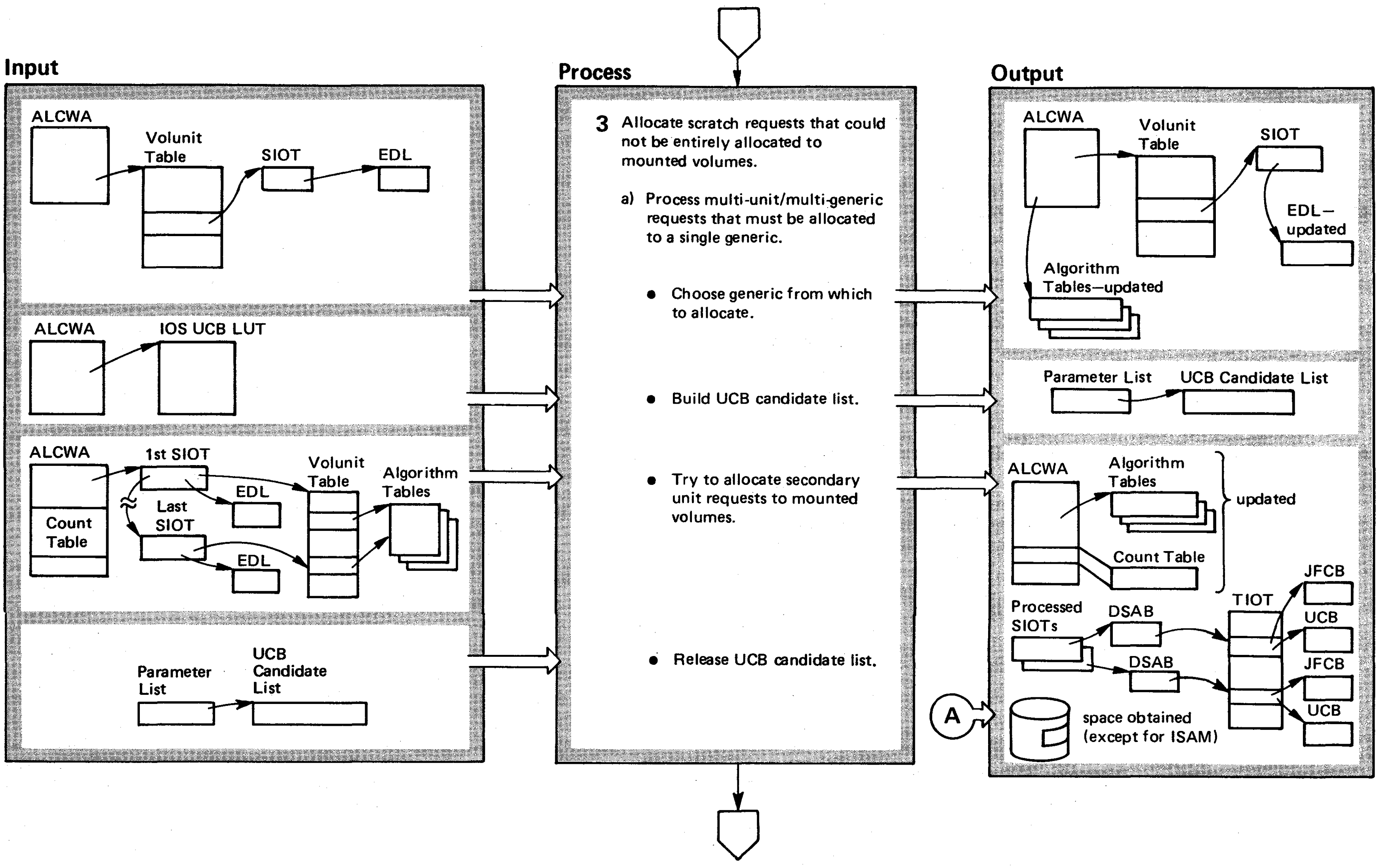
If all requests are satisfied, return to caller.

Return to Common Allocation Control (IEFAB421)

Diagram 14-12. IEFAB485 – Recovery Allocation (Part 4 of 8)

Extended Description	Module	Segment
2 This step handles the following situation: two or more nonspecific direct-access volume requests specified volume affinity to each other; at least one of these requests was successfully allocated, but a subsequent request could not be allocated to the same volume because of insufficient space (that is, a recoverable DADSM error). Recovery Allocation unallocates the request(s) that were allocated and tries to find a volume to which all the requests specifying affinity to each other can be allocated.	IEFAB485	DADAFFER
Recovery Allocation searches the volunit table for entries that indicate a recoverable DADSM error occurred while allocating an affinity request (VUDADSME=1). The following steps are performed for these requests:		
a) IEFAB477 unallocates the requests that were successfully allocated: this routine updates the TIOT, volunit table, SIOT, and UCB; Common Unallocation Control releases the direct access space.	IEFAB477 IEFAB4A0	
b) Recovery Allocation uses the SYSALLDA entry of the EDT to search the IOS UCB LUT for direct access units with mounted volumes that are eligible to satisfy nonspecific volume requests. To be eligible, a UCB must meet the following conditions: • The UCB is online, ready, and not pending offline. • The UCB is not in use by a system task (UCBNALOC=0). • No unload or mount is pending for the UCB. • The volume on the unit is shareable and has a use attribute of public or storage. • No other request is waiting for this unit to be unallocated so that the volume mounted on the unit can be moved to another unit. Pointers to eligible UCBs are placed in a UCB candidate list; this list is used by Nonspecific Volume Allocation Control to determine the units eligible to satisfy unallocated requests.	IEFAB485	DAFERLST
c) Recovery Allocation calls Nonspecific Volume Allocation Control to allocate: • Storage volume requests if there are any. • Public volume requests if there are any. The parameter list for Nonspecific Volume Allocation Control includes a function map that indicates what type of request should be processed and a pointer to the UCB candidate list. For details on Nonspecific Volume Allocation Control, see the M.O. diagram Nonspecific Volume Allocation Control (IEFAB436).	IEFAB485 IEFAB436	DADAFFER
d) Recovery Allocation issues a FREEMAIN macro instruction to release the UCB candidate list.	IEFAB485	DADAFFER

Diagram 14-12. IEFAB485 – Recovery Allocation (Part 5 of 8)



3 Allocate scratch requests that could not be entirely allocated to mounted volumes.

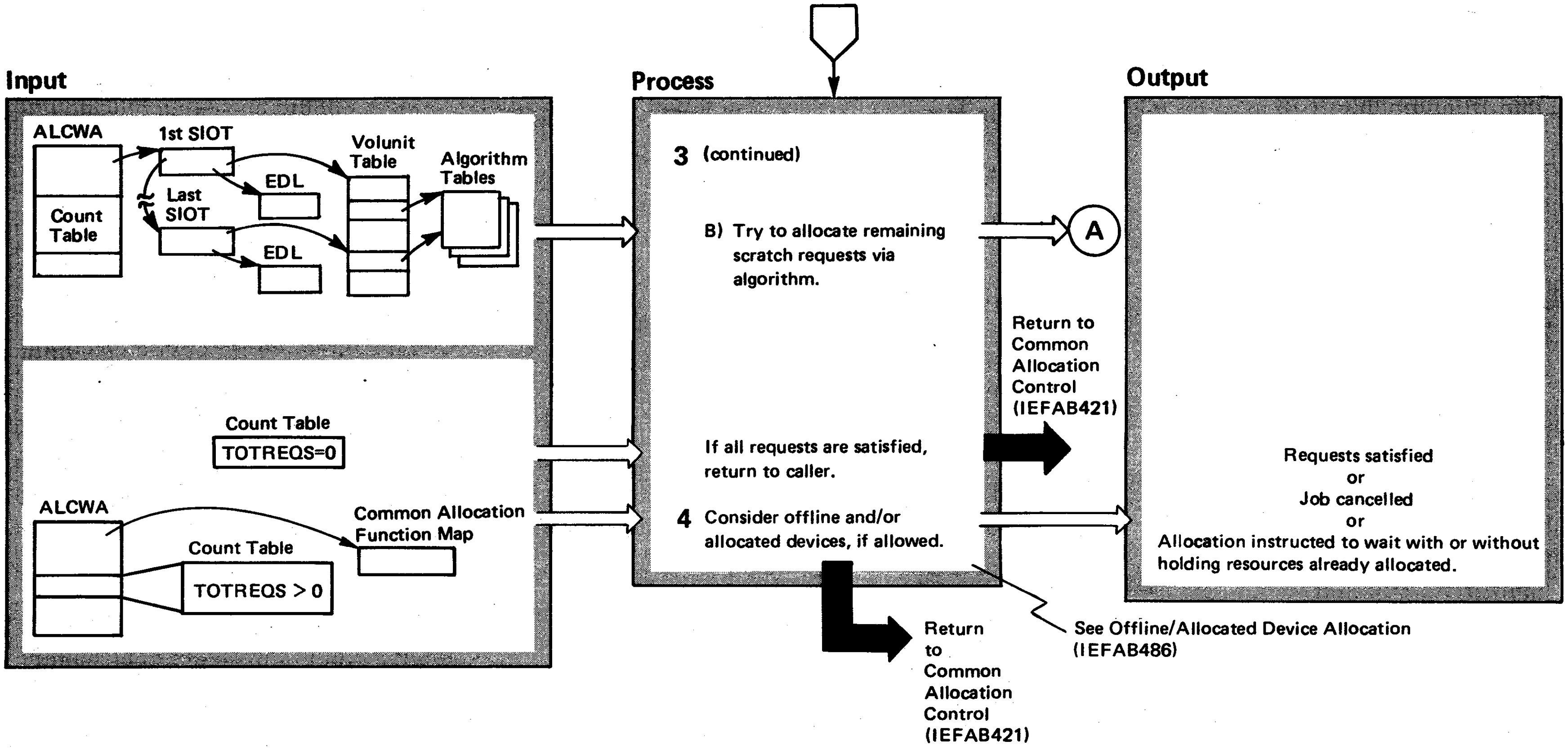
a) Process multi-unit/multi-generic requests that must be allocated to a single generic.

- Choose generic from which to allocate.
- Build UCB candidate list.
- Try to allocate secondary unit requests to mounted volumes.
- Release UCB candidate list.

Diagram 14-12. IEFAB485 – Recovery Allocation (Part 6 of 8)

Extended Description	Module	Segment
3 Generic allocation allocated scratch requests only if they could be completely allocated to mounted volumes – for example, if a SIOT needed two volumes and two units, it was allocated only if two mounted volumes were available. In this step, Recovery Allocation processes scratch requests not allocated during generic allocation. This step is not performed if volume mounting is not allowed (as indicated in the function map of the common allocation parameter list – see figure 2-27).	IEFAB485	SCRATALG
a) Recovery Allocation first processes multi-unit/multi-generic requests (that is, multi-unit requests that are eligible to more than one generic) that must be allocated to a single generic. (Allocation of a single request to more than one generic device type is allowed only for tape requests; this is possible because of the dual-density feature.) Nonspecific Volume Allocation Control will try to allocate as many of the secondary unit requests as possible to mounted volumes. The following steps are performed:	IEFAB485	SCRATALG
• For each multi-unit/multi-generic request, Recovery Allocation determines which generic eligible to the request contains the most mounted volumes that can be allocated to scratch requests. (If no mounted volumes are found in any of the eligible generics, the SIOT is marked ineligible (SIOTGIGN=1) so that Nonspecific Volume Allocation Control will not try to process it.) The EDLNSCNT field of the EDL contains the number of mounted volumes that can be allocated to nonspecific requests. IEFAB481 (Eliminate Ineligible Groups) updates the EDL and the algorithm tables by marking as ineligible all generics except the generic selected.	IEFAB485 IEFAB481	PICSCRAG
• Recovery Allocation searches the IOS UCB LUT for units eligible to satisfy scratch requests. To be eligible, a UCB must meet the following conditions: the unit is an unallocated tape or a shareable direct access device; the unit is online, ready, and not pending offline; no mount or unload is pending for the unit; the unit is not in use by a system task UCBNALOC=0); the use attribute is public or storage, not private; and no other request is waiting for this unit. Pointers to eligible UCBs are placed in a UCB candidate list to be used by Nonspecific Volume Allocation Control.	IEFAB485	SCRAMLST
• Nonspecific Volume Allocation Control allocates as many secondary unit requests as possible to mounted volumes. (The function map in the parameter list for Nonspecific Volume Allocation Control indicates "partially allocate" – that is, allocate only secondary unit requests.) For details on Nonspecific Volume Allocation Control, see the M.O. diagram Nonspecific Volume Allocation Control (IEFAB436).	IEFAB436	
• Recovery Allocation issues a FREEMAIN macro instruction to release the UCB candidate list.	IEFAB485	SCRATALG

Diagram 14-12. IEFAB485 – Recovery Allocation (Part 7 of 8)



3 (continued)

B) Try to allocate remaining scratch requests via algorithm.

If all requests are satisfied, return to caller.

4 Consider offline and/or allocated devices, if allowed.

Requests satisfied
or
Job cancelled
or
Allocation instructed to wait with or without holding resources already allocated.

Diagram 14-12. IEFAB485 – Recovery Allocation (Part 8 of 8)

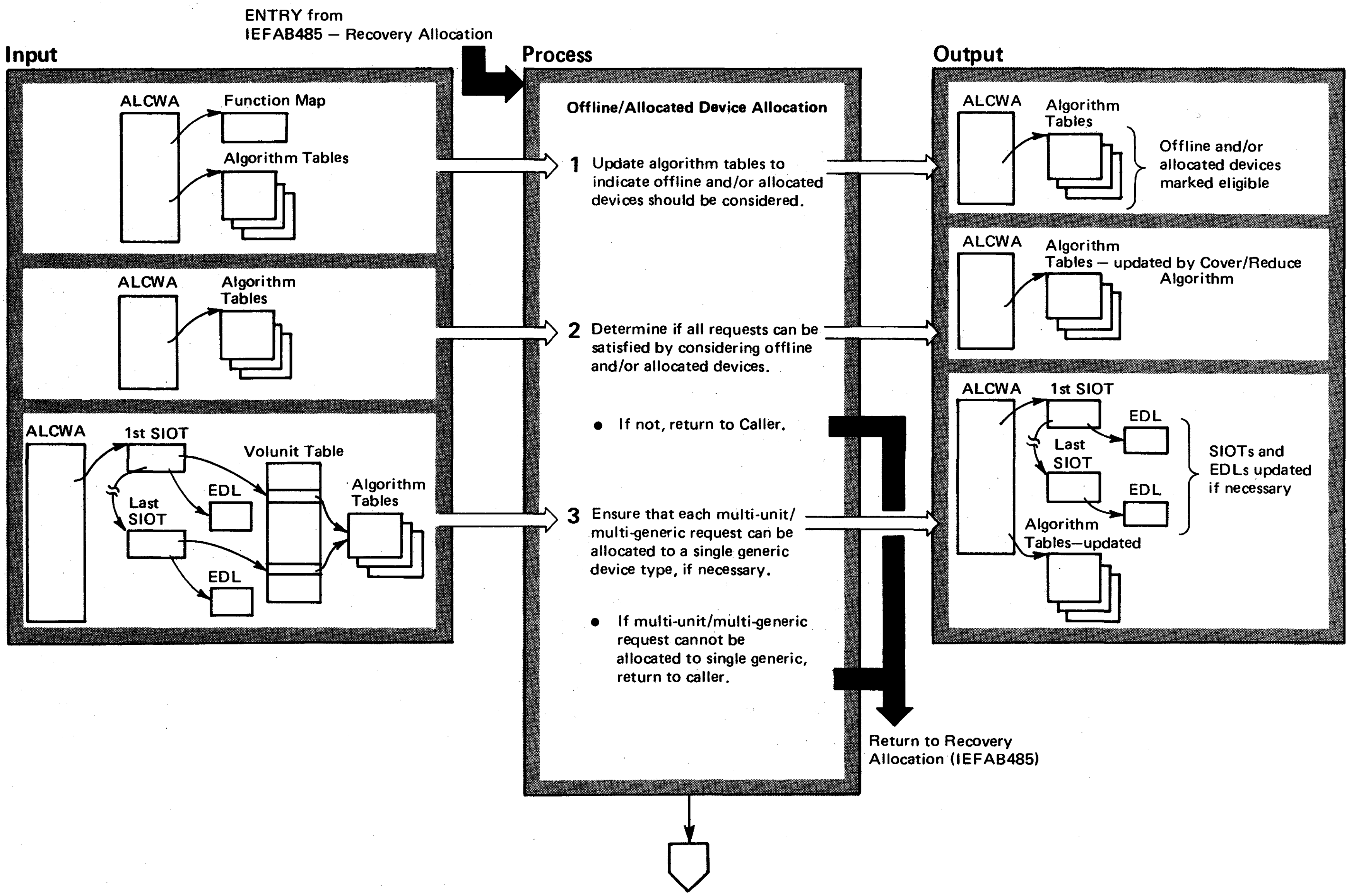
Extended Description	Module	Segment
3 (Continued)		
b) Allocation via Algorithm attempts to allocate all remaining scratch requests, including those secondary unit requests that Nonspecific Volume Allocation Control was unable to allocate. For details on Allocation via Algorithm, see the M.O. diagram Allocation via Algorithm (IEFAB476).	IEFAB476	
4 This step is performed only if: • all requests still have not been allocated; and • this allocation is allowed to consider offline and/or allocated devices (as indicated in the function map of the common allocation parameter list – see figure 2-27).		
If requests remain to be allocated, but neither offline nor allocated devices can be considered, Recovery Allocation returns to the caller and further processing is terminated. Otherwise, Offline/Allocated Device Allocation receives control.	IEFAB485	
Offline/Allocated Device Allocation first determines if all requests can be satisfied by considering offline and/or allocated devices. If not, this allocation is terminated. If all requests can be satisfied, the following processing occurs:	IEFAB486	
a) Requests eligible to available devices are allocated.	IEFAB478	
b) For each request that cannot be allocated to an available device, IEFAB487 (Allocation Recovery Interface with Operator) queries the operator for instructions. The operator can: • Cancel the job. Recovery Allocation returns to Common Allocation Control. • Instruct allocation to use an offline device, which allocation will then bring online. • Instruct allocation to wait for an allocated device(s), holding resources. Recovery Allocation returns to Common Allocation Control, which calls IEFAB491 (Wait Holding Resources). IEFAB491 waits for the needed unit(s) to be unallocated and then allocates the outstanding requests. For details, see the M.O. diagram Common Allocation Control (IEFAB421). • Instruct allocation to wait for an allocated device(s), without holding resources. Recovery Allocation returns to Common Allocation Control. Common Allocation Cleanup handles the wait-without-holding-resources situation – see the M.O. diagram Common Allocation Cleanup (IEFAB490).	IEFAB487	
For details on Offline/Allocated Device Allocation, see the M.O. diagram Offline/Allocated Device Allocation (IEFAB486).		

Error Processing

An error in any routine causes control to be returned to the calling routine.

In the event of an abnormal termination, the ESTAE exit routine established by IEFAB421 performs any necessary cleanup.

Diagram 14-13. IEFAB486 – Offline/Allocated Device Allocation (Part 1 of 12)



Offline/Allocated Device Allocation

1 Update algorithm tables to indicate offline and/or allocated devices should be considered.

2 Determine if all requests can be satisfied by considering offline and/or allocated devices.

- If not, return to Caller.

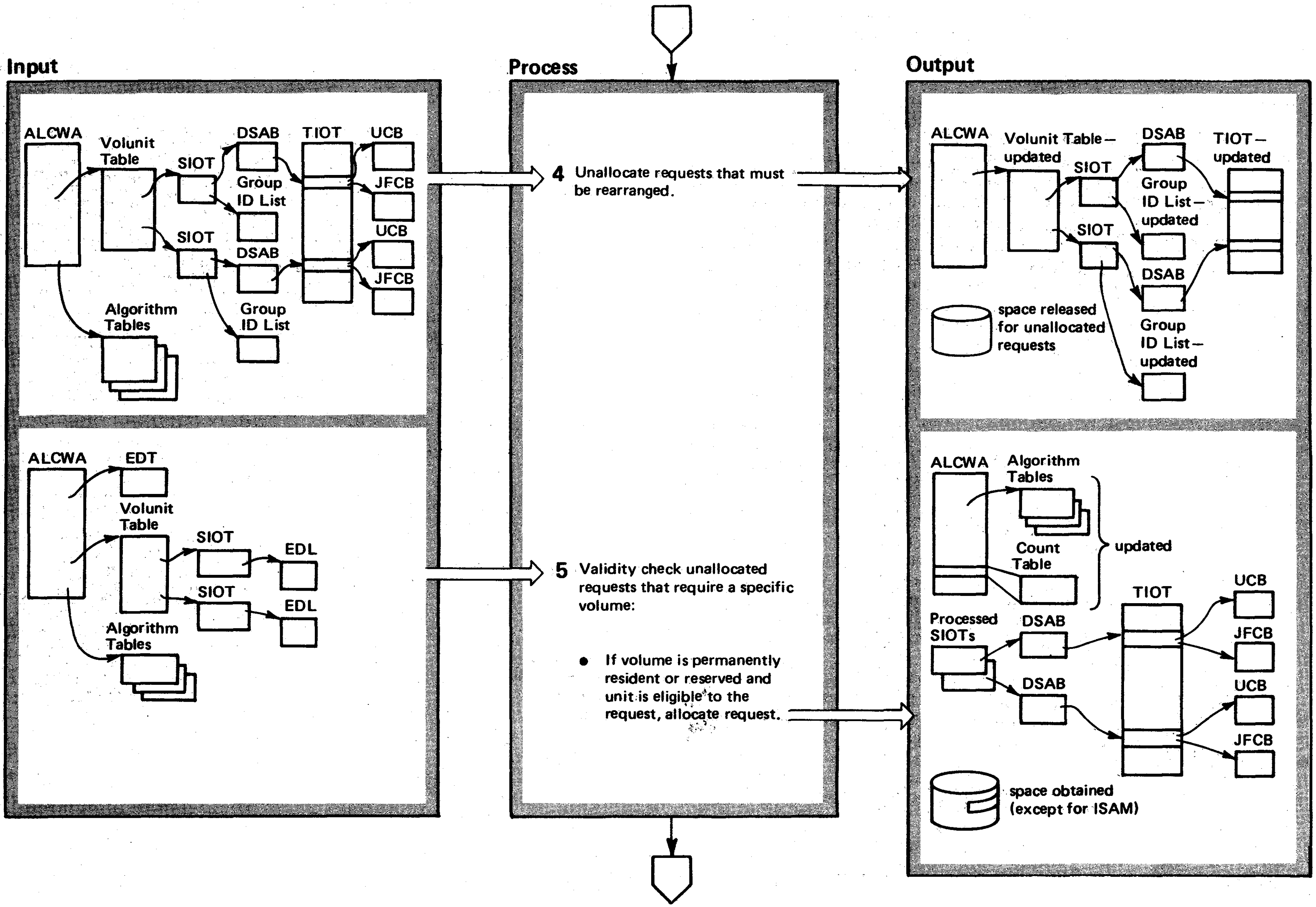
3 Ensure that each multi-unit/multi-generic request can be allocated to a single generic device type, if necessary.

- If multi-unit/multi-generic request cannot be allocated to single generic, return to caller.

Diagram 14-13. IEFAB486 – Offline/Allocated Device Allocation (Part 2 of 12)

Extended Description	Module	Segment
ENTRY Offline/Allocated Device Allocation is called by Recovery Allocation (IEFAB485) only if the following conditions are true:		
• all requests still have not been allocated; and,		
• this allocation is allowed to consider offline and/or allocated device (as indicated in the function map of the common allocation parameter list – see figure 2-27).		
1 Offline/Allocated Device Allocation updates the algorithm tables to indicate that:	IEFAB486	ALGAPREP
• Allocated devices can be considered if the function map indicates this allocation can wait for allocated devices.		
• Offline devices can be considered if the function map indicates this is allowed.		
Note: The algorithm tables are always updated in at least one of the preceding ways; if neither offline nor allocated devices could be considered, Offline/Allocated Device Allocation would not have received control.		
2 The Cover/Reduce Algorithm is called to determine:	IEFAB480	
• If all outstanding requests can be satisfied by considering offline and/or allocated devices.		
• The device groups from which unallocated requests should be allocated. The algorithm's selections are indicated in the algorithm tables.		
If all requests cannot be satisfied, this allocation is terminated; Offline/Allocated Device Allocation returns to Recovery Allocation.	IEFAB486	
3 When the algorithm chooses devices to be allocated to requests, it does not ensure that a multi-unit request eligible to more than one generic is allocated to a single generic. (Only multi-unit tape requests can be allocated to more than one generic, because of the dual density feature. Each multi-unit request that must be allocated to a single generic – each MUG request – was assigned an id (SIOTAFID≠ 0 in the SIOT) by IEFAB472 during generic allocation. Requests that specify affinity to a MUG request were assigned the same SIOTAFID.) The purpose of this step is to ensure that each MUG request will be allocated to a single generic. If this is not possible, the allocation is failed.		
Two processes are performed to ensure that MUG requests can be allocated correctly:		
a) IEFAB474 (Process Multi-Unit/Generic Requests) locates and processes each MUG request assigned to more than one generic by the algorithm.	IEFAB474	HOWALGC
IEFAB474 tries to satisfy each such request by considering the units that the algorithm selected for the requests and the excess (unused) units in each acquired generic to which the request is eligible. If one of the eligible generics can satisfy the request (by considering only the excess units and the units already assigned to the request), IEFAB474 indicates its selection in the algorithm tables;	IEFAB474	FORCEGEN
IEFAB481 (Eliminate Ineligible Groups) updates the EDL and the algorithm tables to mark every other generic ineligible (thereby preventing the algorithm from rearranging this request to other generics) and sets to zero the SIOTAFID field (thereby indicating that the MUG request is successfully processed).	IEFAB481	
If IEFAB474 cannot successfully process the MUG request, it updates the algorithm tables to cancel the algorithm's selection of units for this request; the request will then be handled in step 3b.	IEFAB474	GIVEBACK
IEFAB481 is also called to update the algorithm tables and EDL and to set to zero the SIOTAFID field for MUG requests that the algorithm did assign to a single generic.	IEFAB481	
b) Offline/Allocated Device Allocation processes any MUG requests that were not successfully handled by the algorithm or by IEFAB474 (SIOTAFID≠0). For each request, IEFAB486 considers each generic eligible to the request: it temporarily marks all other generics ineligible in the algorithm tables and then recalls the algorithm. This is repeated for each eligible generic until a generic is found that can satisfy the request or until all eligible generics have been considered. If no single generic can satisfy this request, the allocation is failed; IEFAB486 returns to Recovery Allocation.	IEFAB486	FORCMULT
Otherwise, IEFAB481 updates the algorithm tables and the EDL to permanently mark every other generic ineligible and sets to zero the SIOTAFID field.	IEFAB481	

Diagram 14-13. IEFAB486 – Offline/Allocated Device Allocation (Part 3 of 12)



4 Unallocate requests that must be rearranged.

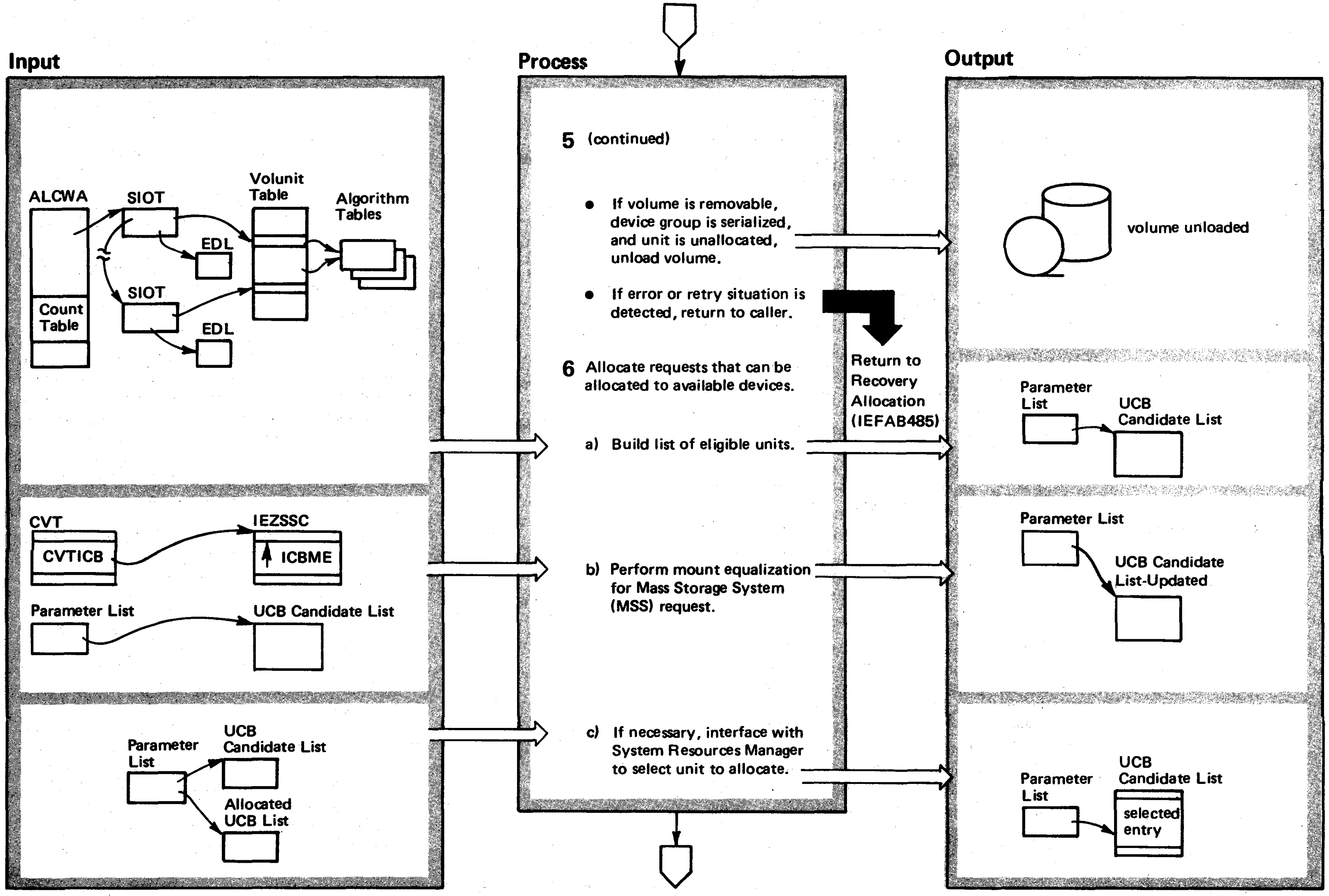
5 Validity check unallocated requests that require a specific volume:

- If volume is permanently resident or reserved and unit is eligible to the request, allocate request.

Diagram 14-13. IEFAB486 – Offline/Allocated Device Allocation (Part 4 of 12)

Extended Description	Module	Segment
4 In order to find sufficient units to allocate a request, the algorithm (called in step 2 and step 3b) might have had to rearrange requests already allocated – that is, indicate that an allocated request must be assigned to a different device group in order to free units needed to satisfy an unallocated request. Requests to be rearranged are indicated in the request list entry of the algorithm tables – the number of units already allocated (CVRGRPAL) is greater than the number of units the algorithm selected for allocation (CVRGALL). The purpose of this step is to unallocate requests that must be rearranged.		
IEFAB477 (Unallocate Requests to be Rearranged) updates the TIOT, volunit table, SIOT and the UCB for each request to be unallocated; Common Unallocation Control releases any space obtained for the requests.	IEFAB477 IEFAB4A0	
5 The purpose of this step is to validity check unallocated requests: to determine if a volume needed by an unallocated request is mounted, and if the volume is mounted, whether the request can be allocated. Offline/Allocated Device Allocation scans the volunit table to find unallocated requests that require a specific volume; IEFAB441 (Volume Validity Checker) performs the validity check for each needed volume.	IEFAB486 IEFAB441	ENDVALID
IEFAB441 scans the EDT group entries to determine if the volume is mounted. If the volume is not mounted, the validity check is unnecessary and processing continues with step 6. Otherwise, IEFAB441 performs the following checks:		
a) The unit containing the volume must not be in use by a system task (UCBNALOC=0). If it is, this allocation is failed.		
b) The device type of the unit containing the volume is compatible with the requested device type. If not, this allocation is failed.	IEFAB441	CHEKTYPE
c) If the unit containing the volume is in a serialized device group and the volume is permanently resident or reserved, the UCB must be included in the EDL for this request – that is, the unit is eligible to this request. If not, this allocation is failed. If the unit is eligible, the following processing is performed:	IEFAB441 IEFAB441	 SERCHEDL
• If another request(s) indicates unit affinity to this request, IEFAB442 cancels the unit affinity by increasing the number of units required. (Unit affinity can be either implied or explicit – see "Selected Terms Used in Allocation/Unallocation" in the "Introduction to Allocation/Unallocation.") If, as a result of increasing the unit requirements, a SIOT would require more than 59 units, the allocation is failed. Otherwise, the unit requirements are increased and IEFAB4F2 updates the algorithm tables to reflect the changed unit requirements.	IEFAB442 IEFAB442 IEFAB4F2	 INCRUNIT
• IEFAB434 (Allocate Request to Unit) allocates the request. For details on IEFAB434, see the M.O. diagram Allocate Request to Unit (IEFAB434).	IEFAB434	
• IEFAB441 (Volume Validity Checker) marks the volunit entry as allocated and decreases the appropriate counts in the count table. If the SIOT is now completely allocated, the SIOT is also marked allocated and the TOTREQS field in the count table is decreased.	IEFAB441	DOARURTN
d) If the volume is not permanently resident or reserved, IEFAB441 (Volume Validity Checker) determines if the device group containing the unit has been serialized:	IEFAB441	
• For the direct access device class, retry is necessary if the device group on which the volume is mounted is not serialized. For the tape device class, the group on which the volume is mounted may be serialized, but a retry is still necessary if the devices are compatible, but the device on which the volume is mounted belongs to a generic device type other than the first device type in the eligible devices list (EDL). Offline/Allocated Device Allocation returns to Recovery Allocation; the retry situation will be handled by Common Allocation Cleanup (see the M.O. diagram Common Allocation Cleanup (IEFAB490).)		
• If the device group is serialized, but the unit allocated to another user, IEFAB441 (Volume Validity Checker) indicates that this allocation must wait for the unit. (The wait condition is processed in step 9 of this diagram.)		
• If the device group is serialized and the unit is not allocated, IEFAB49C unloads the volume.	IEFAB49C	

Diagram 14-13. IEFAB486 – Offline/Allocated Device Allocation (Part 5 of 12)



5 (continued)

- If volume is removable, device group is serialized, and unit is unallocated, unload volume.
- If error or retry situation is detected, return to caller.

6 Allocate requests that can be allocated to available devices.

a) Build list of eligible units.

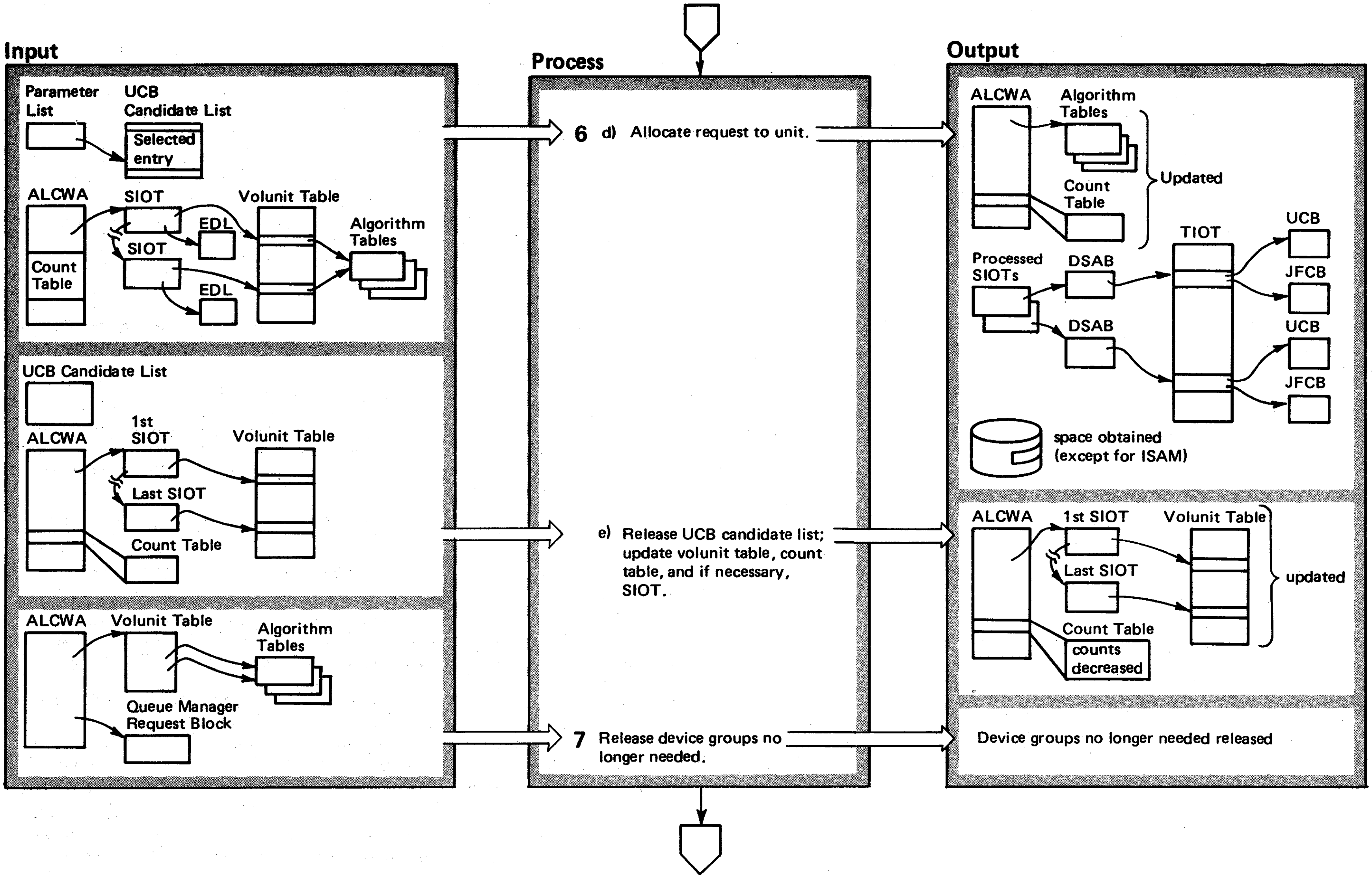
b) Perform mount equalization for Mass Storage System (MSS) request.

c) If necessary, interface with System Resources Manager to select unit to allocate.

Diagram 14-13. IEFAB486 – Offline/Allocated Device Allocation (Part 6 of 12)

Extended Description	Module	Segment
6 IEFAB478 (Allocate from Groups the Algorithm Picked) allocates requests that can be allocated at this time to available devices. IEFAB478 scans the SIOT chain for unallocated SIOTs; for each unallocated volunit entry, the following steps are performed:	IEFAB478	
a) IEFAB478 builds a list of UCBs (UCB candidate list) that are included in a device group selected by the algorithm and that meet the following conditions: ● The UCB is online and unallocated. ● The UCB is not in use by a system task. ● The volume on the unit is not permanently resident or reserved. The EDL for the SIOT contains the addresses of UCBs in the device groups eligible to the request. If the UCB candidate list contains no entries – that is, no eligible units meet the preceding conditions – IEFAB478 locates the next unallocated volunit entry to be processed.	IEFAB478	GENAPREP
b) If the request is for a Mass Storage System (MSS) device, IEFAB478 interfaces with ICBME, which will return a preferred list of UCBs. This list will then be used to merge into and update the UCB candidate list. (See *OS/VS2 Mass Storage System Communicator (MSSC) Logic* for information on module ICBME.)	IEFAB478 ICBME	MONTEQAL
c) If the UCB candidate list contains more than one entry, IEFAB478 interfaces with the System Resources Manager to select the device to be allocated. IEFAB440 builds a list of allocated UCBs. The System Resources Manager uses this list and the UCB candidate list to determine which unit should be allocated.	IEFAB478 IEFAB440	GALGALOC

Diagram 14-13. IEFAB486 – Offline/Allocated Device Allocation (Part 7 of 12)



6 d) Allocate request to unit.

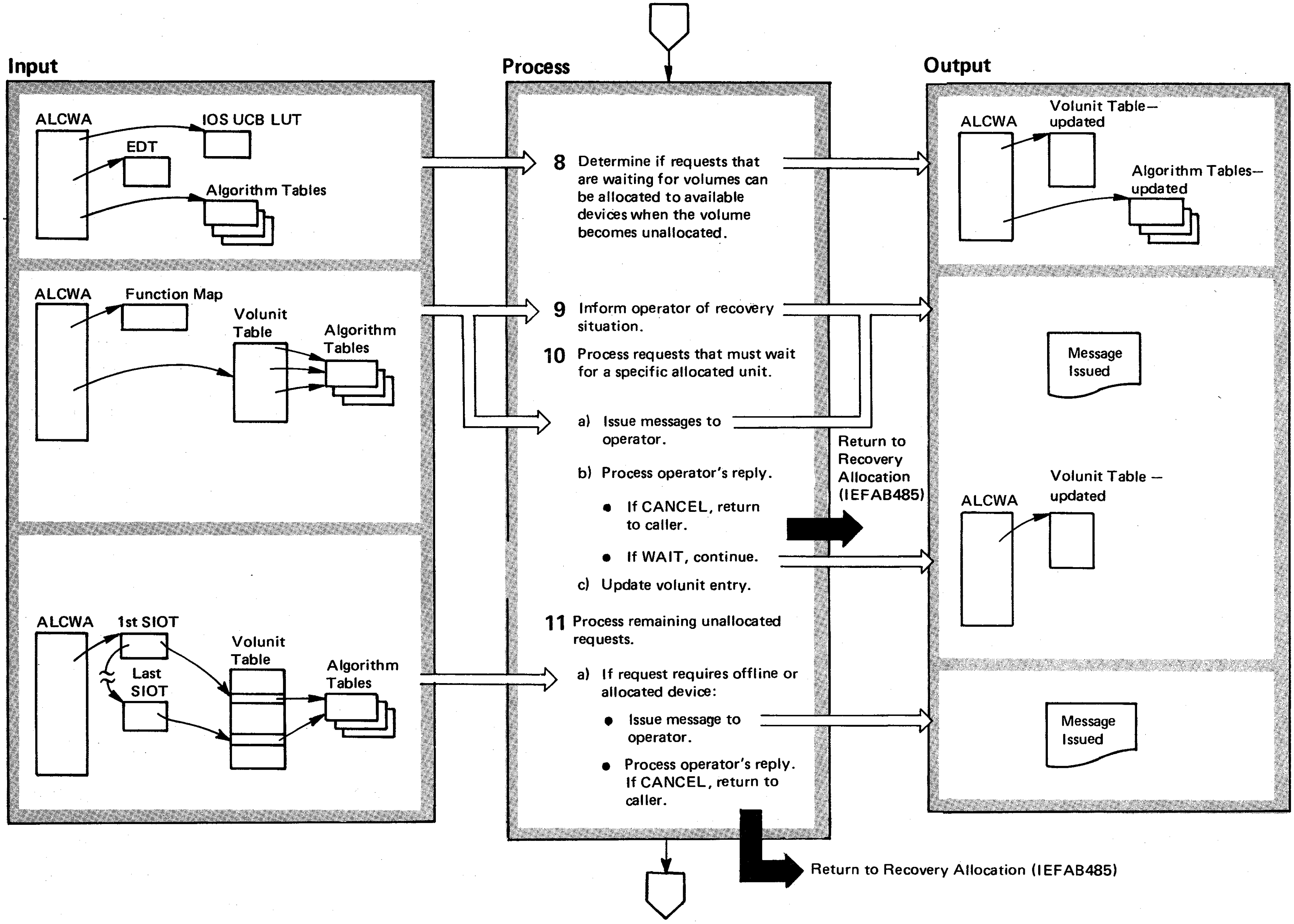
e) Release UCB candidate list; update volunit table, count table, and if necessary, SIOT.

7 Release device groups no longer needed.

Diagram 14-13. IEFAB486 – Offline/Allocated Device Allocation (Part 8 of 12)

Extended Description	**Module**	**Segment**
6 (Continued)		
d) IEFAB434 (Allocate Request to Unit) allocates the unit. For details, see the M.O. diagram, Allocate Request to Unit (IEFAB434).	IEFAB434	
e) IEFAB478 releases the UCB candidate list, marks the volunit entry as allocated, and decreases the TOTVOLUN field in the count table. If the SIOT is completely allocated, IEFAB478 also marks the SIOT as allocated and decreases the TOTREQS field in the count table.	IEFAB478 IEFAB478 IEFAB478	GENAPREP GARURTN
7 Offline/Allocated Device Allocation determines which device groups are no longer needed and can be released. Device groups that must remain serialized include those that: ● The algorithm indicates might be used to satisfy an unallocated request. ● Include a specific offline or allocated unit that is needed by this allocation. ● Include an allocated volume that is needed by this allocation (for example, the volume must be moved to another unit when it is unallocated). ● Include a direct access unit that requires a scratch volume to be mounted.	IEFAB486	FREGROUP
The Allocation Queue Manager is called to release the device groups that are no longer needed.	IEFAB4FA	

Diagram 14-13. IEFAB486 – Offline/Allocated Device Allocation (Part 9 of 12)



8 Determine if requests that are waiting for volumes can be allocated to available devices when the volume becomes unallocated.

9 Inform operator of recovery situation.

10 Process requests that must wait for a specific allocated unit.

a) Issue messages to operator.

b) Process operator's reply.

- If CANCEL, return to caller.
- If WAIT, continue.

c) Update volunit entry.

11 Process remaining unallocated requests.

a) If request requires offline or allocated device:

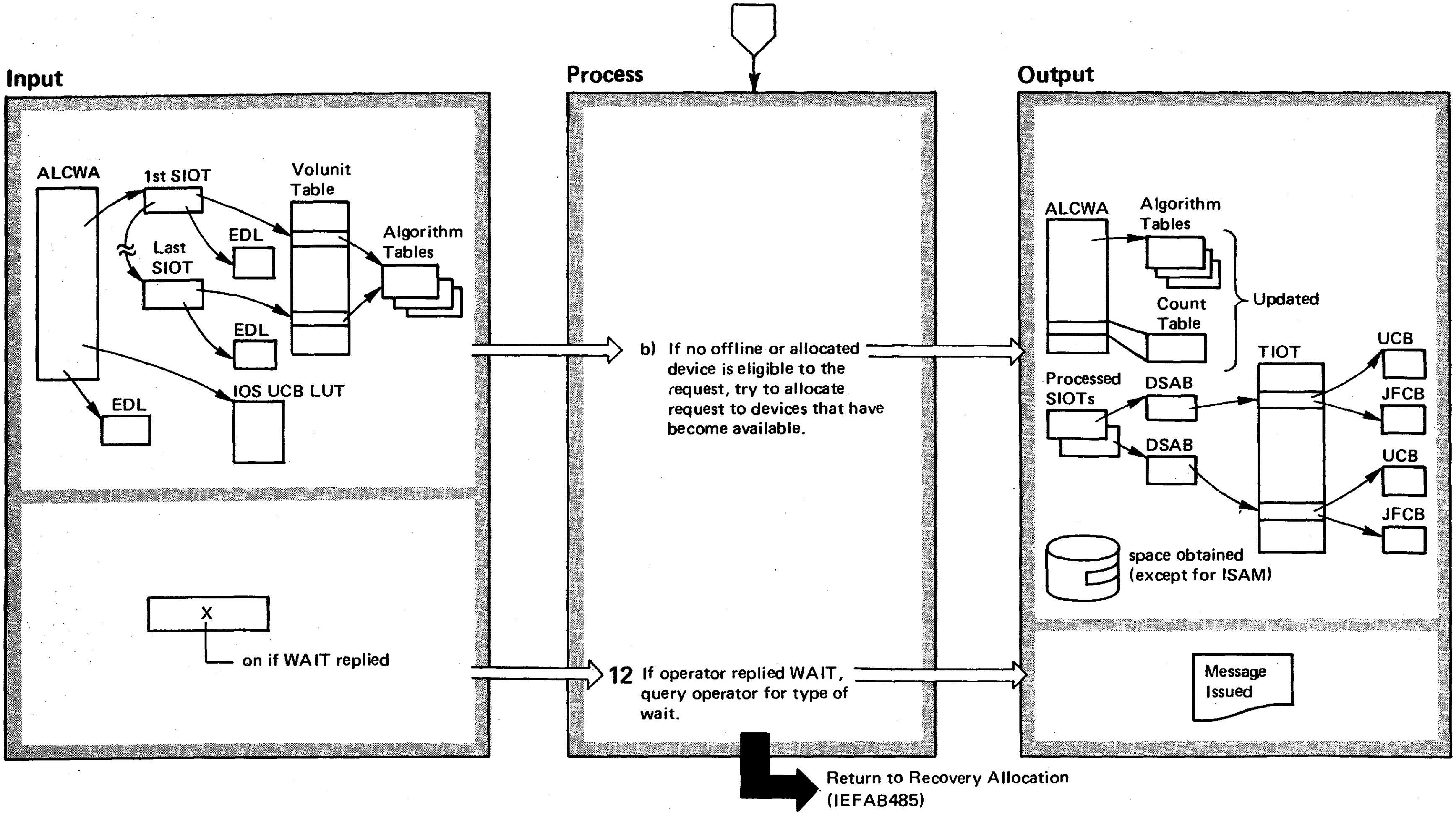
- Issue message to operator.
- Process operator's reply. If CANCEL, return to caller.

Diagram 14-13. IEFAB486 – Offline/Allocated Device Allocation (Part 10 of 12)

Extended Description	Module	Segment
8 This step determines if requests that must wait for a needed volume(s) to be unallocated (so the volume can be moved to an eligible device) can be allocated to an available unit, once the volume is unallocated. IEFAB486 searches: • the units in each serialized device group (by means of the EDT and IOS UCB LUT) to locate available units; and, • the algorithm tables to determine if an available unit can be used to allocate a request waiting for a volume. For each request that can be allocated to an available unit, IEFAB486 updates the algorithm tables to bind the request to the device group containing the available unit, and updates the volunit table to indicate that the request can be allocated to an available unit (VURCVYPR=1).	IEFAB486	SETHANDL
The Cover/Reduce Algorithm is called to limit its final selection of device groups from which to allocate.	IEFAB480	
9 IEFAB487 (Allocation Recovery Interface with Operator) issues message IEF244I, informing the operator of the number of units needed to complete this allocation. If there are requests that require either an offline or an allocated unit (but are not eligible to both), a second line of the message is issued, indicating the minimum number of offline and/or allocated devices that are needed. This message is for information only and does not require an operator response. To determine the number of units needed, IEFAB487 searches the volunit table (for demand requests) and the algorithm tables (for non-demand requests) for each unallocated volunit entry.	IEFAB487	RHDRMSGS
10 IEFAB487 (Allocation Recovery Interface with Operator) first processes demand requests if the needed unit is allocated, and requests for a specific volume that is mounted on an ineligible allocated unit. IEFAB487 searches volunit entries of each unallocated SIOT to locate these requests.	IEFAB487	SPECWAIT
a) IEFAB487 issues message IEF488I for each request, informing the operator and appropriate device pools of the allocated unit or the volume on an allocated unit that is required. No operator response is required to this message. IEFAB488 (Allocation Recovery Reply Options Processor) then issues message IEF238D, requesting a reply from the operator. In this case (specific waits), the only valid replies are CANCEL or WAIT.	IEFAB487 IEFAB488	SPECWAIT BLDRMSG
b) IEFAB488 processes the operator's reply: • If the reply is invalid (it is not CANCEL or WAIT), IEFAB488 issues an invalid reply message (IEF490I) and then reissues message IEF238D. • If the reply is CANCEL, IEFAB488 returns to the caller; this allocation will be failed. • If the reply is WAIT, IEFAB488 sets an indicator to note a reply of WAIT was processed; this allocation continues to process unallocated requests.	IEFAB488	
c) IEFAB487 updates the volunit entry of each request requiring a specific allocated unit to indicate that no further recovery processing should be done (VURCVYPR=1).	IEFAB487	SPECWAIT
11 IEFAB48A (Process Offlines/Allocateds) searches the volunit entries of each unallocated SIOT to locate requests that have not yet been processed (VURCVYPR=0). Remaining requests fall into two categories: a) an offline or allocated device is needed to satisfy the request; b) the request is not allocated but is not eligible to any offline or allocated devices. (This situation can occur if a needed device was brought online before this step.)	IEFAB48A	
a) If an offline or allocated device is needed for the request, IEFAB48A issues the following messages to the operator (and to appropriate device pools):	IEFAB48A	
• Message IEF489I informs the operator of the number of units that must be made available before the request can be allocated.	IEFAB48A	BLDDDHDR
• If offline devices are needed, message IEF247I lists the devices that could be allocated to the request, if the status of the devices were changed. The message indicates which devices are offline or not accessible (for example, the channel is offline).	IEFAB48A	BLDOLMSG

Step 11 continued on Part 11

Diagram 14-13. IEFAB486 – Offline/Allocated Device Allocation (Part 11 of 12)



b) If no offline or allocated device is eligible to the request, try to allocate request to devices that have become available.

12 If operator replied WAIT, query operator for type of wait.

Extended Description	Module	Label
11 a) (Continued) IEFAB488 issues message IEF238D to the operator, listing the possible replies, and then processes the operator's reply:	IEFAB488	BLDRMSG
• If the reply is invalid, IEFAB488 issues message IEF490I, indicating an invalid reply, and then reissues message IEF238D.		
• If the reply is CANCEL, this allocation is failed; IEFAB488 returns to its caller.		
• If the reply is WAIT, the Cover/Reduce Algorithm is called to select the device group to wait for and to update the algorithm tables; an indicator is set to note that a WAIT reply was processed. This allocation continues to process unallocated requests.	IEFAB488	PRCWAIT

Step 11 continued on Part 12

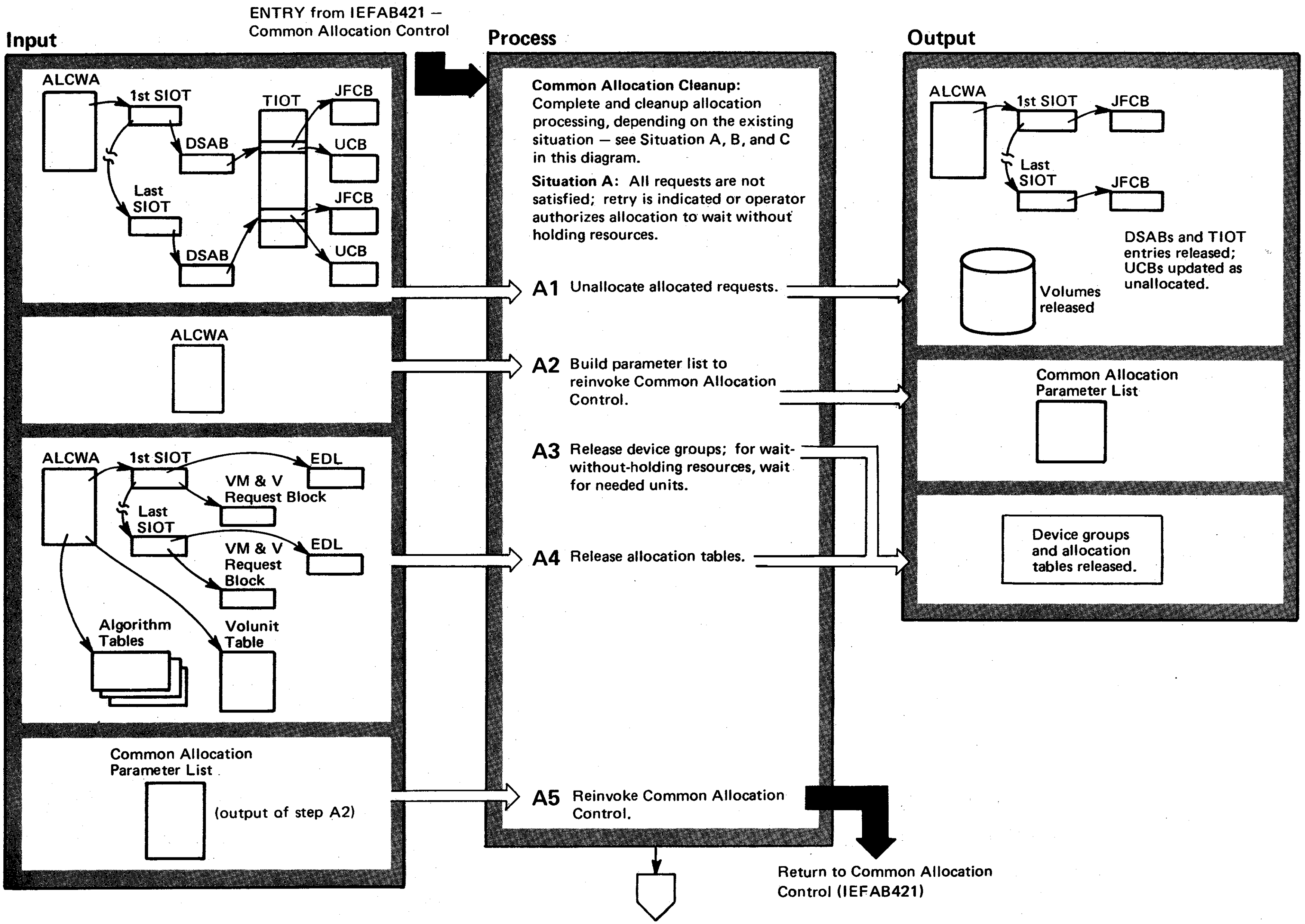
Diagram 14-13. IEFAB486 – Offline/Allocated Device Allocation (Part 12 of 12)

Extended Description	Module	Segment
11 a) (Continued)		
• If the reply is a valid device name, IEFAB488 checks to ensure that the device is useable and is needed by an unallocated request. If not, IEFAB488 issues message IEF490I, indicating an invalid reply, and then reissues message IEF238D. If the device is useable and needed, IEEVDEV determines if the device can be brought online and IEFAB4F8 brings the device online (if the operator did not). IEFAB489 allocates the device – IEFAB489 searches the rest of the SIOT chain to try to allocate any remaining requests to available devices. For details on the processing of IEFAB489, see step 11b.	IEFAB488 IEFAB488 IEEVDEV IEFAB4F8 IEFAB489	PRCDEV INVLRPLY
b) If no offline or allocated device is eligible to the request, an eligible offline unit must have been brought online by the operator. IEFAB489 (Recovery Allocation of Online Devices) tries to allocate this request and any other unallocated requests to online, unallocated units. (IEFAB489 searches the SIOT chain to allocate as many requests as possible to devices that have become available.)	IEFAB489	
IEFAB489 first attempts to allocate demand requests that needed an offline unit. For each demand request, IEFAB489 determines if the needed unit is now online. If it is, IEFAB434 (Allocate Request to Unit) allocates the request. (For details on IEFAB434, see the M.O. diagram Allocate Request to Unit (IEFAB434).)	IEFAB489 IEFAB434	CHKDMNDS
IEFAB489 then determines, for each device group in the system, if the number of offline units in the group has decreased. For each device group, IEFAB489 builds a count of offline units by examining the group entry in the EDT. If the count of offline devices is less than the number of offline devices in the algorithm tables (GRPOFFLN), IEFAB489 updates the algorithm tables and attempts to allocate non-demand requests eligible to that device group:	IEFAB489	
• IEFAB489 determines if any of the unallocated volunit entries are eligible to the device group being processed (that is, the device group whose count of offline units has decreased). For each eligible request, IEFAB489 builds a UCB candidate list containing the units that are eligible, available, and not needed by a demand request. (Before allocating from the UCB candidate list, IEFAB489 ensures that it will not be allocating an available unit intended for a request that is waiting for a volume – see step 8.)	IEFAB489 IEFAB489 IEFAB489	CHKNDMND ALOCREQ CHKCANDS
• If the request is for a Mass Storage System (MSS) device, IEFAB489 interfaces with ICBME, which will return a preferred list of UCBs. This list will then be used to merge into and update the UCB candiate list. (See *OS/VS2 Mass Storage System Communicator (MSSC) Logic* for information on module ICBME.)	IEFAB489 ICBME	MONTEQAL
• If the UCB candidate list contains more than one entry IEFAB489 interfaces with the System Resources Manager. IEFAB440 builds an allocated UCB list. The System Resources Manager uses this list and the UCB candidate list to determine which unit should be allocated.	IEFAB489 IEFAB440	ALOCENT
• IEFAB434 (Allocate Request to Unit) allocates the request. For details, see the M.O. diagram Allocate Request to Unit (IEFAB434).	IEFAB434	
• The Cover/Reduce Algorithm updates the algorithm tables.	IEFAB480	
12 If the reply to any of the requests processed in steps 10 or 11 was WAIT, IEFAB487 issues message IEF433D, asking the operator if this allocation should wait with or without holding resources. ALCWA is updated to indicate the type of wait (INDWAIT=1 or INDREQV=1) and IEFAB487 returns to its caller.	IEFAB487	GWAITYPE
If the reply is NOHOLD (indicating wait *without* holding resources), Common Allocation Cleanup will unallocate requests that have been allocated and reattempt the allocation – see the M.O. diagram Common Allocation Cleanup (IEFAB490). If the reply is HOLD, IEFAB491 (Wait Holding Resources) waits until the needed devices are unallocated and then allocates the requests – see the M.O. diagram Common Allocation Control (IEFAB421).	IEFAB490 IEFAB491	

Error Processing

An error in any routine causes control to be returned to the calling routine. In the event of an abnormal termination, the ESTAE exit routine established by IEFAB421 performs any necessary cleanup.

Diagram 14-14. IEFAB490 – Common Allocation Cleanup (Part 1 of 8)



Common Allocation Cleanup: Complete and cleanup allocation processing, depending on the existing situation – see Situation A, B, and C in this diagram.

Situation A: All requests are not satisfied; retry is indicated or operator authorizes allocation to wait without holding resources.

A1 Unallocate allocated requests.

A2 Build parameter list to reinvoke Common Allocation Control.

A3 Release device groups; for wait-without-holding resources, wait for needed units.

A4 Release allocation tables.

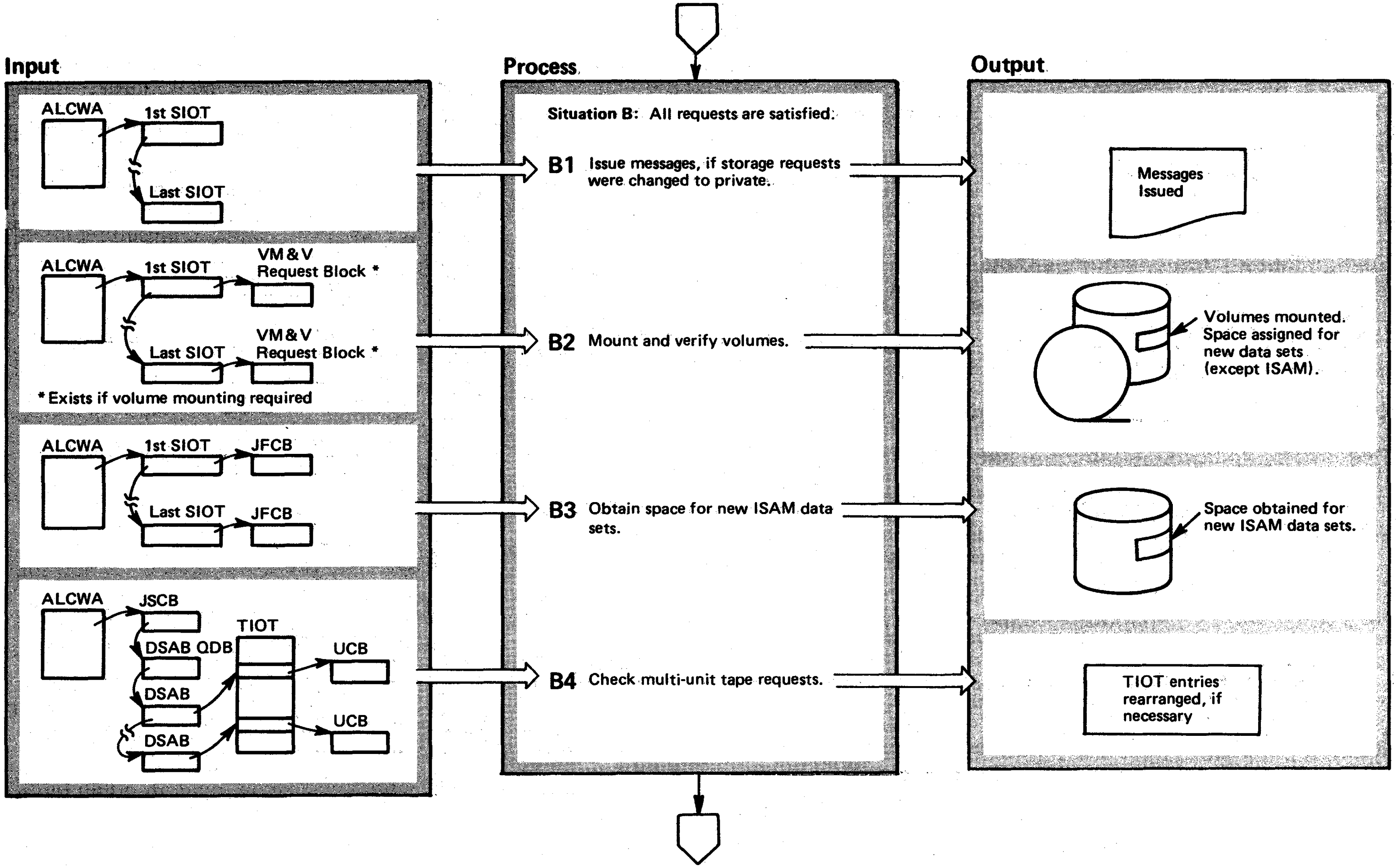
A5 Reinvoke Common Allocation Control.

Diagram 14-14. IEFAB490 – Common Allocation Cleanup (Part 2 of 8)

Extended Description	Module	Segment
Note: The processing performed by Common Allocation Cleanup depends on the situation that exists when Common Allocation Cleanup is called by Common Allocation Control. One of three distinct situations exists:		
A All requests still are not satisfied *and* either retry is indicated or the operator authorized allocation to wait without holding resources. See Situation A (steps A1-A5).		
B All requests are satisfied. See Situation B (steps B1-B8).		
C A terminating error occurred during allocation or the operator cancelled the allocation. See Situation C (steps C1-C2).		
SITUATION A If not all requests have been satisfied, the allocation will be reattempted in two cases:		
• Retry is indicated in ALCWA (INDRETRY=1). Retry situations are detected within Generic Allocation Control and within Recovery Allocation: a needed volume is mounted on a unit 1) included in a device group that has not been serialized, or 2) in the case of tapes, on a unit not of the generic device type being considered. The allocation will be reattempted and all needed device groups will be serialized. For a further explanation of retry, see "The Retry Situation" in the "Introduction to Allocation/Unallocation."		
• A request(s) could not be satisfied because a required unit or volume is allocated to another user. If the operator authorizes, Common Allocation Cleanup will wait without holding resources until the needed unit or volume is unallocated. At that time, the allocation will be reattempted. (The operator can still cancel the job while allocation is waiting for the required resources to become available.)		

Extended Description	Module	Segment
A1 Common Unallocation Control is called by Common Allocation Cleanup to unallocate all requests already satisfied except for: (1) dummy, subsystem, and VIO requests; and (2) requests that were completely allocated to permanently resident or reserved direct access volumes. For details on Common Unallocation Control, see the M.O. diagram "IEFAB4A0 – Common Unallocation Control."	IEFAB4A0	
A2 Common Allocation Cleanup builds the Common Allocation Parameter List, which will be the input to Common Allocation Control when it is reinvoked. For retry, the function map indicates that *all* device groups in a generic must be serialized; during allocation processing, the pertinent generic is determined by checking the generic device types to which a SIOT marked for retry (SIOTRTRY=1) is eligible.	IEFAB490	FINISALC
A3 The Allocation Queue Manager releases any device groups still serialized by this allocation. If this allocation is to wait for a needed unit(s) to become available, the Allocation Queue Manager is informed of the device groups from which an allocated unit is needed. Processing continues when the needed units are unallocated.	IEFAB4FA	
A4 Common Allocation Cleanup issues FREEMAIN macro instructions to release the volunit table, the EDLs, any volume mount and verify request blocks, the algorithm tables, and ALCWA.	IEFAB490	FREEGETS
A5 Common Allocation Cleanup reinvokes Common Allocation Control. (See the M.O. diagram "IEFAB421 – Common Allocation Control.")	IEFAB490	

Diagram 14-14. IEFAB490 – Common Allocation Cleanup (Part 3 of 8)



Situation B: All requests are satisfied.

B1 Issue messages, if storage requests were changed to private.

B2 Mount and verify volumes.

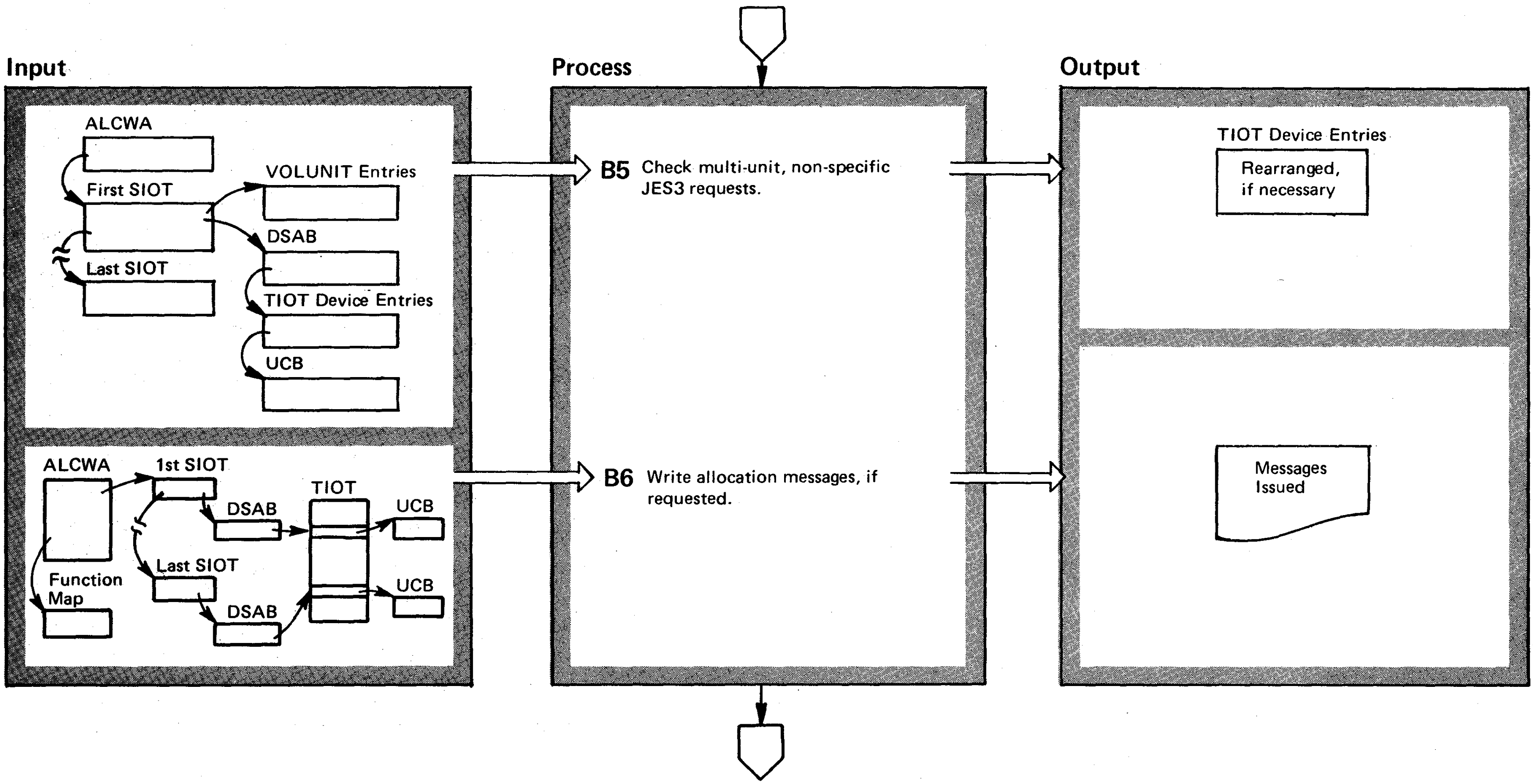
B3 Obtain space for new ISAM data sets.

B4 Check multi-unit tape requests.

Diagram 14-14. IEFAB490 – Common Allocation Cleanup (Part 4 of 8)

Extended Description	Module	Segment
SITUATION B Steps B1-B8 are performed if all requests have been satisfied.		
B1 Common Allocation Cleanup searches the SIOT chain for requests that were changed from storage to private (SPVTAMSG=1 in the SIOT). (If insufficient storage volumes were available to satisfy storage requests, Nonspecific Volume Allocation Control changed the requests to private.)	IEFAB490	ASUMPVTS
The System Message Interface Routine issues a "private-assumed" message for each such request.	IEFAB4FD	
B2 The Allocation/Volume Mount & Verify (VM&V) Interface (IEFAB492) receives control to mount and verify needed volumes. During allocation, a VM&V request block was built for every request that required a volume to be mounted. For details, see the M.O. diagram Allocation/Volume Mount & Verify Interface (IEFAB492).	IEFAB492	
B3 Different areas of ISAM data sets (index, prime, overflow) can be defined on separate DD statements. Because allocation does not necessarily allocate requests in the order they were coded, space for ISAM data sets could not be obtained as each SIOT (representing a single DD statement) was allocated. Now that all requests are satisfied, Common Allocation Cleanup can obtain space for ISAM data sets.		
An ISAM request is indicated in the JFCB, pointed to by the request's SIOT. Common Allocation Cleanup checks for the following error conditions in ISAM requests: • Four or more ISAM data sets are concatenated. • ISAM and non-ISAM data sets are concatenated. • New and old ISAM data sets are concatenated. • Data sets with and without automatic data set protection are concatenated. If any of these errors is detected, this allocation is failed – the cleanup processing described in Situation C is performed.	IEFAB490	CHEKISAM
If no error is found, Common Allocation Cleanup interfaces with DADSM for space for new ISAM data sets.	IEFAB490	ISAMSPAC
B4 This step is performed for multi-unit tape data sets that have been allocated to both single-density and dual-density devices. Common Allocation Cleanup switches device entries in the TIOT, if necessary, to ensure that the first device entry in the TIOT entry for this data set is a single-density device. (This is necessary to ensure that a useable density will be the default, in the event that no density was coded.)	IEFAB490	DUALTIOT

Diagram 14-14. IEFAB490 – Common Allocation Cleanup (Part 5 of 8)



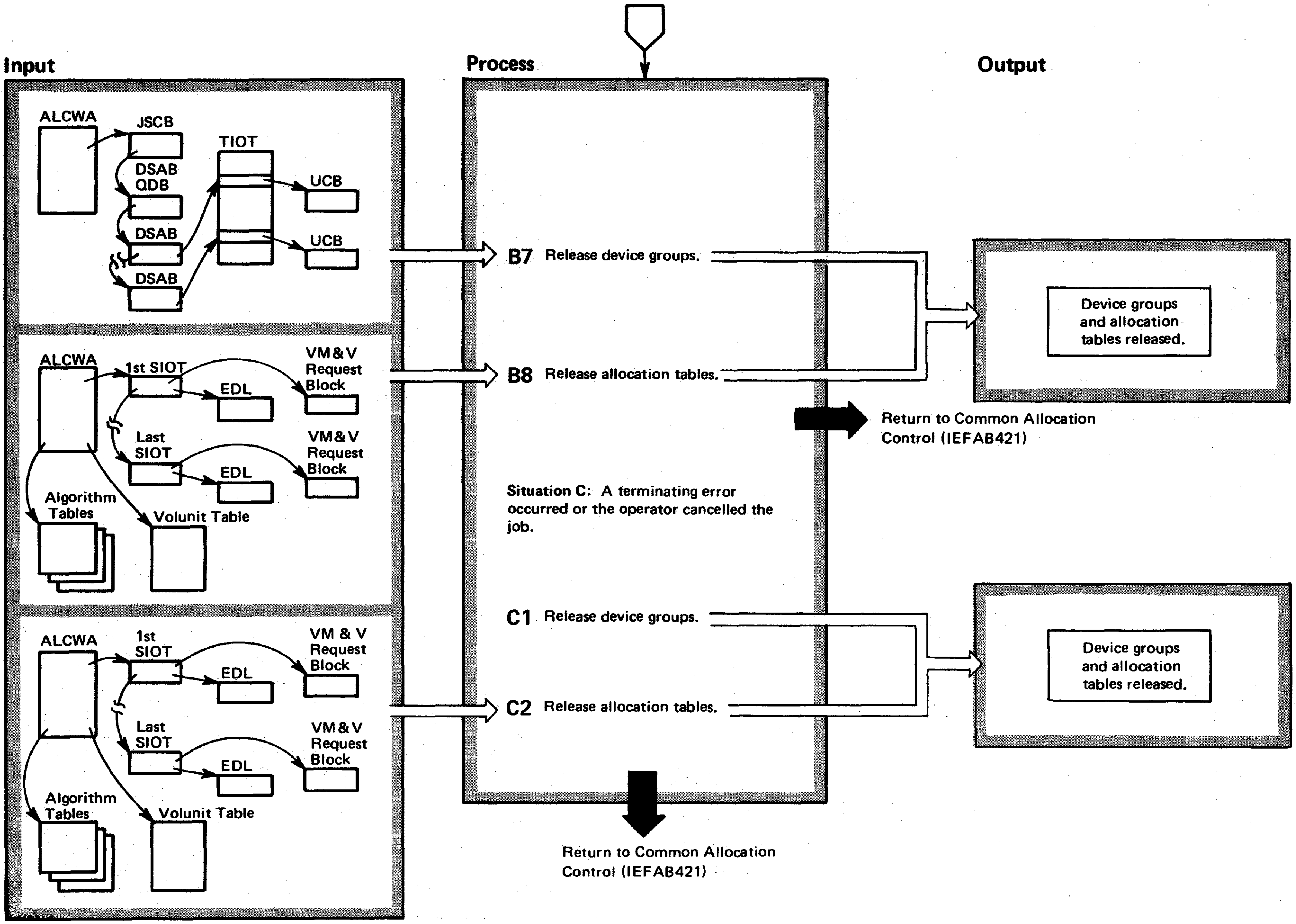
B5 Check multi-unit, non-specific JES3 requests.

B6 Write allocation messages, if requested.

Diagram 14-14. IEFAB490 – Common Allocation Cleanup (Part 6 of 8)

Extended Description	Module	Segment
B5 This step is performed if JES3 selected devices for a request that required more than one nonspecific volume. Common Allocation Cleanup switches device entries in the TIOT, if necessary, to ensure that the order of the TIOT device entries corresponds to the order in which JES3 selected the devices.	IEFAB490	MOVETIOT
B6 The purpose of this step is to build and issue allocation messages if they were requested (as indicated in the common allocation function map – see figure 17).		
The Allocation Message Routine (IEFAB4EE) builds the allocation messages by scanning the SIOT chain and locating the device entries in the TIOT for the units allocated to each SIOT. The System Message Interface Routine issues the messages.	IEFAB4EE IEFAB4FD	

Diagram 14-14. IEFAB490 – Common Allocation Cleanup (Part 7 of 8)



B7 Release device groups.

B8 Release allocation tables.

Return to Common Allocation Control (IEFAB421)

Situation C: A terminating error occurred or the operator cancelled the job.

C1 Release device groups.

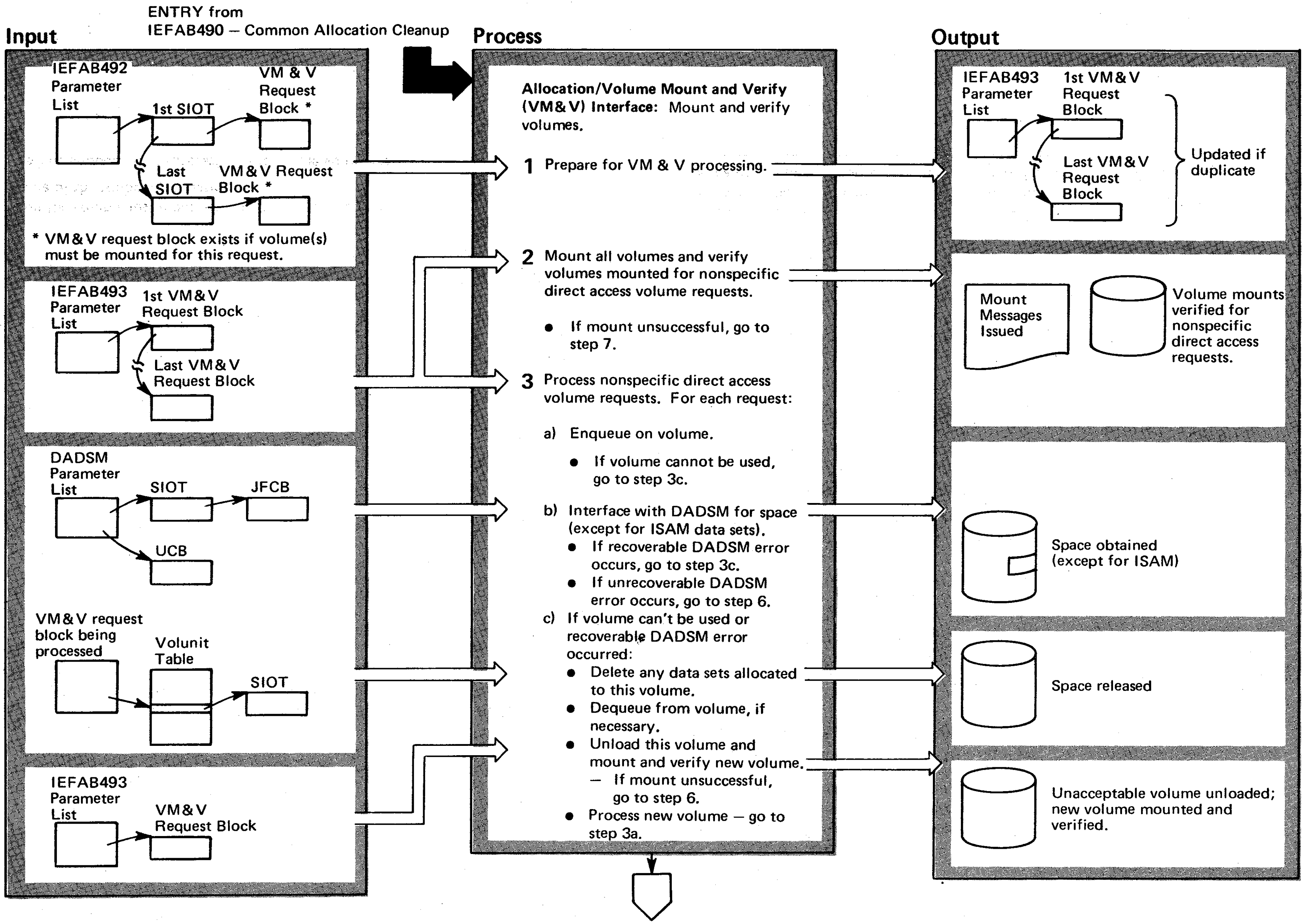
C2 Release allocation tables.

Return to Common Allocation Control (IEFAB421)

Diagram 14-14. IEFAB490 – Common Allocation Cleanup (Part 8 of 8)

Extended Description	Module	Segment
B7 The Allocation Queue Manager (IEFAB4FA) releases any device groups still serialized by this allocation. (Device groups that contained units on which nonspecific volumes were to be mounted have not yet been released.)	IEFAB4FA	
B8 Common Allocation Cleanup issues FREEMAIN macro instructions to release the volunit table, the EDLs, any volume mount & verify request blocks, the algorithm tables, and ALCWA.	IEFAB490	FREEGETS
SITUATION C Steps C1 and C2 are performed if the operator cancelled the allocation or if an error occurred during allocation processing.		
C1 The Allocation Queue Manager (IEFAB4FA) releases any device groups still serialized by this allocation.	IEFAB4FA	
C2 Common Allocation Cleanup (IEFAB490) issues FREEMAIN macro instructions to release the volunit table, the EDLs, any volume mount & verify request blocks, the algorithm tables, and ALCWA.	IEFAB490	FREEGETS
Error Processing		
Any errors during cleanup processing cause the processing described under Situation C to be performed.		
In the event of an abnormal termination, the ESTAE exit routine established by IEFAB421 performs any necessary cleanup.		

Diagram 14-15. IEFAB492 – Allocation/Volume Mount and Verify (VM&V) Interface (Part 1 of 4)



Allocation/Volume Mount and Verify (VM&V) Interface: Mount and verify volumes.

1 Prepare for VM & V processing.

2 Mount all volumes and verify volumes mounted for nonspecific direct access volume requests.

- If mount unsuccessful, go to step 7.

3 Process nonspecific direct access volume requests. For each request:

a) Enqueue on volume.
  - If volume cannot be used, go to step 3c.

b) Interface with DADSM for space (except for ISAM data sets).
  - If recoverable DADSM error occurs, go to step 3c.
  - If unrecoverable DADSM error occurs, go to step 6.

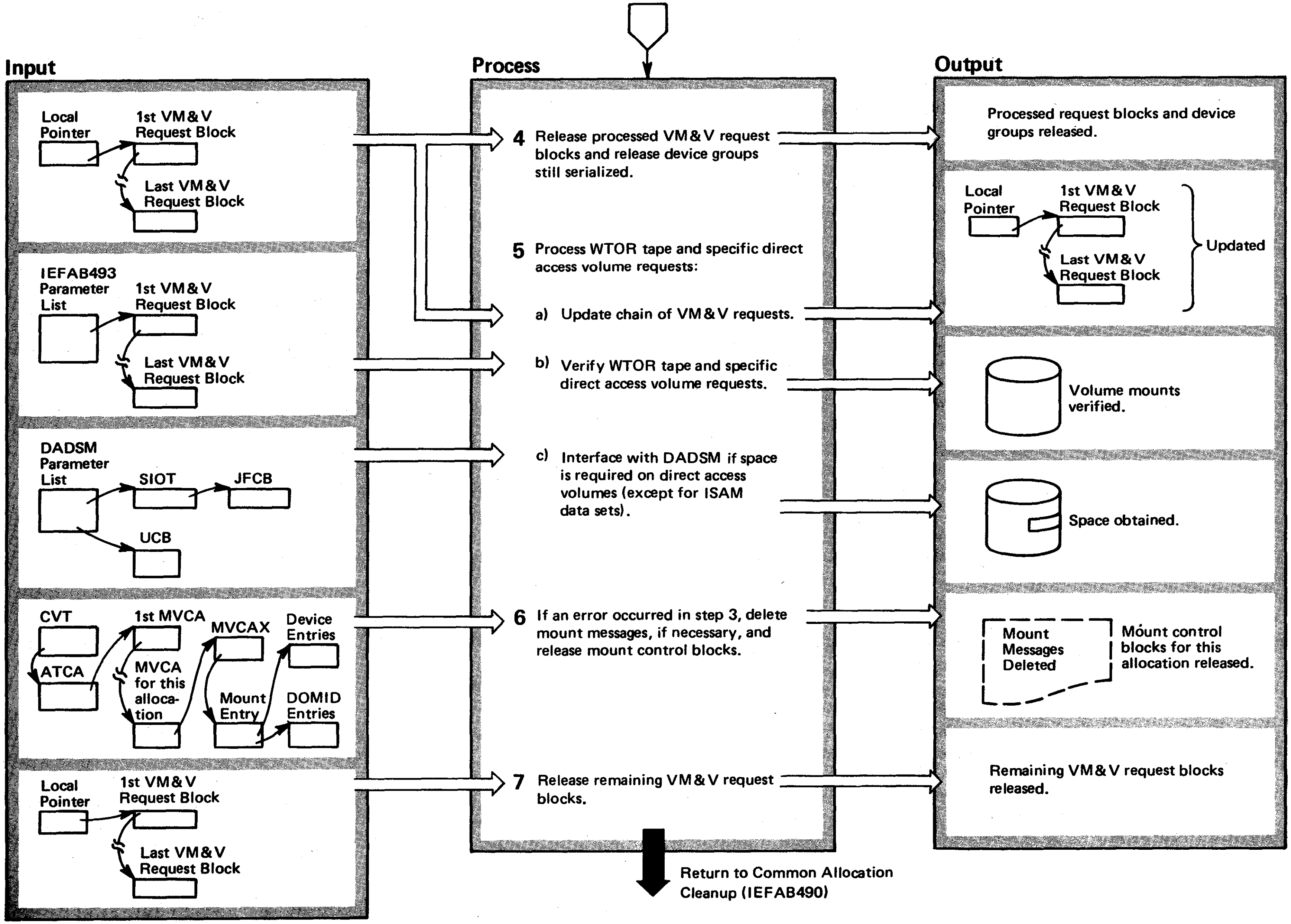
c) If volume can't be used or recoverable DADSM error occurred:
  - Delete any data sets allocated to this volume.
  - Dequeue from volume, if necessary.
  - Unload this volume and mount and verify new volume.
    - – If mount unsuccessful, go to step 6.
  - Process new volume – go to step 3a.

Diagram 14-15. IEFAB492 – Allocation/Volume Mount and Verify (VM&V) Interface (Part 2 of 4)

Extended Description	Module	Label
ENTRY The Allocation/Volume Mount & Verify (VM&V) Interface (IEFAB492) is called by Common Allocation Cleanup (IEFAB490) to process requests that require volume mounting. IEFAB492 prepares for VM&V processing, calls VM&V Control to mount and verify volumes, and interfaces with DADSM for space for new data sets on direct access devices.		
1 To prepare for VM&V processing, the Allocation/VM&V Interface:	IEFAB492	
• Searches the SIOT chain for VM&V request blocks, chains them together, and places a pointer to the first request block in a parameter list.	IEFAB492	VMVCHAIN
• If more than one request block needs the same volume, marks as duplicates all but the first request block for that volume.	IEFAB492	VMVDUPCK
2 VM&V Control is called to issue all mount messages and to verify volumes mounted for nonspecific direct access volume requests. For details on VM&V Control, see the M.O. diagram VM&V Control (IEFAB493).	IEFAB493	
If the return code from VM&V Control indicates that the operator cancelled the job (if the allocation is batch), cancelled this request (if the allocation is dynamic), or the mount failed for a Mass Storage System (MSS) volume, control is passed to step 7. This allocation will be failed.	IEFAB492	
3 The Allocation/VM&V Interface completes the processing of all nonspecific direct access volume requests. For each request:	IEFAB492 IEFAB492	NSPDACTL NSPDAVOL
a) IEFAB4F0 (Conditional ENQ/DEQ Routine) enqueues on the volume just mounted. The enqueue results in one of the following situations:	IEFAB4F0	
• The enqueue is unsuccessful because the volume is already owned by this job. The volume can be used if the enqueue is share and no unallocated specific volume requests need this volume.		
• The enqueue is unsuccessful because another user owns this volume and this allocation cannot share the volume; the volume cannot be used.		
• The enqueue is successful; the volume can be used.		
If the volume cannot be used, processing continues with step 3c.		

Extended Description	Module	Segment
b) If the volume can be used and the data set is not an ISAM data set, the Allocation/VM&V Interface interfaces with DADSM for space. (Space for new ISAM data sets is obtained by Common Allocation Cleanup – see the M.O. diagram Common Allocation Cleanup (IEFAB490). One of the following conditions results:	IEFAB492	DADSMINT
• The space is successfully obtained; the next nonspecific direct access request is processed. If all requests are processed, go to step 4.		
• A recoverable DADSM error occurred (for example, the volume does not contain sufficient space to satisfy the request). Processing continues with step 3c.	IEFAB492	DADSERR3
• An unrecoverable DADSM error occurred (for example, the SPACE parameter was not coded). Processing continues with step 6.		
c) This step receives control if the volume cannot be used because of an enqueue error or a recoverable DADSM error. The following processing is performed in the event of a recoverable DADSM error:	IEFAB492	
• If any other requests were allocated to this volume, IEFAB4A0 (Common Unallocation Control) releases the space obtained for those requests. (This situation occurs when more than one nonspecific request needs the same volume and one or more of the duplicate requests has already been successfully allocated to this volume.)	IEFAB4A0	
• IEFAB4F0 dequeues from the volume.	IEFAB4F0	
For both enqueue and recoverable DADSM errors, the Allocation/VM&V Interface rebuilds the VM&V request block to indicate the current volume must be unloaded and a new volume mounted and verified. IEFAB493 receives control to perform the unload, mount, and verify. Step 6 receives control if the operator cancels this job (for a batch allocation) or the request (for a dynamic allocation), or if the mount or verify fails for an MSS volume. Otherwise, the newly mounted volume is processed – go to step 3a.	IEFAB492 IEFAB493	VMVRQBLD

Diagram 14-15. IEFAB492 – Allocation/Volume Mount and Verify (VM&V) Interface (Part 3 of 4)



4 Release processed VM&V request blocks and release device groups still serialized.

5 Process WTOR tape and specific direct access volume requests:

a) Update chain of VM&V requests.

b) Verify WTOR tape and specific direct access volume requests.

c) Interface with DADSM if space is required on direct access volumes (except for ISAM data sets).

6 If an error occurred in step 3, delete mount messages, if necessary, and release mount control blocks.

7 Release remaining VM&V request blocks.

Return to Common Allocation Cleanup (IEFAB490)

Diagram 14-15. IEFAB492 – Allocation/Volume Mount and Verify (VM&V) Interface (Part 4 of 4)

Extended Description	Module	Label
4 After all nonspecific direct access volume requests are successfully processed, the Allocation/VM&V Interface releases the request blocks associated with them.	IEFAB492	NSPDACTL
The Allocation Queue Manager (IEFAB4FA) then releases all device groups that are still serialized for this allocation.	IEFAB4FA	
5 The Allocation/VM&V Interface next processes all tape and specific direct access volume requests:	IEFAB492	
a) The Allocation/VM&V Interface updates the chain of VM&V request blocks so that it contains only specific direct access requests and WTOR tape requests. (If the mount message for a tape was issued in the form of a WTO, it is unnecessary to verify the volume – the volume verification is done when the data set is opened. No further processing is required for these requests and therefore they are released from the chain of VM&V requests.)	IEFAB492	UPDCHAIN
b) VM&V Control verifies all remaining requests -- specific direct access volume requests and WTOR tape volume requests. For details, see the M.O. diagram Volume Mount & Verify Control (IEFAB493).	IEFAB493	
c) The Allocation/VM&V Interface interfaces with DADSM for space if the data set is new and is not ISAM. (Space for new ISAM data sets is obtained by Common Allocation Cleanup – see the M.O. diagram Common Allocation Cleanup (IEFAB490).) If a DADSM error occurs, this allocation will be failed.	IEFAB492 IEFAB492	DADSMINT DADSERR2

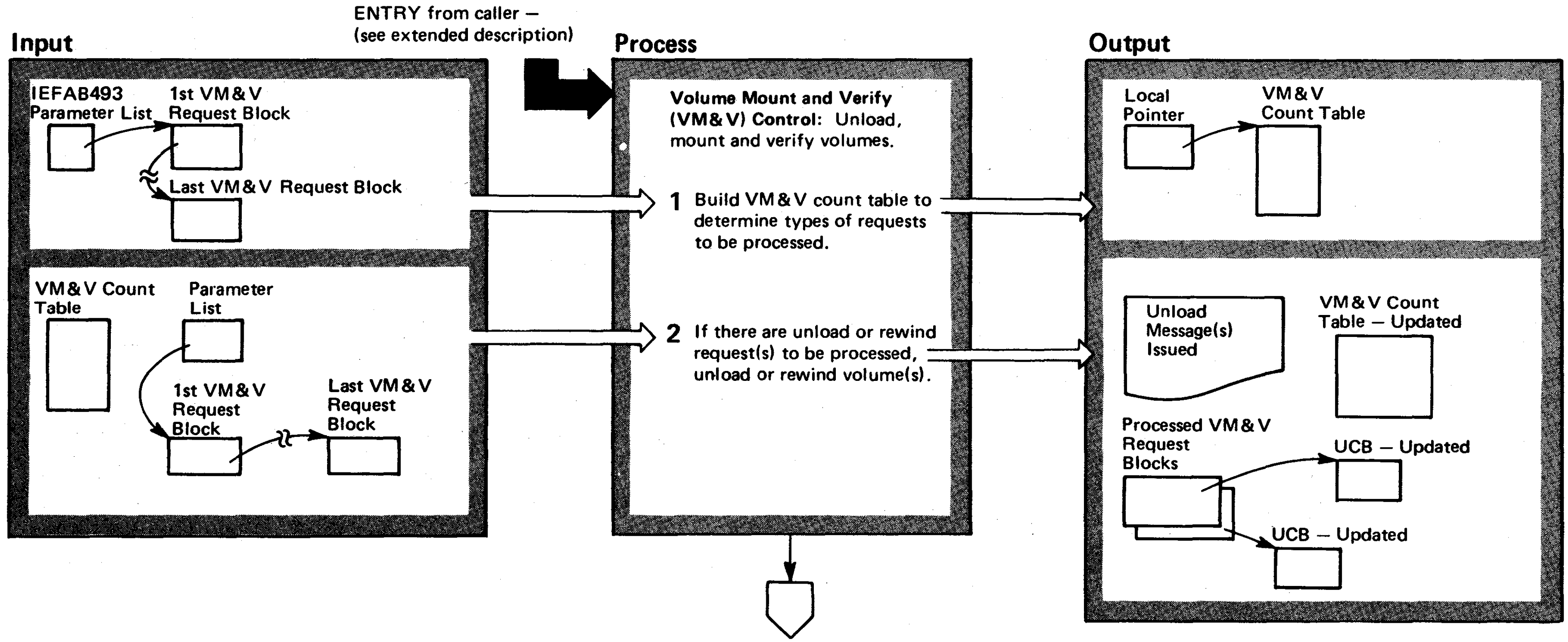
Extended Description	Module	Segment
6 This step is performed only if an error occurred in step 3. In step 2, IEFAB493 issued all mount messages and verified volumes mounted for nonspecific volume requests; to verify volumes, IEFAB493 created mount control blocks. (For details on both the mount control blocks and when they are created, see the M.O. diagram Volume Mount & Verify Control (IEFAB493).) If an error occurs in step 3, the mount control blocks must be released.		
IEFAB49A (VM&V DOMR and Cleanup Routine) receives control to delete mount messages for direct access and WTOR tape volume requests; when all messages have been deleted, IEFAB498 (MVCA Chain Processor) releases the mount control blocks for this allocation.	IEFAB49A IEFAB498	
7 The Allocation/VM&V Interface issues a FREEMAIN macro instruction to release remaining VM&V request blocks.	IEFAB492	

Error Processing

An error in any routine causes control to be returned to the calling routine.

An ESTAE exit routine established by IEFAB493 deletes mount control blocks for this allocation if an abnormal termination occurs in step 3.

Diagram 14-16. IEFAB493 – Volume Mount and Verify (VM&V) Control (Part 1 of 6)



Volume Mount and Verify (VM&V) Control: Unload, mount and verify volumes.

1 Build VM&V count table to determine types of requests to be processed.

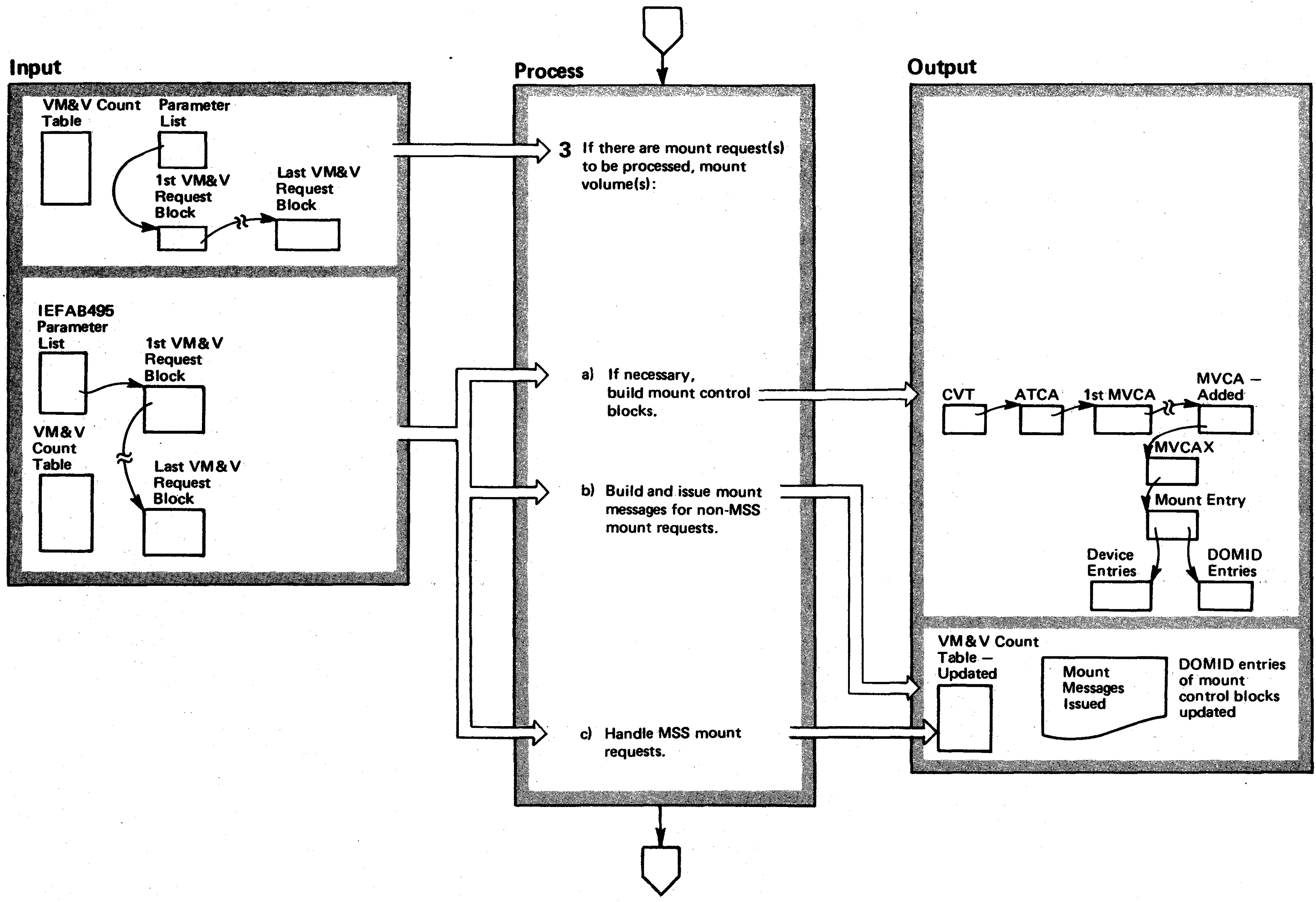
2 If there are unload or rewind request(s) to be processed, unload or rewind volume(s).

Diagram 14-16. IEFAB493 – Volume Mount and Verify (VM&V) Control (Part 2 of 6)

Extended Description	Module	Label
ENTRY Volume Mount & Verify (VM&V) Control has three functions: to unload or rewind a volume; to mount a volume; and to verify that a volume mounted by the Mount Control Routine is an acceptable volume (verify label or verify device end). Not all of these functions are necessarily performed each time VM&V Control is called – the functions performed depend on the caller:		
• The Unload Interface (IEFAB49C) calls VM&V Control to unload or rewind a volume.		
• The Allocation/VM&V Interface (IEFAB492) calls VM&V Control to do one of the following: mount all volumes and verify those volumes mounted for nonspecific volume requests; unload an unacceptable volume and mount and verify a new volume; verify volumes mounted for specific direct access volume requests and WTOR tape volume requests.		
• The Verify Control Routine (IEFAB496) calls VM&V Control to unload an unacceptable volume and to mount an acceptable volume.		
The parameter list passed to VM&V Control includes a pointer to the chain of VM&V request blocks – the VM&V request blocks indicate what functions must be performed.		
1 To determine what functions must be performed (unload/rewind, mount, or verify), VM&V Control searches through the chain of VM&V request blocks and builds a VM&V count table. The VM&V count table contains fields that indicate the number of:	IEFAB493	VMVSETUP
• Unload or rewind requests.		
• Mount requests.		
• Verify requests (to verify a direct access label or to verify that a device has become ready).		
The VM&V count table also includes a field called the DOM count. The DOM count represents both the number of mount messages to be deleted and the number of volume mounts to be verified – direct access volume mounts and WTOR tape volume mounts are verified; after the mount is verified, the message is deleted. (For WTO tape mounts, the data management OPEN routine verifies the volume and deletes the mount message.)		

Extended Description	Module	Segment
2 If the count table indicates that unload or rewind requests exist, IEFAB494 (Volume Unload Control) receives control. IEFAB494 searches the chain of VM&V request blocks to locate unload/rewind requests.	IEFAB494	
For each unload request, the following steps are performed:		
• For volumes not used for the Mass Storage System (MSS), IEFAB499 (VM&V WTO/R Format Routine) builds the unload message; IEFAB494 issues the message. (The message text exists in the message module IEFAB4M4.)	IEFAB499 IEFAB494	
• If a tape volume is being unloaded, IEFAB494 initializes and issues the channel commands necessary to have the tape rewound and unloaded.	IEFAB494	ISUEEXCP
If an MSS volume is being unloaded, IEFAB494 interfaces with the 3850 Mass Storage System to demount the volume.	IEFAB494	VIRTDEMT
• IEFAB494 clears all fields in the UCB that pertain to this volume.	IEFAB494	UCBCLEAN
• IEFAB494 decreases the count of unload/rewind requests in the VM&V count table. If this count reaches 0, IEFAB494 returns to VM&V Control.	IEFAB494	
For each rewind request, IEFAB494:		
• Initializes and issues the channel commands necessary to have the tape rewound.	IEFAB494	ISUEEXCP
• Clears the file sequence number and file sequence count in the UCB.	IEFAB494	
• Decreases the count of unload/rewind requests in the VM&V count table. If this count reaches 0, IEFAB494 returns to VM&V Control.	IEFAB494	

Diagram 14-16. IEFAB493 – Volume Mount and Verify (VM&V) Control (Part 3 of 6)



3 If there are mount request(s) to be processed, mount volume(s):

a) If necessary, build mount control blocks.

b) Build and issue mount messages for non-MSS mount requests.

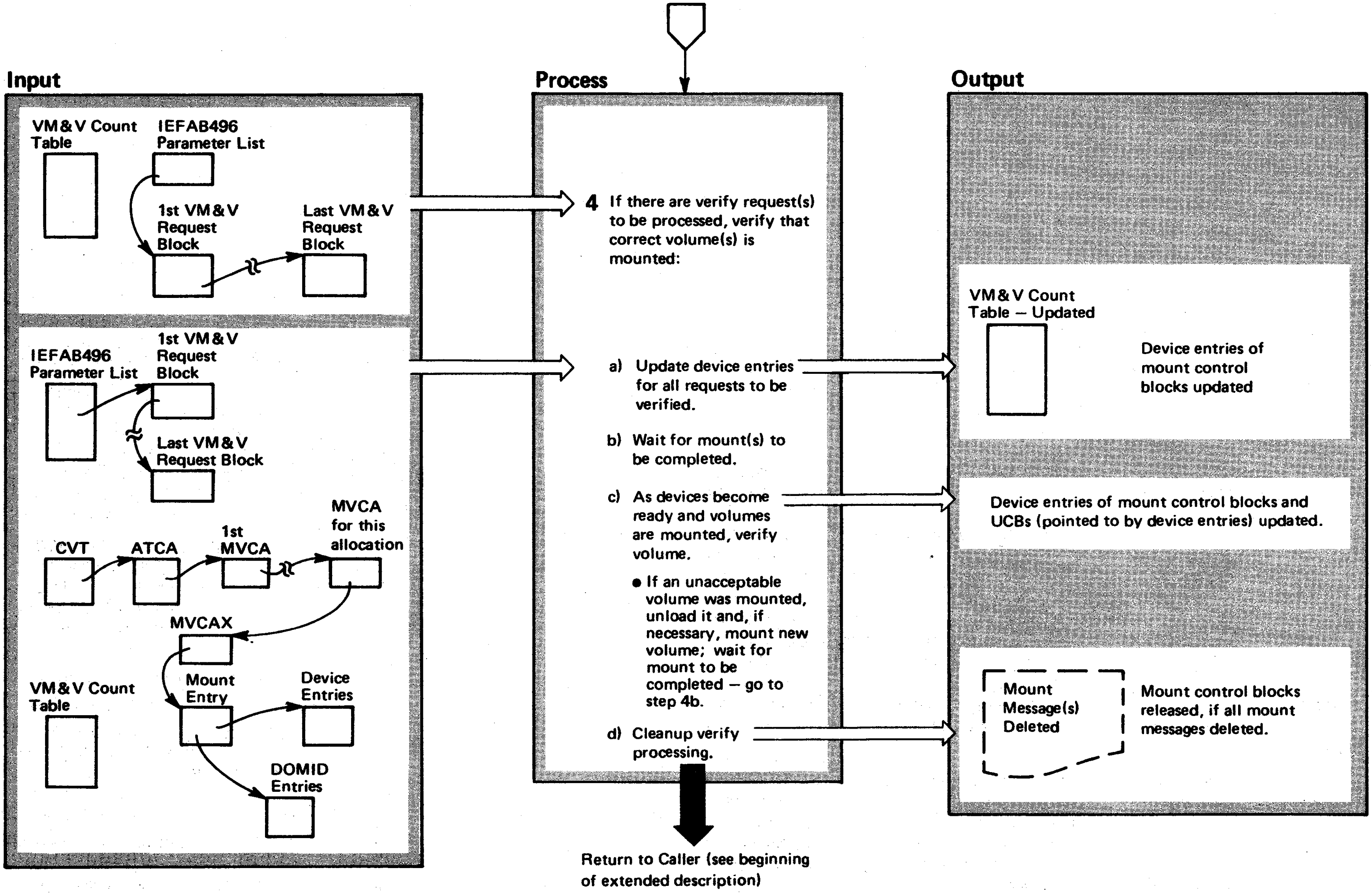
c) Handle MSS mount requests.

Diagram 14-16. IEFAB493 – Volume Mount and Verify (VM&V) Control (Part 4 of 6)

Extended Description	Module	Label
3 If the count table indicates that mount requests must be processed, IEFAB495 (Mount Control Routine) receives control. The following steps are performed:	IEFAB495	
a) If the DOM count in the VM&V count table does not equal 0 (that is, mount requests will have to be verified), mount control blocks are built if none exist. (Mount control blocks are necessary to keep track of which direct access and WTOR tape mount requests are not yet completed and what messages must be deleted.)		
The mount control blocks include:		
• An MVCA and MVCA extension (MVCAX), which include general information such as pointers to ECBs and pointers to other MVCAs on the MVCA chain. These blocks are created the first time IEFAB495 is called by this allocation, if they are necessary (the DOM count does not equal 0).		
• Mount entries. One mount entry exists for each call to IEFAB495; each mount entry points to a list of device entries, which identify the units on which volumes must be mounted and verified, and a list of domid entries, which identify the messages that must be deleted once the volumes are mounted and verified.		
For details on the mount control blocks, see *OS/VS2 Data Areas*, SYB8-0606.		

Extended Description	Module	Segment
IEFAB498 (MVCA Chain Processor) searches the existing MVCA chain to determine if an MVCA exists for this allocation. (An MVCA will not exist if this is the first time IEFAB495 is called by this allocation.)	IEFAB498	
If an MVCA does not exist, IEFAB495 obtains space for the mount control blocks and IEFAB498 adds them to the chain.	IEFAB495 IEFAB498	B495MSPC
b) For each non-MSS mount request, IEFAB499 builds a mount message; IEFAB495 then issues the message. If the mount must be verified and the message deleted (that is, if it is a direct access or WTOR tape mount message), IEFAB495 places the id of the message (DOMID) into the list of domid entries (which is pointed to by the mount entry – see step 3a).	IEFAB499 IEFAB495	 B495IWTO
As each mount message is issued, IEFAB495 decreases the mount request count in the VM&V count table – when the reaches 0, IEFAB495 returns to VM&V Control.	IEFAB495	
c) For each MSS request IEFAB495 interfaces with the 3850 Mass Storage System to mount the volume.	IEFAB495	VIRTMONT

Diagram 14-16. IEFAB493 – Volume Mount and Verify (VM&V) Control (Part 5 of 6)



4 If there are verify request(s) to be processed, verify that correct volume(s) is mounted:

a) Update device entries for all requests to be verified.

b) Wait for mount(s) to be completed.

c) As devices become ready and volumes are mounted, verify volume.

- If an unacceptable volume was mounted, unload it and, if necessary, mount new volume; wait for mount to be completed – go to step 4b.

d) Cleanup verify processing.

Return to Caller (see beginning of extended description)

Device entries of mount control blocks updated

Device entries of mount control blocks and UCBs (pointed to by device entries) updated.

Mount control blocks released, if all mount messages deleted.

Diagram 14-16. IEFAB493 – Volume Mount and Verify (VM&V) Control (Part 6 of 6)

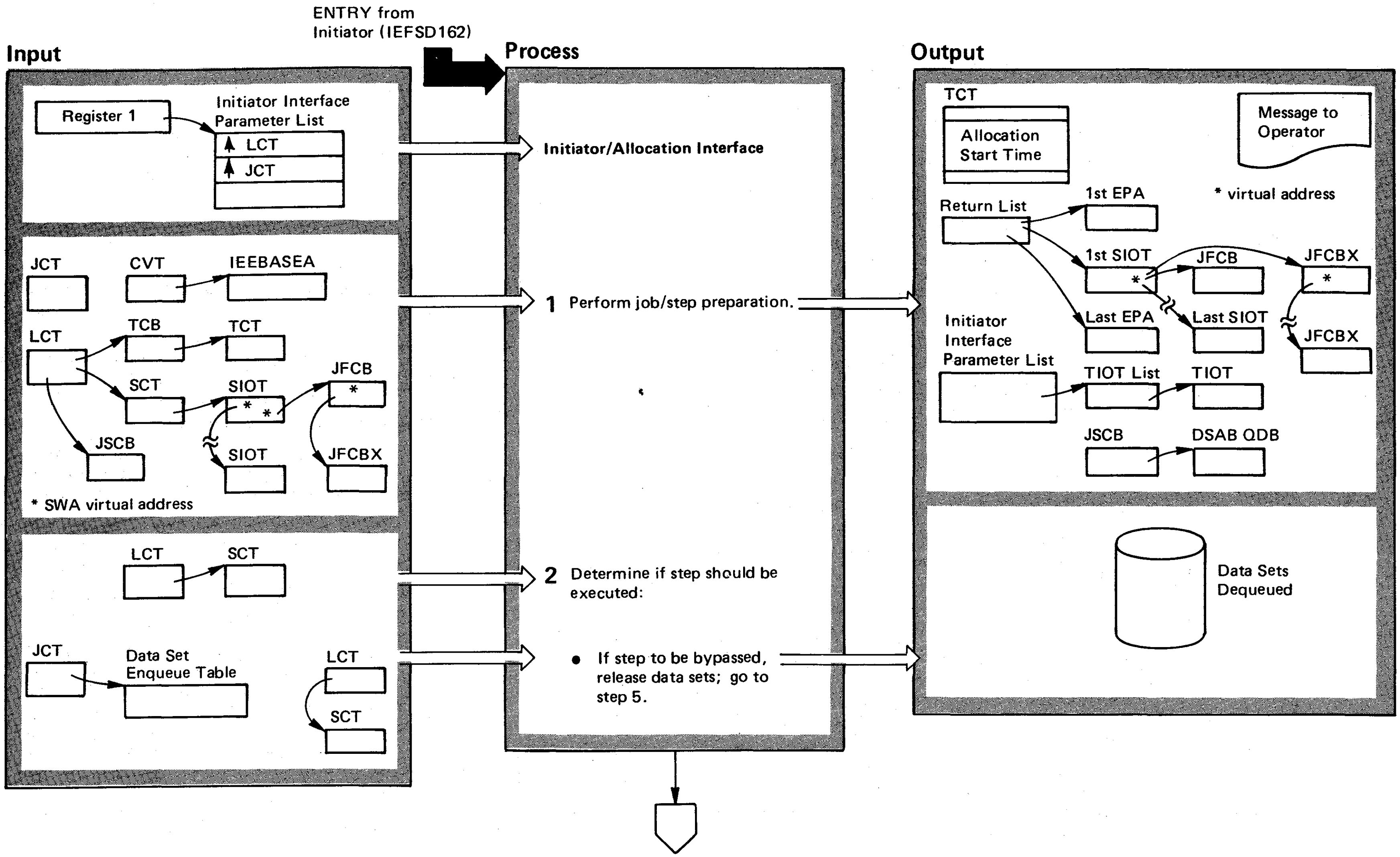
Extended Description	Module	Segment
4 If the count table indicates that verify requests must be processed, IEFAB496 (Verify Control Routine) receives control. Verify processing includes four steps:	IEFAB496	
a) The device entries of all requests to be verified are updated to indicate that verify processing is being performed; IEFAB498 locates the mount control blocks and IEFAB496 locates and updates the device entry. IEFAB496 also notes if the UCB pointed to by the device entry is ready (that is, a volume is mounted on it).	IEFAB498 IEFAB496	B496SRUP
b) IEFAB496 waits for all mounts that have not been completed and that need to be verified. As each mount is completed, step 4 is performed to verify that an acceptable volume was mounted.	IEFAB496	B496WAIT
c) If any waits are completed or if devices were found ready in step 4a, IEFAB49B (Device End Post Handler) receives control. The following processing occurs:	IEFAB496 IEFAB49B	B496POST
• If the label is to be verified, IEFAB4F8 reads the direct access label. (Tape lables are verified by the data management OPEN routine. The only verification done by allocation for tape volumes is to ensure that the device is ready, if the mount message was issued in the form of a WTOR as indicated by the common allocation parameter list — see figure 2-27.)	IEFAB4F8	
The volume can not be used if any of the following conditions occur:		
• IEFAB4F8 was not able to read the direct access label.		
• For specific volume requests, the volume serial number of the mounted volume does not match the volume serial number in the UCB.		
• For non-specific volume requests, the volume serial number of the mounted volume duplicates the volume serial number of another mounted volume.		
If the third condition occurs for an MSS volume, or if any of the conditions occur for a non-MSS volume, IEFAB493 is called to unload the volume and to mount an acceptable volume. If the first or second condition occurs for an MSS volume request, IEFAB493 is called to unload the volume and the job is terminated.	IEFAB493	
If none of the above conditions occurs, the volume is acceptable, and IEFAB496 updates the UCB with the volume serial number (for non-specific requests), updates the device entry to indicate verify processing is complete, and decreases the count of mounts being waited for.	IEFAB496	
• If the request is to verify that a device is ready, IEFAB496 updates the device entry to indicate that verify processing is complete and decreases the count of mounts being waited for.	IEFAB496	
As each request is successfully verified, IEFAB496 indicates that cleanup is needed for that request (see step 4d).		
If the operator cancels the job (batch allocation) or replies NO to a requested mount (dynamic allocation), IEFAB496 indicates that complete cleanup is necessary; this allocation will be failed.	IEFAB496	
d) IEFAB49A performs cleanup processing in two cases:	IEFAB49A	
• A request has been successfully mounted and verified. The mount message for the request is deleted; if all messages have been deleted, IEFAB498 releases the mount control blocks for this allocation.	IEFAB49A IEFAB498	
• The allocation is being failed. IEFAB49A deletes all mount messages for this allocation and IEFAB498 releases the mount control blocks.	IEFAB49A IEFAB498	

Error Processing

An error in any routine causes control to be returned to the calling routine.

When IEFAB493 receives control, it creates an ESTAE environment so that its exit routine receives control if an abnormal termination occurs.

Diagram 14-17. IEFBB401 – Initiator/Allocation Interface (Part 1 of 6)



Initiator/Allocation Interface

1 Perform job/step preparation.

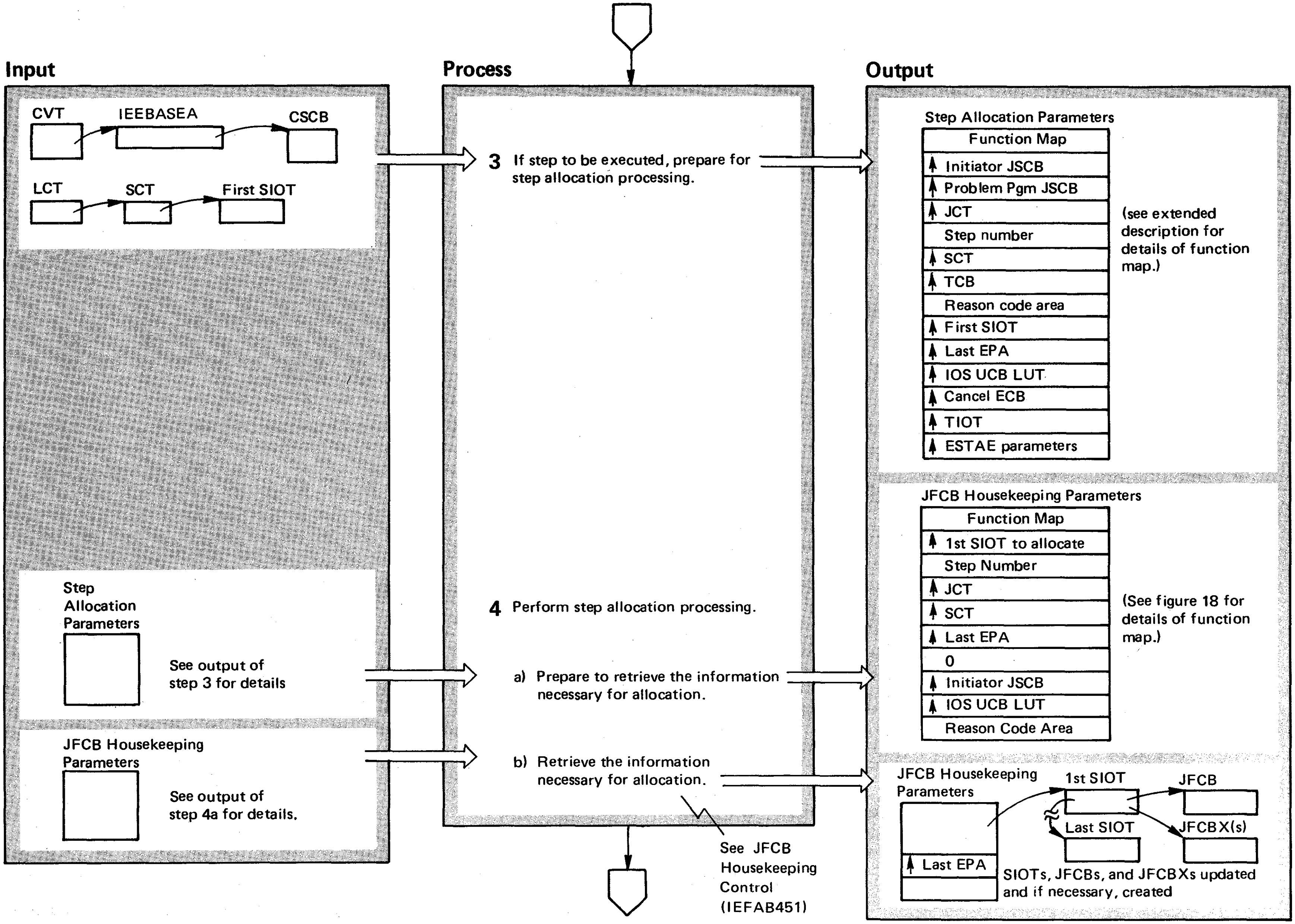
2 Determine if step should be executed:

- If step to be bypassed, release data sets; go to step 5.

Diagram 14-17. IEFBB401 – Initiator/Allocation Interface (Part 2 of 6)

Extended Description	Module	Segment
ENTRY The Initiator/Allocation Interface initializes and controls batch allocations; that is, jobs and logon requests.		
1 To prepare the job/step for allocation, the Initiator/Allocation Interface:	IEFBB401	JSTPPREP
• Issues one of three job status messages, if necessary: job not run, job started, or user logged on. The message issued depends on several factors. If the job failed prior to allocation of the first step (determined by checking the step number, which is included in the LCT, and the failure indicator in the JCTJSTAT field of the JCT), message IEF452I is issued: job not run. If the job didn't fail prior to allocation of the first step and if this is the first step, IEFBB401 determines if MONITOR JOBNAMES or SESSIONS is active. (IEEBASEA includes three fields – BASFL, BAMONITR, and MSBTN – that indicate whether MONITOR JOBNAMES or SESSIONS is active.) If so, it issues either message IEF125I, job started, or message IEF403I, user logged on.	IEFBB401 IEFBB401	JOBINIT BUILDMSG
• Issues the time macro instruction and places allocation start time in the TCT (timing control table), if it exists.	IEFBB401	STEPINIT
• Locates SIOTs, JFCBs, and JFCBXs in SWA, via the SWA Manager, and chains them together by means of the virtual addresses the SWA Manager returns. In SWA, each of these control blocks includes a prefix to which the SWA virtual address (SVA) points. The Initiator/Allocation Interface creates a SWA Manager external parameter area (EPA) for each control block; the EPA includes both the SVA and the virtual address of the actual beginning of the block, not including the prefix.	IEFAB4FE	
• Builds the TIOT Manager request block; the TIOT size (in multiples of 4K) is in the LCTTSIZ field of the LCT; if this is 0, the Initiator/Allocation Interface sets the TIOT size to 32K.	IEFBB401	STEPINIT
• Calls the TIOT Manager to create and initialize the TIOT and DSAB QDB (queue descriptor block).	IEFAB4FC	
2 To determine if a step should be executed, the Initiator/Allocation Interface processes step condition codes as specified in the COND parameter of the EXEC statement. Return codes from previous steps are included in the SCT.	IEFBB402	
If the step is to be bypassed, Data Set Release releases data sets enqueued by the initiator. The data set enqueue table includes, for each data set, the step number of the last job step that needs the data set. The SCT contains the current step number. If the current step number matches the step number associated with a data set in the data set enqueue table, Data Set Release issues the DEQ macro instruction for that data set. After the data sets are released, control is passed to step 5.	IEFAB4A6	

Diagram 14-17. IEFBB401 – Initiator/Allocation Interface (Part 3 of 6)



3 If step to be executed, prepare for step allocation processing.

4 Perform step allocation processing.

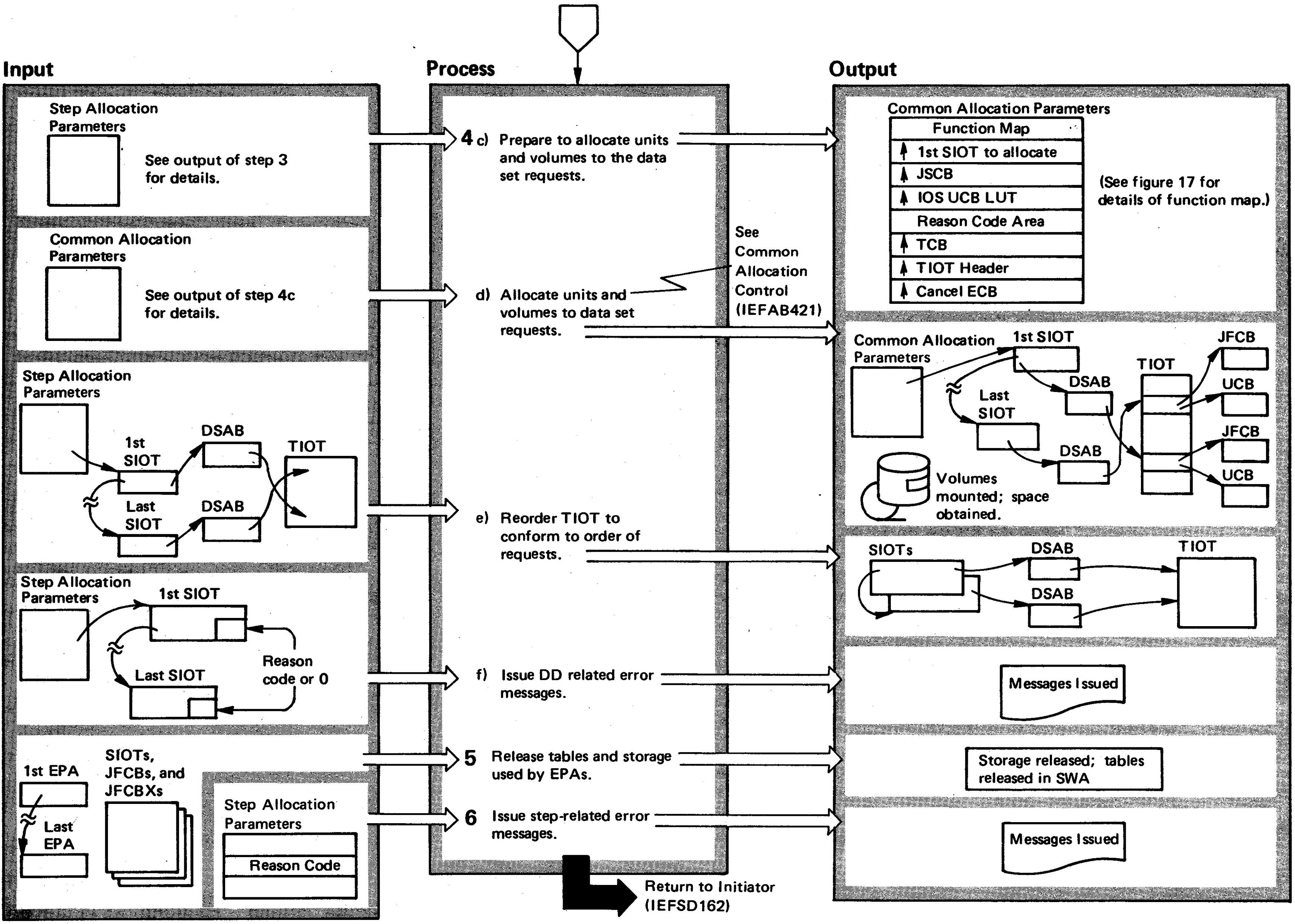
a) Prepare to retrieve the information necessary for allocation.

b) Retrieve the information necessary for allocation.

Diagram 14-17. IEFBB401 – Initiator/Allocation Interface (Part 4 of 6)

Extended Description	Module	Segment
3 If the step is to be executed, the Initiator/Allocation Interface constructs the parameter list for step allocation processing. The function map indicates if allocation should: ● Wait for units. This indicator is set on if the request is not a logon. ● Wait for volumes or data sets. This indicator is set on if the request is not a logon. ● Consider offline devices for recovery. This indicator is always set on by the Initiator/Allocation Interface. ● Issue an allocation message to the operator for unit record devices. This indicator is set on if MONITOR JOBNAMES is active. The CHTRKID field of the CSCB indicates if this request is a logon; the BASFL field in IEEBASEA indicates if MONITOR JOBNAMES is active.	IEFBB401	CALLALOC
4 Step Allocation Control (IEFBB404) performs step allocation processing:	IEFBB404	
a) Step Allocation Control (IEFBB404) constructs the parameter list for JFCB Housekeeping Control. For details on the function map, see figure 2-28.	IEFBB404	SETFUNMP
b) JFCB Housekeeping Control (IEFAB451) places the information necessary for allocation in the SIOTs, JFCBs, and JFCBXs, and creates additional tables if necessary. For details, see the M.O. diagram JFCB Housekeeping Control (IEFAB451).	IEFAB451	
c) Step Allocation Control (IEFBB404) constructs the parameter list for Common Allocation Control. For details of the function map, see figure 2-27.	IEFBB404	SETFUNMP
d) Common Allocation Control (IEFAB421) allocates units and volumes to the data set requests. For details on Common Allocation Control, see the M.O. diagram Common Allocation Control (IEFAB421).	IEFAB421	
e) The TIOT Manager (IEFAB4FC) compresses the TIOT and re-orders the TIOT entries and DSABs to conform to the order of the SIOTs.	IEFAB4FC	
f) If steps 4b or 4d indicated errors (return codes of 4 or 8 were returned), Step Allocation Control searches each SIOT for a non-zero reason code and the System Message Interface Routine issues the appropriate error message.	IEFBB404 IEFAB4FD	

Diagram 14-17. IEFBB401 – Initiator/Allocation Interface (Part 5 of 6)



4 c) Prepare to allocate units and volumes to the data set requests.

d) Allocate units and volumes to data set requests.

See Common Allocation Control (IEFAB421)

e) Reorder TIOT to conform to order of requests.

f) Issue DD related error messages.

5 Release tables and storage used by EPAs.

6 Issue step-related error messages.

Return to Initiator (IEFSD162)

Diagram 14-17. IEFBB401 – Initiator/Allocation Interface (Part 6 of 6)

Extended Description	Module	Segment
5 The Initiator/Allocation Interface:	IEFBB401	B401CLNP
• Releases control of the tables in SWA used during allocation.	IEFAB4F7	
• Issues the FREEMAIN macro instruction to release the storage used by each EPA.	IEFBB401	B401CLNP
6 The System Message Interface Routine (IEFAB4FD) issues step-related messages for any error (non-zero) reason code returned by means of the step allocation parameter list.	IEFAB4FD	

RETURN The Initiator/Allocation Interface returns to the Initiator. The Initiator Interface Parameter List is updated with a pointer to the TIOT list, which points to the TIOT created by the Initiator/Allocation Interface. The LCTERROR field of the LCT indicates if:

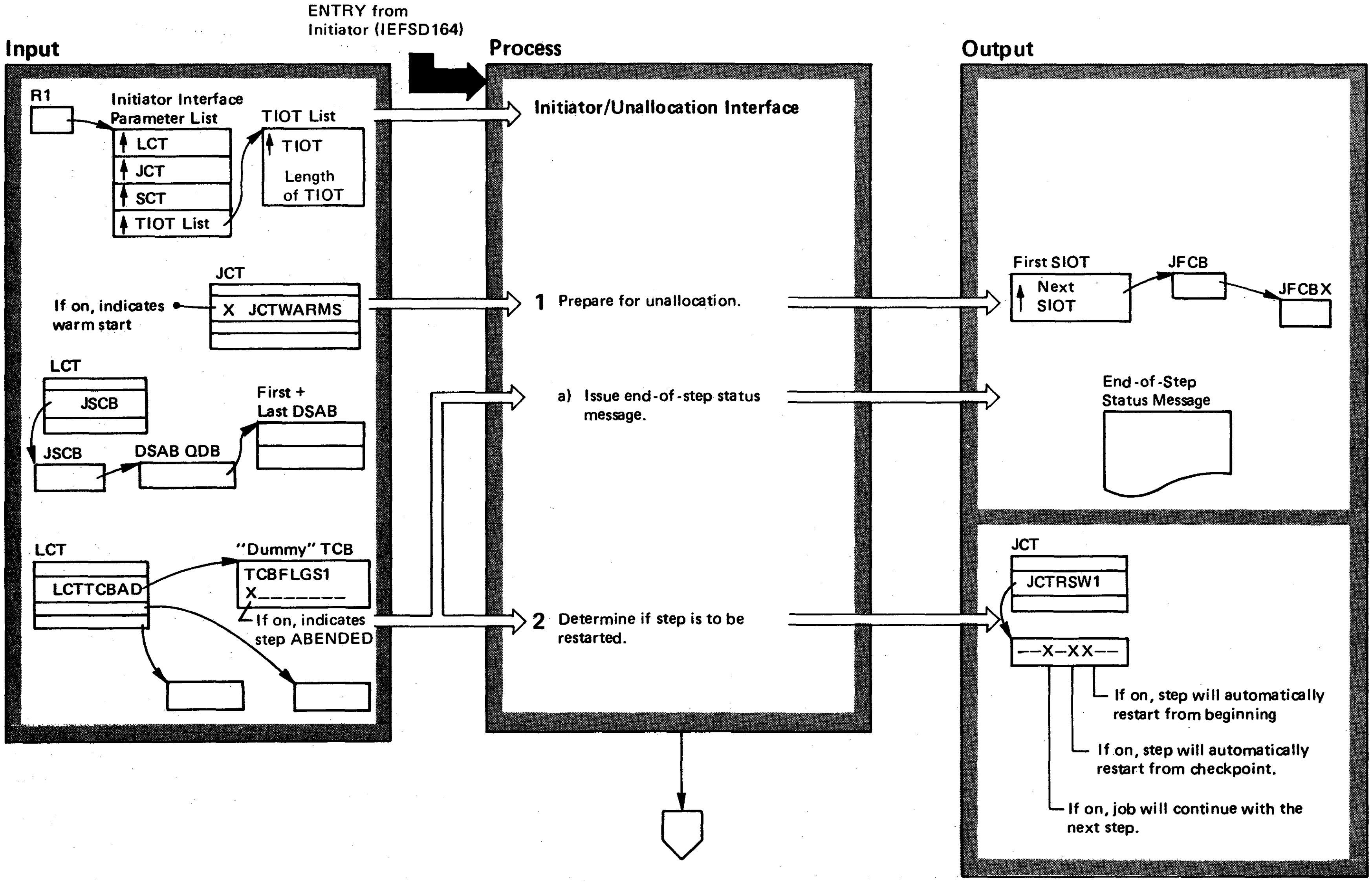
- The job failed.
- Any data sets were allocated for the job.
- Any data sets were allocated for the step.
- The step was not run due to condition codes.

Error Processing

An error in any routine causes control to be returned to the calling routine.

When IEFBB401 receives control, it creates an ESTAE environment so that its exit routine receives control if the program abnormally terminates.

Diagram 14-18. IEFBB410 – Initiator/Unallocation Interface (Part 1 of 8)



Initiator/Unallocation Interface

1 Prepare for unallocation.

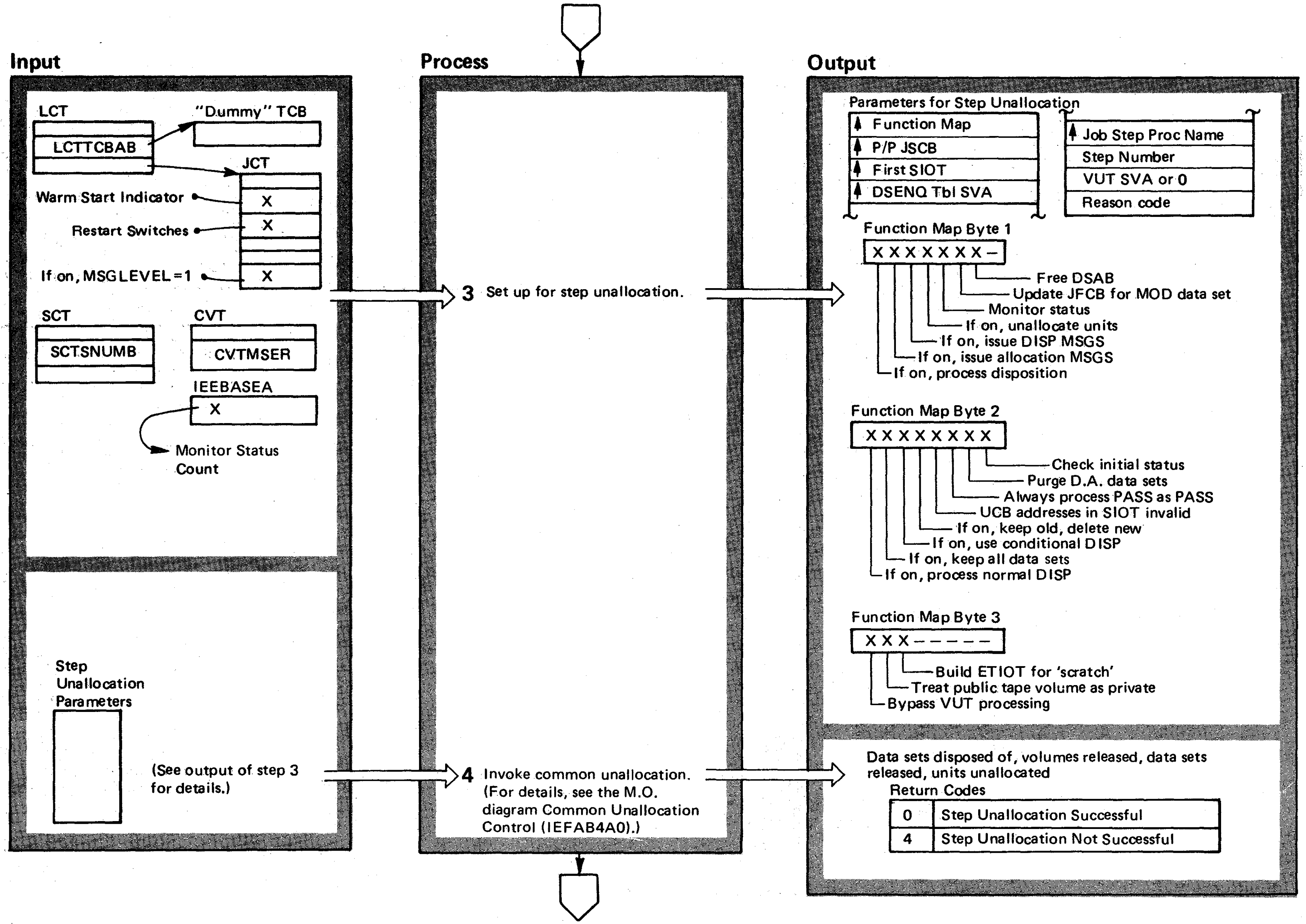
a) Issue end-of-step status message.

2 Determine if step is to be restarted.

Diagram 14-18. IEFBB410 – Initiator/Unallocation Interface (Part 2 of 8)

Extended Description	Module	Segment
ENTRY The Initiator/Unallocation Interface initializes and controls batch unallocations, that is, job step and logoff requests. Step unallocation is called if at least one of the step's data sets was allocated. During step unallocation, data set dispositions are processed, units are unallocated, and data sets and volumes are released. If the step being processed is the last step or if the job has been failed because of a job condition code, job unallocation is invoked. During job unallocation, disposition processing is done for all passed, unreceived data sets, and the job's private volumes are unloaded. Also, at end of job, this routine issues a generic dequeue to release all volumes and data sets associated with this job.	IEFBB410	
The major functions of this routine are to:		
• Use the SWA manager read/locate function to obtain the addresses of SIOTs, JFCBs, and JFCB extensions (JFCBX) and chain them together.		
• Issue step status messages.		
• Determine whether steps will automatically restart.		
• Call step unallocation.		
• Call SMF to perform end-of-step processing.		
• Free the TIOT.		
• Process job condition codes.		
• Call job unallocation if the present step is the last step to be executed, or if the job is being failed.		
• If the job is ending or failing, call SMF to perform end-of-job processing, and issue job status message.		
• Use the SWA manager to release the control of tables used by unallocation.		
1 The function of this step is to locate in the scheduler work area (SWA) all SIOTs, JFCBs, and JFCBXs. The job control table (JCT) contains an indicator (JCTWARMS) that indicates if the job is in a warm start environment. If the environment is a warm start, the SIOTs are read via the first SIOT pointer in the step control table (SCT) and the chain pointers in the SIOT themselves; otherwise, they are read via the chain of data set association blocks (DSABs). An ESTAE environment is created so that the exit routine receives control if the program abnormally terminates.	IEFBB410	B410INIT
a) One of the following messages is sent to the programmer:	IEFBB410	ISSUMSGS
• Step was executed.		
• Step was not executed.		
• Step was abnormally terminated. (This message is also issued to the operator.)		
2 In this step the "DUMMY" TCB is checked to see if the job has abnormally terminated (except for cancel ABEND). If it has, IEFRPREP is invoked, IEFRPREP will determine if a restart is possible and is authorized by the operator. It then sets the proper indicators in the JCT. Then the restart information from the SCT and JCT is used to decide whether the job is eligible for an automatic step or checkpoint restart, or a continue restart.	IEFRPREP	
Note: Because the original TCB was removed by a DETACH command from the initiator, a "DUMMY" TCB created by the initiator contains the ABEND indicator and completion code.		

Diagram 14-18. IEFBB410 – Initiator/Unallocation Interface (Part 3 of 8)



3 Set up for step unallocation.

4 Invoke common unallocation. (For details, see the M.O. diagram Common Unallocation Control (IEFAB4A0).)

0	Step Unallocation Successful
4	Step Unallocation Not Successful

Diagram 14-18. IEFBB410 – Initiator/Unallocation Interface (Part 4 of 8)

Extended Description	Module	Segment
3 The function of this routine is to use the information in the LCT, JCT, SCT, and CVT; and to construct the parameter list for step unallocation, indicating the unallocation processing to be done.	IEFBB410	CALLUNAL
4 The common unallocation function map and common unallocation request blocks are built requesting common unallocation to release volumes and data sets, unallocate units, and process data set dispositions. (For details, see the M.O. diagram: Common Unallocation (IEFAB4A0), Disposition Processing (IEFAB4A2), and Unit Unallocation Process (IEFAB4A4).	IEFBB414 IEFBB414	BLDCMRBS SETRBDSP

Diagram 14-18. IEFBB410 – Initiator/Unallocation Interface (Part 5 of 8)

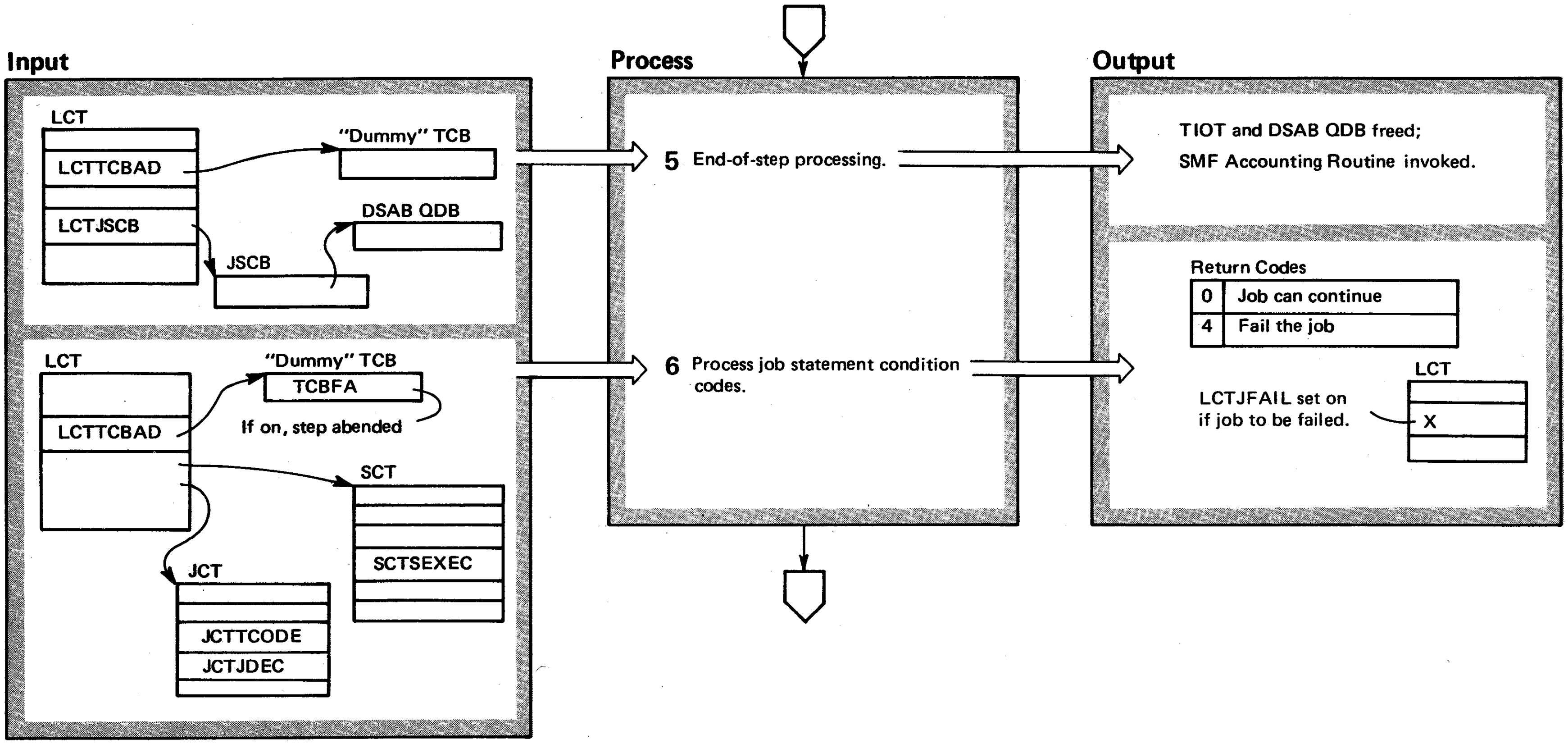
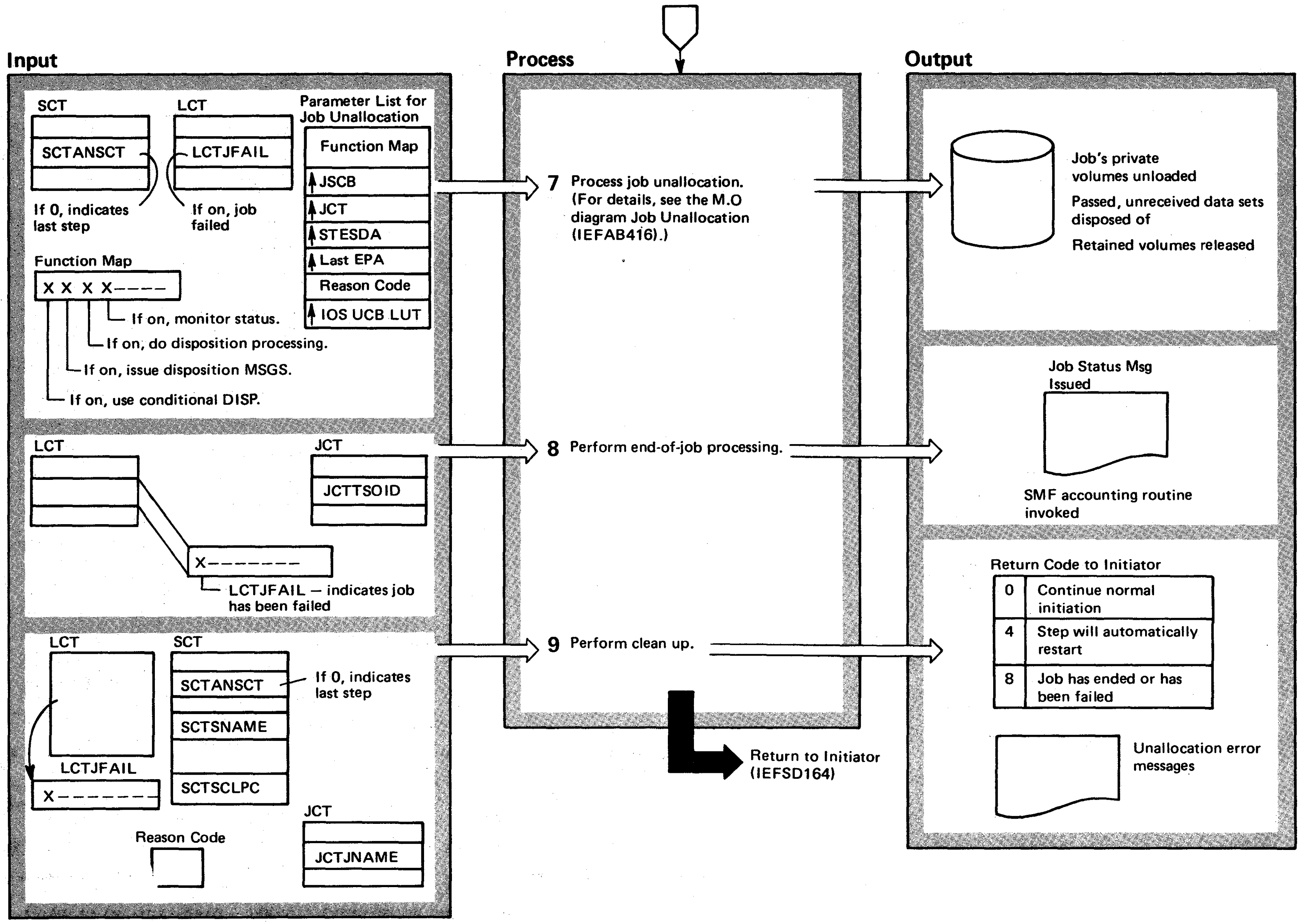


Diagram 14-18. IEFBB410 – Initiator/Unallocation Interface (Part 6 of 8)

Extended Description	Module	Segment
5 In this step the TIOT manager (IEFAB4FC) is invoked to free the TIOT and the DSAB QDB (queue descriptor block). The SMF accounting routine is invoked to do end-of-step processing.	IEFBB410	ENDOSTEP
6 Job statement condition codes from the JCT are checked against the return code of the current step to determine whether the job is to continue.	IEFBB412	

Diagram 14-18. IEFBB410 – Initiator/Unallocation Interface (Part 7 of 8)



7 Process job unallocation. (For details, see the M.O diagram Job Unallocation (IEFAB416).)

8 Perform end-of-job processing.

9 Perform clean up.

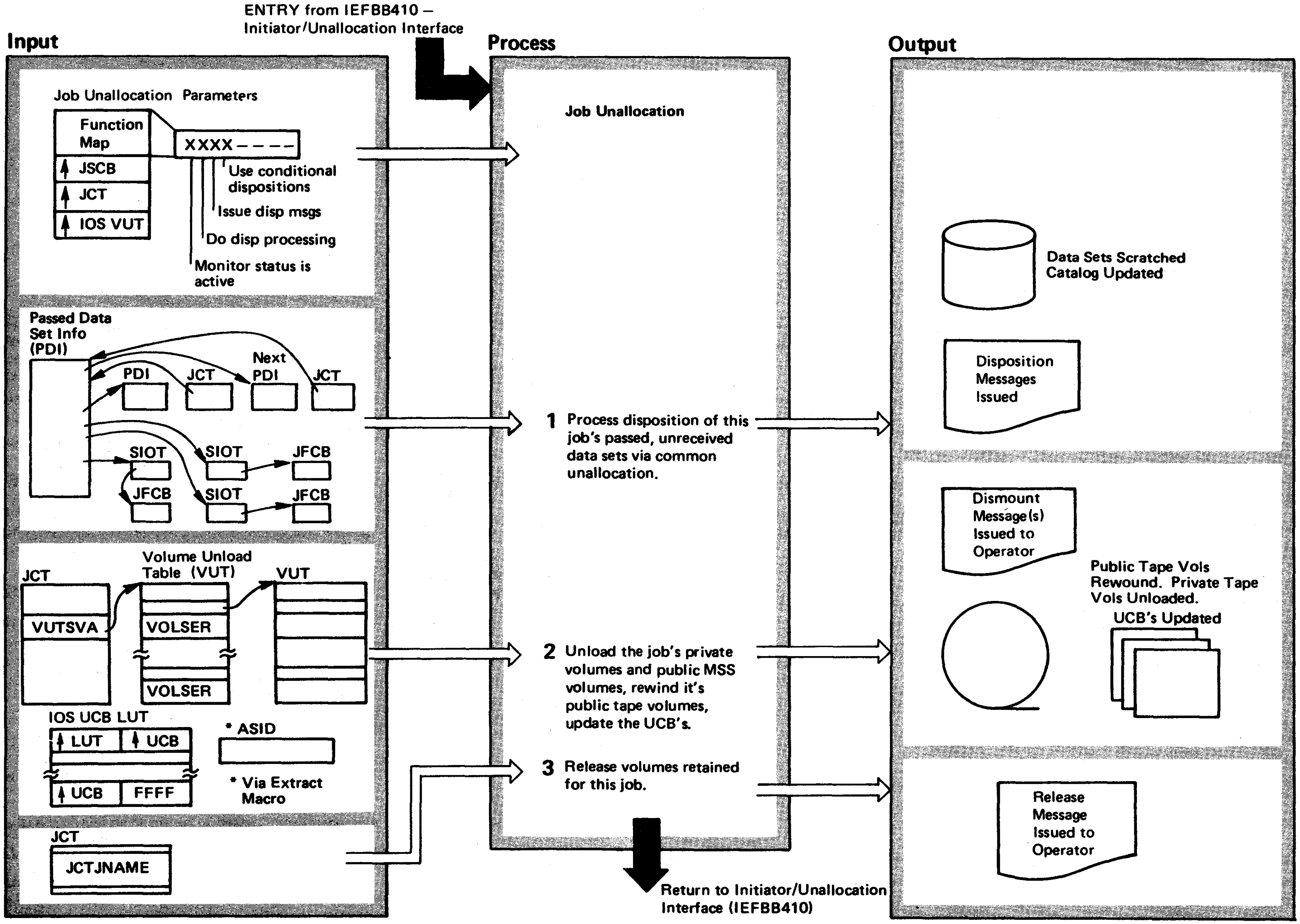
Return Code to Initiator

0	Continue normal initiation
4	Step will automatically restart
8	Job has ended or has been failed

Diagram 14-18. IEFBB410 – Initiator/Unallocation Interface (Part 8 of 8)

Extended Description	Module	Segment
7 Job Unallocation performs the following functions: • Process final dispositions of passed, unreceived data sets. • Unload the job's private volumes. • Release any volumes retained for the job. This step is performed only if the job is ending. For details, see the M.O. diagram Common Unallocation (IEFAB4A0) and Job Unallocation Processing (IEFBB416). **Note:** Common unallocation is called to do any disposition processing necessary.	IEFBB416 IEFAB4A0	
8 The SMF/User Accounting (IEFBB410) routine is invoked to issue an end-of-job record. One of the following messages is issued to the operator: "logged off," "job ended," "job failed – JCL error." This step is performed only if the job is terminating.	IEFBB410 IEFBB410	ENDOFJOB ENDJMSG
9 The SWA manager (IEFBB410) is used to write in locate mode all records read in for unallocation processing. If any error conditions occur during the unallocation process that required messages, the messages are now issued. The reason code(s) set during unallocation processing determines what messages to issue. The ESTAE environment is cancelled. Any unallocation messages still in write-to-programmer buffers are written.	IEFBB410	B410CLNP

Diagram 14-19. IEFBB416 – Job Unallocation (Part 1 of 2)



1 Process disposition of this job's passed, unreceived data sets via common unallocation.

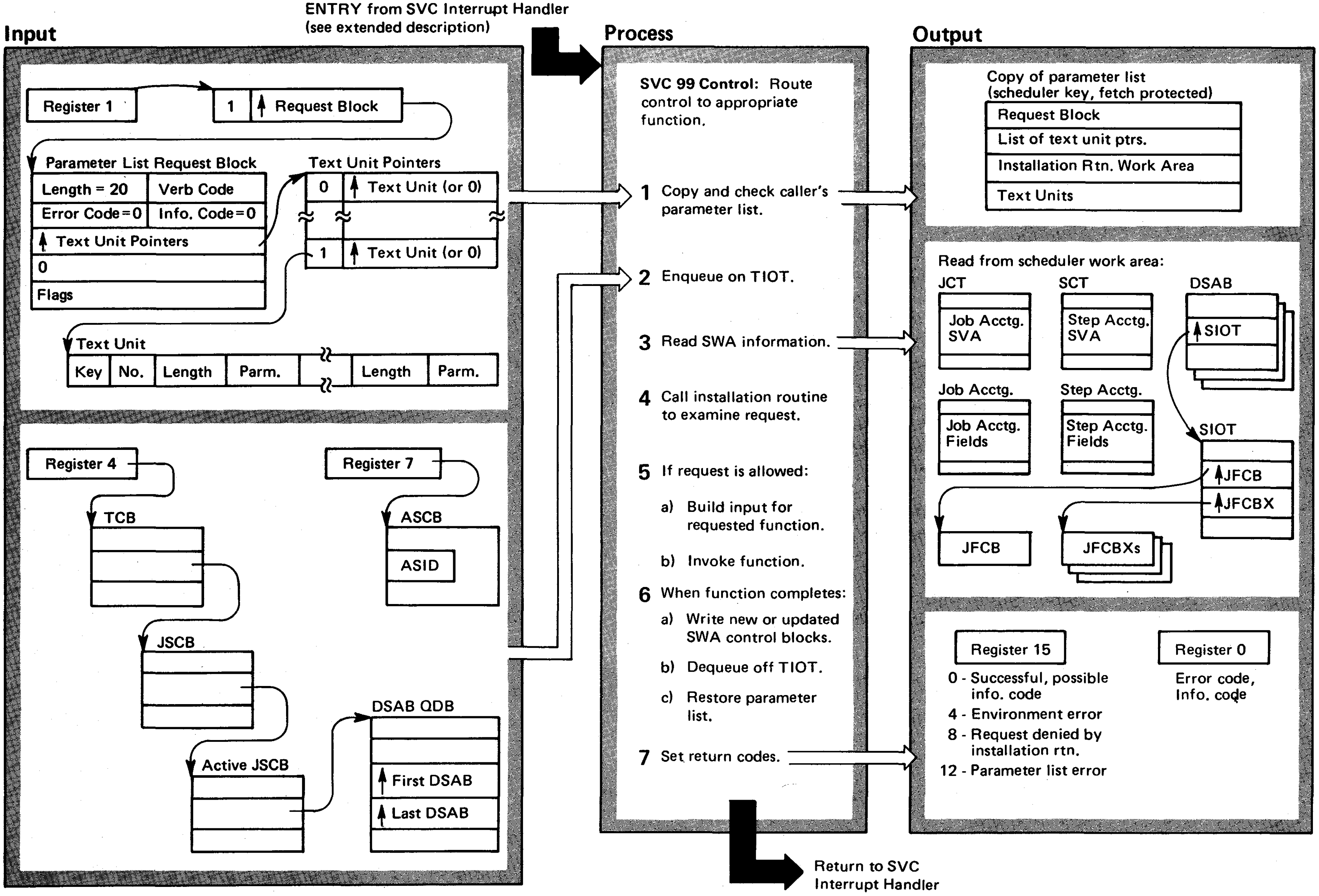
2 Unload the job's private volumes and public MSS volumes, rewind it's public tape volumes, update the UCB's.

3 Release volumes retained for this job.

Diagram 14-19. IEFBB416 – Job Unallocation (Part 2 of 2)

Extended Description	Module	Segment
ENTRY The functions of job unallocation are to process final dispositions of unreceived passed data sets and to perform volume clean-up: ● Unload the job's private volumes and rewind scratch tapes. ● Issue messages for released volumes.	IEFBB416	
1 All passed data set information (PDI) blocks are located via the SWA manager locate function. There is one PDI entry per data set passed within the job. For each passed, unreceived data set, the associated SIOT and JFCB are located via the SWA manager read/locate function. The PDI entry, SIOT, and JFCB are used to construct a common unallocation request block for each data set. When all the request blocks are built and chained, they are passed as input to the common unallocation routine (IEFAB4A0). Common unallocation performs all disposition processing and issues any disposition messages for the data set(s) associated with this job. For details, see the M.O. diagram "IEFAB4A0 – Common Unallocation."	IEFBB416 IEFAB416	READPDIS BLDCOMRB
2 The volume unload table (VUT) blocks are located and chained via the SWA manager. The VUT contains the volume serial numbers for all private volumes, public MSS volumes, and for volumes containing passed data sets used within the job. A list of unique volume serial numbers is built. This list is used to search the UCBs for this job's volumes, which should be rewound and unloaded.	IEFBB416 IEFBB416 IEFBB416 IEFBB416	VOLCLNP BVOLLIST UPDUCBS SCANUCBS
To serialize with other allocations, a list of UCBs is built. (See note below.) When serialization is complete, volume mount and verify request blocks are built and chained together.	IEFAB4EC	LOCKINIT
When the entire IOS look-up table has been searched, Volume Mount & Verify Control is called (via the Unload Interface) to rewind public tapes, rewind and unload demountable private tapes, and issue "KEEP" messages for demountable private volumes, both tape and direct access. (For details on Volume Mount & Verify Control, see the M.O. diagram Volume Mount & Verify Control (IEFAB493).)	IEFAB493	
The UCBs are updated to reflect the unloading and rewinding of volumes via volume mount and verify. After all rewinding and unloading is completed, units which were targetted for scratch tape mounts, but didn't have tape volumes mounted, are reset by setting the VOLSER and volume status in these units UCBs back to zero.		
Note: During this processing, serialization with other component's device processing such as the VARY OFFLINE, UNLOAD, and DDR, is accomplished by enqueueing sharable on major name SYSIEFSD, minor names CHNGDEVS, Q4, DDRTPUR, and DDRDA, and locking on the groups represented in a list of UCBs via the allocation Queue Manager.	IEFAB4EC	
3 Private volumes or volumes containing passed data sets may have been dismounted during this job's processing. If they were, a retain-type dismount message was issued to the operator. If such volumes are still not mounted, a message to the operator is issued indicating that the volumes are no longer needed by this job. Volumes mounted on units for the 3850 Mass Storage System will not appear in this message since these volumes do not require any operator action.	IEFBB416	RLSEMSG

Diagram 14-20. IEFDB400 – SVC 99 Control (Part 1 of 2)



SVC 99 Control: Route control to appropriate function.

1 Copy and check caller's parameter list.

2 Enqueue on TIOT.

3 Read SWA information.

4 Call installation routine to examine request.

5 If request is allowed:

- a) Build input for requested function.
- b) Invoke function.

6 When function completes:

- a) Write new or updated SWA control blocks.
- b) Dequeue off TIOT.
- c) Restore parameter list.

7 Set return codes.

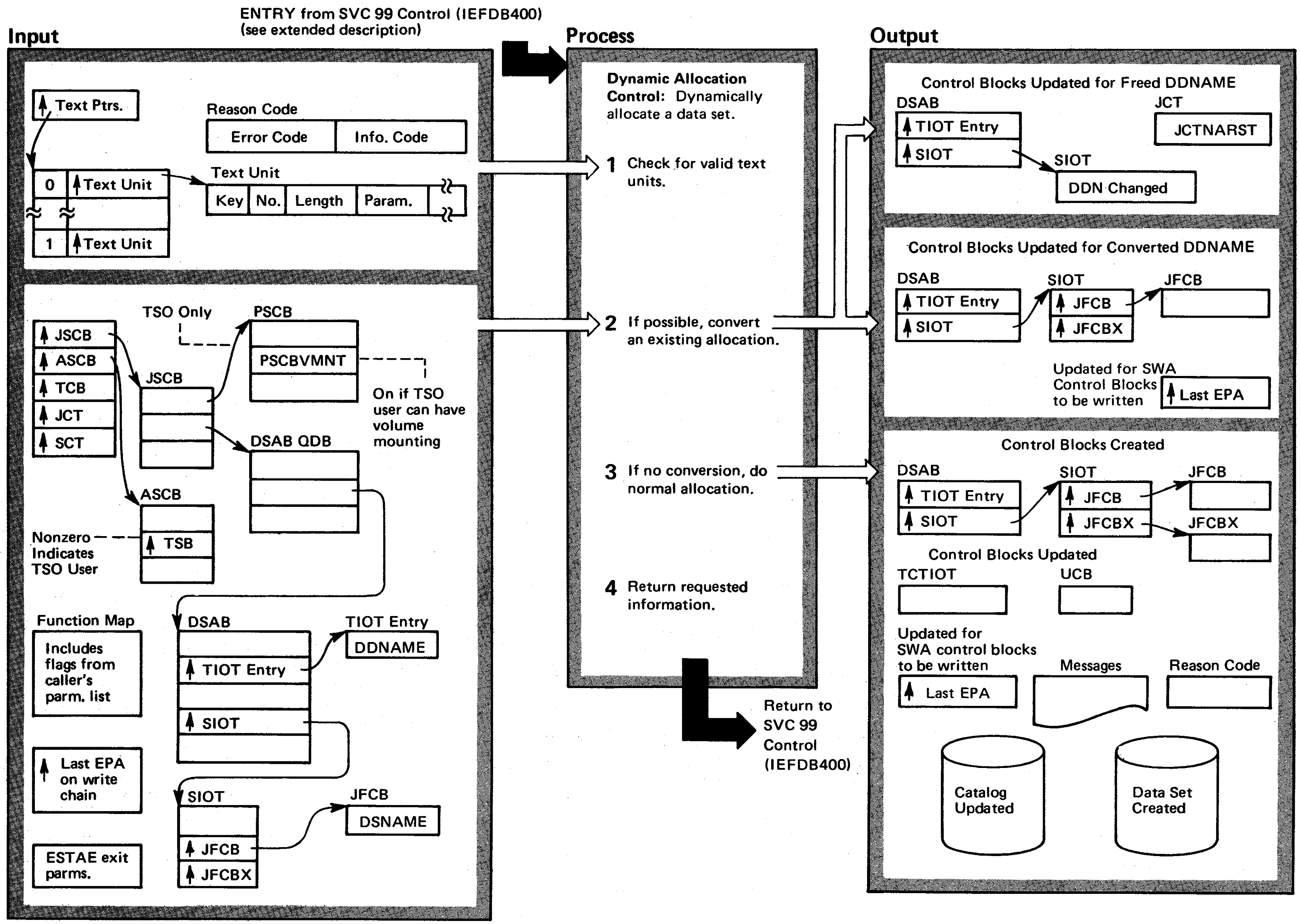
Register 15
0 - Successful, possible info. code
4 - Environment error
8 - Request denied by installation rtn.
12 - Parameter list error

Register 0
Error code, Info. code

Diagram 14-20. IEFDB400 – SVC 99 Control (Part 2 of 2)

Extended Description	Module	Segment
ENTRY SVC 99 control (IEFDB400) routes control from the SVC interrupt handler to a function that processes a request from a currently processing job step. The request may be from a background job step or from a foreground TSO job step.		
1 The ESTAE environment is established so that the exit (IEFDB402) that will receive control if there is an abnormal termination, such as an abend, cancel, or machine check. If one of these situations occurs, the exit routine will clean up and release resources that were obtained by SVC 99 Control.		
The caller's parameter list (request block, text unit pointer list, and text units) is copied into a scheduler-key, fetch-protected storage area to ensure the integrity of the parameter list and to allow reference by SVC 99, which runs in the scheduler key.	IEFDB400 IEFDB400	SCAN COPY
The format of the request block pointer, and the request block, is verified. If privileged functions are requested, the caller's authorization is checked. Any error found in these checks will cause step 7 to be done next.		S99CHKRB
2 To serialize dynamic allocation with data management and other dynamic allocations under the same job step, SVC 99 control enqueues on the caller's TIOT, if not already enqueued (shown by bit 5 in FLAGSON). The major name is SYSZTIOT, and the minor name is the two-byte ASID field of the caller concatenated to the address of the DSAB queue descriptor block.		
3 SVC 99 Control reads the SWA information by calling an allocation service routine to interface with the SWA manager, which reads in the JCT, SCT, and ACT for the job and step. Another service routine is called to read in the SIOTs, JFCBs, and JFCBXs from the SWA, chaining them to each other and to the DSABs. This is done either on the first execution of SVC 99 Control for a job step, or on the first execution (that is, in which processing abnormally terminated following an ESTAE exit).	IEFDB400 IEFAB4F7 IEFAB4FE	READSWA
4 To provide each installation the capability of checking the accounting information of a dynamic request and the capability of examining or modifying the SVC 99 parameter list, SVC 99 Control calls an installation replaceable routine. Return codes from the installation routine can accept or reject the allocation request.	IEFDB401	
5 If the dynamic request is allowed, SVC 99 Control sets up the input to the requested function and calls one of the following routines:		
• Dynamic Allocation Control	IEFDB410	
• Dynamic Unallocation Control	IEFDB4A0	
• Concatenate	IEFDB450	
• Deconcatenate	IEFDB460	
• Ddname Allocation	IEFDB490	
• Remove in-use Control	IEFDB480	
• Dynamic Information Retrieval.	IEFDB470	
Each of these routines is described in a separate M.O. diagram – see Figure 11: Allocation/Unallocation Functions and Related Method-of-Operation Diagrams.		
6 After the requested function has completed, any updated SWA control blocks are written to the SWA. SVC 99 Control dequeues off the caller's TIOT (if it enqueued) and restores the parameter list with any returned information. The ESTAE environment is canceled.	IEFAB4F7	RESTORE
7 After the return code is set in register 15 and the reason code (two-byte error code and two-byte information code) is set in register 0, SVC 99 Control returns to the SVC interrupt handler.		

Diagram 14-21. IEFDB410 – Dynamic Allocation Control (Part 1 of 2)



Dynamic Allocation Control: Dynamically allocate a data set.

1 Check for valid text units.

2 If possible, convert an existing allocation.

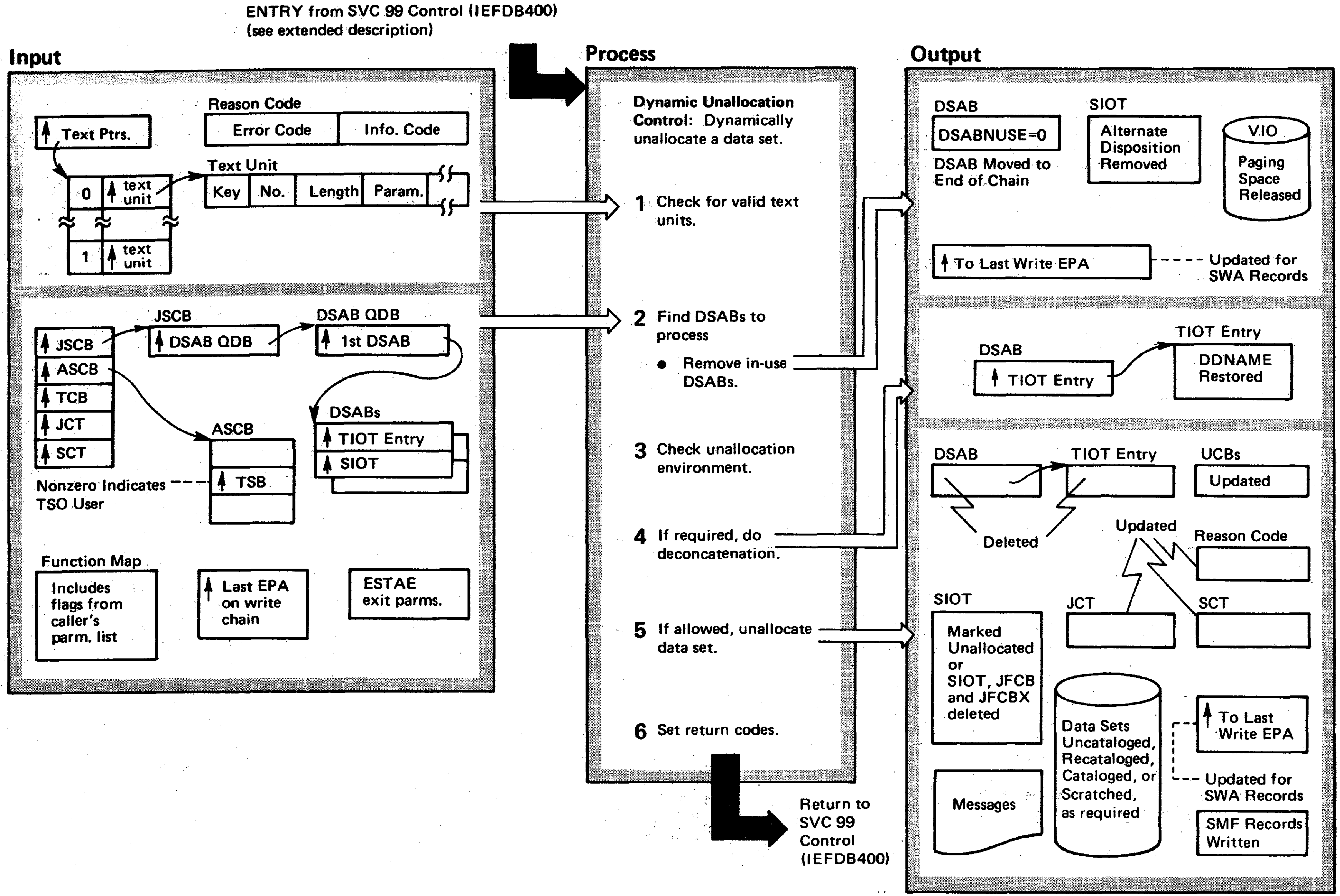
3 If no conversion, do normal allocation.

4 Return requested information.

Diagram 14-21. IEFDB410 – Dynamic Allocation Control (Part 2 of 2)

Extended Description	Module	Segment
ENTRY Dynamic allocation control (IEFDB410) is called by SVC 99 control to allocate a data set that was requested by a currently processing job step.		
1 The internal text is checked for format errors and for invalid, duplicate, mutually exclusive or inclusive keys, or for conflicting values specified for the keys. A key table of text unit pointers is built with the addresses of the specified text units at fixed table offsets. If a keyword is not specified, the table pointer for that key is set to zero. After the key table is built, the text units are edited to supply default values or to modify values as required.	IEFDB412 IEFDB410	 FXSYNTX
2 If possible, dynamic allocation control will convert an existing allocation to satisfy a request, rather than perform a normal allocation. The existing allocation environment is checked to see if this is possible.	IEFDB410 IEFDB411	CKCONVRT
3 If an existing allocation cannot be converted, normal allocation control is called to create and fill in the necessary control blocks and perform the new allocation. To retrieve additional information required for allocation, IEFDB413 calls JFCB Housekeeping Control (see the M.O. diagram JFCB Housekeeping Control (IEFAB451).) To perform the allocation, IEFDB413 calls Common Allocation Control (see the M.O. diagram Common Allocation Control (IEFAB421).)	IEFDB413 IEFDB413 IEFDB413	 HSKPINTF ALLOCINT
4 Following conversion, or a normal allocation, dynamic allocation control returns to SVC 99 control. The text units will indicate the ddname, DSNAME, DSORG, and VOLSER parameters used to satisfy the request, if they were requested by the caller. The error and information code fields of the reason code contain appropriate information.	IEFDB410	CLEANUP

Diagram 14-22. IEFDB4A0 – Dynamic Unallocation Control (Part 1 of 2)



Dynamic Unallocation Control: Dynamically unallocate a data set.

1 Check for valid text units.

2 Find DSABs to process

- Remove in-use DSABs.

3 Check unallocation environment.

4 If required, do deconcatenation.

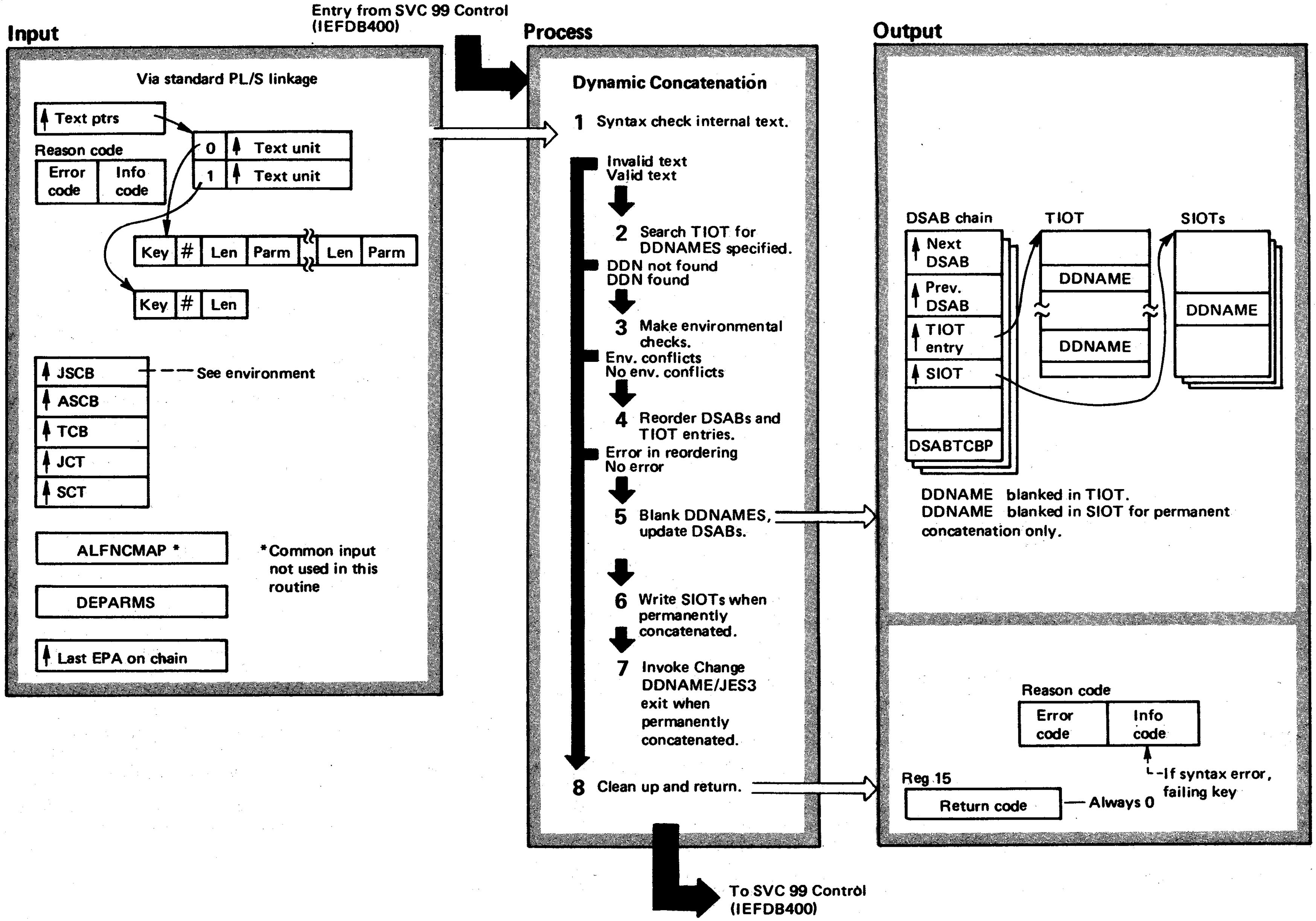
5 If allowed, unallocate data set.

6 Set return codes.

Diagram 14-22. IEFDB4A0 – Dynamic Unallocation Control (Part 2 of 2)

Extended Description	Module	Segment
ENTRY Dynamic unallocation control (IEFDB4A0) is called by SVC 99 control to unallocate a data set that was requested by a currently processing job step.		
1 The Dynamic Unallocation Control routine first checks the input request for valid text units by calling the syntax checker. The input is checked for invalid, mutually exclusive or duplicate keys, and for invalid number and length values. The key table, which contains pointers to specified keys, is filled in by the syntax checker. If there were no errors found in the previous checks, inclusive keys and invalid parameter values are checked. Any error condition detected in this step will cause step 6 to be done next.	IEFDB4A0 IEFDB4FF	VALIDCHK
2 Depending on the request, the ddname or dsname search routine is used to find the DSABs to be unallocated. If both dsname and ddname were specified, the search is by ddname, and the dsname is checked to ensure that it matches. For each DSAB found, processing is as follows:	IEFDB4FC IEFDB4FA	DDNPRCSS DSNPRCSS
• If unallocation or remove-in-use was specified, the appropriate processing is done.		
• If neither of the above options was specified and the DSAB is not permanently allocated, unallocation processing is done; otherwise, remove-in-use processing is done.		
• If the DSAB is for a nonpermanently allocated non-&dsname data set with a disposition of delete, it is unallocated regardless of the input option specified.		
If there are DSABs for remove-in-use, the remove-in-use processor is called.	IEFDB481	
3 Dynamic Unallocation Control scans the DSABs to be unallocated. It checks for open or cataloged data sets, or overriding dispositions of delete for shared data sets. If any of these conditions is detected, the DSAB is not unallocated.	IEFDB4A0	UNAENVCK
For a dsname specified, Dynamic Unallocation checks each DSAB to be unallocated to ensure that a permanently concatenated group is either a GDGALL dsname group or a VSAM group spanning device types. Otherwise, the data set is not unallocated.	IEFDB4A0	PCATCHKS
4 Deconcatenation (IEFDB460) is called to process any DSABs to be unallocated that are members of a concatenated group.	IEFDB460	
5 Dynamic Unallocation Control calls the Unallocation Processor to do the required unallocation for the eligible DSABs.	IEFDB4A1	
6 On completion of processing, the reason code field is set (two-byte error code and two-byte information code) and control is returned to the SVC 99 control routine.		

Diagram 14-23. IEFDB450 – Dynamic Concatenation (Part 1 of 2)



Dynamic Concatenation

1 Syntax check internal text.

Invalid text
Valid text

2 Search TIOT for DDNAMES specified.

DDN not found
DDN found

3 Make environmental checks.

Env. conflicts
No env. conflicts

4 Reorder DSABs and TIOT entries.

Error in reordering
No error

5 Blank DDNAMES, update DSABs.

6 Write SIOTs when permanently concatenated.

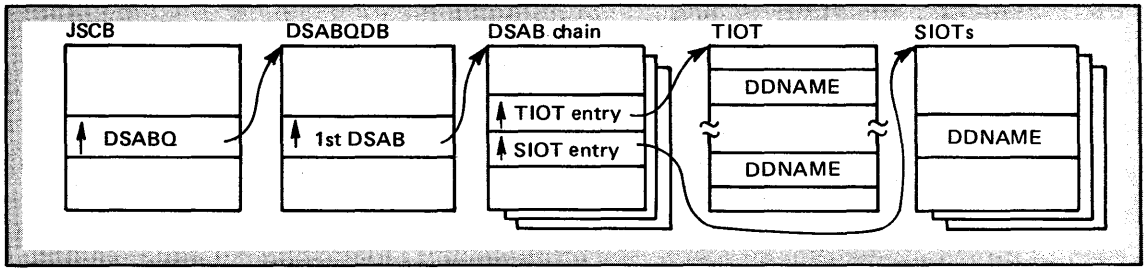
7 Invoke Change DDNAME/JES3 exit when permanently concatenated.

8 Clean up and return.

DDNAME blanked in TIOT.
DDNAME blanked in SIOT for permanent concatenation only.

Diagram 14-23. IEFDB450 – Dynamic Concatenation (Part 2 of 2)

Environment



Extended Description	Module	Label
Dynamic concatenation provides the user with a means of logically connecting allocated data sets into a concatenated group. These data sets must not be OPEN, or the request for dynamic concatenation is denied and an error return code is returned to the user.	IEFDB450	
All members of the dynamically concatenated group will be assigned the 'In-Use' attribute. The permanently concatenated attribute may optionally be assigned. If one member of the permanently concatenated group is permanently allocated the entire group becomes permanently allocated.		
1 Text is checked for invalid syntax or duplicate text units. If the ddname key is not specified or one of the specified ddnames is blank or JOBLIB, STEPLIB, JOBCAT, or STEPCAT, an error code is set.	IEFDB4FF	CCSYNCHK
2 The DSAB chain is searched for a TIOT entry with the specified ddname (done for each ddname supplied). If a ddname is not found or a duplicate ddname was specified, an error code is set.	IEFDB4FC	CCDDSRCH
3 An error code is set if any of the specified ddnames is associated with an OPEN data set.		CCDDSRCH
4 TIOT entries and DSABs are re-ordered prior to concatenation, i.e., entries are made contiguous in the order of ddname specification.	IEFAB4FC IEFAB4FC	CONCATIO CCATDSAB
5 The ddnames in the TIOT are blanked. DSABDCAT, DSABNUSE and DSABCATM are set to 1. The TCB address (from the input parameters) is stored in DSABTCBP. If permanent concatenation was requested, (DSABPCAT is set to 1); if any member of a PCAT group is permanently allocated (DSABPALC is set to 1), all members of the group are marked permanently allocated. To reflect the concatenation, the TCTIOT is updated and an SMF record 40 is written.	IEFAB4FC IEFDB450 IEFDB4F9	CONCATIO CONCAT
6 For a permanent concatenation request, SIOT ddnames are blanked and the SIOTs are written.	IEFAB4F7	
7 For a permanent concatenation request, invoke the Change DDNAME/JES3 exit to notify the subsystem of any changes in DDNAME and relative position number.	IEFDB4FB	
8 Return is made to the caller with register 15 set to 0.		

Diagram 14-24. IEFDB460 – Dynamic Deconcatenation (Part 1 of 2)

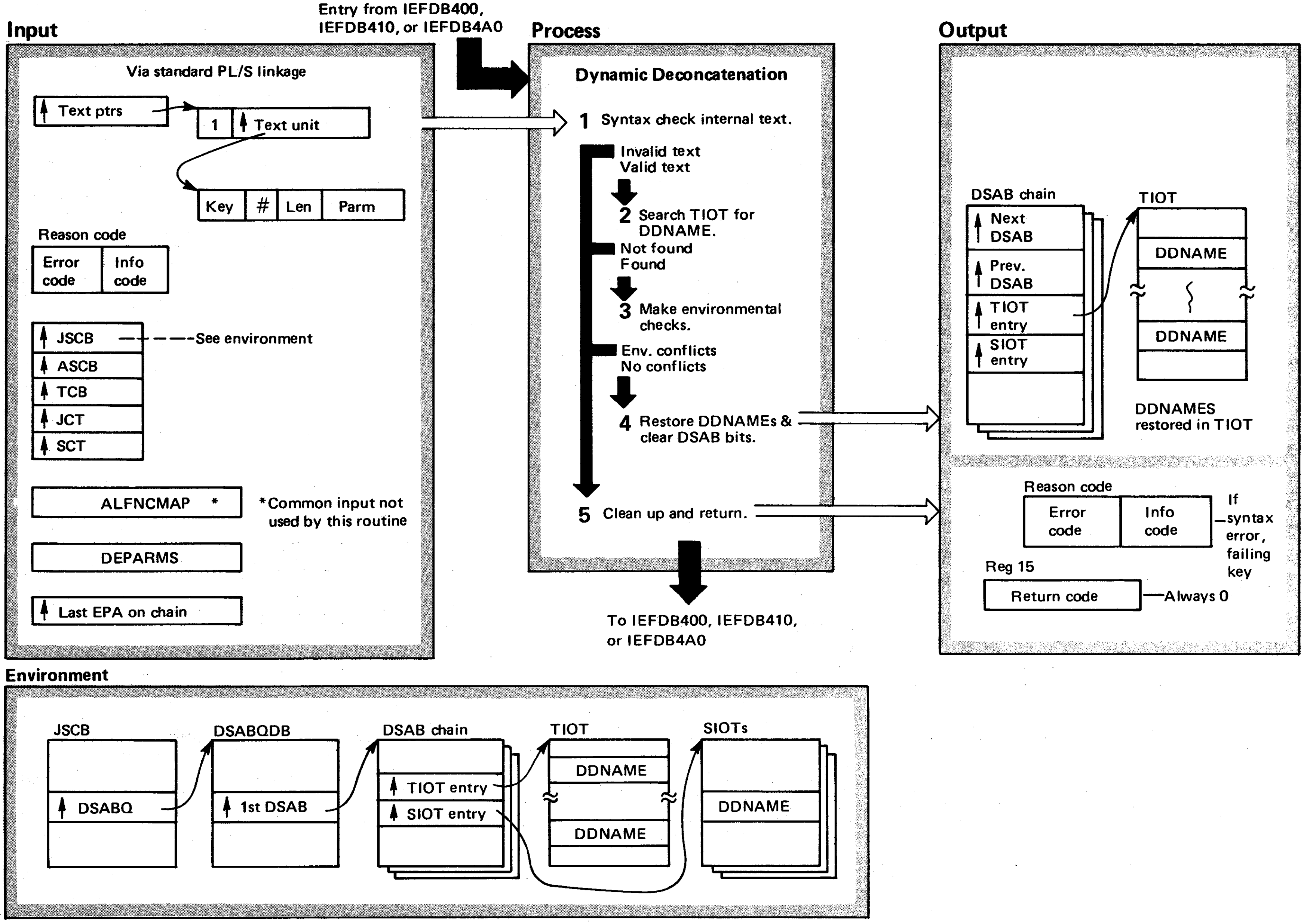


Diagram 14-24. IEFDB460 – Dynamic Deconcatenation (Part 2 of 2)

Extended Description	Module	Label
Dynamic deconcatenation provides the user with a means of logically disconnecting the members of a concatenated group. The user must specify the ddname of the concatenated group. No data set within the concatenated group may be OPEN. A permanently concatenated group or members of a concatenated group which are permanently concatenated will remain concatenated. When a concatenated group is dynamically deconcatenated, the ddnames that were associated with the data sets before they were concatenated are restored unless this would result in duplicate ddnames. This situation could arise if a dynamic allocation with the ddname to be restored occurred after a dynamic concatenation. In this case the deconcatenation request is failed. **Note:** Dynamic deconcatenation has no effect on the "In-Use" attribute.	IEFDB460	

Extended Description	Module	Label
1 Text is checked for invalid syntax or duplicate text units. If the ddname key is not specified, or the specified ddname is blank or is JOBLIB, STEPLIB, JOBCAT, or STEPCAT, an error code is set.	IEFDB4FF	SYNCK460
2 The DSAB chain is searched for a TIOT entry with the specified ddname. If the ddname is not found, an error code is set.	IEFDB4FC	ENVIRNCK
3 An error code is set if the specified ddname is associated with any OPEN data set or if deconcatenation would cause duplicate ddnames in the TIOT.		ENVIRNCK SIOTDDCK
4 For permanently concatenated members, DSABDCAT is set to 0. For temporarily concatenated members, DSABDCAT and DSABCATM are set to 0 and ddnames from the appropriate SIOT are restored in the appropriate TIOT entry.		
5 Return is made to the caller with register 15 set to 0.		

Diagram 14-25. IEFDB470 – Dynamic Information Retrieval (Part 1 of 2)

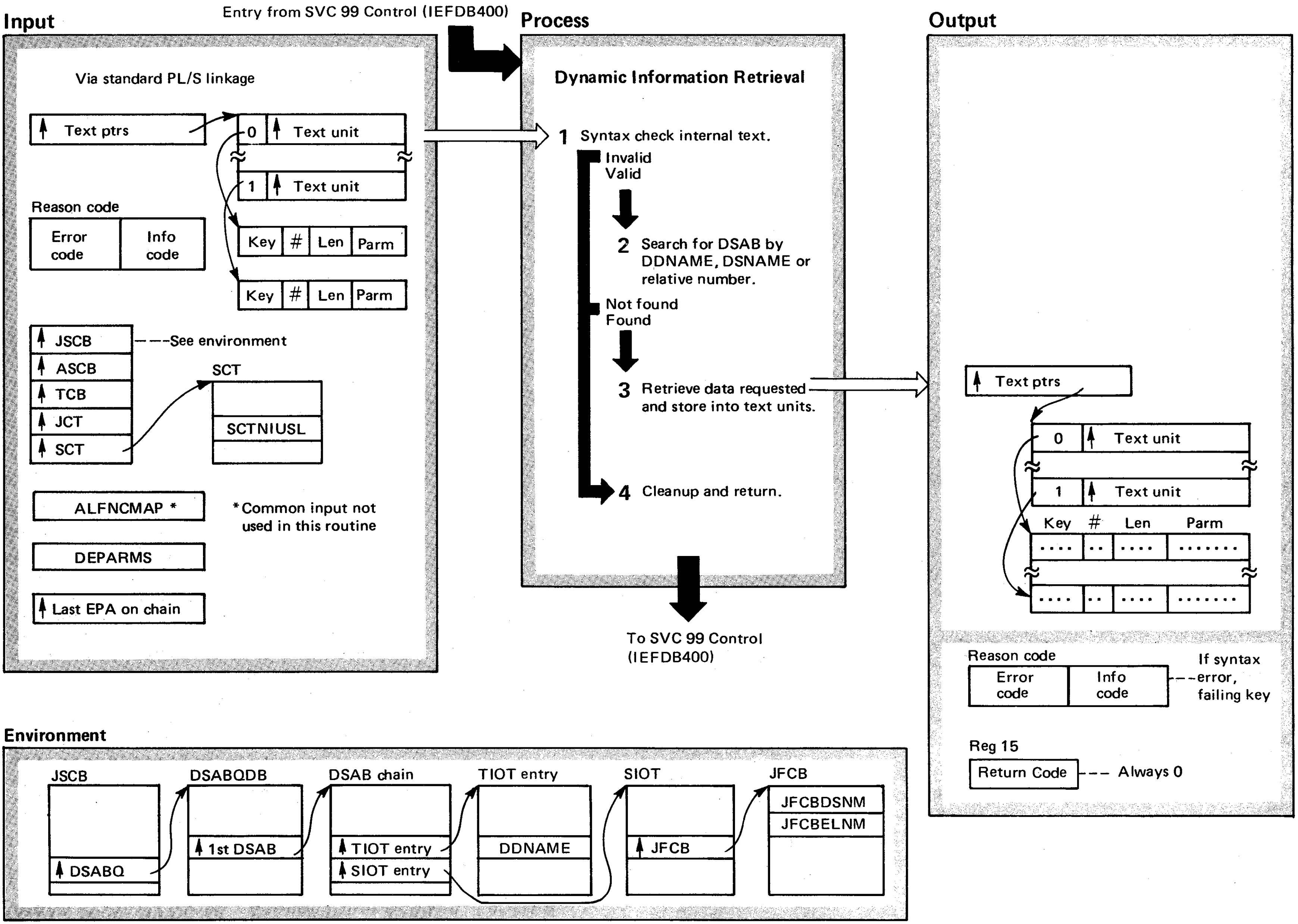
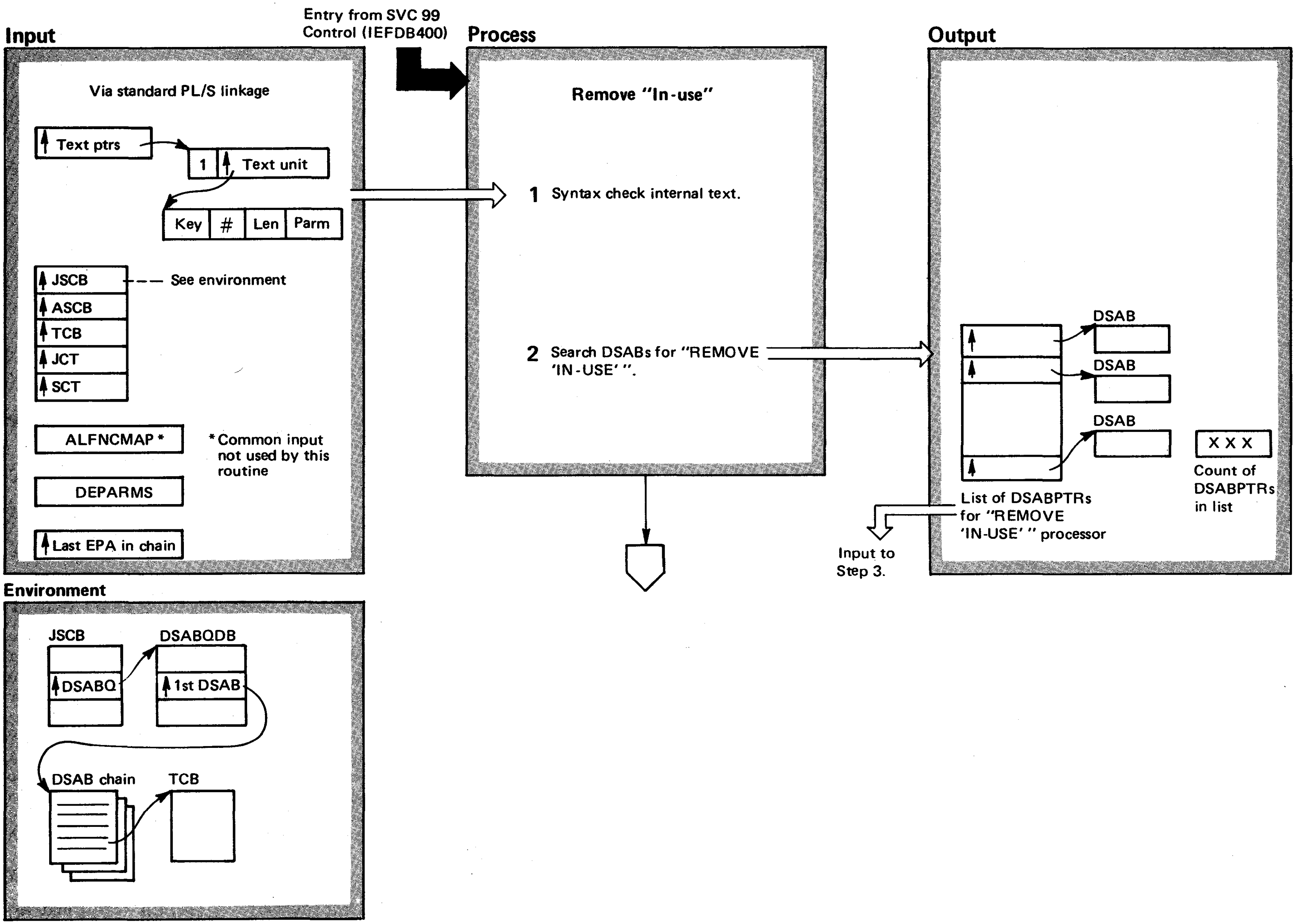


Diagram 14-25. IEFDB470 – Dynamic Information Retrieval (Part 2 of 2)

Extended Description	Module	Label
Information retrieval provides the user with information about his current allocation environment. The user can request the information via the ddname or dsname. In addition, the user may ask for information about any of his currently allocated requests by specifying a relative entry number. For example, information about all requests can be acquired by asking for information about the first, second, and successive entries in order. A unique return code is provided when information is requested for a non-existent relative entry number.	IEFDB470	
1 Text is checked for invalid syntax or duplicate text units	IEFDB4FF	DINSYNCK
2 The DSAB chain is searched by ddname, dsname, or relative number. If the entry is not found, an error code is set.		DINRTSRC
3 Requested data from appropriate control block (see environment) is stored into text units.		RETRIEVE
4 Return is made to the caller with register 15 set to 0.		

Diagram 14-26. IEFDB480 – Remove In-Use Attribute (Part 1 of 4)



1 Syntax check internal text.

2 Search DSABs for "REMOVE 'IN-USE' ".

Diagram 14-26. IEFDB480 – Remove In-Use Attribute (Part 2 of 4)

Extended Description	Module	Label
The function of the Remove "In-Use" routine is to handle dynamic allocation (SVC 99) requests for Remove "In-Use" processing.	IEFDB480	
1 Checks are made to ensure that exactly one of the two valid remove in-use keys was specified, and that valid number and length fields were also specified for it. The syntax checker is called to assist in these checks. If an error is detected, step 4 is processed.	IEFDB4FF	
2 The DSABs for remove in-use processing are determined as follows:		
• If the current task option key was specified, a scan up the mother TCB chain to the initiator's TCB (or, if 0, the job step TCB) is made. Looking at each in-use DSAB in the DSAB chain not for a private catalog, remove in-use processing will be done for any of these DSABs whose TCB address does not match that of a TCB found in the TCB scan.		CUTSKSCH
• If the TCB address key was specified, a scan of the DSAB chain is made and remove in-use processing will be done for any in-use, non-private catalog DSAB which has the specified TCB address.		TCBADSCH

Diagram 14-26. IEFDB480 – Remove In-Use Attribute (Part 3 of 4)

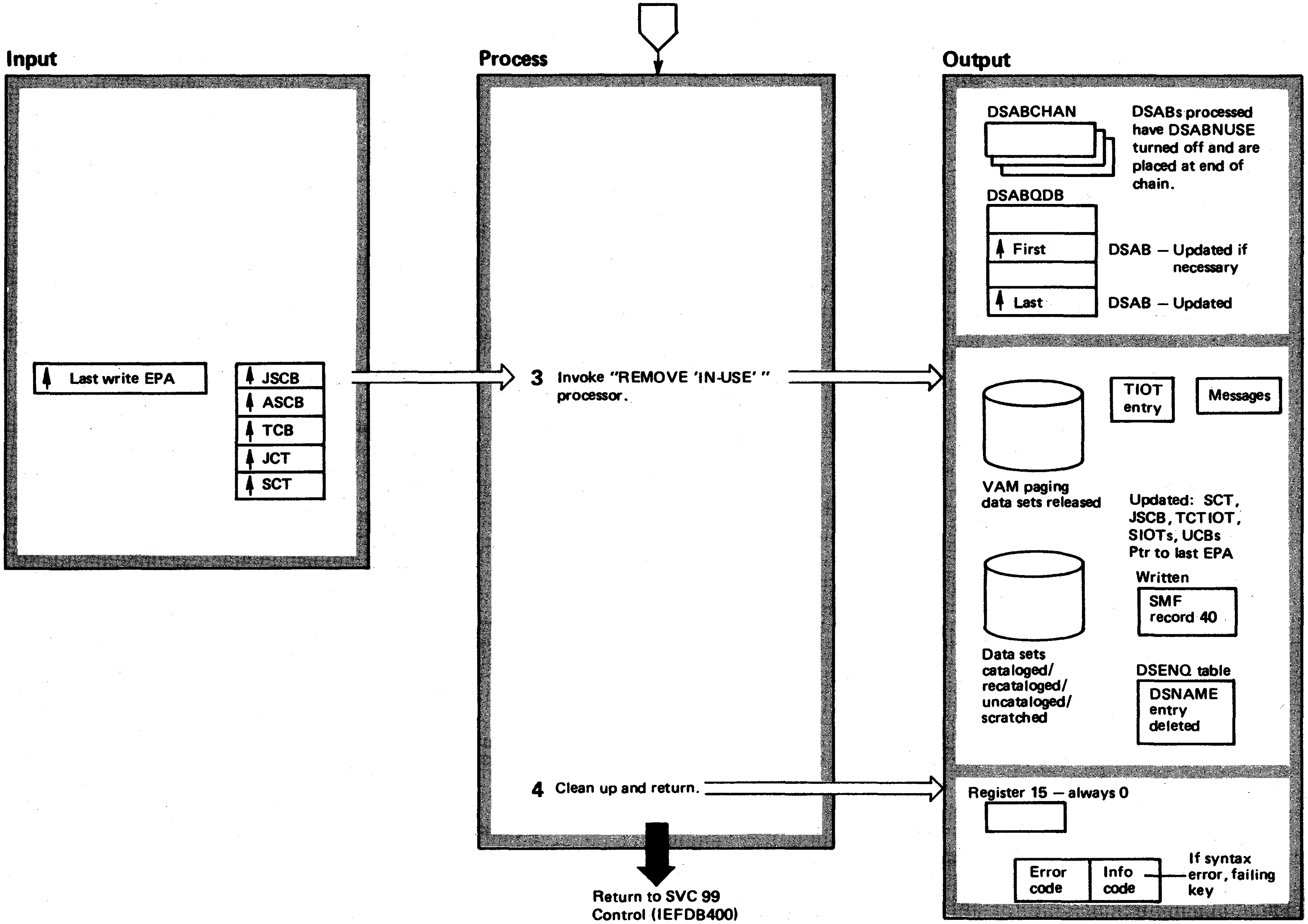
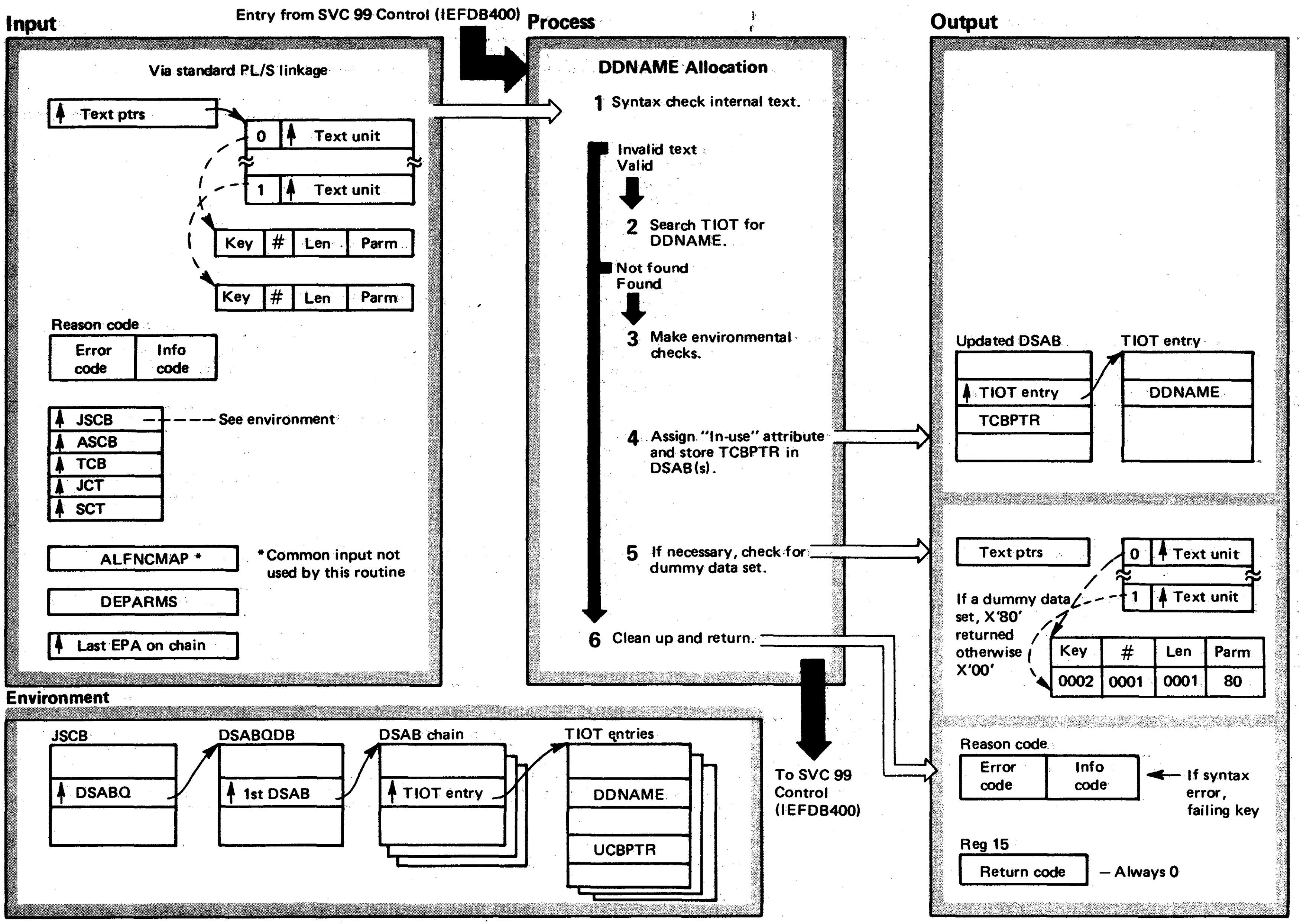


Diagram 14-26. IEFDB480 – Remove In-Use Attribute (Part 4 of 4)

Extended Description	Module	Label
3 The remove in-use processor is invoked for any DSABs found as a result of the search in step 2. It performs the following functions:		
a. When a DSAB pointed to in the input list is either OPEN or a member of a concatenated group, functions b and c are bypassed.	IEFDB481	
b. When a DSAB is not permanently allocated with a DISP of DELETE:		
● If the DSAB is for a non-&DSN its address will be placed in the list for unallocation and the unallocation count updated. Functions c and d are bypassed for this DSAB.		
● If the DSAB is for an &DSN VAM data set, VAM paging space is scratched and function d is processed.		VAMSCTCH
c. When a DSAB is either not permanently allocated, or has a disposition other than DELETE, a check is made to see if the conditional disposition was specified. If it was specified and is different from the normal disposition, it is removed from the SIOT and an EPA for the updated SIOT is placed on the write chain.		DISPPRCR
d. The in-use bit in the DSAB is turned off and the DSAB is moved to the end of the DSAB chain. (If the DSAB is concatenated, this processing is done for all members of the group.)		DSABCHNR
e. If there are DSABPTRs in the list for unallocation from function b above, the unallocation processor is invoked.	IEFDB4A1	
4 Return is made to the caller.		

Diagram 14-27. IEFDB490 – Ddname Allocation (Part 1 of 2)



DDNAME Allocation

1 Syntax check internal text.

2 Search TIOT for DDNAME.

3 Make environmental checks.

4 Assign "In-use" attribute and store TCBPTR in DSAB(s).

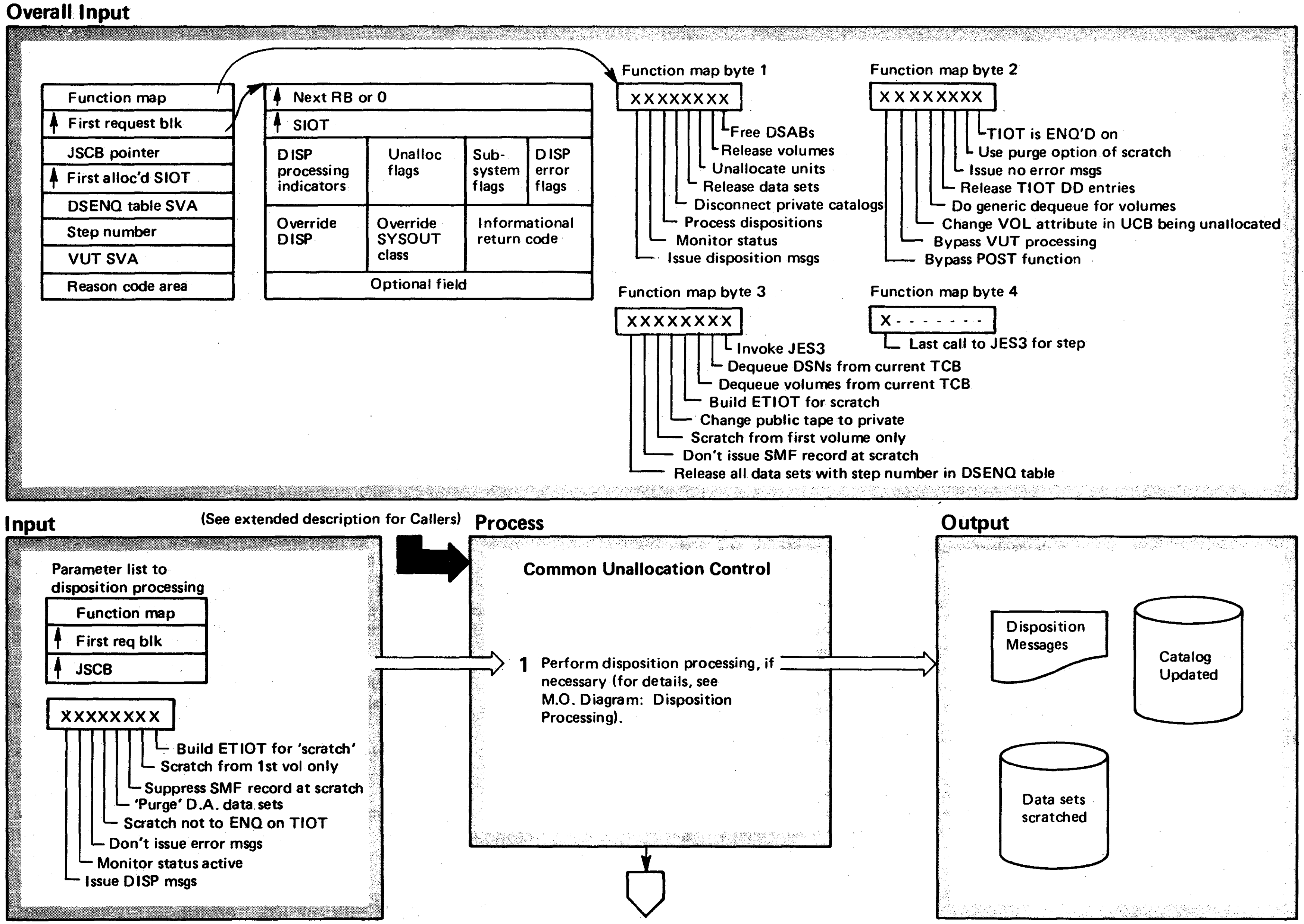
5 If necessary, check for dummy data set.

6 Clean up and return.

Diagram 14-27. IEFDB490 – Ddname Allocation (Part 2 of 2)

Extended Description	Module	Label
This module assigns the "In-Use" attribute to the resources associated with the specified ddname. This function can be specified using the verb code X'06' with one or both valid text keys. • Key X'0001' – This key is used to specify the ddname to be allocated. This key must be specified. • Key X'0002' – This key is used to request an indication that a DUMMY data set is being associated with the specified ddname.	IEFDB490	
1 A subroutine is invoked to check text for invalid or duplicate keys. If an error is found, step 6 is processed. If key '0001' is not present, an error code is set and step 6 is processed. If specified ddname is JOBLIB, STEPLIB, JOBCAT, or STEPCAT, an error code is set and step 6 is processed.	IEFDB490 IEFDB4FF	DDNLCCHK
2 A subroutine is invoked to search the DSAB chain for a TIOT entry with the specified ddname. If the ddname is not found, an error code is set and step 6 is processed.	IEFDB4FC	
3 The following checks are made on the DSAB entry: If entry is not in-use . . . . DSABNUSE=0 If entry is permanently allocated . . . DSABPALC=1 If entry is not convertible . . . . . DSABCONV=0 If entry is member of concatenated . . DSABCATM=1 If group entry is not OPEN . . . . DSABOPCT=0 If any of the above are not true, an error code is set and step 6 is processed.	IEFDB490	DDNLCCHK
4 The "In-Use" attribute is assigned. If member of a concatenated group, the in-use bit is turned on for each entry. The TCB address is stored in the DSAB(s).		
5 If text key '0002' is present, a check is made to determine if data set is a dummy. If it is, '80'X is returned in parameter field of key '0002'. Otherwise, '00'X is returned. **Note:** *All* members of a concatenated group must be dummy in order to have the dummy indicator returned.		
6 Return is made to the caller with register 15 set to zero.		

Diagram 14-28. IEFAB4A0 – Common Unallocation Control (Part 1 of 10)

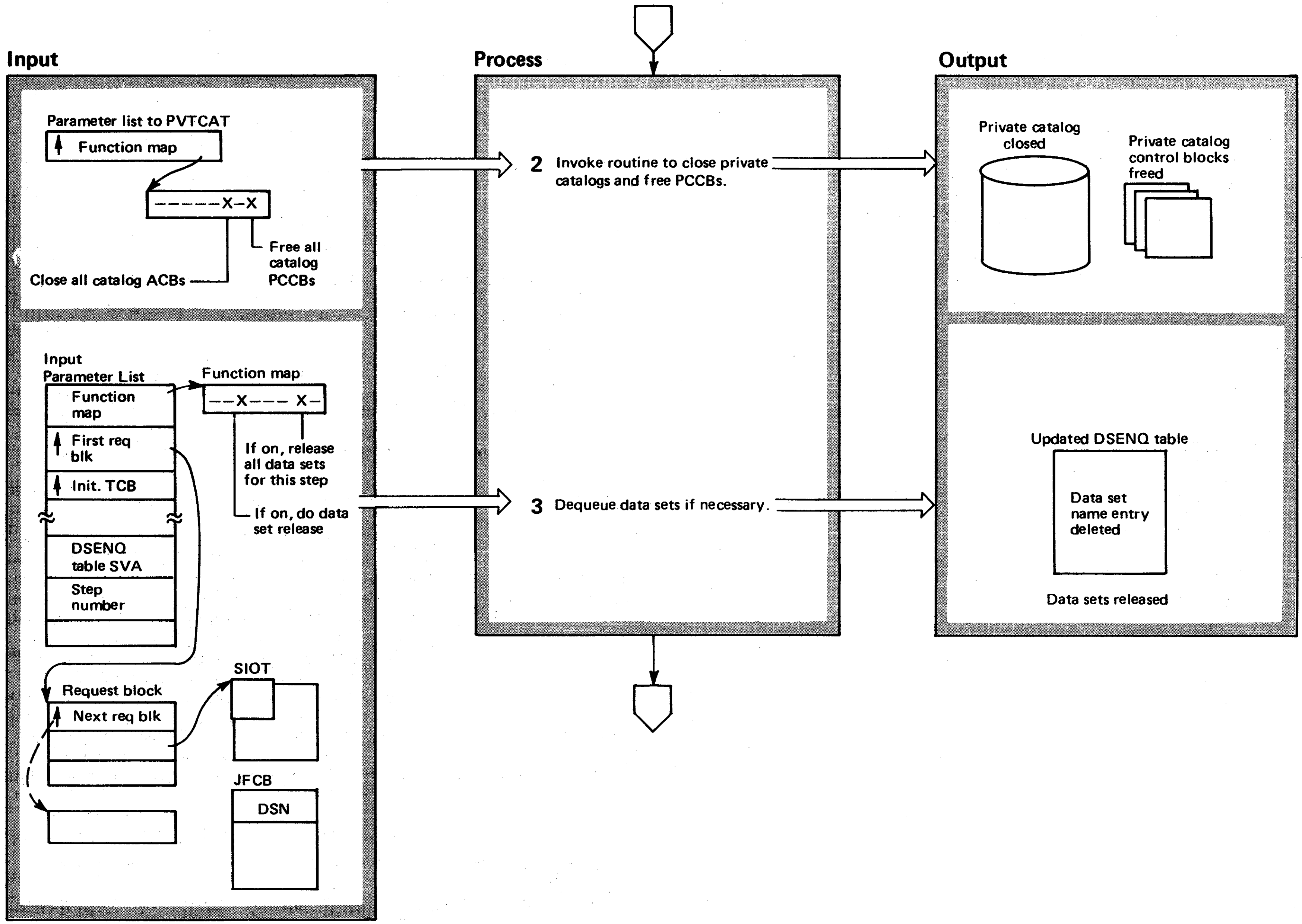


1 Perform disposition processing, if necessary (for details, see M.O. Diagram: Disposition Processing).

Diagram 14-28. IEFAB4A0 – Common Unallocation Control (Part 2 of 10)

Extended Description	Module	Label
The functions of the common unallocation routine are: • Data set disposition – Disposing of a data set as directed by the user through the DISP parameter in the job control language or as specified through the dynamic interface. • Data set release – Releasing the data set for use by other jobs, after the data set's last use by this job. • Unit unallocation – Unallocating the unit(s) to which a data set is allocated. • Volume release – Releasing volumes allocated to requests. **Note:** All four processes are not always performed for every request for unallocation. For example, a unit record device may not have a volume associated with it, or a data set may not be released if it is required in a later step of the job. The callers of this routine are: IEFBB414, IEFBB416, IEFAB477, IEFAB490, IEFAB4DE, IEFAB492, IEFDB4A1, IEFDB413, and IEFDB418.	IEFAB4A0	
1 Disposition processing is invoked unless the input function map indicates that it is to be bypassed. The subsystem interface is invoked for all subsystem data sets, i.e., SYSOUT and SYSIN data sets.	IEFAB4A2	

Diagram 14-28. IEFAB4A0 – Common Unallocation Control (Part 3 of 10)



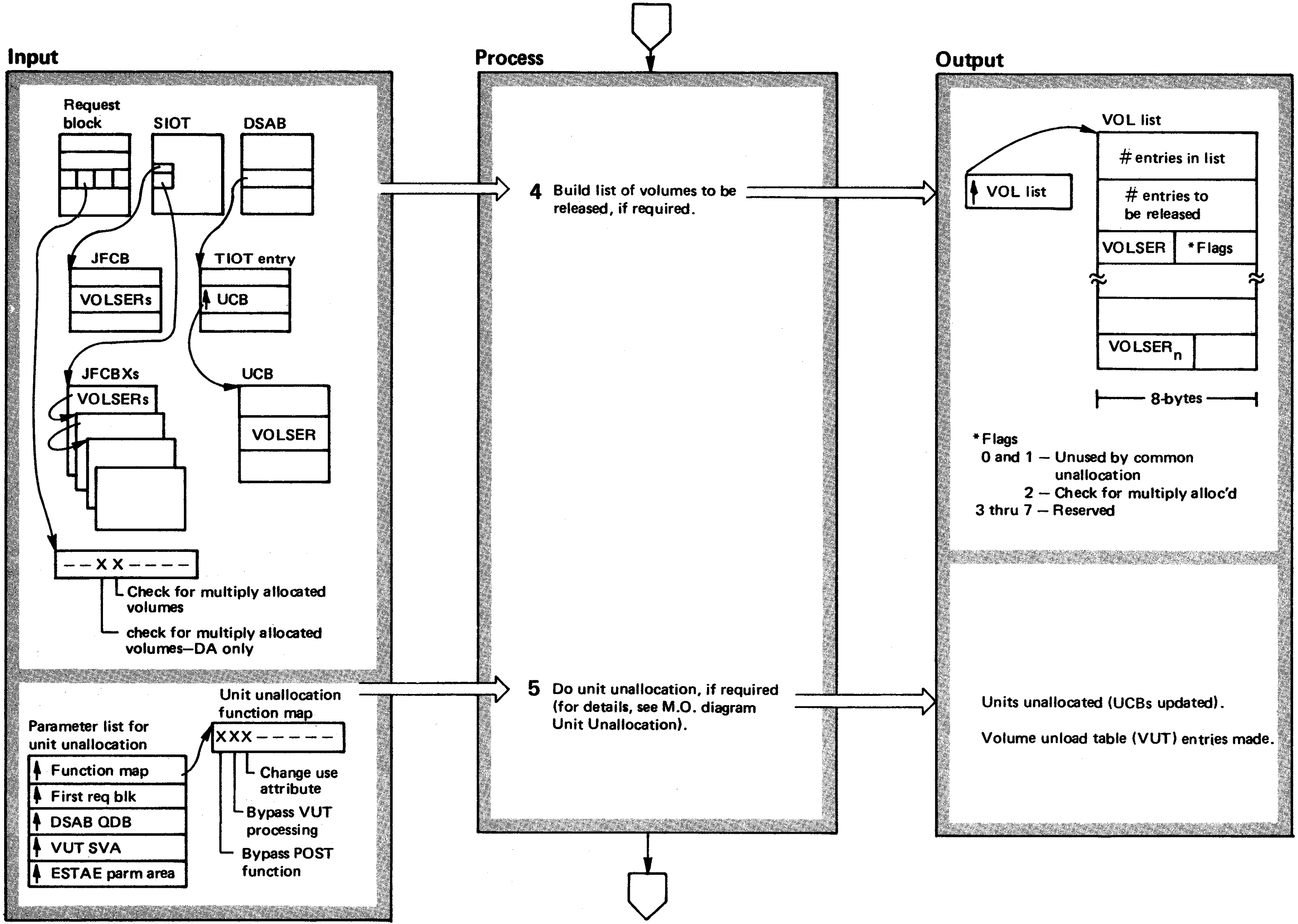
2 Invoke routine to close private catalogs and free PCCBs.

3 Dequeue data sets if necessary.

Diagram 14-28. IEFAB4A0 – Common Unallocation Control (Part 4 of 10)

Extended Description	**Module**	**Label**
2 A routine is invoked to close private catalogs and free PCCBs.	IEFAB4A0 IEFAB4F4	DSCPCATS
3 If all data sets for the step are being released, the entire DSENQ table is located via the SWA manager read/locate function, and the DEQ parameter list is built. Each data set used for the last time by the job in the current step is placed in the DEQ parameter list. If a subset of the current step's data sets (in the form of a data set name list) is to be released, the DSENQ table is read until all the data sets in the subset have been found or the DSENQ table has been exhausted.	IEFAB4A6	A4A6INIT READDSNQ DEQLBILD
When the DEQ parameter list is completed, a conditional DEQ is issued for all names in the DEQ parameter list. The input TCB pointer is used to direct the DEQ to the appropriate task. Data sets being dequeued are removed from the DSENQ table.	IEFAB4A6	A4A6CLNP DEQDSNS

Diagram 14-28. IEFAB4A0 – Common Unallocation Control (Part 5 of 10)



4 Build list of volumes to be released, if required.

5 Do unit unallocation, if required (for details, see M.O. diagram Unit Unallocation).

Units unallocated (UCBs updated).

Volume unload table (VUT) entries made.

Diagram 14-28. IEFAB4A0 – Common Unallocation Control (Part 6 of 10)

Extended Description	Module	Label
4 If volume release is to be done, a list of volumes to be released is built unless a generic DEQ is to be issued. The list contains all the VOLSERs in the JFCBs and JFCBXs for the request as well as those from the UCBs of the unit(s) allocated to the request.	IEFAB4A0	BVOLRLST GETVLSER
5 If unit unallocation is to be done, the following functions are performed: 1. Updating the unit control blocks (UCBs) associated with the requests being unallocated. 2. Creating/updating the volume unload table (VUT). 3. Posting generic allocation via the allocation Q-manager, for all device groups unallocated.	IEFAB4A4	

Diagram 14-28. IEFAB4A0 – Common Unallocation Control (Part 7 of 10)

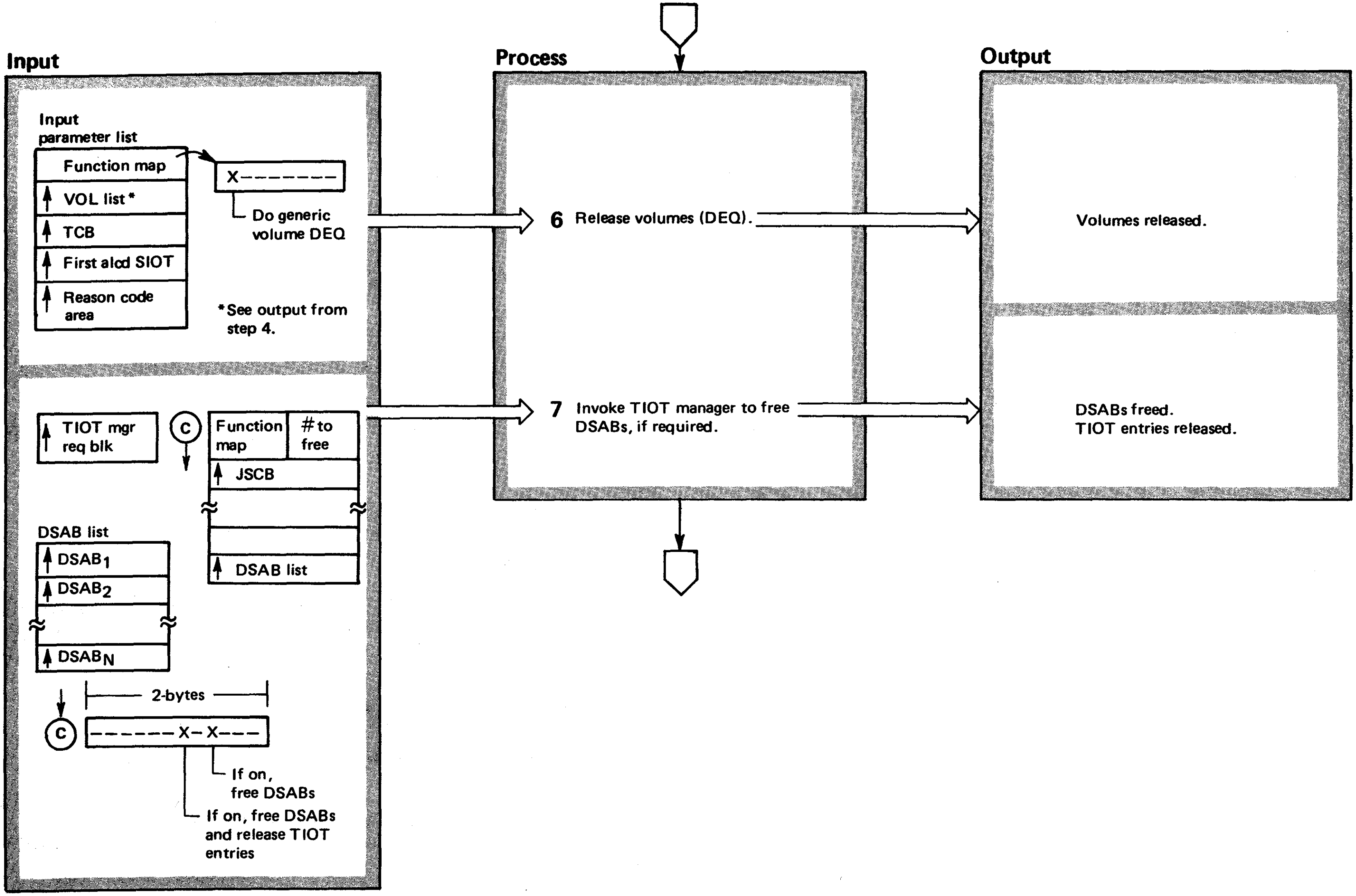
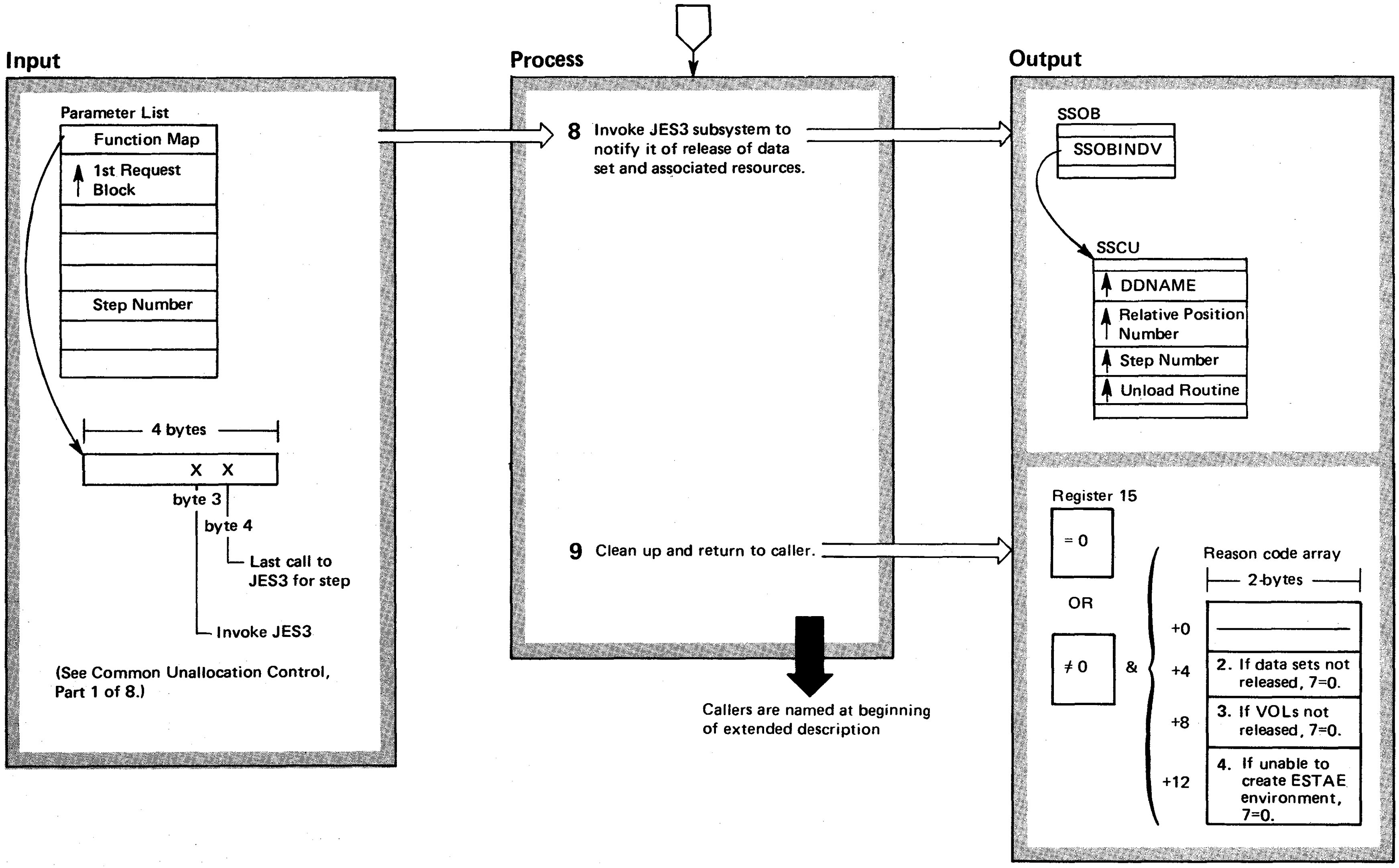


Diagram 14-28. IEFAB4A0 – Common Unallocation Control (Part 8 of 10)

Extended Description	Module	Label
6 If a generic DEQ is requested the DEQ is issued specifying the major name SYSZVOLs. Otherwise, the DEQ parameter list is built and a conditional DEQ is issued for the volumes that can be released (Volumes still in use by the job cannot be released). The major name used is SYSZVOLs and the minor names are the VOLSERs that can be released. The input TCB pointer is used to direct the DEQ to the owning task.	IEFAB4A8	
7 If DSABs are to be freed, a TIOT manager request block is built. If storage is available for the list of DSABs, the list is built, and the TIOT manager (IEFAB4FC) is invoked. The core for the DSABs is freed. The TIOT DD entries are released if required.	IEFAB4A0 IEFAB4FC	FDSABINT UNALCTIO

Diagram 14-28. IEFAB4A0 – Common Unallocation Control (Part 9 of 10)



8 Invoke JES3 subsystem to notify it of release of data set and associated resources.

9 Clean up and return to caller.

Callers are named at beginning of extended description

(See Common Unallocation Control, Part 1 of 8.)

Diagram 14-28. IEFAB4A0 – Common Unallocation Control (Part 10 of 10)

Extended Description	Module	Label
8 If the JES3 subsystem is to be invoked, build the subsystem interface and invoke JES3 once for each request block (except for DUMMY, TERM, QNAME, SYSIN, and SYSOUT). JES3 tables are updated to reflect the unallocation.	IEFAB4A0	EXITJES3
9 Control is returned to the caller with a return code in register 15.	IEFAB4A0	

Diagram 14-29. IEFAB4A2 – Disposition Processing (Part 1 of 3)

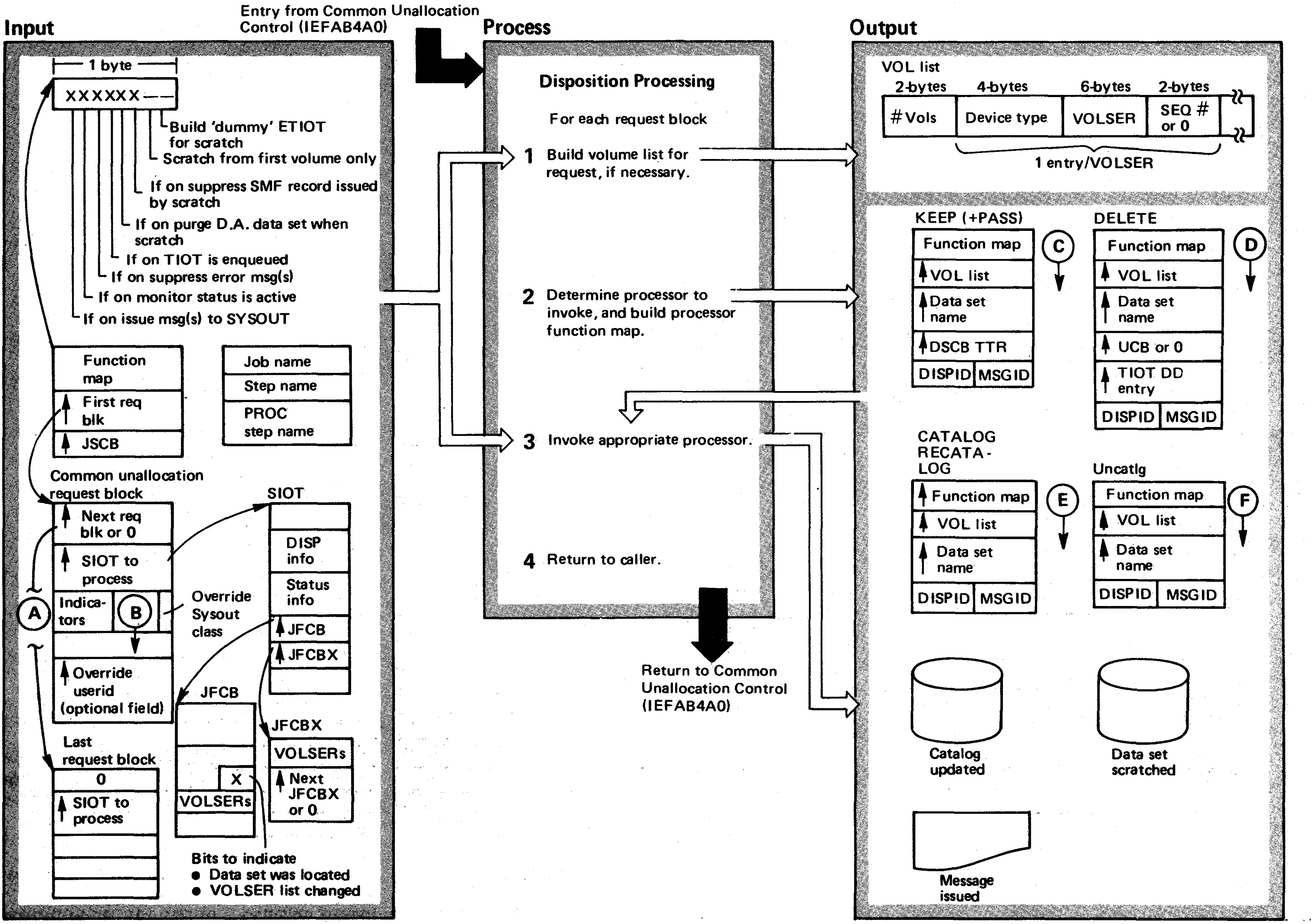


Diagram 14-29. IEFAB4A2 – Disposition Processing (Part 2 of 3)

Input

Indicators

Ⓐ

Byte 1

Bit 0 Bypass DISP proc
1 Use normal DISP
2 Use cond. DISP
3 Use override DISP
4 Use override userid for SYSOUT
5 Don't check UCB status
6 Reserved
7 Reserved

Byte 2 (SS interface flag)

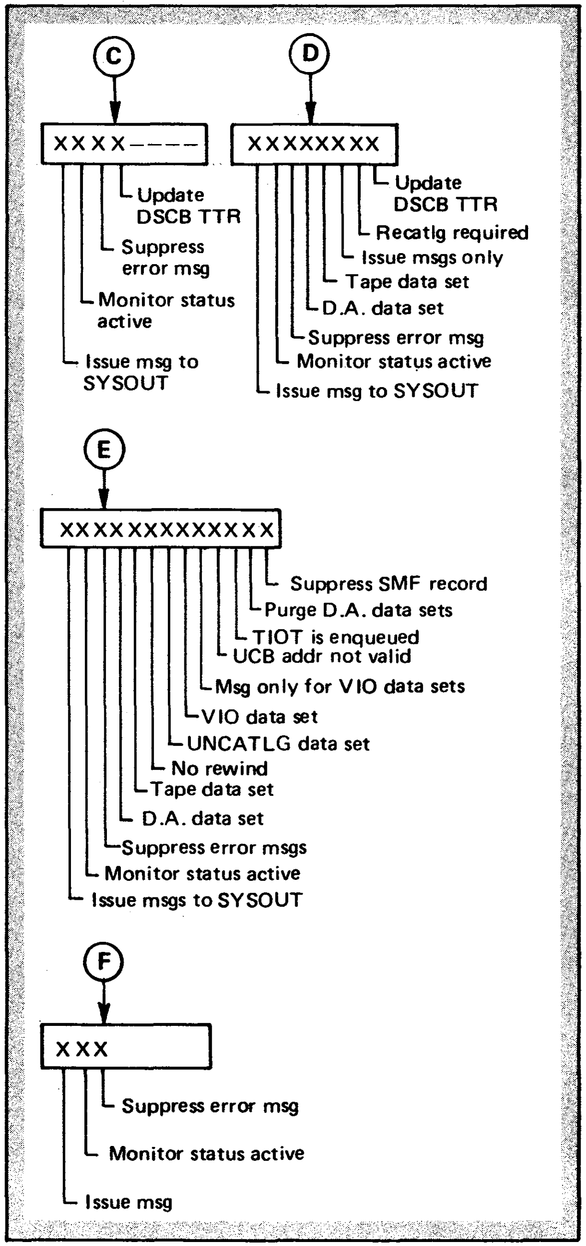
Bit 0 Delete at unallocation
1 Hold at unallocation
2 Dynamic unallocation
3 WAIT
4 Use override SYSOUT class
5-7 Reserved

Override DISP

Ⓑ

Bit 0-3 Reserved
4 KEEP
5 DELETE
6 CATLG
7 UNCATLG

Output



Extended Description

Module	Label
IEFAB4A2	

This routine controls the processing required to dispose of the data sets associated with the input requests.

Before any disposition processing can be done on a non-subsystem data set, a volume list must be successfully built.

If the volume list is successfully built, this routine will perform the appropriate process from among the following:

- Process KEEP or PASS disposition, and build and issue the appropriate message.
- Process DELETE disposition, and build and issue the appropriate message.
- Process CATLG disposition, and build and issue the appropriate message.
- Process UNCATLG disposition, and build and issue the appropriate message.
- Unallocate subsystem request, and build and issue appropriate message.

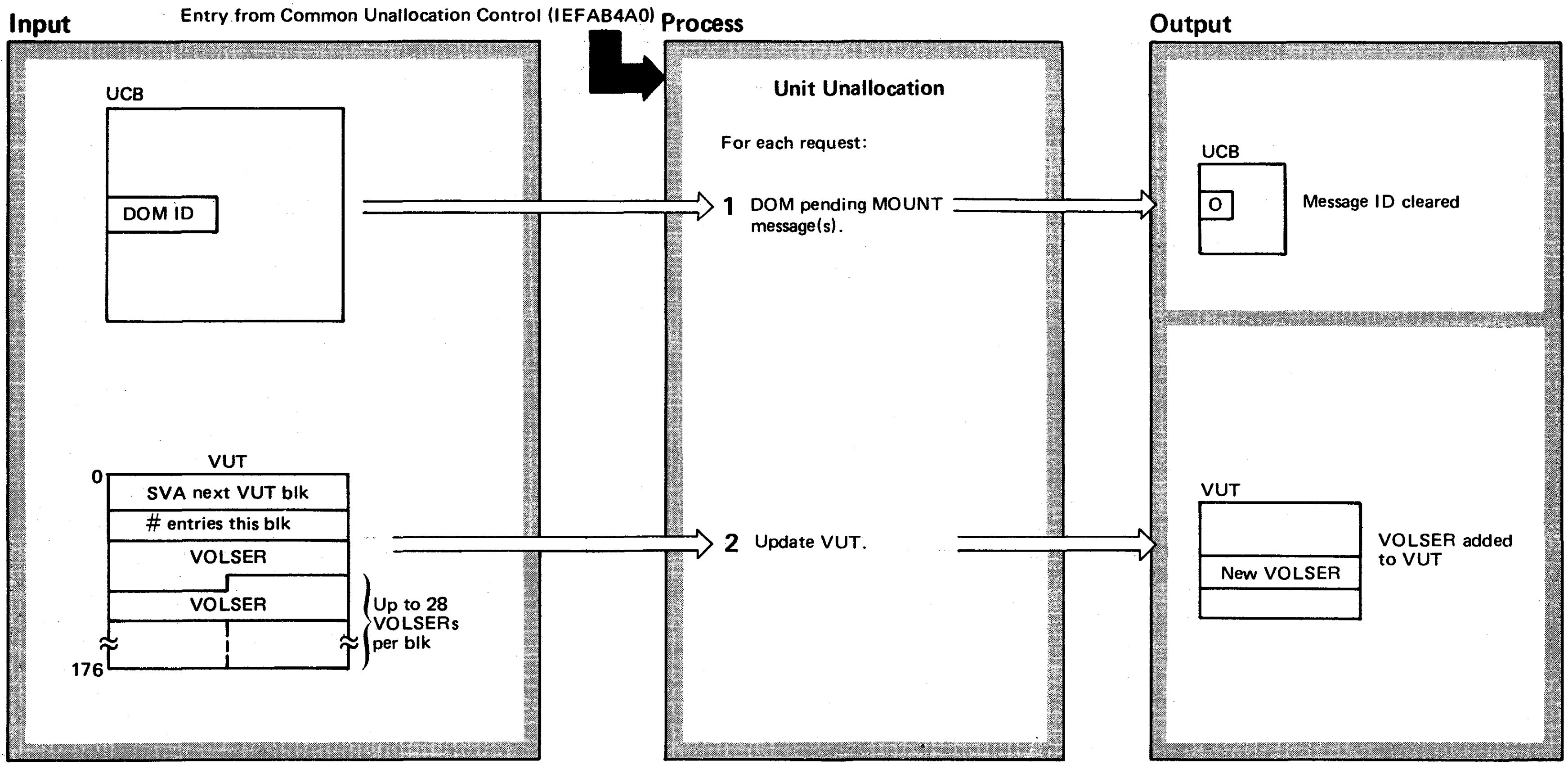
If the volume list is not successfully built the routine will:

- Process DELETE (tape only), KEEP, PASS, and UNCATLG.
- Build a disposition message, using JFCBs and JFCBXs via the alternate disposition message routine (IEFAB4B2).

Diagram 14-29. IEFAB4A2 – Disposition Processing (Part 3 of 3)

Extended Description	Module	Label
1 Unless the data set is a subsystem data set, a volume list is built containing the VOLSERs and device type of the current data set. The device type is obtained from a UCB allocated to the data set, unless the data set is to be recataloged, in which case the cataloged device type is used. The volume list is used as an interface to the various disposition processors and to the message processing routine, (IEFAB4B0). Function maps and other indicators are set as required for the disposition to be processed.	IEFAB4A2	BVOLLST
Note: For a DADSM scratch only data set, a dummy ETIOT entry will be created if it is indicated in the common unallocation function map.	IEFAB4A3	
2 Each request block is checked to determine what processor should be invoked. One of the following processes will be performed:	IEFAB4A2	PRCSDSP BDSPINT
3		
a. KEEP (and PASS) Identify message to issue (KEPT or PASSED). Invoke the message processor, IEFAB4B0. Update DSCB TTR in catalog, if required, using RECATLG interface in segment CTLGPRCR.		KEEPPRCR
b. DELETE disposition If the data set is direct access: Set up for DADSM SCRATCH. Invoke DADSM SCRATCH function. Whether the data set is direct access or not: If the data set was cataloged invoke UNCATLG. Invoke IEFAB4B0, to issue DELETE and/or NOT DELETED message(s). A data set may be partially deleted, in which case both messages will be issued. If tape, indicate 'REWIND NEEDED LATER' in the UCB(s).		DELPRCR
c. CATLG disposition If tape data set: Check density and update device type field of VOL LIST, to allow the widest possible range of devices, unless the data set is being recataloged, in which case use the cataloged device type. If necessary invoke catalog management to catalog (or recatalog) the data set. Determine the message to issue, if any, and call IEFAB4B0 to issue message.		CTLGPRCR
Note: If CTLGPRCR is being used only to update the DSCB TTR in the catalog, no message will be issued. If data sets were cataloged when allocated, the "cataloged" message will be issued, but catalog management will not be invoked.		
d. UNCATALOG disposition Indicate uncatalog in parameter list for CATLG management. Invoke CATLG management to uncatalog data set. Invoke message processor, IEFAB4B0, if disposition messages are to be issued.		UNCPRCR
e. Subsystem data sets Build the subsystem interface, using information from the data set's SIOT, JFCB, DSAB, and common unallocation request block. Issue IEFSSREQ macro to generate a CALL statement for the subsystem interface routine, SSREQ, so that the subsystem can dispose of the data set.		PRCSSSDS
Note: If no core is available for the volume list, a special message routine, IEFAB4B2, is invoked, which uses the JFCBs and JFCBXs to obtain volume information for the disposition message. If the disposition being processed is DELETE (DASD data sets only), CATALG, or RECATLG, a "NOT DISP 5" message is issued but no processor (i.e., Catalog Management or DADSM) is invoked.		NVLSTPRC
4 Control is returned to Common Unallocation Control (IEFAB4A0).		

Diagram 14-30. IEFAB4A4 – Unit Unallocation (Part 1 of 3)



For each request:

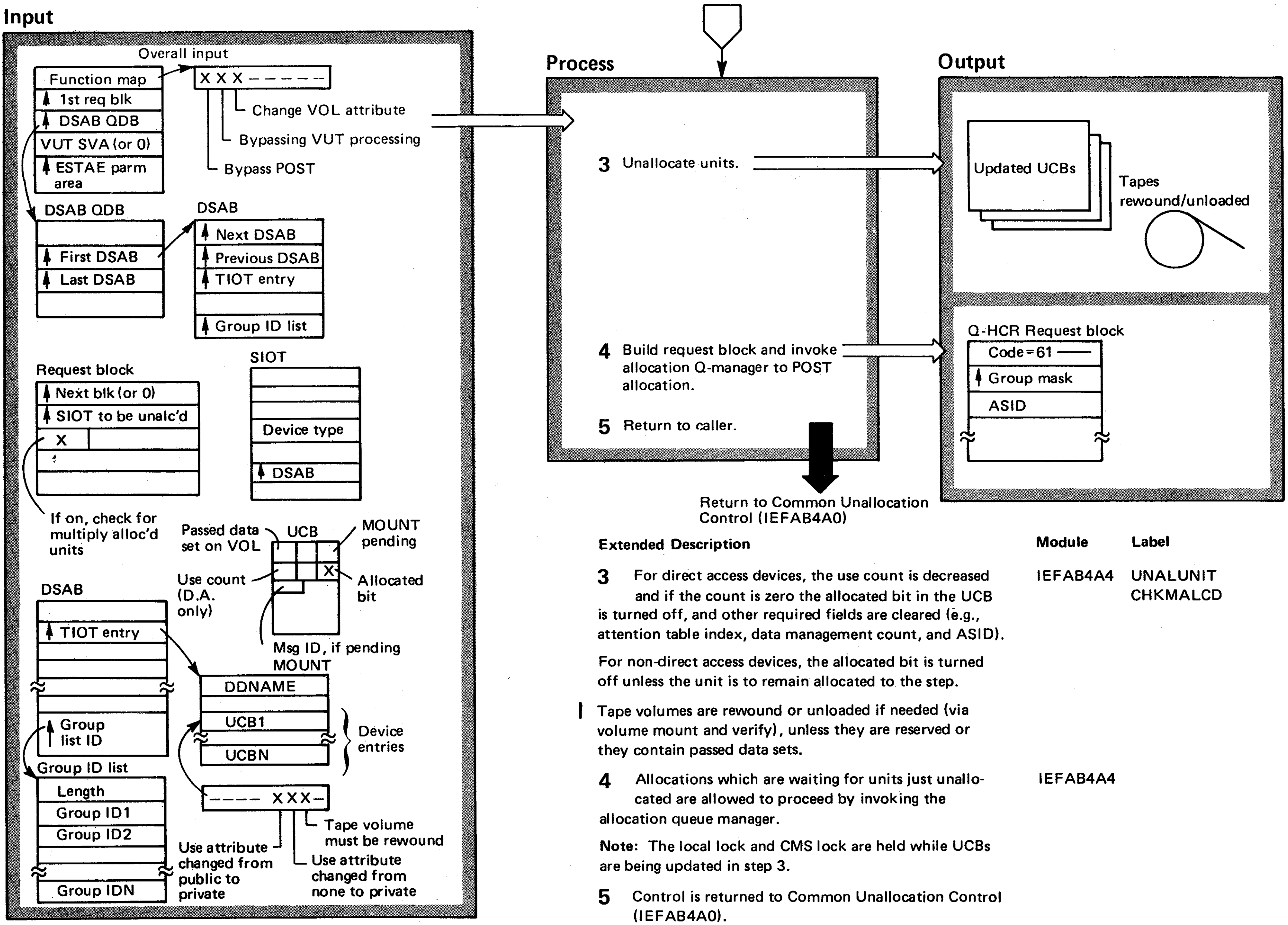
1 DOM pending MOUNT message(s).

2 Update VUT.

Diagram 14-30. IEFAB4A4 – Unit Unallocation (Part 2 of 3)

Extended Description	Module	Label
This routine unallocates the devices associated with each input request. Functions performed are: ● Unallocating units no longer needed – A direct access unit is not unallocated until the use count becomes zero. ● Rewinding tape volumes – Volumes on tape units being unallocated will be rewound or unloaded if necessary via the volume mount and verify function. ● Deleting outstanding messages – Any mount message pending for a tape data set is deleted. ● Identifying volumes to be unloaded – VOLSERs for private volumes, for public MSS volumes, and for volumes containing passed data sets are placed into the volume unload table (VUT).	IEFAB4A4	
1 If there is a mount message pending for tape, it is deleted using the DOM macro, and the DOM identifier and mount pending indicator in the UCB are cleared.	IEFAB4A4	DOMMSG
2 If VUT processing is requested, the volume serial of any volume which is private, public MSS, or contains a passed data set is placed in the VUT.	IEFAB4A4	CRVUTAB

Diagram 14-30. IEFAB4A4 – Unit Unallocation (Part 3 of 3)



Extended Description	Module	Label
3 For direct access devices, the use count is decreased and if the count is zero the allocated bit in the UCB is turned off, and other required fields are cleared (e.g., attention table index, data management count, and ASID).	IEFAB4A4	UNALUNIT CHKMALCD
For non-direct access devices, the allocated bit is turned off unless the unit is to remain allocated to the step.		
Tape volumes are rewound or unloaded if needed (via volume mount and verify), unless they are reserved or they contain passed data sets.		
4 Allocations which are waiting for units just unallocated are allowed to proceed by invoking the allocation queue manager.	IEFAB4A4	
Note: The local lock and CMS lock are held while UCBs are being updated in step 3.		
5 Control is returned to Common Unallocation Control (IEFAB4A0).		

System Management Facilities

System management facilities (SMF) routines collect data, provide for user-supplied data collection routines, and record the collected data in a data set. There are two cataloged direct access data sets, SYS1.MANX and SYS1.MANY, that are filled alternately. While the system records on one of the data sets, the other may be written out (or 'dumped').

Master scheduling task routines initialize the SMF routines. Then, the scheduling task uses the ATTACH macro instruction to establish the SMF task. The following components contain SMF data collection routines and exits for user-supplied data collection routines:

- The interpreter.
- The initiator/terminator.
- The command processor.
- The timer supervisor.
- The real and auxiliary storage manager.
- The real and virtual storage manager.
- The system resource manager.
- The JES2 subsystem.

As these various components record and collect data, they are building records. Eventually they use SVC 83 to transfer the records to an SMF buffer. The SVC 83 routine also writes the records to the SMF data set and, as needed, initializes and switches data sets.

The following list contains the types of record information that may be recorded on the SMF data sets:

- Records describing the resources used by tasks the system processes.
- Records describing changes in status of the system, such as changes brought about by use of VARY and HALT commands.
- Records describing the usage of data sets and volumes by users.
- Records describing the resources used by a TSO user.

Some components collect a single data item and accumulate the value of the item in an SMF control block. Some components format a record of various data items and transfer the record to a central SMF buffer. Other components provide control program interfaces to installation-supplied exit routines for data-collection functions.

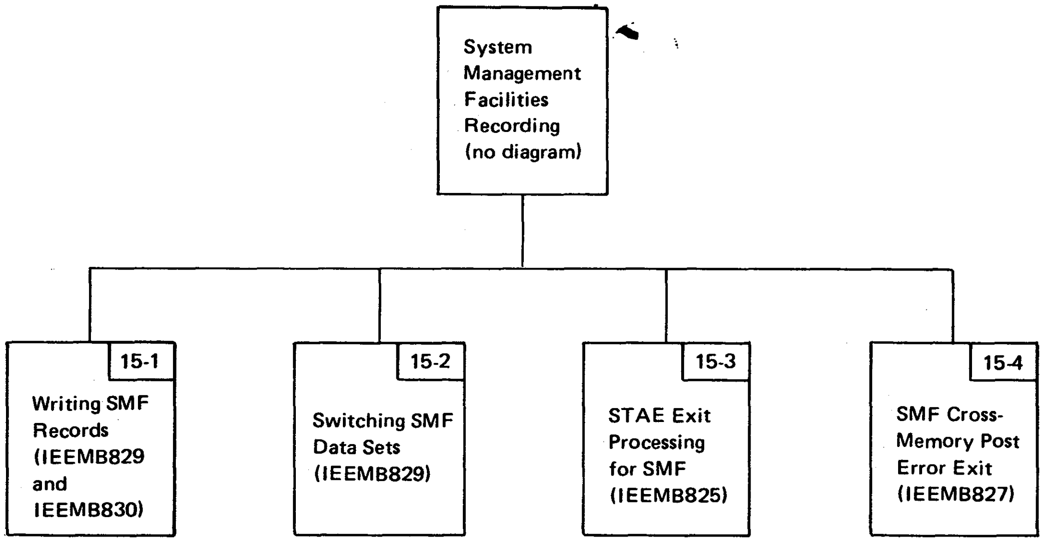
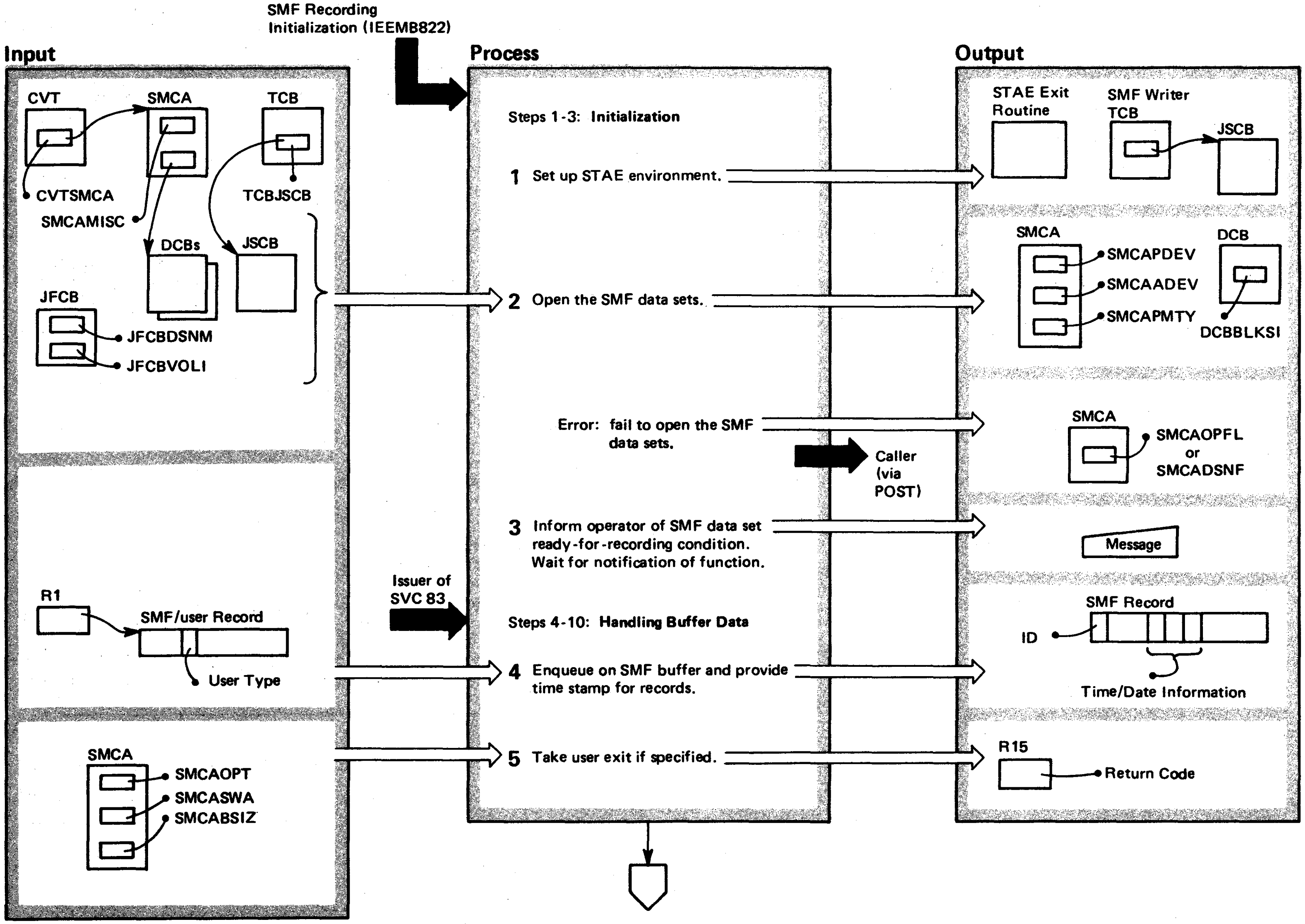


Figure 2-29. System Management Facilities (SMF) Recording: Visual Contents

Diagram 15-1. Writing SMF Records (IEEMB829 and IEEMB830) (Part 1 of 4)



Steps 1-3: **Initialization**

1 Set up STAE environment.

2 Open the SMF data sets.

Error: fail to open the SMF data sets.

3 Inform operator of SMF data set ready-for-recording condition. Wait for notification of function.

Steps 4-10: **Handling Buffer Data**

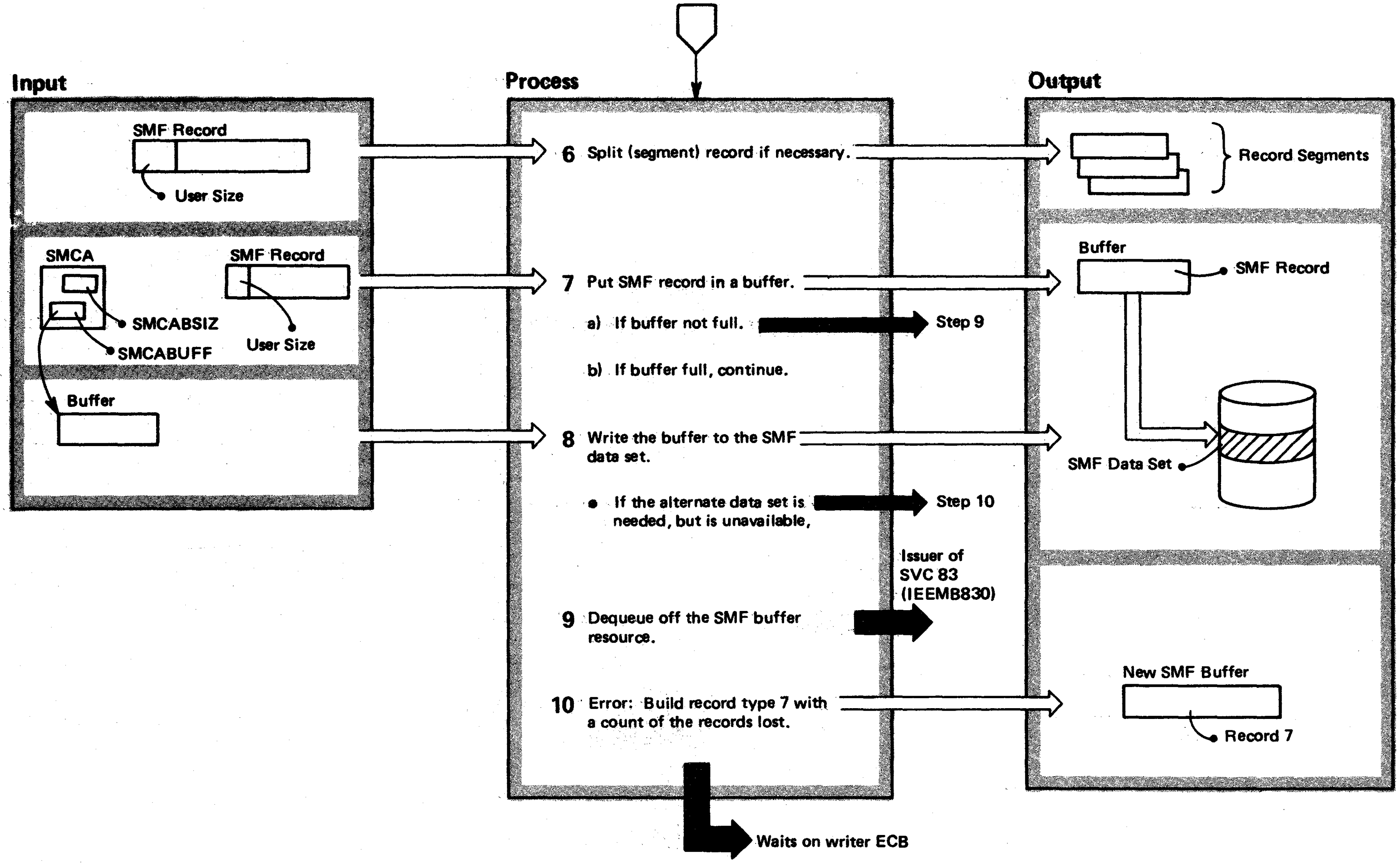
4 Enqueue on SMF buffer and provide time stamp for records.

5 Take user exit if specified.

Diagram 15-1. Writing SMF Records (IEEMB829 and IEEMB830) (Part 2 of 4)

Extended Description	Module	Label
This routine places SMF records in a buffer and writes the buffer to the recording SMF data set.		
Initialization		
1 This environment protects the SMF record writing function by handling abnormal end situations.	IEEMB829	WTRINIT
2 The data sets SYS1.MANX and SYS1.MANY are opened (via OPENJ) and one is selected for recording data.		INITOPEN
3 The routine uses a WTO macro instruction to inform the operator that SMF is recording.		WRITEMSG
Note: Modules IEEMB829 and IEEMB830 communicate with each other via the cross-memory posting facility. After the initialization phase, module IEEMB829 enters the wait state, having been set dispatchable (via the STATUS SVC) by module IEEMB822. When module IEEMB829 completes its processing, it issues a POST macro instruction for module IEEMB830, then waits for the next function to process. If an error occurs in the cross-memory posting process, module IEEMB827 gets control to perform cleanup processing. This eventually leads to a termination of the SMF function.		
Handling Buffer Data		
4 This procedure serializes the SMF record processing resource. (A time stamp is omitted from records 4, 5, 34, 35, and 128-255.)	IEEMB830	IEEMB830
Note: For a record for a HALT (EOD) situation, the active SMF data set is closed and a switch is made to the alternate data set.		
5 User exits may be specified at initialization. If exits are specified (via field SMCAOPT), the exit IEFU83 is taken.		USEREXT

Diagram 15-1. Writing SMF Records (IEEMB829 and IEEMB830) (Part 3 of 4)



6 Split (segment) record if necessary.

7 Put SMF record in a buffer.

a) If buffer not full. → Step 9

b) If buffer full, continue.

8 Write the buffer to the SMF data set.

- If the alternate data set is needed, but is unavailable, → Step 10

9 Dequeue off the SMF buffer resource. → Issuer of SVC 83 (IEEMB830)

10 Error: Build record type 7 with a count of the records lost.

Waits on writer ECB

Diagram 15-1. Writing SMF Records (IEEMB829 and IEEMB830) (Part 4 of 4)

Extended Description	Module	Label
Handling Buffer Data (Continued)		
6 Record splitting occurs if the record size is greater than the buffer size. Before splitting a record, the following steps occur:	IEEMB830	SPLITREC
• The SMF buffer is emptied.		EMPTYBUF
• The routine determines whether the number of segments required to hold the record will fit into the space remaining in the actively-recording data set.	IEEMB829	SPACECHK
• The routine switches to the alternate SMF data set if the current data set lacks the necessary space for the record. (Record truncating may be necessary if a segmented — split — record is too large for an empty SMF data set.) (See the diagram Switching SMF Data Sets for data set switching.)		DSWITCH
7 This assumes that the record size is less than or equal to the empty buffer size.	IEEMB830	RECPROC
8 The routine uses a BSAM write mode to transfer the buffer contents. Before writing, the following processing occurs:	IEEMB829	BWRITER
• A check is made for available space on the currently-recording SMF data set.		RECWRITE
• If the current data set lacks space for the record waiting to be written, data set switching occurs. (See the diagram Switching SMF Data Sets.)		DSSWITCH
• The routine closes the former (currently active) 'primary' data set and issues a message to have the operator dump the full data set.		SMFOPEN
• The routine opens the new 'primary' data set.		SMFOPEN
The writing of the buffer occurs if either of the following conditions are true:		
• The current size of the record waiting to be written is greater than the remaining buffer space. (That is, the buffer must be emptied before it can hold another record.)		
• A HALT command is being processed and the buffer contains at least one record.		
• After writing the buffer to the recording data set, a TCLOSE macro instruction is used to close the active data set for the writing of an end-of-file label that will preserve any already accumulated data in case a subsequent system failure occurs. The next buffer written to the same data set will overlay the label and another TCLOSE action is performed.		
9 This procedure removes the serialization protection on the buffer resource. (That is, the routine no longer controls the use of the resource.)	IEEMB830	
10 Recovery will occur when an SMF data set becomes available after being dumped and successfully re-opened.		

Diagram 15-2. Switching SMF Data Sets (IEEMB829) (Part 1 of 4)

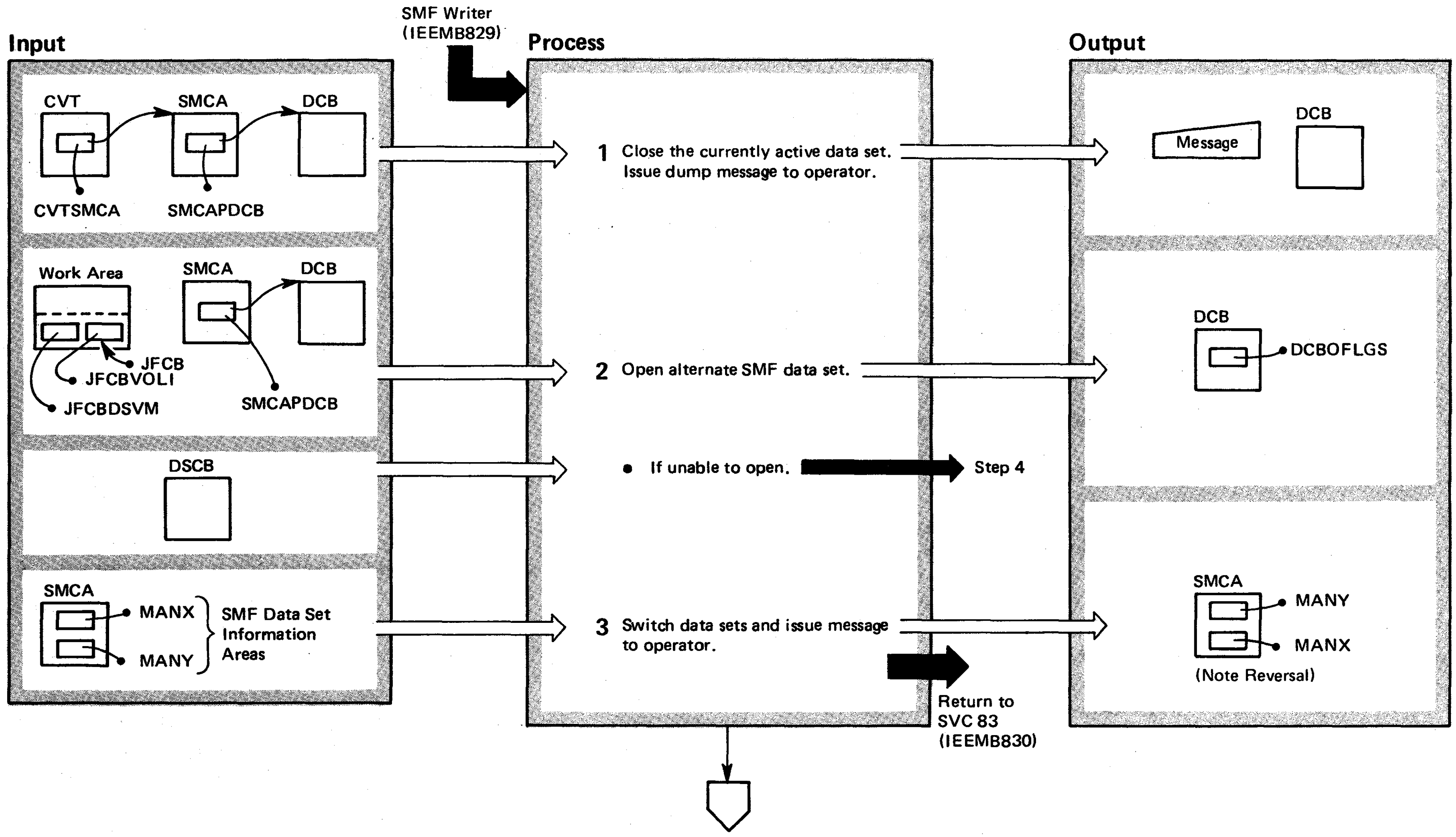
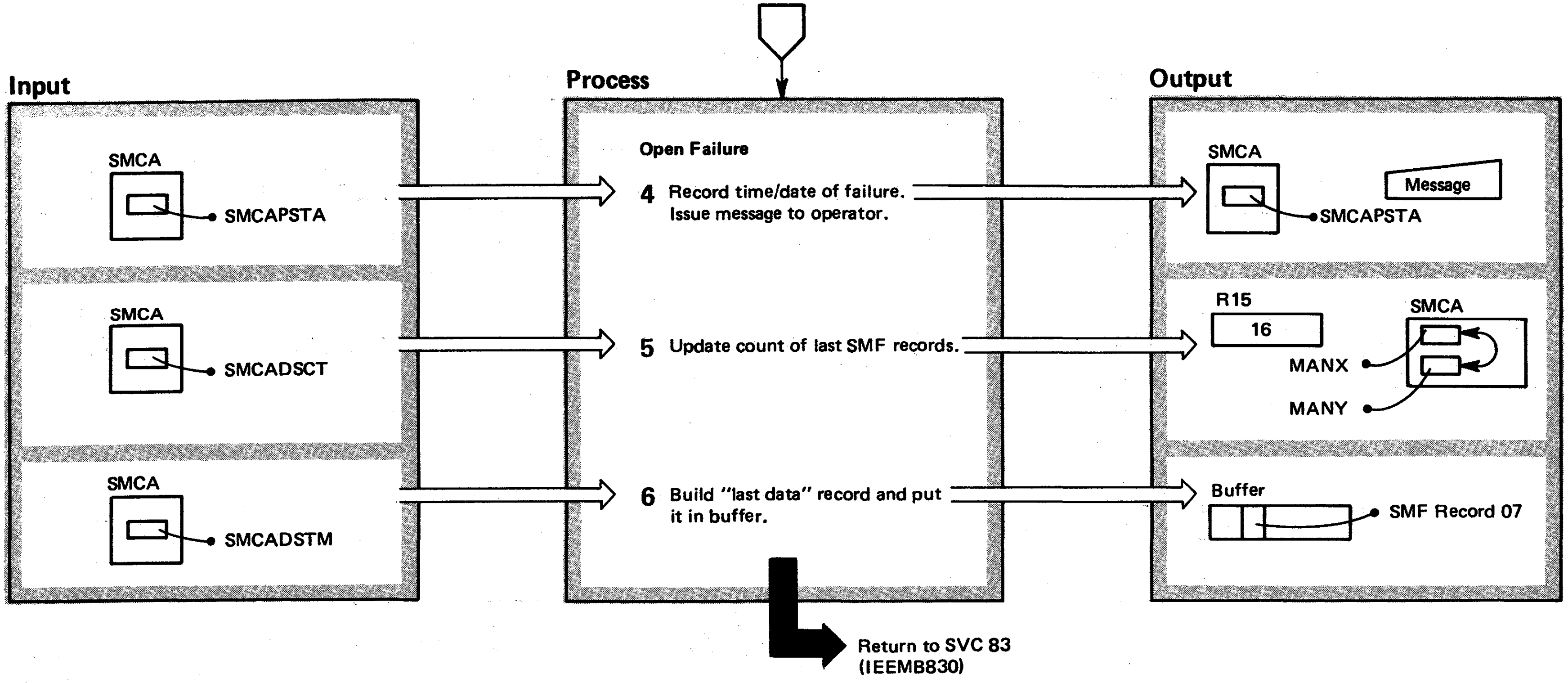


Diagram 15-2. Switching SMF Data Sets (IEEMB829) (Part 2 of 4)

Extended Description	Module	Label
This routine switches recording data sets for SMF records. If switching is impossible, lost records are counted.		
1 The WTO message requests the operator to dump the full data set.	IEEMB829	DDSWITCH
2 The routine checks the DSCB to see if an alternate data set is available for recording. A data set must be empty before it is opened. The routine uses an OPENJ macro instruction to open the currently non-recording SMF data set.		SMFOPEN
3 The currently-recording data set is the one listed first in the SMCA. The indicated areas (MANX and MANY) in the SMCA contain information about the active and alternate data sets. A WTO message informs the operator that SMF recording now occurs on the alternate data set.		WRITEMSG

Diagram 15-2. Switching SMF Data Sets (IEEMB829) (Part 3 of 4)



Open Failure

4 Record time/date of failure. Issue message to operator.

5 Update count of last SMF records.

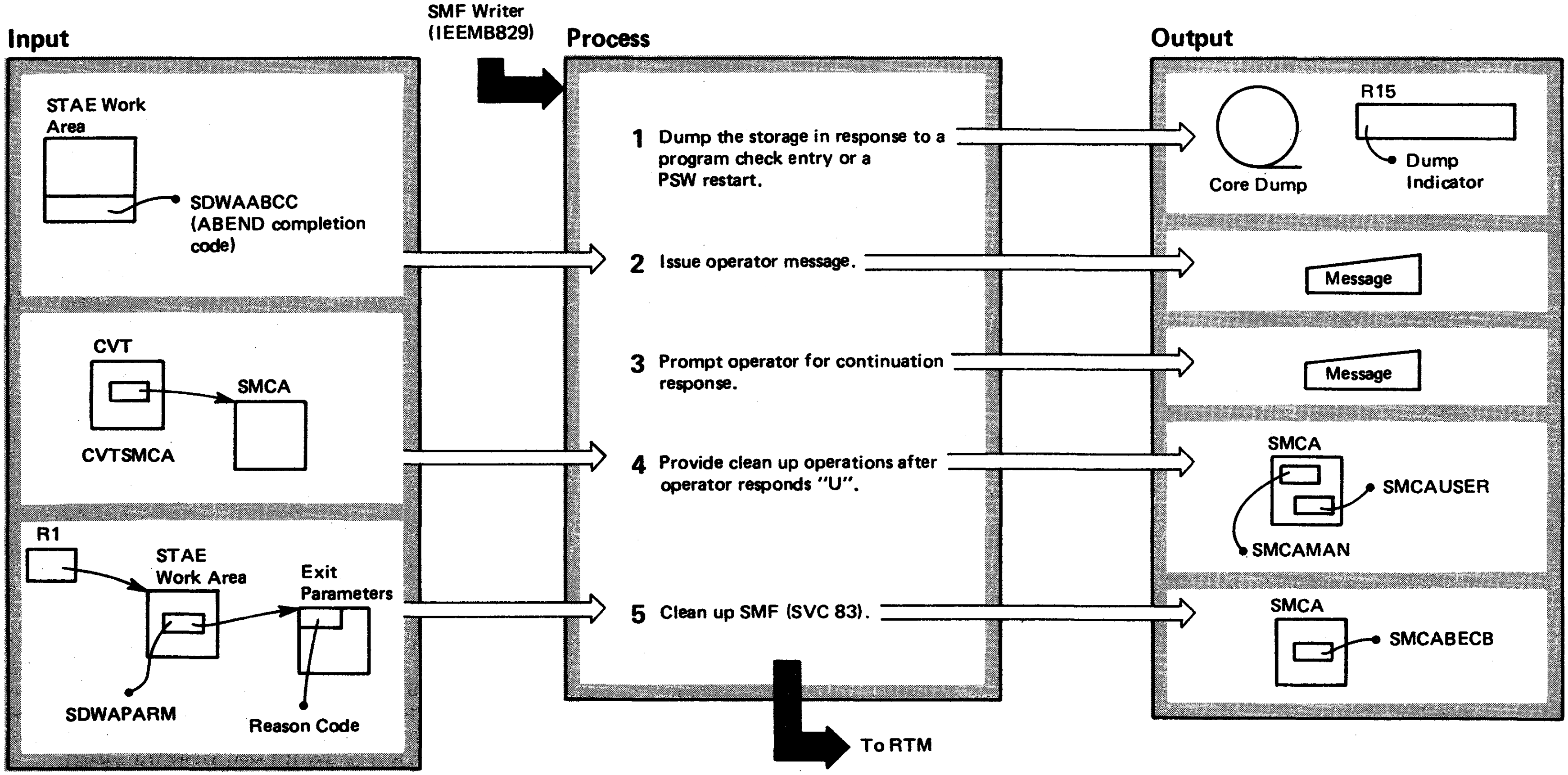
6 Build "last data" record and put it in buffer.

Return to SVC 83 (IEEMB830)

Diagram 15-2. Switching SMF Data Sets (IEEMB829) (Part 4 of 4)

Extended Description	Module	Label
4 If the alternate data set contains data, switching is prohibited. A WTO message informs the operator and an indicator is set in the field SMCAPSTA.	IEEMB829	WRITEMSG
5 For each subsequent attempt, beyond the first, to write an SMF record, the count of lost records is updated. A return code of 16 indicates the failure to write a record. The data sets are then switched for the next attempt.		RECOVERY
6 SMF data set recovery occurs when a data set is dumped and successfully opened. At that time, recording on the data set is possible. The "lost data" record includes the count of SMF records lost during the non-recording time, and it resides in a new SMF buffer.		RECOVERY

Diagram 15-3. STAE Exit Processing for SMF (IEEMB825) (Part 1 of 2)



1 Dump the storage in response to a program check entry or a PSW restart.

2 Issue operator message.

3 Prompt operator for continuation response.

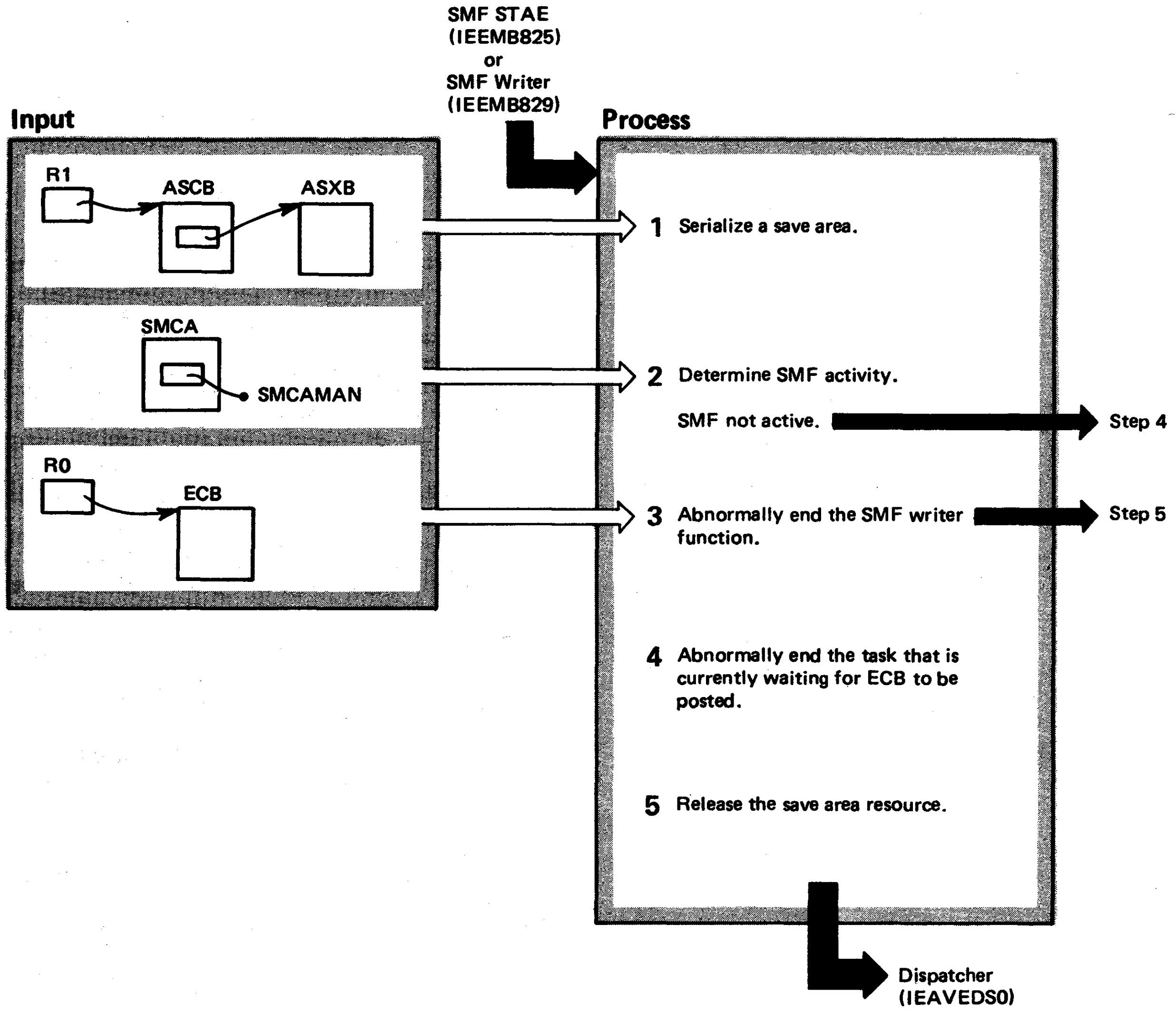
4 Provide clean up operations after operator responds "U".

5 Clean up SMF (SVC 83).

Diagram 15-3. STAE Exit Processing for SMF (IEEMB825) (Part 2 of 2)

Extended Description	**Module**	**Label**
This routine provides clean up processing for abnormal end situations that occur during SMF writer processing. (The same routine handles ABEND occurrences during SMF initialization – see the publication, *OS/VS2 System Initialization Logic.*		
1 The SVC Dump service (via the SDUMP macro instruction) provides this function. Register 15 = 0 indicates a successful dump.	IEEMB825	
2 A WTO macro instruction issued to the operator indicates the ABEND completion code, the existence of a termination condition, and an indication of a successful dump (if R 15 = 0).		
3 The message (step 2) prompts the operator to respond with a "U" or a re-IPL. Until he responds with a "U", other job terminations are suspended. A non-"U" response results in a reprompting.		
4 The routine frees the SMF buffers and frees the DCB storage. The recording flags in the SMCA are also set to zero.	IEEMB825	
5 If the reason code (see step 2) is 0, the STAE routine is operating for the SMF writer and the SMF SVC (83) routine must be cleaned up. The routine issues a conditional ENQ macro instruction for the SMF buffer resource cleanup. The writer routine must then post the SVC routine before the SVC routine can continue. (The flag SMCABECB carries the posted indication.) If the ENQ is not obtained, the SVC routine is not waiting for the SMF writer routine to complete the function that failed and the buffer resource cleanup is unnecessary.		

Diagram 15-4. SMF Cross-Memory POST Error Exit (IEEMB827) (Part 1 of 2)



1 Serialize a save area.

2 Determine SMF activity.

SMF not active.

3 Abnormally end the SMF writer function.

4 Abnormally end the task that is currently waiting for ECB to be posted.

5 Release the save area resource.

Diagram 15-4. SMF Cross-Memory POST Error Exit (IEEMB827) (Part 2 of 2)

Extended Description	Module	Label
This routine handles the abnormal ending of the SMF cross-memory posting function.		
1 Routine gets a local lock to serialize the resource in the ASCB extension. This routine was entered to handle a failure in the SMF Writer (module IEEMB829).	IEEMB827	
2 If SMF recording is inactive, the failure occurred when the SMF STAE routine (IEEMB825) posted the SVC 83 routine after an abnormal end situation occurred in the SMF writer routine. If SMF recording is active, the failure occurred either when the SMF Recording (SVC 83) routine posted the SMF writer routine or when the writer routine posted the SVC 83 routine.		
3 The routine issues a CALLRTM macro instruction. This causes the SMF STAE routine to receive control to stop the system's SMF function.		
4 This routine was entered from the SMF STAE routine (IEEMB825). The routine issues a CALLRTM macro instruction to set up an ABEND for the current task that is waiting in the SMF recording (SVC 83) routine.		
5 The routine releases the local lock.		

System Log

The system log provides a record of system activity. The log function handles the logging of messages from a system operator, from user routines, and from the operating system routines. All MVS systems contain the log function. Previous systems had to request the log function as a SYSGEN option. Now, after the job entry subsystem (JES2) is active, module IEEVWAIT (the master scheduler wait routine) automatically attaches the system log task. The system procedure library (SYS1.LINKLIB) contains the process defining the initialization of the system log. After the system log data set is open, it may receive messages resulting from a WTL macro instruction or a LOG command. When a given data set is full (based on a limit value previously supplied in the parameter library member IEASYSxx), a new log data set is obtained.

For MVS systems, the log task operates in the master scheduler's region and has its own job identification. To replace the system data sets (SYS1.LOGX and SYS1.LOGY) used in previous systems, the log task dynamically produces internally-created data sets which the job entry subsystem (JES2) and JES3 processes as output data sets.

For MVS, a WRITELOG command with the START operand causes reactivation of the system log should the log become inactive due to system failure or the issuance of a WRITELOG CLOSE command. A re-IPL procedure is unnecessary.

In order to close a system log that is also the system hardcopy device, it is necessary for an operator to issue a VARY command to re-define the hardcopy device. Then the operator may close the log by issuing a WRITELOG CLOSE command.

Users communicate with the system log through the use of the WTL (Write-to-Log) macro instruction and the commands LOG and WRITELOG. The WTL macro instruction (which results in an SVC 36 instruction) is used to schedule the entering of information into the log. To enter information into the system log from an operator's console, a user issues a LOG command. By using a WRITELOG command, a user may request the closing of the currently recording log data set (with subsequent queueing of the data set to a SYSOUT writer). If the system log is acting as the hardcopy log, write-to-operator/reply (WTO/WTOR) messages and the system and operator LOG and WRITELOG commands with their responses may be entered on the system log. Depending on the source of the message or command, either the communications task or the command scheduler (SVC 34) routines convert the message/command into a Write-to-Log macro instruction in order to enter it into the system log.

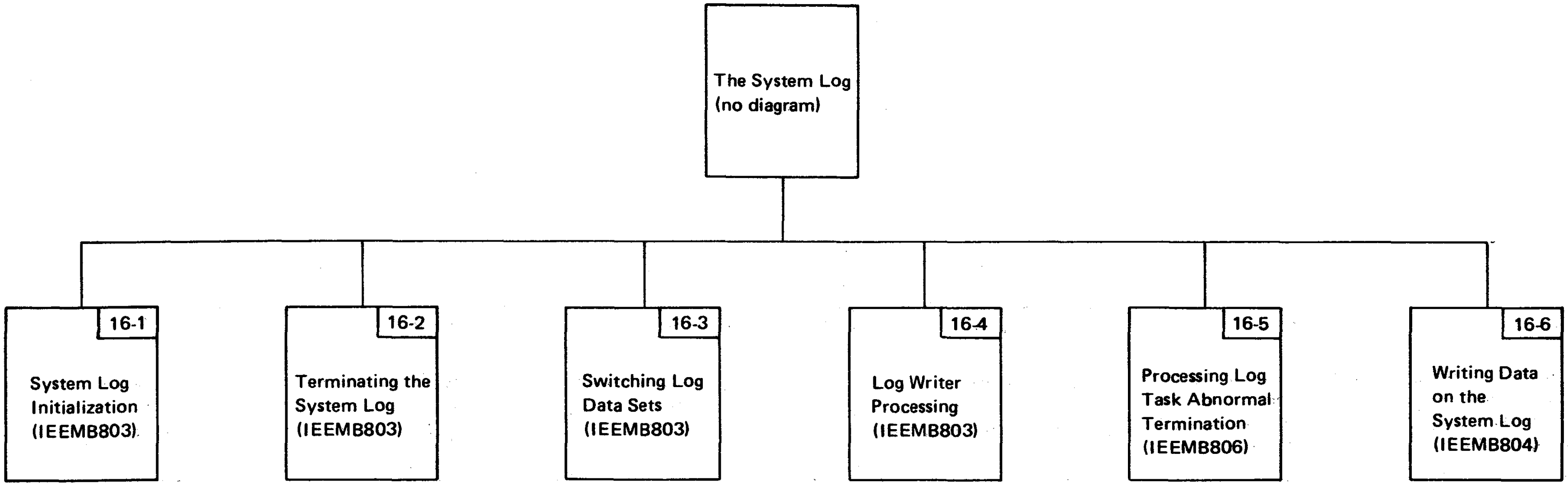
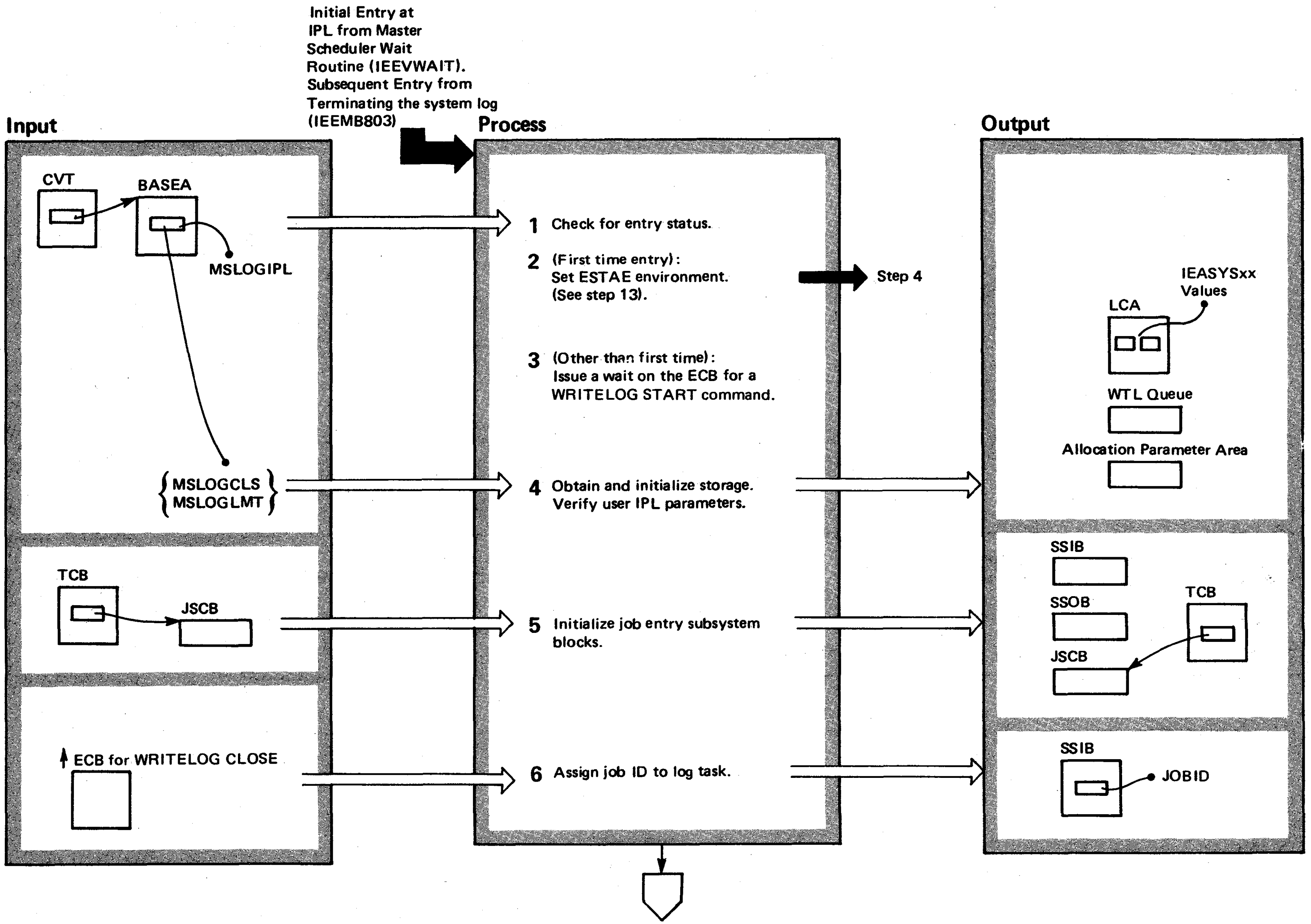


Figure 2-30. System Log Visual Contents

Diagram 16-1. System Log Initialization (IEEMB803) (Part 1 of 4)



1 Check for entry status.

2 (First time entry): Set ESTAE environment. (See step 13).

3 (Other than first time): Issue a wait on the ECB for a WRITELOG START command.

4 Obtain and initialize storage. Verify user IPL parameters.

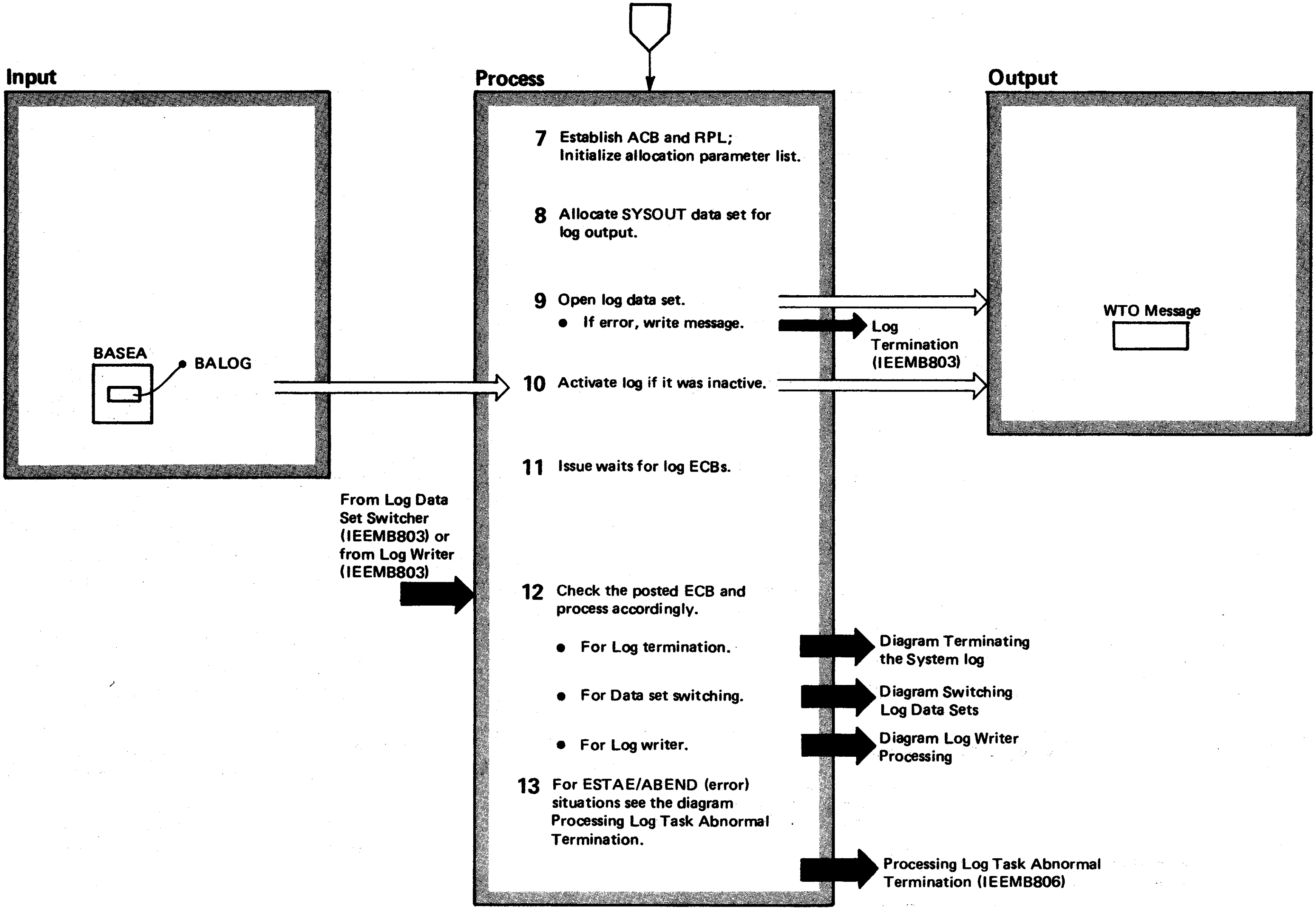
5 Initialize job entry subsystem blocks.

6 Assign job ID to log task.

Diagram 16-1. System Log Initialization (IEEMB803) (Part 2 of 4)

Extended Description	Module	Label
This routine initializes the system log; opens and closes the log data set; writes information to the log data set; and shuts down the system log.		
1 Entrance is either IPL-initiated or via WRITELOG. (For WRITELOG, the log is waiting on a POST macro instruction.)	IEEMB803	IEEMB803
2 The ESTAE routine will handle abnormal (error) situations.		
3 The routine waits for indication that a WRITELOG START command is ready to be executed.		LOGSTART
4 Parameters LOGMT and LOGCLS refer to WTL number and output class. They are in the PARMLIB member. The format of the allocation parameter list (used when terminating the system log) is as follows: ↑SYSOUT class 4 ↑Ddname of log data set to be unallocated 4	IEEMB805	LOGPVRTN
5 Storage comes from subpool 0 (LSQA).		GETCBLKS
6 This permits JES2 to process the log data sets as SYSOUT data sets. The macro instruction IEFSSREQ causes JES2 to be invoked for this purpose.		

Diagram 16-1. System Log Initialization (IEEMB803) (Part 3 of 4)



7 Establish ACB and RPL; Initialize allocation parameter list.

8 Allocate SYSOUT data set for log output.

9 Open log data set.
- If error, write message.

10 Activate log if it was inactive.

11 Issue waits for log ECBs.

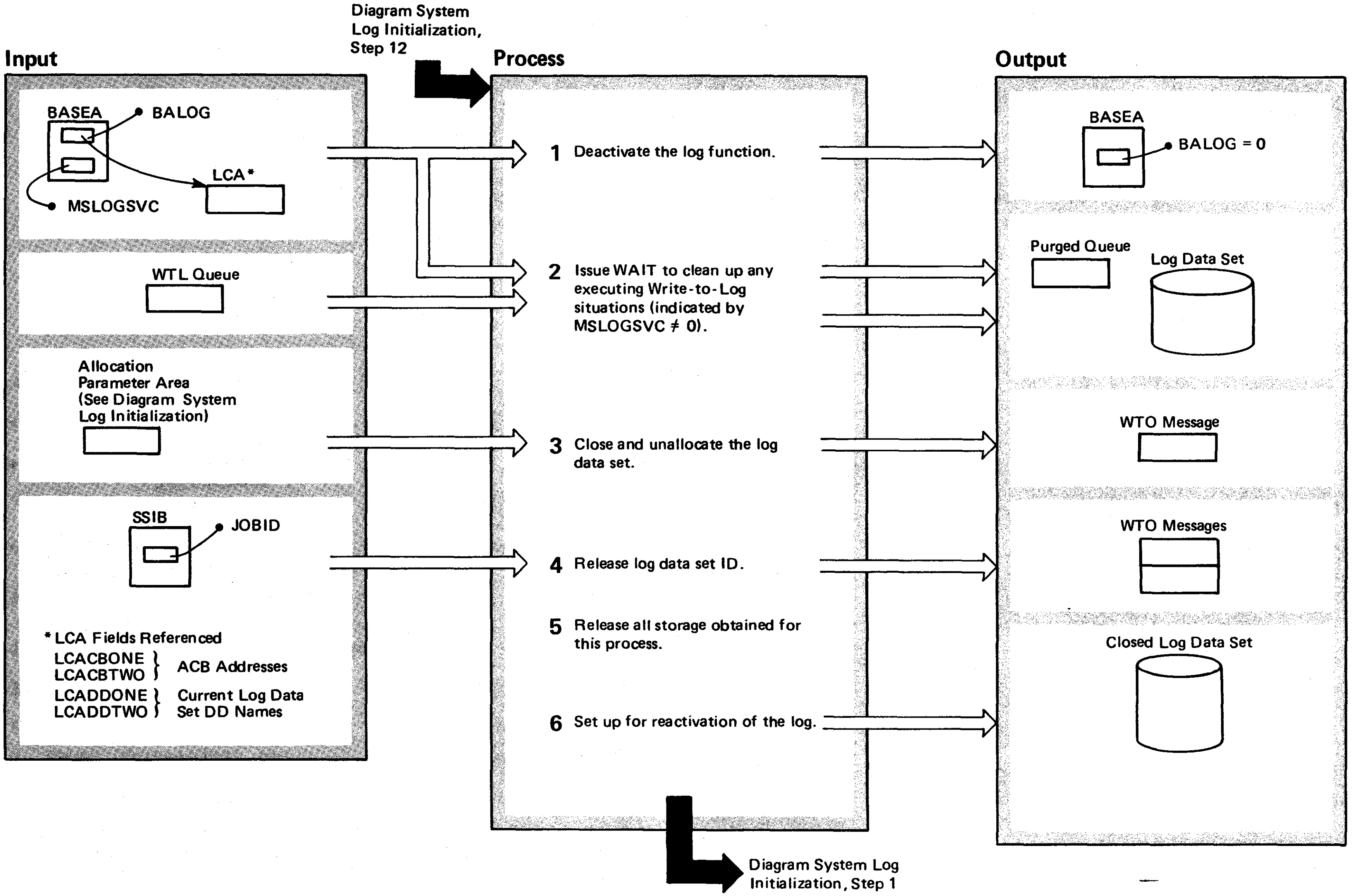
12 Check the posted ECB and process accordingly.
- For Log termination.
- For Data set switching.
- For Log writer.

13 For ESTAE/ABEND (error) situations see the diagram Processing Log Task Abnormal Termination.

Diagram 16-1. System Log Initialization (IEEMB803) (Part 4 of 4)

Extended Description	**Module**	**Label**
7 VSAM routines use the access control block (ACB) information to build a DCB. The log writer uses the request parameter list (RPL).	IEEMB803	ACBINIT RPLINIT
8 The routine issues SVC 99 for this allocation function.		
9 Use an access control block "open".	IEEMB803	OPENRTN
a. Error routine writes message.	IEEMB807	
10 Post the communications task.		NEWLOG
11 The log function to be performed determines the routine that may post the ECB.		
12 Module IEEMB803 contains subroutines to perform each task.		CHKPOST
• Log termination: IEEMB804 IEE1603D IEE70110 JES2 processing	IEEMB803	LOGTERM
• Switching log data sets: IEEMB804 IEE1603D		SWITCHDS
• Log writer: IEEMB804		WTLREC
13 The abnormal end situations are handled by the ESTAE routine.		

Diagram 16-2. Terminating the System Log (IEEMB803) (Part 1 of 2)



1 Deactivate the log function.

2 Issue WAIT to clean up any executing Write-to-Log situations (indicated by MSLOGSVC ≠ 0).

3 Close and unallocate the log data set.

4 Release log data set ID.

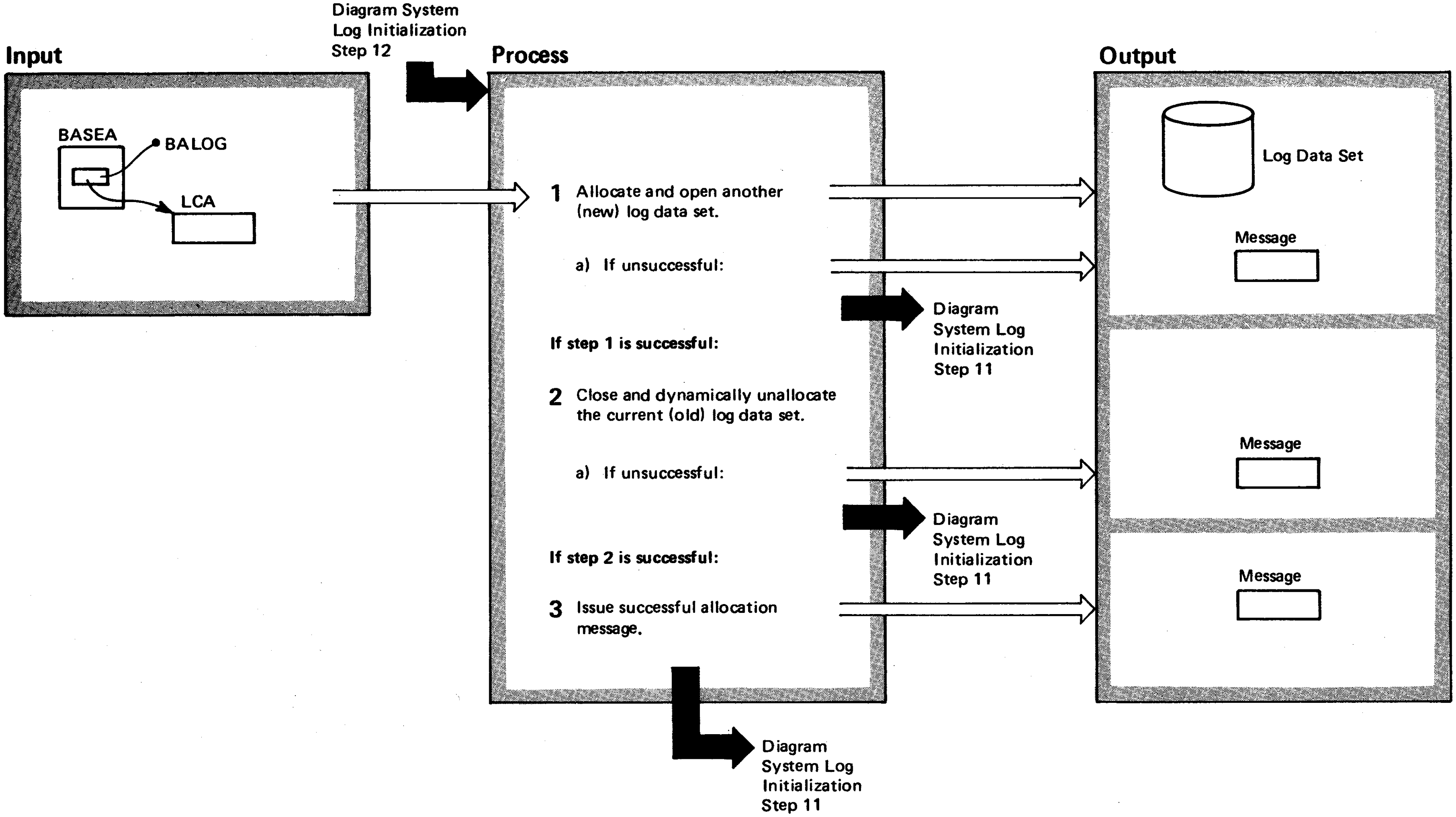
5 Release all storage obtained for this process.

6 Set up for reactivation of the log.

Diagram 16-2. Terminating the System Log (IEEMB803) (Part 2 of 2)

Extended Description	Module	Label
This routine closes the system log.		
1 Set BALOG field to zero. Issue a POST macro instruction to the Communications task.	IEEMB803	SHUTDOWN
2 The last WTL will post the wait condition.	IEEMB803	
The WTL queue is emptied by writing to the log data set.		WTLREC PUTREC
3 Unallocation may fail, in which case the error routine receives control.	IEEMB803	CLOSRTN UNALLOC
4 This job ID was assigned by the job entry subsystem and is now released by JES2 or JES3.		FRESUBS
5 The routine uses a FREEMAIN macro instruction.		WTLCLOSE
6 A WAIT is issued for a WRITELOG START command.	IEEMB803	LOGSTART

Diagram 16-3. Switching Log Data Sets (IEEMB803) (Part 1 of 2)



1 Allocate and open another (new) log data set.

a) If unsuccessful:

If step 1 is successful:

2 Close and dynamically unallocate the current (old) log data set.

a) If unsuccessful:

If step 2 is successful:

3 Issue successful allocation message.

Diagram 16-3. Switching Log Data Sets (IEEMB803) (Part 2 of 2)

Extended Description	Module	Label
This routine switches log data sets when the currently recording set is full.	IEEMB803	SWITCHDS
1 This will be done after a specified limit of write-to-log commands have been executed or after an operator issues a WRITELOG command.	IEEMB803	ALLOCRTN OPENRTN
a. Issue write-to-operator message. Continue to use existing log data set.	IEEMB807	
2 Write-to-operator message. An unallocation failure may occur before the log data set is queued to an output class. This may result in a loss of data.	IEEMB807	
3 The routine was able to close the system log data set and queue it to a SYSOUT class for disposition by JES. Message is issued by IEEMB807.	IEEMB803	CLOSERTN UNALLOC

Diagram 16-4. Log Writer Processing (IEEMB803) (Part 1 of 2)

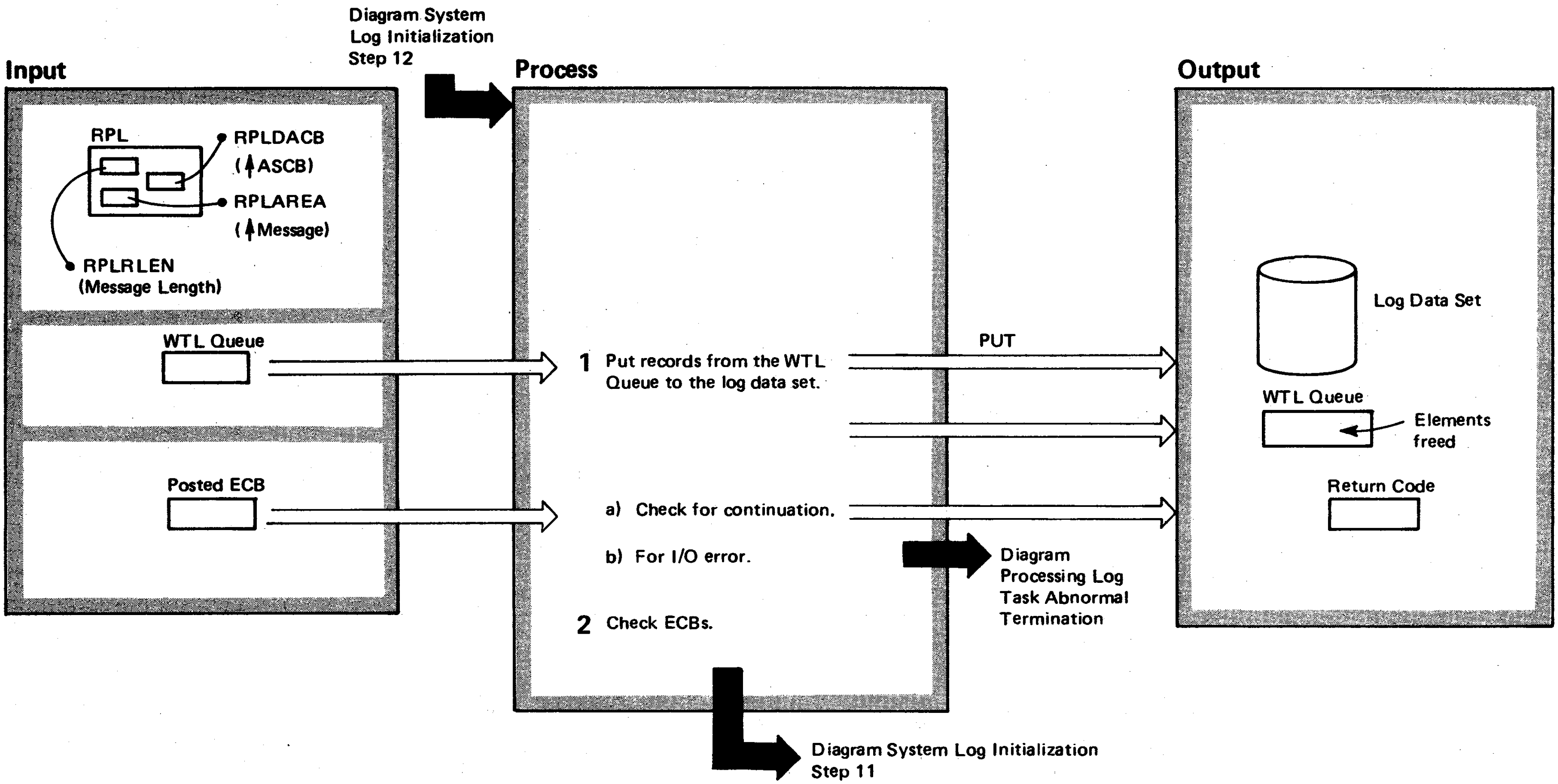
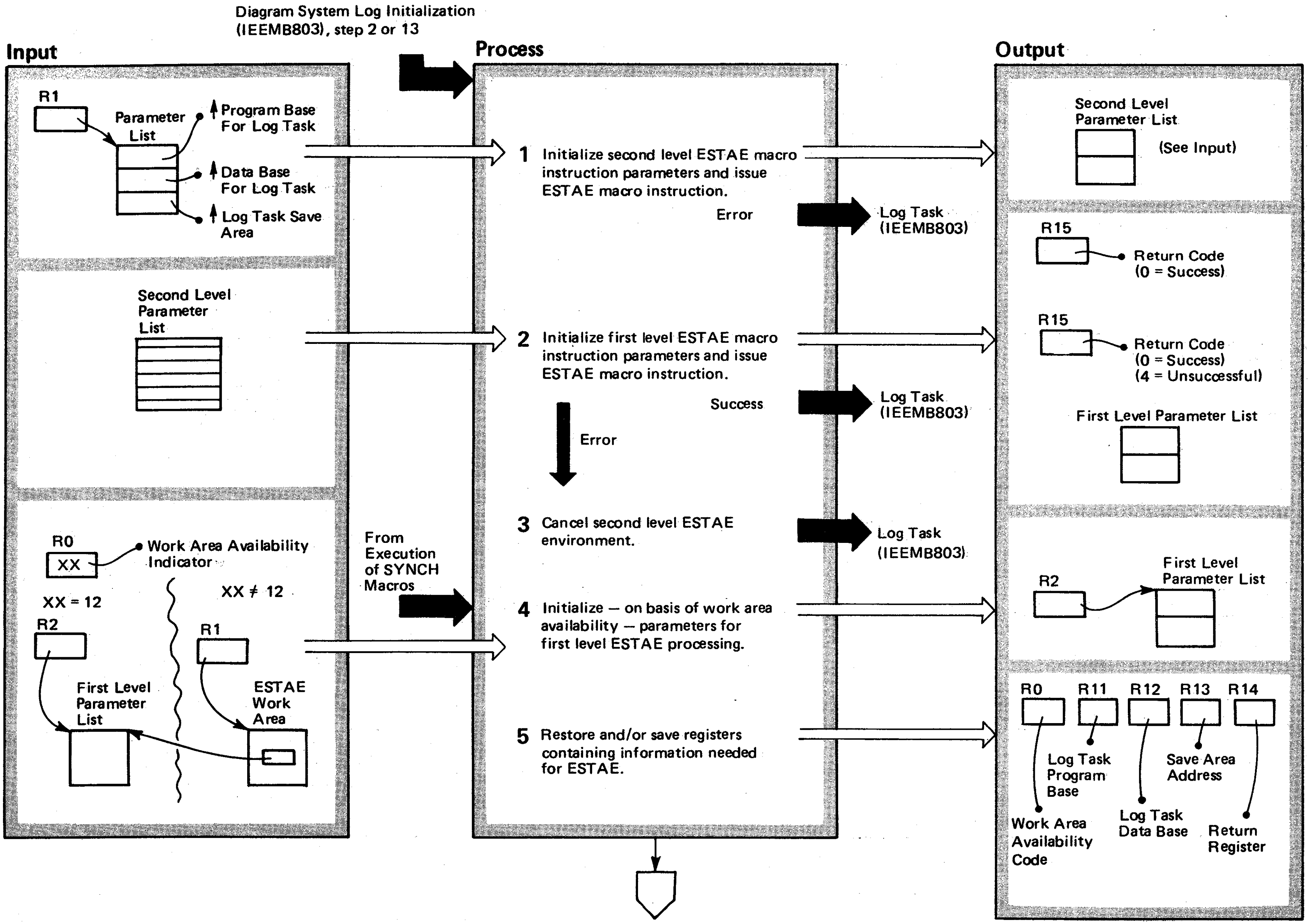


Diagram 16-4. Log Writer Processing (IEEMB803) (Part 2 of 2)

Extended Description	Module	Label
This routine writes records to the log data set.		
1 The routine first moves the queue element having the lowest chain pointer in the LCA. (Remaining pointers are decremented after each element (record) is put on the log.)	IEEMB803	WTLREC
a. After each record is put on the log, the termination and data-set-switch ECBs are tested for posting.		CHKPOST
b. If an I/O error occurs during this process, a SDUMP macro instruction (SVC 51) provides a dump. Then an ABEND macro instruction causes control to go to the ESTAE routine (IEEMB806). The ESTAE (-ABEND) control interface routine then returns control to IEEMB803.	 IEEMB806	PUTREC ESTAE1
2 After all records on the WTL queue are put on the log data set, control loops back to check ECB posting.	IEEMB803	CHCKPOST

Diagram 16-5. Processing Log Task Abnormal Termination (IEEMB806) (Part 1 of 4)



1 Initialize second level ESTAE macro instruction parameters and issue ESTAE macro instruction.

2 Initialize first level ESTAE macro instruction parameters and issue ESTAE macro instruction.

3 Cancel second level ESTAE environment.

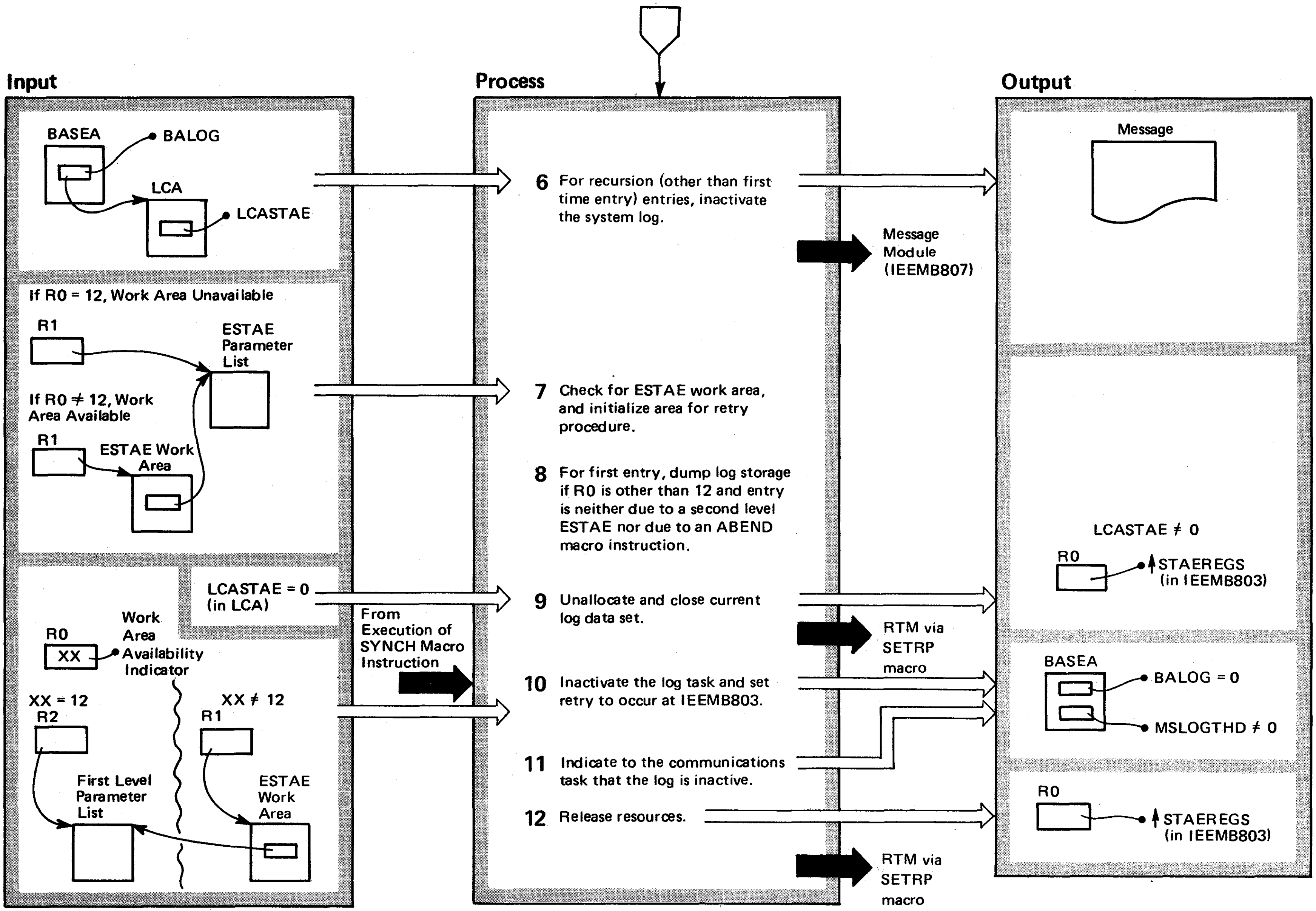
4 Initialize — on basis of work area availability — parameters for first level ESTAE processing.

5 Restore and/or save registers containing information needed for ESTAE.

Diagram 16-5. Processing Log Task Abnormal Termination (IEEMB806) (Part 2 of 4)

	Extended Description	Module	Label
	This processing sets up and uses ESTAE environments to handle abnormal termination conditions that may exist in log task processing.		
1	This level of ESTAE handles ABEND situations that may occur during the first level ESTAE processing.	IEEMB806 (Load Module IEEMB803)	IEEMB806
2	This level of ESTAE handles ABEND situations that may occur during the system log task processing.		
3	Second level ESTAE, sometimes referred to as "percolation ESTAE," is unneeded since the first level ESTAE routine is inoperative.		
4	The inputs to this step reflect whether the ESTAE/ABEND interface (that is, the ESTAE macro instruction) obtained a work area. Entry at this point indicates an ABEND occurring in the log module IEEMB803.		ESTAE1
5	These registers are restored from the ESTAE parameter list.		

Diagram 16-5. Processing Log Task Abnormal Termination (IEEMB806) (Part 3 of 4)



6 For recursion (other than first time entry) entries, inactivate the system log.

7 Check for ESTAE work area, and initialize area for retry procedure.

8 For first entry, dump log storage if R0 is other than 12 and entry is neither due to a second level ESTAE nor due to an ABEND macro instruction.

9 Unallocate and close current log data set.

10 Inactivate the log task and set retry to occur at IEEMB803.

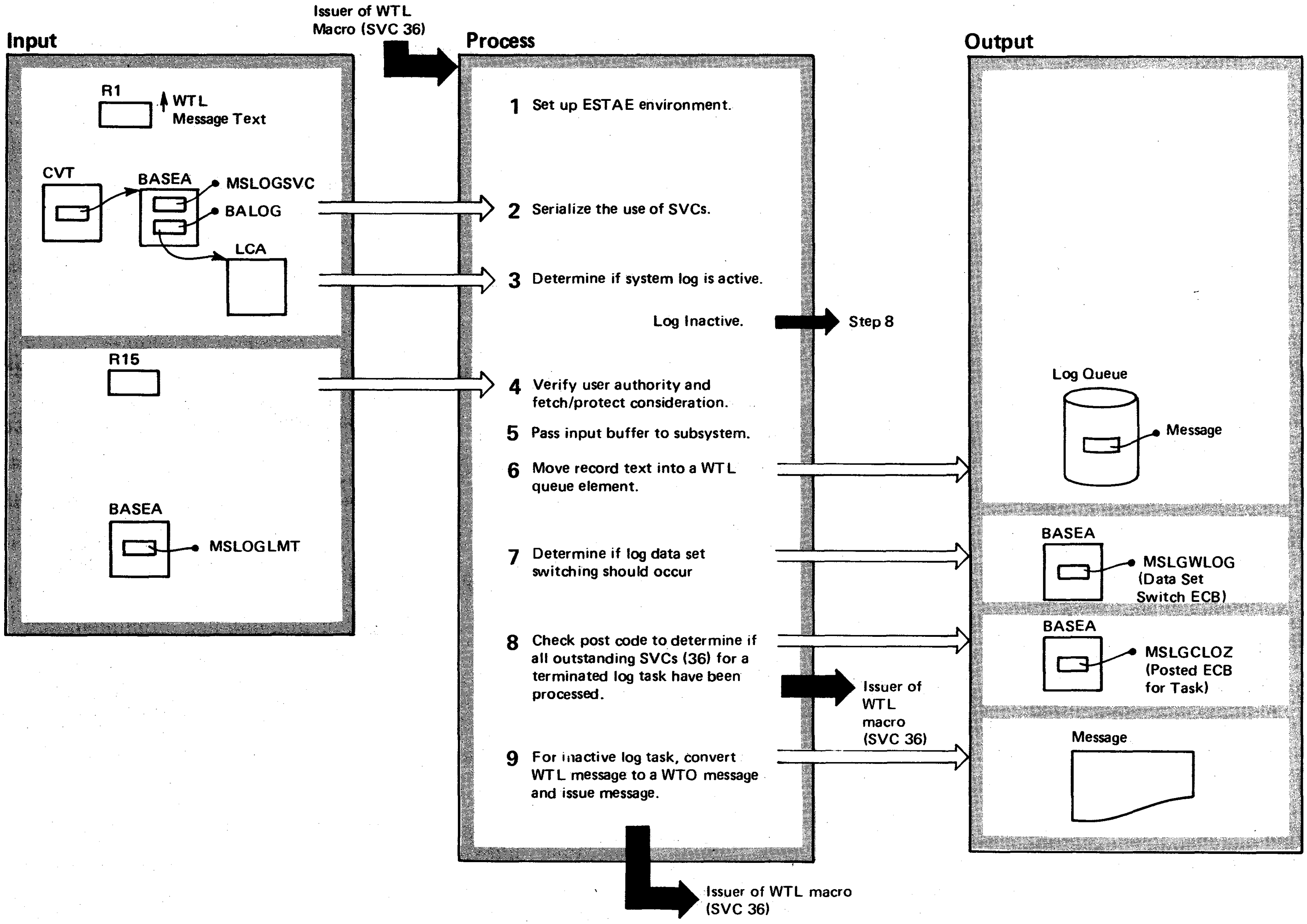
11 Indicate to the communications task that the log is inactive.

12 Release resources.

Diagram 16-5. Processing Log Task Abnormal Termination (IEEMB806) (Part 4 of 4)

Extended Description	Module	Label
6 Log task subroutines will set indicators, write the WTL queue to the log data set, unallocate and close the log data set, and free resources.	IEEMB806 IEEMB803	SHUTDOWN WTLREC LOGERROR WTLCLOSE
7 This routine uses the SETRP macro instruction for this retry initializing. For further description of the macro instruction, see the Recovery Termination Manager section of this book.	IEEMB806	RESTRT
8 The routine uses SVC 51 to produce the dump. A dump is not taken if entry is due to percolation or to an ABEND macro.		
9 The routine turns on the recursion switch (LCASTAE), establishes return codes, and returns control to the ESTAE/ABEND interface.	IEEMB806	
10 Entry at this point indicates an ABEND occurring in the first level ESTAE exit routine. See label ESTAE1, step 4.		ESTAE2
11 The switch MSLOGTHD (in BASEA) indicates the log activity condition.		
12 The log task module handles cleanup.	IEEMB803	WTLCLOSE

Diagram 16-6. Writing Data on the System Log (IEEMB804) (Part 1 of 2)



1 Set up ESTAE environment.

2 Serialize the use of SVCs.

3 Determine if system log is active.

4 Verify user authority and fetch/protect consideration.

5 Pass input buffer to subsystem.

6 Move record text into a WTL queue element.

7 Determine if log data set switching should occur

8 Check post code to determine if all outstanding SVCs (36) for a terminated log task have been processed.

9 For inactive log task, convert WTL message to a WTO message and issue message.

Diagram 16-6. Writing Data on the System Log (IEEMB804) (Part 2 of 2)

Extended Description	Module	Label
This routine processes requests resulting from the use of a WTL macro instruction.		
1 The ESTAE routine will handle Fetch/protect errors involving user data.	IEEMB804	IEE0003F
2 Routine uses a COMPARE and SWAP (CS) instruction to achieve serialization of the SVC resource.		
3 A zero in field BALOG indicates an inactive log.		
4 The TESTAUTH macro instruction and CS instruction are used here. R15=0 for valid authority.		
5 The IEFSSREQ interface passes the record to the subsystem. JES3 will send a return code to continue processing the record (RC=0) or not to continue because JES3 has processed it (RC ≠ 0). JES2 does not support this function.		
6 The routine truncates all record characters beyond 130 bytes. When the number of elements on the queue reaches 20, the routine posts the log writer to write the queue to the log data set.		
7 Posting the ECB will allow closing the current log data set and opening another one.		
8 The termination of a log task may have occurred during the processing of an SVC 36.	IEEMB804	
9 Routine issues a WTO macro instruction.		

The job scheduler restart facility consists of routines that collect job-related information and process this information in the event of a job or system failure. This information is recorded either at programmer-designated checkpoints or whenever SWA-contained control blocks that are critical to a job's processing are updated. This permits termination of active jobs in the event of a system failure. It also allows the restarting of a job step from the beginning, its most recent checkpoint if automatic restart is requested, or from a programmer-designated checkpoint if deferred restart is requested.

DSDR Processing

DSDR processing routines use data from a checkpoint data set to update the control blocks in the scheduler work area (SWA). A CHKPT macro instruction results in the issuing of a checkpoint SVC to save information in the checkpoint data set. Checkpoint/restart routines use this information when a job restarts at the designated checkpoint. The information required by the scheduler is saved in the DSDR.

The Job Journal

The job journal, a logical sequential data set residing on JES2's or JES3's direct access spool volume, provides backup direct access storage for the scheduler work area. It contains copies of SWA control blocks that are critical to the restart processing of a job. Each job has its own job journal, which is a temporary data set that exists for the life of the job. For each job, the initial entry to the job journal is a Job Header Record (JHR). A Step Header Record (SHR) is journalled just prior to the job step allocation processing for each step. At the completion of allocation processing, the critical SWA control blocks for the job step are written to the job journal.

As SWA updating occurs during the processing of each job step, a copy of each critical control block for the step is written to the job journal. Thus, an audit trail of the necessary control blocks for each job is maintained to provide for the reconstruction of the SWA for the following forms of restart:

Automatic checkpoint restart.
Automatic step restart.
System restart.
Continue restart.

To support multiple subsequent restarts, all critical blocks for all steps up to the failing step are re-journalled.

Job-processing routines write the following critical SWA control blocks to the job journal:

JCT (Job Control Table).
SCT (Step Control Table).
SIOT (Step Input/Output Table).
JFCB (Job File Control Block).
JFCBE (Job File Control Block Extension for 3800 printer).
JFCBX (Job File Control Block Extension).
PDI (Passed Data Set Information Block).
GDGNT (Generation Data Group Name Table).
ACT (Account Control Table).
VUT (Volume Unload Table).
VDSCB (Virtual Data Set Control Block).
DSPCT (Data Set Page Control Table Header).

Journal Routines

The journal write routine is responsible for maintaining the job journal. The journal write routine determines which control blocks are necessary for restart and writes those blocks to the job journal. As critical blocks are altered during the processing of a job step (for example, due to an open, scratch, close, checkpoint, or dynamic allocation procedure) the journal write routine writes the updated blocks to the journal.

The SWA resides in virtual (pageable) storage and contains the control blocks used by the scheduler during job processing. When initiator routines perform termination processing at job step failure time, they free the SWA. If a failing job step is authorized for automatic restart, the SWA must be reconstructed using the information preserved in the job journal. The SWA contents are unrecoverable in the event of a system failure. For this situation, restart routines use the information preserved in the job journal to reconstruct the SWA in order to perform termination processing for all active jobs.

For system and automatic restart processing, the journal merge routine reconstructs the SWA so it appears as follows:

- When used for automatic checkpoint restart, the SWA contains the control blocks in effect at the time the checkpoint was taken.

- When used for automatic step restart, the SWA contains the control blocks in effect at the beginning of the failing step.

- When a system failure occurs, the SWA contains the control blocks in effect at the point of failure. If a system failure occurs during job step termination processing, the job is reenqueued for step-continue processing.
- When used for step-continue processing, the SWA contains the control blocks necessary to permit a restart at the next job step.

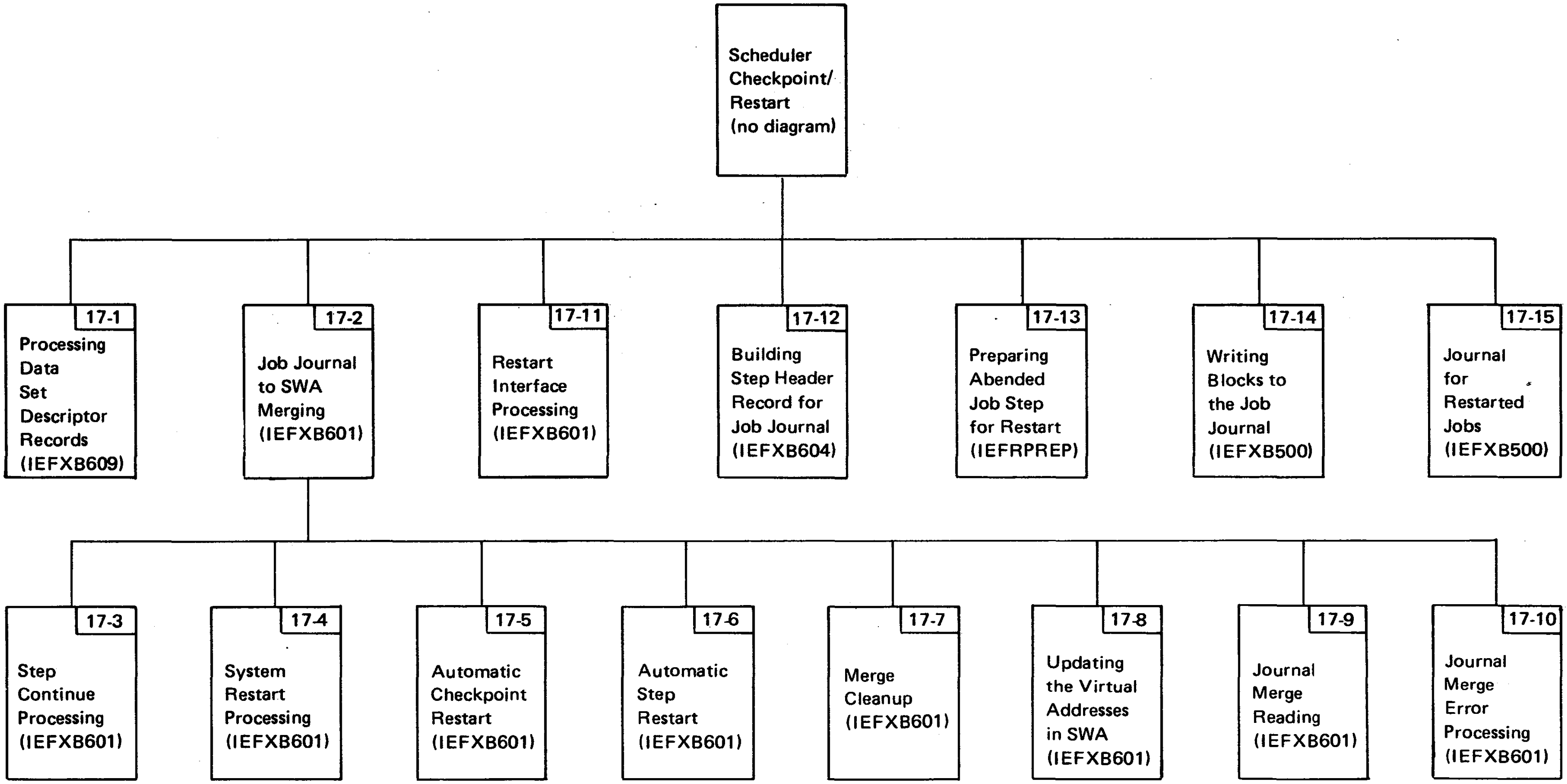
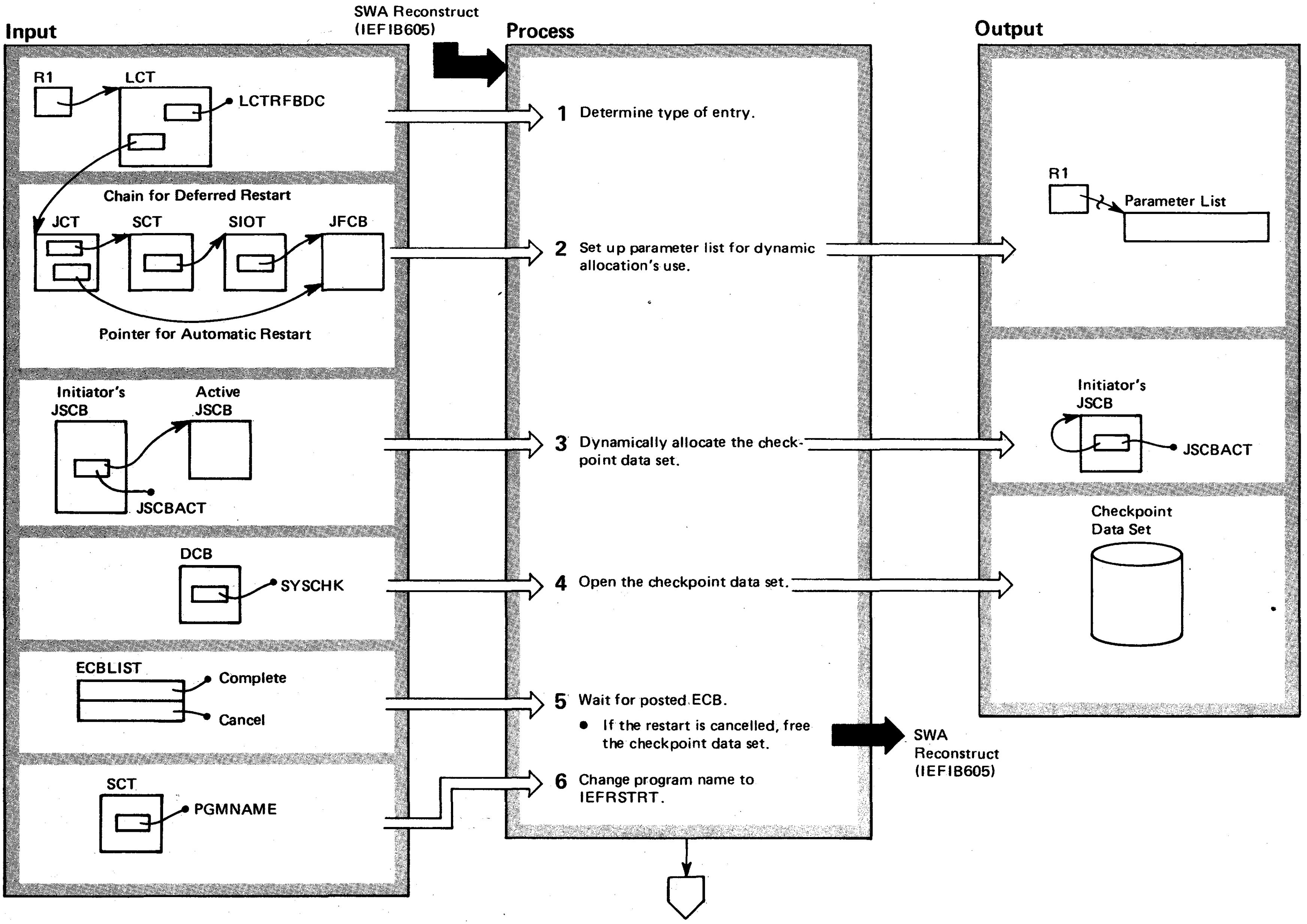


Figure 2-31. Job Scheduler Checkpoint/Restart Visual Contents

Diagram 17-1. Processing Data Set Descriptor Records (IEFXB609) (Part 1 of 6)



1 Determine type of entry.

2 Set up parameter list for dynamic allocation's use.

3 Dynamically allocate the checkpoint data set.

4 Open the checkpoint data set.

5 Wait for posted ECB.

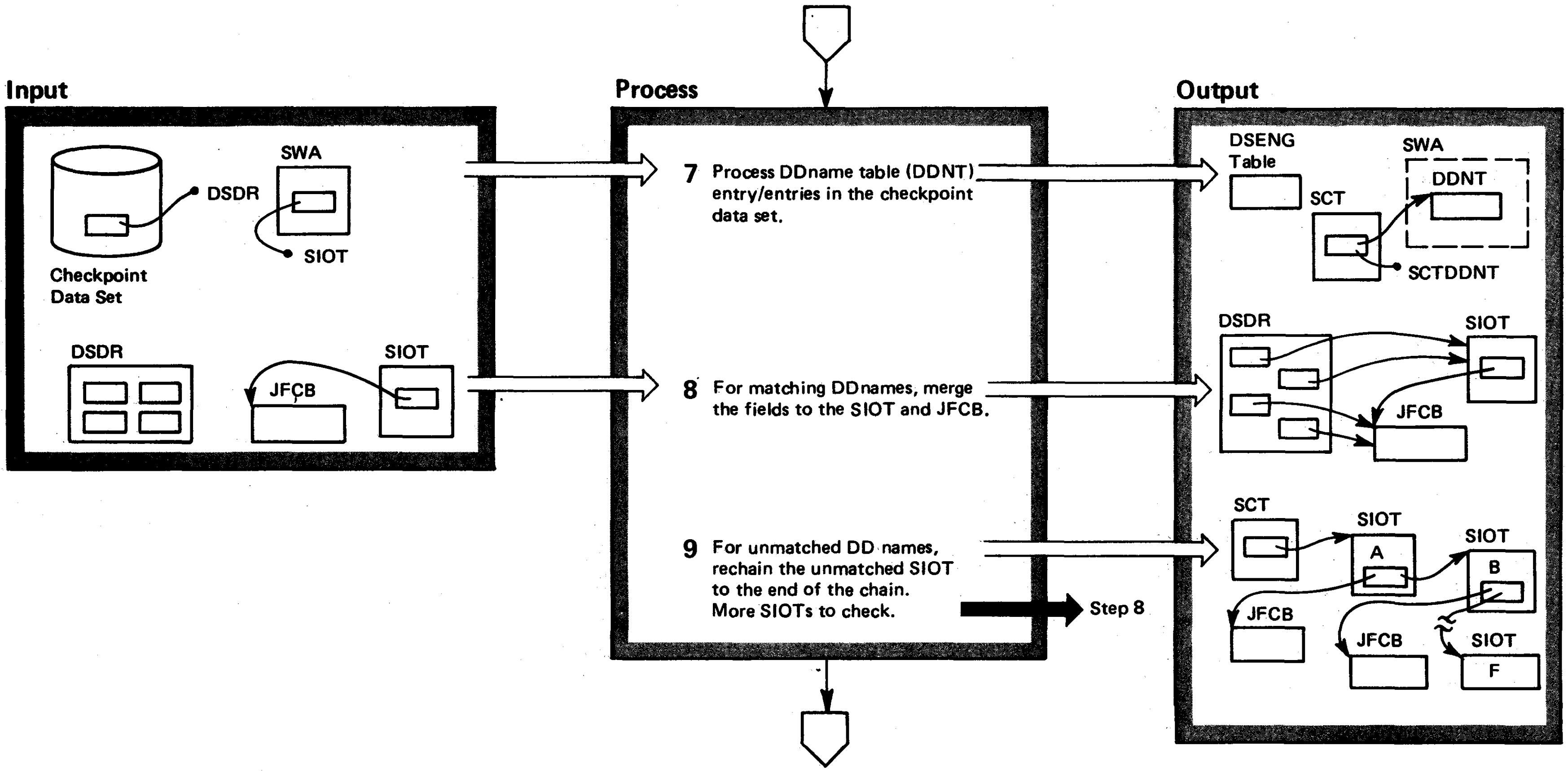
- If the restart is cancelled, free the checkpoint data set.

6 Change program name to IEFRSTRT.

Diagram 17-1. Processing Data Set Descriptor Records (IEFXB609) (Part 2 of 6)

Extended Description	**Module**	**Label**
0 This routine processes the SWA information in the checkpoint data set DSDR records so that the SWA entries reflect the checkpoint environment.		
1 The field LCTRFDBC indicates whether the entry is for automatic restart or deferred restart. ● If the entry is for automatic restart, the JCT contains the virtual address of the JFCB used for the checkpoint data set. The SWA merge routine (IEFXB601) has already merged the job journal to the SWA. The JFCB information will be used in dynamically allocating the checkpoint data set. ● If the entry is for deferred restart, interpreter routines have created the control blocks for the step. Control block pointers will locate the JFCB for the checkpoint data set to be dynamically allocated.	IEFXB609	SETUP
2 Dynamic allocation routines use this list, which contains information from the JFCB. Entries in the parameter list include the ddname (SYSCHK), the data set name (DSNAME), the volume serial number, and the unit specification.		ALOCCHEK
3 The routine issues the DYNALLOC macro instruction (via SVC 99). Prior to issuing the macro instruction, the routine sets the initiator's JSCB to point to itself as the active JSCB. This permits dynamic allocation routines to use the initiator's SWA. After the checkpoint data set is allocated, IEFXB609 resets the pointer so the active JSCB pointer indicates the problem program's SWA. If the dynamic allocation fails, the DAIRFAIL routine (IKJEFF18) is invoked to issue a write to programmer (WTP) indicating the nature of the failure.		ALOCCHEK
If VSAM private catalogs exist, they are allocated to ensure proper catalog search during DSDR merge processing.	IEFXB609	ALOCAT
4 The routine attaches the module IEFXB610 to open the data set. Then it waits on the ECBs (see step 5).	IEFXB609	OPENCHEK
5 The ECBs indicate either successful 'open' processing or an external job cancellation during the 'open' processing. If cancelling occurs, the routine posts the subtask, (IEFXB610) that was to open the checkpoint data set. This permits module IEFXB610 to terminate its processing.		OPENCHEK
6 The program name is that of the restart program that issues the SVC RESTART to cause data sets to be repositioned.	IEFXB609	RNAMEPGM

Diagram 17-1. Processing Data Set Descriptor Records (IEFXB609) (Part 3 of 6)



7 Process DDname table (DDNT) entry/entries in the checkpoint data set.

8 For matching DDnames, merge the fields to the SIOT and JFCB.

9 For unmatched DD names, rechain the unmatched SIOT to the end of the chain. More SIOTs to check.

Diagram 17-1. Processing Data Set Descriptor Records (IEFXB609) (Part 4 of 6)

Extended Description **Module** **Label**

7 Prior to processing the ddname table, the routine processes the checkpoint header record (CHR) and saves the CHR for the restart parameter list.

IEFXB609 DDNTPROC

The format of the checkpoint header record (CHR) appears below. (The *OS/VS2 Checkpoint/Restart Logic* publication, Form SY26-3820, contains further information about the CHR.)

Number of checkpoints taken	2	Length of checkpoint data set entry's ID	2
ID of checkpoint data set entry	16		
Ddname of checkpoint data set	8		
Beginning (low order) address of problem program storage	4		
Size of problem program area	4		
Checkpoint data set user's blocksize	2	TIOT length	2
Flagbyte 1*	1	Checkpoint work area size	3
Flagbyte 2**	1	↑ Checkpoint work area	3
↑ SVRB used by checkpoint routines	4		
Unused	8		
System ID	1	CHR identifier	19

Flagbyte 1	*Meaning*
X'40'	Track overflow is specified.
X'20'	Checkpoint data set is on tape.
X'08'	Task is in real storage (that is, request is V = R).
X'02'	Checkpoint modules opened the checkpoint data set.
X'01'	The checkpoint is using (was opened for) BPAM.

***Flagbyte 2*	*Meaning*
X'04'	A user has supplied the checkpoint data set entry.
Other than X'04'	The operating system supplied the checkpoint data set entry.

The DDNT contains ddnames for all JCL-specified data sets dynamically unallocated prior to the checkpoint. The routine insures that allocation for these data sets does not occur at restart time.

The module places the DDNT record in the SWA. After all DDNTs are processed, they are chained together in the SWA. The module also updates the SIOT chain and the DSENQ table. The updated data set enqueue (DSENQ) table reflects data set changes resulting from dynamic allocation processing.

DDNTSIOT

8 The first available SIOT in the SWA is obtained. Then, the module matches the ddname in the SIOT with the ddname field in the DSDR being processed. The format of the DSDR appears below.

Identifier	2	JFCB at checkpoint time	196
DDnames from TIOT (or from SIOT if the entry is dynamically concatenated)	8		
Unit type(s) descriptors	4		
SCT fields (from SIOT)**	3	flags	1
DEB volume sequence	1		

**Flag Bit*	*Meaning*
7	Data set dynamically allocated (from DSAB)
6	Data set dynamically concatenated (from DSAB)
5	Data set was open at checkpoint time
0-4	reserved

**SCT fields (1 byte each) in order from low to high block location.
Status field from SIOT; Disposition field from SIOT;
Conditional disposition field from SIOT

Status and data set disposition fields are merged from the DSDR to the SIOT. Non-volume information is moved from the DSDR to the JFCB in the SWA. Volume information (for example, VOLSERs) is conditionally moved to the JFCB to reflect the SWA environment at checkpoint time.

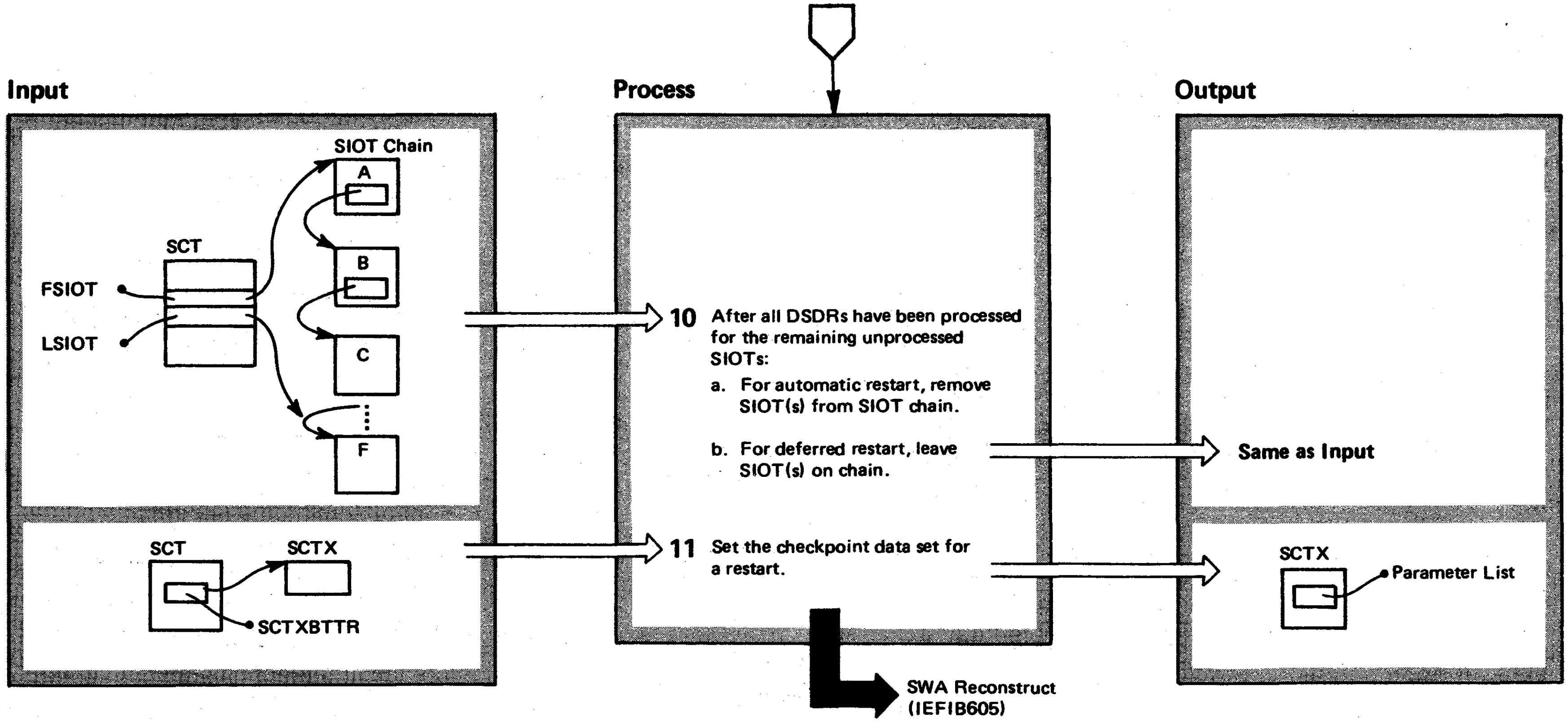
SIOTPROC
JFCBPROC
VOLSERS
JFCBXJOB

The DEB volume sequence field of the DSDR is used by the checkpoint data set utility to obtain the correct volume of a multiple volume data set.

9 By putting unmatched SIOTs at the end of the chain, the final order of the SIOT chain will be that of the DSDRs on the checkpoint data set. For deferred restarts, a SIOT/JFCB pair is built for a dynamically-allocated data set.

IEFXB609 NOMATCH

Diagram 17-1. Processing Data Set Descriptor Records (IEFXB609) (Part 5 of 6)



10 After all DSDRs have been processed for the remaining unprocessed SIOTs:

a. For automatic restart, remove SIOT(s) from SIOT chain.

b. For deferred restart, leave SIOT(s) on chain.

11 Set the checkpoint data set for a restart.

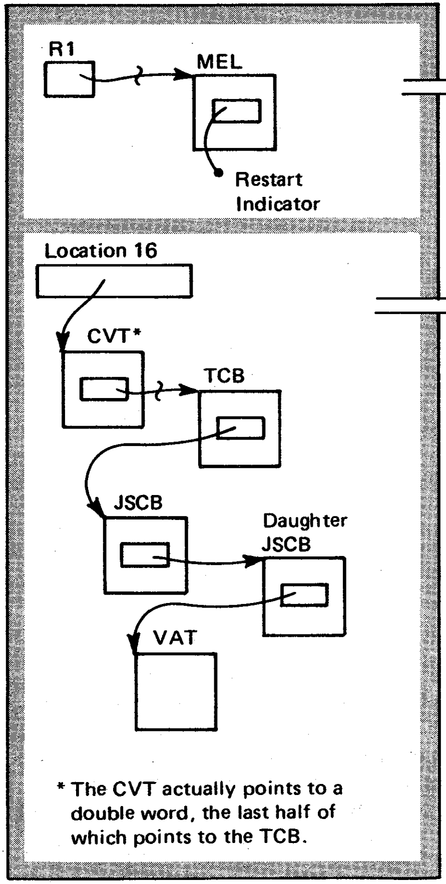
Diagram 17-1. Processing Data Set Descriptor Records (IEFXB609) (Part 6 of 6)

Extended Description	Module	Label
10 These SIOTs are the ones remaining on the SIOT chain after all DSDRs have been processed. a) 'Automatic restart' SIOTs are those created dynamically after the checkpoint was taken. They are deleted so the user can re-allocate them. b) 'Deferred restart' SIOTs represent DD cards added to the JCL input stream when the job was resubmitted. The number of DDs is updated to reflect these SIOTs.		LEFTOVER
11 The routine sets up a parameter list for the restart SVC. The list contains the TTR of the core image record (CIR) on the checkpoint data set and checkpoint header record information. To pass the list to the restart SVC, the routine uses the SCTX control block.		SCTXPROC RNAMEPGM

Diagram 17-2. Job Journal to SWA Merging (IEFXB601) (Part 1 of 2)

SWA Reconstruct
(IEFIB605)

Input



* The CVT actually points to a double word, the last half of which points to the TCB.

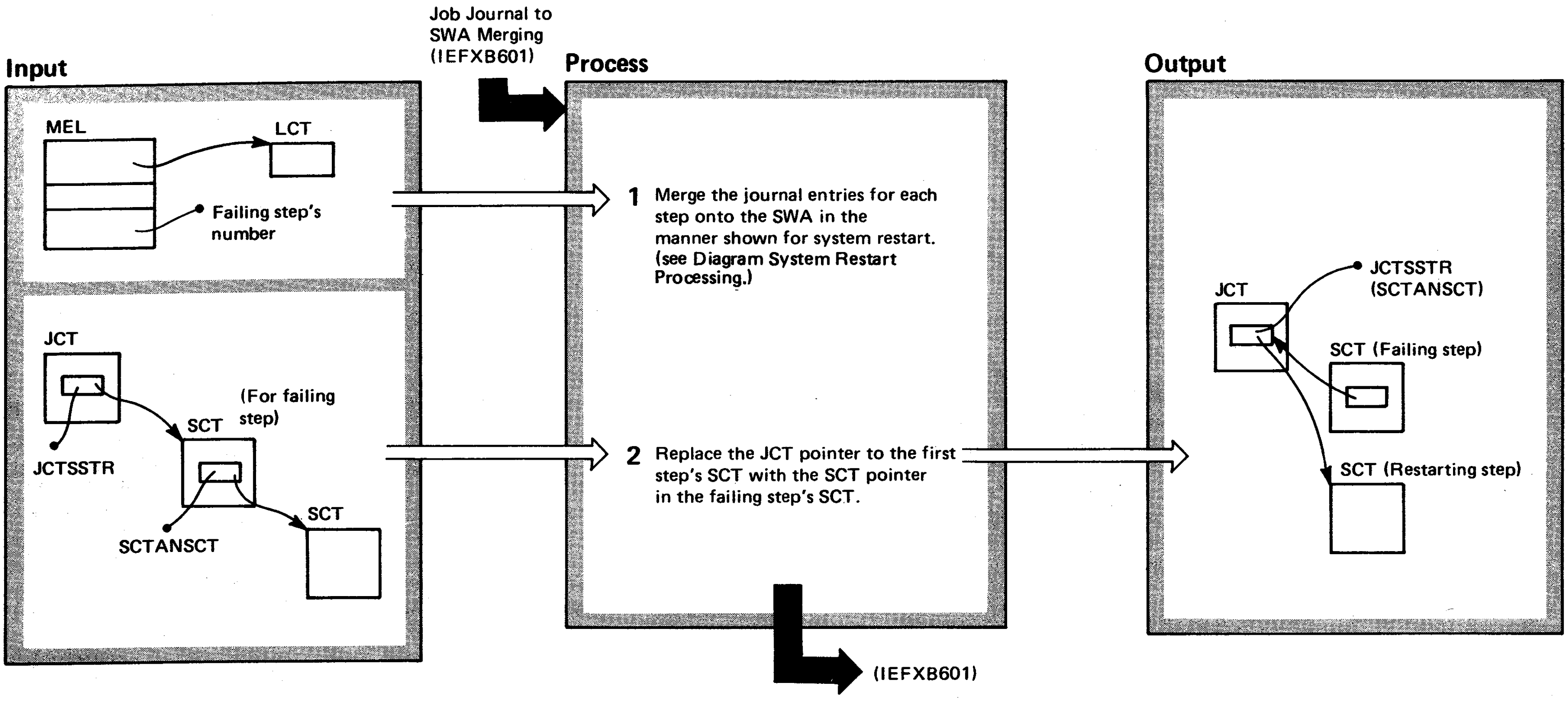
Process	Output
1 Determine type of restart and do either step 2, 3, 4, or 5. Then do steps 6-9.	**See Appropriate Diagrams**
2 For system restart:	Diagram, System Restart Processing
3 For step continue processing:	Diagram, Step Continue Processing
4 For automatic checkpoint restart:	Diagram, Automatic Checkpoint Restart
5 For automatic step restart:	Diagram, Automatic Step Restart
6 Process non-SWA control blocks if system, automatic checkpoint restart, or step continue.	Diagrams, System Restart Processing or Automatic Checkpoint Restart or Step Continue Processing
7 Update chaining fields in SWA.	Diagram, Updating the Virtual Addresses in SWA
8 Error processing.	Diagram, Journal Merge Error Processing
9 Free resources.	Diagram, Merge Cleanup

SWA Reconstruct
(IEFIB605)

Diagram 17-2. Job Journal to SWA Merging (IEFXB601) (Part 2 of 2)

Extended Description	Module	Label
This routine reconstructs the SWA (from the job journal) so it has the control blocks in effect at the time indicated: • For automatic checkpoint restart: Control blocks at time checkpoint was taken. • For automatic step restart: Control blocks at beginning of the failing step. • For system failure: Control blocks at the point of failure. This diagram refers to several other diagrams covering the checkpoint/restart functions. Each of the latter diagrams represents a subroutine (within module IEFXB601) that has a given function to perform. This present diagram contains general module entry-information that is also applicable to these subsequent diagrams. (See also, the introduction to this section.)		
1 In the 6th byte of the merge entrance list (MEL) contains the restart indicator as follows: X'08' = system restart X'20' = step continue X'40' = automatic checkpoint restart X'80' = automatic step restart The MEL also contains the address of the LCT (in the first word) and the failing step's number (in the last two bytes).	IEFXB601	IEFXB601
2 For this case, a full merge of all control blocks for all steps is performed.		SYSMERGE
3, 4, 5 For each case, a full merge of the control blocks for the non-failing steps is performed, and selective merging of fields in critical control blocks for the failing step is performed.		SYSMERGE CKPTMRGE STEPMRGE
6 For each non-SWA control block on the job journal, an appropriate exit routine performs the required processing.		VATPUT VAMPROC
7 The routine updates the SWA control block chaining fields to reflect the new virtual addresses resulting from the SWA reconstruction.	IEFXB601	ADDRUPDT
8 The Routine sets a return code of X'24' in register 15 and sends an appropriate message to the programmer and/or the operator.		ERRPROC
9 The routine releases the virtual address table and any extensions to it. **Note:** There is one entry in the virtual address table (VAT) for each control block that the interpreter writes to the SWA. This entry points to the 16-byte prefix to the control block. When dynamic allocation routines cause the SWA's control block structure (that is, the relative control block addresses) to change during restart, the VAT updating routines insert the new control block addresses (of other journalled control blocks) into the appropriate fields of the control blocks in the SWA.		

Diagram 17-3. Step Continue Processing (IEFXB601) (Part 1 of 2)



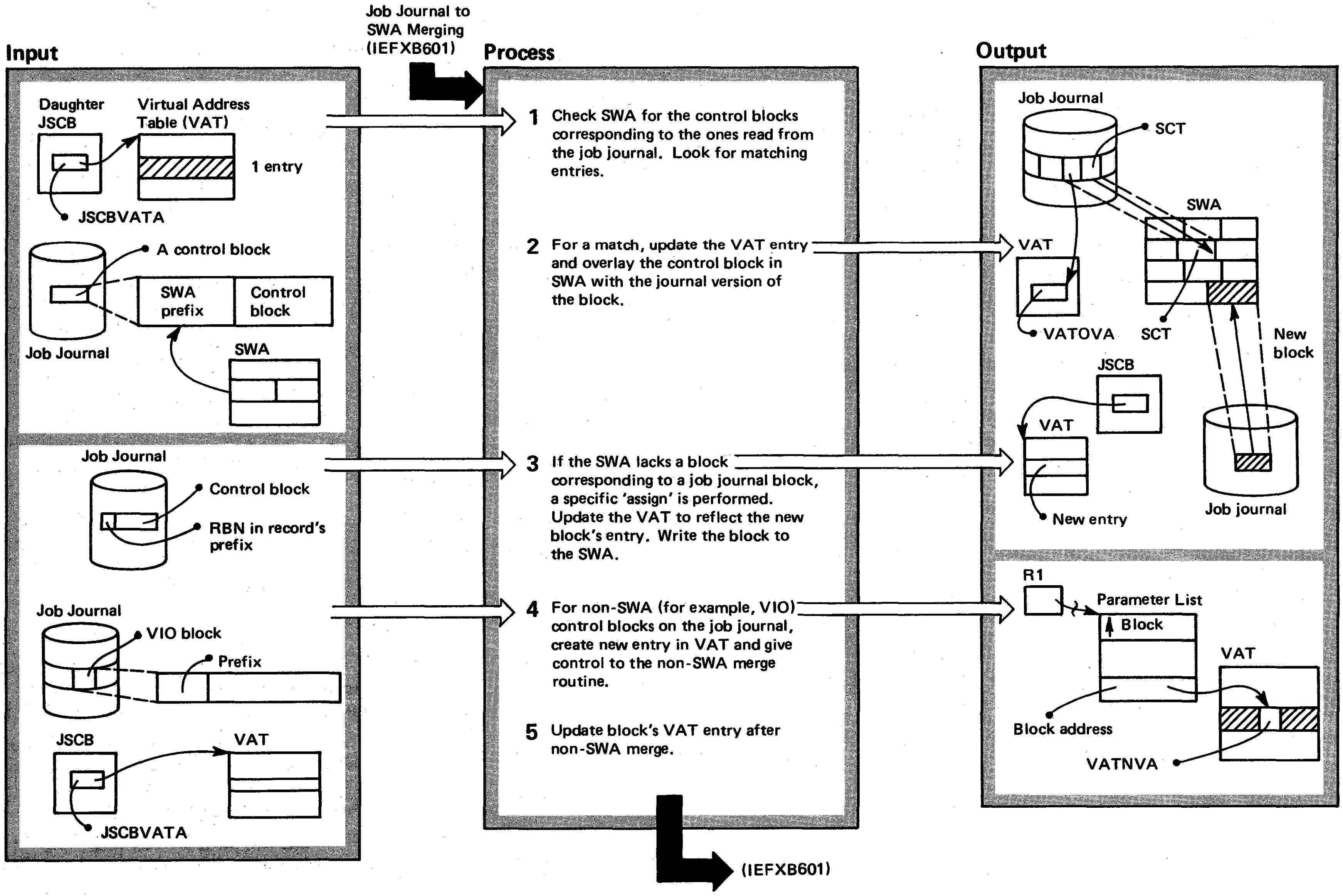
1 Merge the journal entries for each step onto the SWA in the manner shown for system restart. (see Diagram System Restart Processing.)

2 Replace the JCT pointer to the first step's SCT with the SCT pointer in the failing step's SCT.

Diagram 17-3. Step Continue Processing (IEFXB601) (Part 2 of 2)

Extended Description	Module	Label
This routine handles the processing that allows a user's job to continue at the next step.		
1 This processing occurs when a step was being terminated at the time a system failure occurred. Since the job journal entries are complete, they are processed in the same manner as for system restarts.	IEFXB601	SYSMERGE
2 By resetting the JCT pointer (to the SCT), the restart will occur at the job step following the failing step.		CLEANUP

Diagram 17-4. System Restart Processing (IEFXB601) (Part 1 of 2)



1 Check SWA for the control blocks corresponding to the ones read from the job journal. Look for matching entries.

2 For a match, update the VAT entry and overlay the control block in SWA with the journal version of the block.

3 If the SWA lacks a block corresponding to a job journal block, a specific 'assign' is performed. Update the VAT to reflect the new block's entry. Write the block to the SWA.

4 For non-SWA (for example, VIO) control blocks on the job journal, create new entry in VAT and give control to the non-SWA merge routine.

5 Update block's VAT entry after non-SWA merge.

Diagram 17-4. System Restart Processing (IEFXB601) (Part 2 of 2)

Extended Description	Module	Label
For a system restart, all control blocks for all steps in a job will be fully merged from the job journal to the SWA. (See also the diagrams Merge Cleanup and Updating the Virtual Addresses in SWA.)		
1 The SWA prefix (in the block) contains journal record identifications. The VAT contains representations of all blocks in the SWA. The relative block number (RBN) and block ID fields in the SWA prefix (for the block on the journal) are matched against entries in the VAT.	IEFXB601	VATPUT
2 If the RBN and ID fields of the prefix match those in the VAT, the old virtual address is placed in the VAT entry and the job journal form of the control block overlays the corresponding form in the SWA. (In the 'output' part of this diagram, the SCT is used as an example.)		VATPUT FLDMERGE
3 The SWA manager assign routine uses the 'assign' function in the IEFQMREQ macro instruction to get storage for control blocks initially created by allocation routines and JFCB housekeeping routines. The assign routine uses the RBN in the block prefix. An entry for the new block is made in the VAT, and the corresponding block is written to the SWA.		ASGNRITE
4 Based on the control block's ID (in the block prefix) the journal merge routine creates a new entry in the VAT and fills in the RBN, control block ID, and the old virtual address. The merge routine then calls a subroutine (IDDWIMRG or IDAVBPJ2) to merge the block to the SWA.	IEFXB601	VATPUT VAMPROC
The subroutine uses an interface parameter list to obtain the merge information.		

The parameter list used for this appears as follows:

↑ Block being merged	4		
ID of block being merged*	1	Length of block being merged	3
Relative block number	4		
New virtual address**	4		
↑ GETMAIN storage area***	4		

*The block ID field has the following meanings:

Block ID	Control Block	VIO Routine Performing the Merge
X'FE'	Data set page control table (DSPCT) header	Virtual Block Processor (VBP)
X'FC'	Virtual data set control block (VDSCB)	Window Intercept (WI)

**The new virtual address is that passed to the appropriate VIO merge routine for all except the first occurrence of the control block on the job journal.

***The GETMAIN area address is passed back to the journal merge routine by a VIO merge routine when the VIO merge routine issues a GETMAIN macro instruction for the block to be merged.

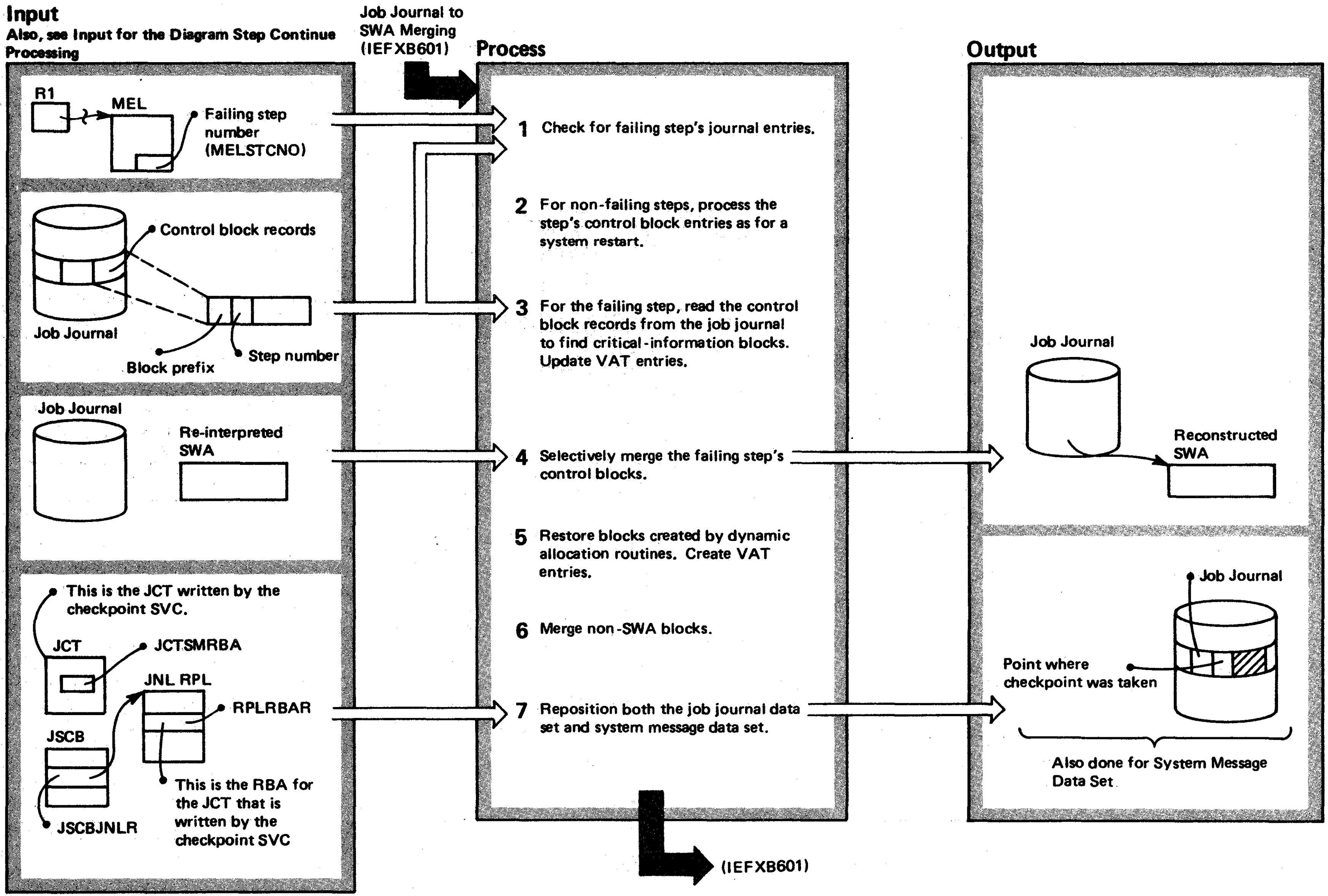
All merges subsequent to the first one (for this non-SWA block) use this information.

The control block's ID given in the block prefix is compared against an internal table of block IDs in the SWA to determine if the block (on the journal) is also in SWA.

The journal version of the block overlays the block as it resides in storage. After the block has been updated, the pointer fields in the block and the block's address (as given in the VAT) are updated.

Extended Description	Module	Label
5 The information returned from the non-SWA merge routine indicates the location of the merged control block. This address is placed in the block's VAT entry.		VAMPROC VATPUT

Diagram 17-5. Automatic Checkpoint Restart (IEFXB601) (Part 1 of 2)



1 Check for failing step's journal entries.

2 For non-failing steps, process the step's control block entries as for a system restart.

3 For the failing step, read the control block records from the job journal to find critical-information blocks. Update VAT entries.

4 Selectively merge the failing step's control blocks.

5 Restore blocks created by dynamic allocation routines. Create VAT entries.

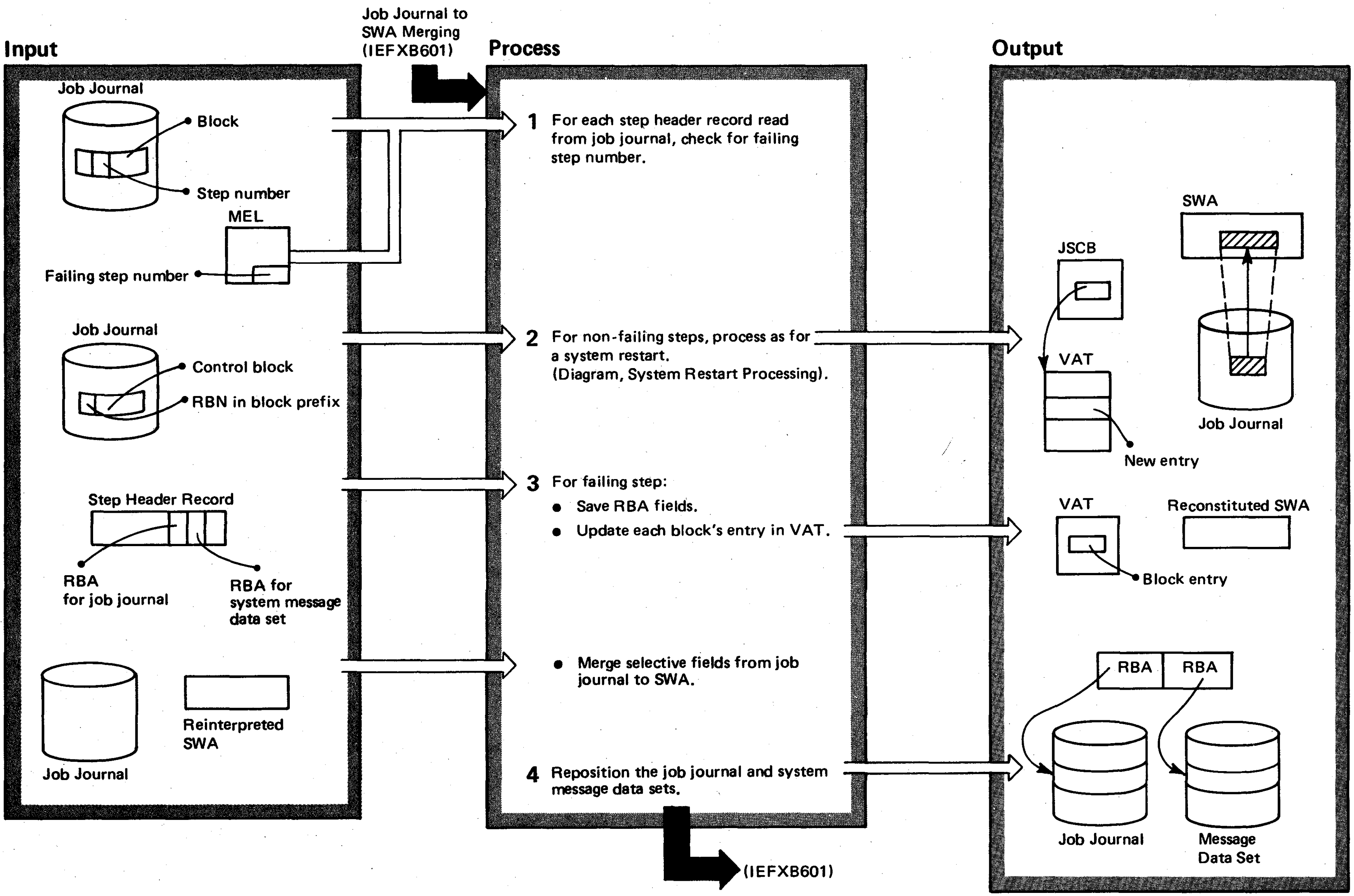
6 Merge non-SWA blocks.

7 Reposition both the job journal data set and system message data set.

Diagram 17-5. Automatic Checkpoint Restart (IEFXB601) (Part 2 of 2)

Extended Description	Module	Label
This routine merges control blocks from the job journal to the SWA for failed jobs that are eligible for an automatic checkpoint restart (checked via indicator in MEL). See also Diagrams, Merge Cleanup and Updating the Virtual Addresses in SWA.		
1 Compare the step number in the step header record with the step number in the MEL. For a step header record, the SWPID field of the block prefix is X'C0'.	IEFXB601	
2 See the Diagram, System Restart Processing for processing of steps prior to the failing step.		
3 For example, the checkpoint and job status information fields of the JCT, the volume and label information fields of the JFCB, and the chain pointer fields of the SCT, SIOT, and JFCB. In addition, the routine updates the old virtual address field of the block's entry in the VAT.		CKPTMRGE VATPUT
4 The fields containing the critical information are merged from the job journal to the SWA.		FLDMERGE
5 See the Diagram, System Restart Processing for assign details. The blocks include the SIOTs, JFCBs, JFCBEs, and JFCBXs created by dynamic allocation and JFCB housekeeping routines. The SWA manager assign and write routines specifically assigns these blocks and writes them to the SWA. The newly-created VAT entries for these blocks contain the RBN, ID, old virtual address, and new virtual address.		ASGNRITE VATPUT
6 See the Diagram, System Restart Processing, step 3.		
7 From the job control table written by the checkpoint SVC routine, save the RBA (relative block address) field for the system message data set. From the journal RPL (JNL RPL), save the RBA of the JCT written by the checkpoint SVC.		CKPTMRGE

Diagram 17-6. Automatic Step Restart (IEFXB601) (Part 1 of 2)



1 For each step header record read from job journal, check for failing step number.

2 For non-failing steps, process as for a system restart. (Diagram, System Restart Processing).

3 For failing step:

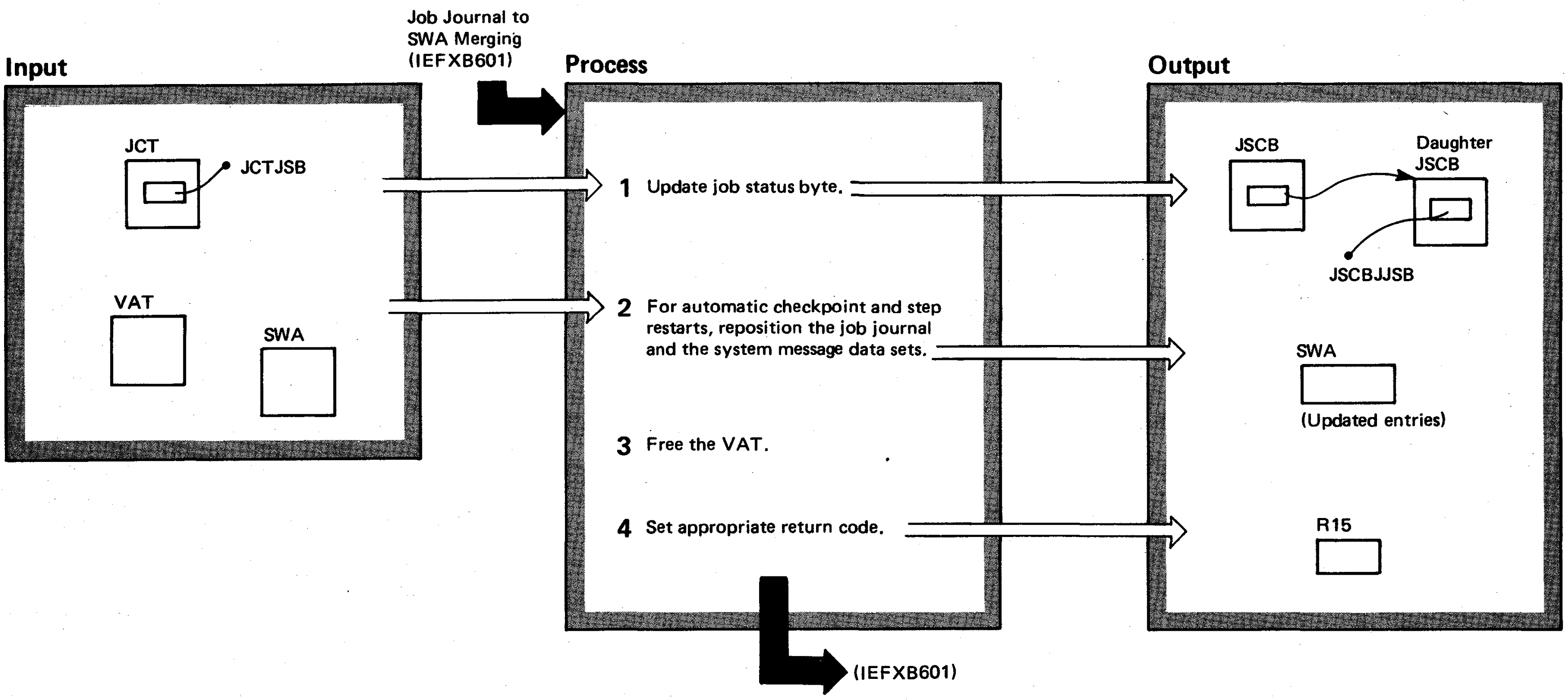
- Save RBA fields.
- Update each block's entry in VAT.
- Merge selective fields from job journal to SWA.

4 Reposition the job journal and system message data sets.

Diagram 17-6. Automatic Step Restart (IEFXB601) (Part 2 of 2)

Extended Description	Module	Label
This routine merges control blocks from the job journal to the SWA for failed jobs that are eligible for an automatic step restart (checked via indicator in MEL). See also Diagrams, Merge Cleanup and Updating the Virtual Addresses in SWA.		
1 Check the step number field in the step header record and compare it against the failing step number given in the merge entrance list (MEL). (See Diagram, Automatic Checkpoint Restart.)	IEFXB601	STEPMRGE
2 See Diagram, System Restart Processing. **Note:** If the step numbers do not match, the step is non-failing.		
3 The RBA fields saved are for the job journal and the system message data set. The fields are located in the step header records.		
For each critical control block associated with the step, the routine updates the old virtual address field in the VAT.		VATPUT
For example, selective merging involves the following fields in the indicated blocks: JCT: job status information and restart switches. JFCB and JFCBX: volume information. JFCB: MOD data set information for TTR and track balance considerations. JFCBE: 3800 printer parameters.		FLDMERGE
4 Pointers are established using the RBAs saved from the step header record. The pointers show the step's entry in each data set.		

Diagram 17-7. Merge Cleanup (IEFXB601) (Part 1 of 2)



1 Update job status byte.

2 For automatic checkpoint and step restarts, reposition the job journal and the system message data sets.

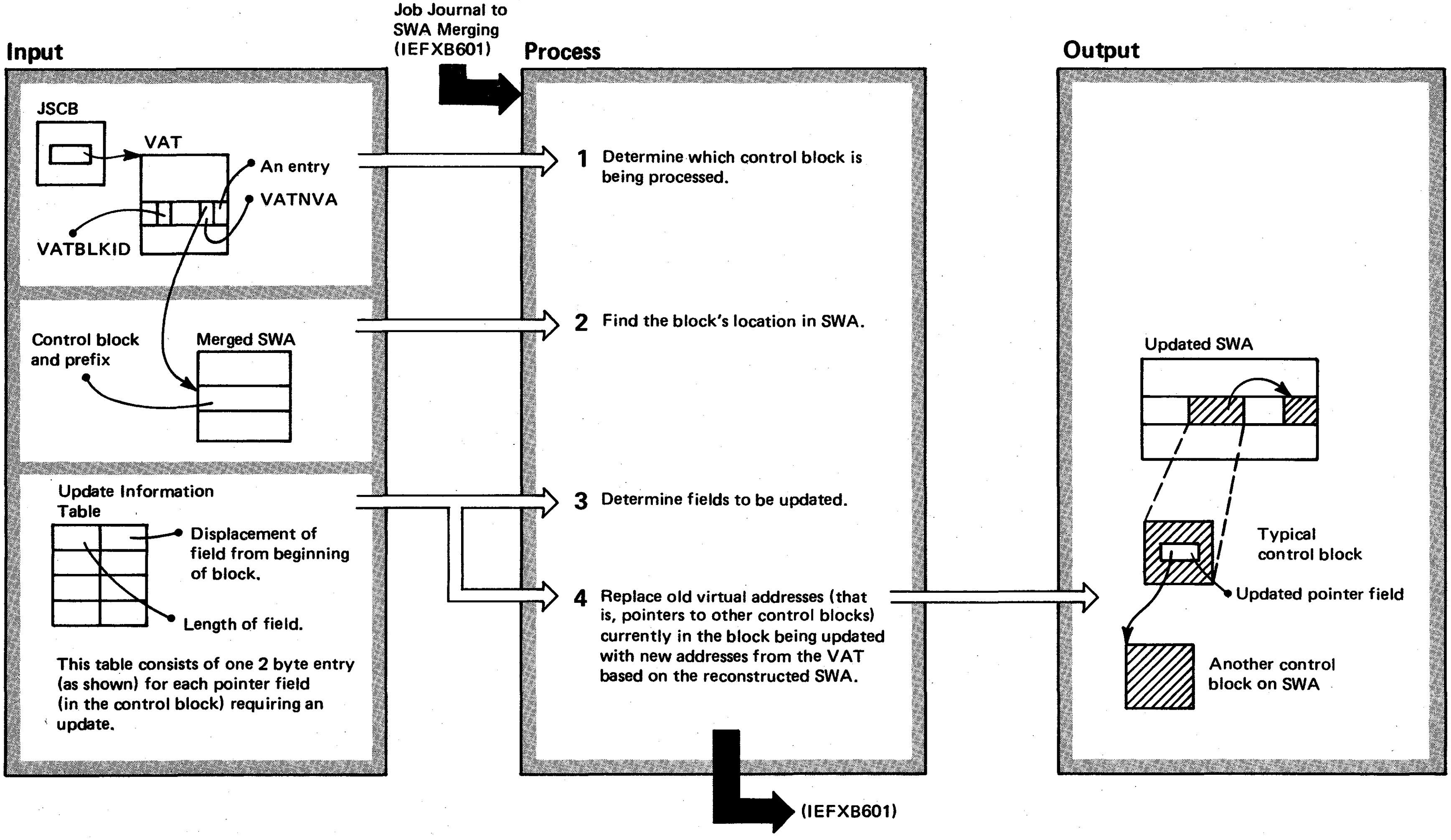
3 Free the VAT.

4 Set appropriate return code.

Diagram 17-7. Merge Cleanup (IEFXB601) (Part 2 of 2)

Extended Description	Module	Label
This routine does the clean-up functions for automatic checkpoint or step restart or for step continue processing.		
1 The latest version of this field information comes from the job journal (the JCT block). The JCTJSB information overlays that in the JSCBJJSB.	IEFXB601	CLEANUP
2 The relative block addresses used for repositioning the data sets are obtained from the step header record for automatic step restart or from the JCT and request parameter list (RPL) for automatic checkpoint restart.		
3 The VAT and any extensions to it are released.		
4 An error return code of X'24' causes the job to be purged from the system. A normal return code of X'00' permits restart to continue.		

Diagram 17-8. Updating the Virtual Addresses in SWA (IEFXB601) (Part 1 of 2)



This table consists of one 2 byte entry (as shown) for each pointer field (in the control block) requiring an update.

1 Determine which control block is being processed.

2 Find the block's location in SWA.

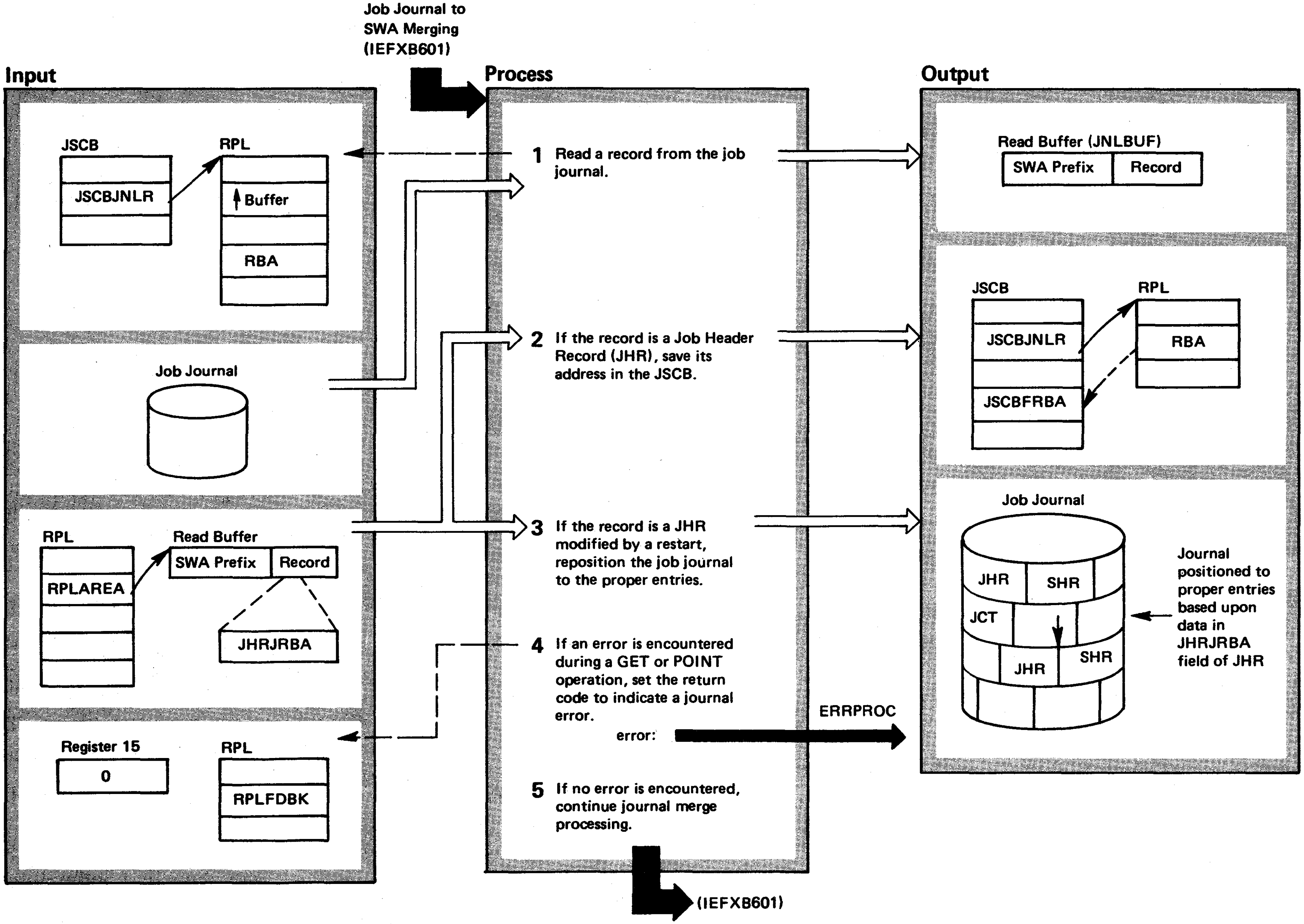
3 Determine fields to be updated.

4 Replace old virtual addresses (that is, pointers to other control blocks) currently in the block being updated with new addresses from the VAT based on the reconstructed SWA.

Diagram 17-8. Updating the Virtual Addresses in SWA (IEFXB601) (Part 2 of 2)

Extended Description	Module	Label
For each entry in the VAT, this routine updates the virtual address of all the block's fields that are changed.		
1 The block ID field in the VAT contains an indication of the control block being processed. The routine then processes consecutively all entries in the VAT.	IEFXB601	ADDRUPDT
2 The new virtual address field in the VAT entry for the block provides the new location in the merged SWA. The routine then reads the control block being processed. The SWPID field in the SWA prefix indicates the control block that is being updated.		
3 An internal table contains the necessary update information. This information includes the displacements and lengths of all fields that require updating. There is one table per control block being updated.		
4 For each address to be updated, the value in the new virtual address field (of the VAT entry for the changed control block field) replaces the existing old virtual address field in the control block.		UPDATE PTRUPDTE

Diagram 17-9. Journal Merge Reading (IEFXB601) (Part 1 of 2)



1 Read a record from the job journal.

2 If the record is a Job Header Record (JHR), save its address in the JSCB.

3 If the record is a JHR modified by a restart, reposition the job journal to the proper entries.

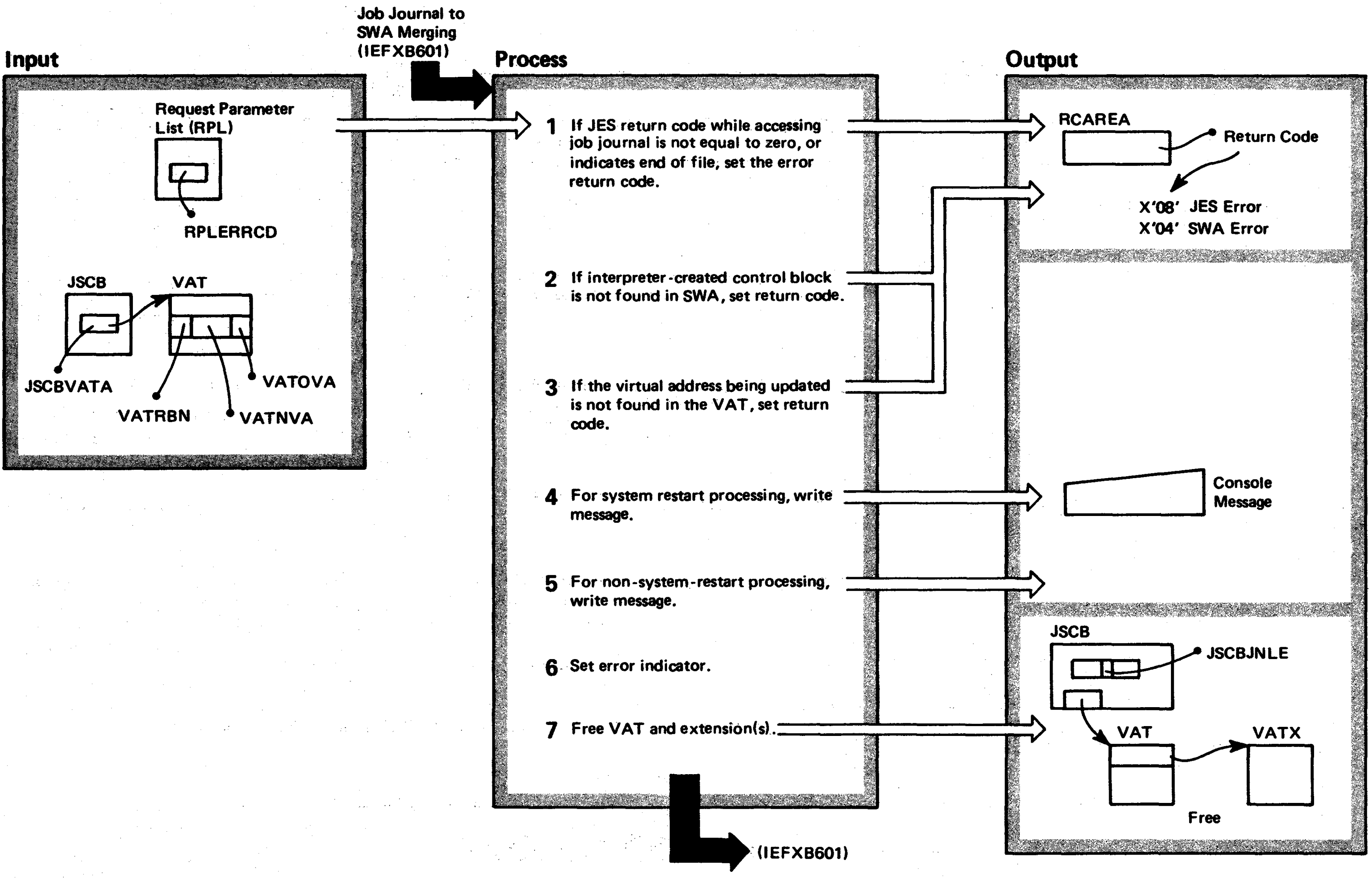
4 If an error is encountered during a GET or POINT operation, set the return code to indicate a journal error.

5 If no error is encountered, continue journal merge processing.

Diagram 17-9. Journal Merge Reading (IEFXB601) (Part 2 of 2)

Extended Description	Module	Label
This routine is responsible for all reading from the job journal required for merge processing.		
1 A record is read from the job journal using the request parameter list (RPL) pointed to by the active JSCB (JSCBJNLR).	IEFXB601	READPROC
2 A job header record has a control block ID (X'C1') in the SWA prefix. Save the address, which was passed back in the RPL, (RPLRBAR) in the active JSCB (JSCBFRBA) for journal data set repositioning.		
3 A job header record written as a result of a restart contains job journal repositioning information in the field JHRJRBA. This value is used to issue the POINT macro to position the job journal to the proper entries.		
4 Any non-zero return code in register 15 (other than a logical error indicating end of file – R15=8, RPLERRCD=0004) is considered an error condition. An error return code is set and ERRPROC receives control. (Refer to Journal Merge Error Processing diagram). If normal return code, journal merge processing is continued.		

Diagram 17-10. Journal Merge Error Processing (IEFXB601) (Part 1 of 2)



1 If JES return code while accessing job journal is not equal to zero, or indicates end of file, set the error return code.

2 If interpreter-created control block is not found in SWA, set return code.

3 If the virtual address being updated is not found in the VAT, set return code.

4 For system restart processing, write message.

5 For non-system-restart processing, write message.

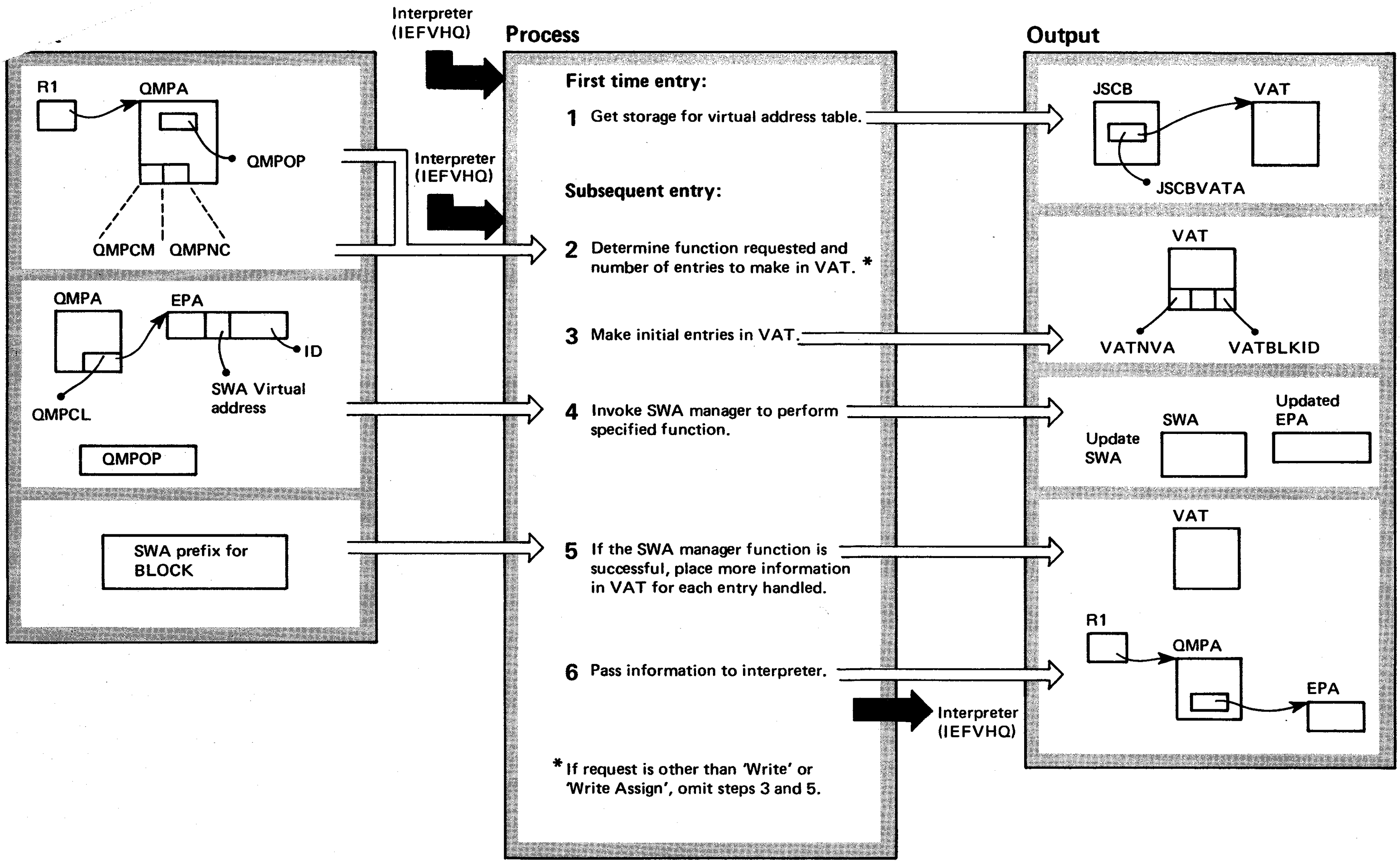
6 Set error indicator.

7 Free VAT and extension(s).

Diagram 17-10. Journal Merge Error Processing (IEFXB601) (Part 2 of 2)

Extended Description	Module	Label
This processing handles errors that may be encountered during SWA reconstruction or in accessing the job journal. It issues an appropriate message and, for either automatic step or automatic checkpoint restart, it informs the operator that the job has been cancelled.		
1 The return code is set to X'08'.	IEFXB601	ERRPROC
2 The return code is set to X'04'.		
3 The return code is set to X'04'.		
4 This message is intended for the programmer and is written to the SYSOUT data set.		
5 The message is written to the programmer via the SYSOUT data set, and a message is written to the operator via the WTO macro instruction.		
6 The journal error bit in the JSCB is turned on.		
7 The routine releases the VAT resource, and returns a code of X'24' in register 15.		

interface Processing (IEFXB602) (Part 1 of 2)



First time entry:

1 Get storage for virtual address table.

Subsequent entry:

2 Determine function requested and number of entries to make in VAT. *

3 Make initial entries in VAT.

4 Invoke SWA manager to perform specified function.

5 If the SWA manager function is successful, place more information in VAT for each entry handled.

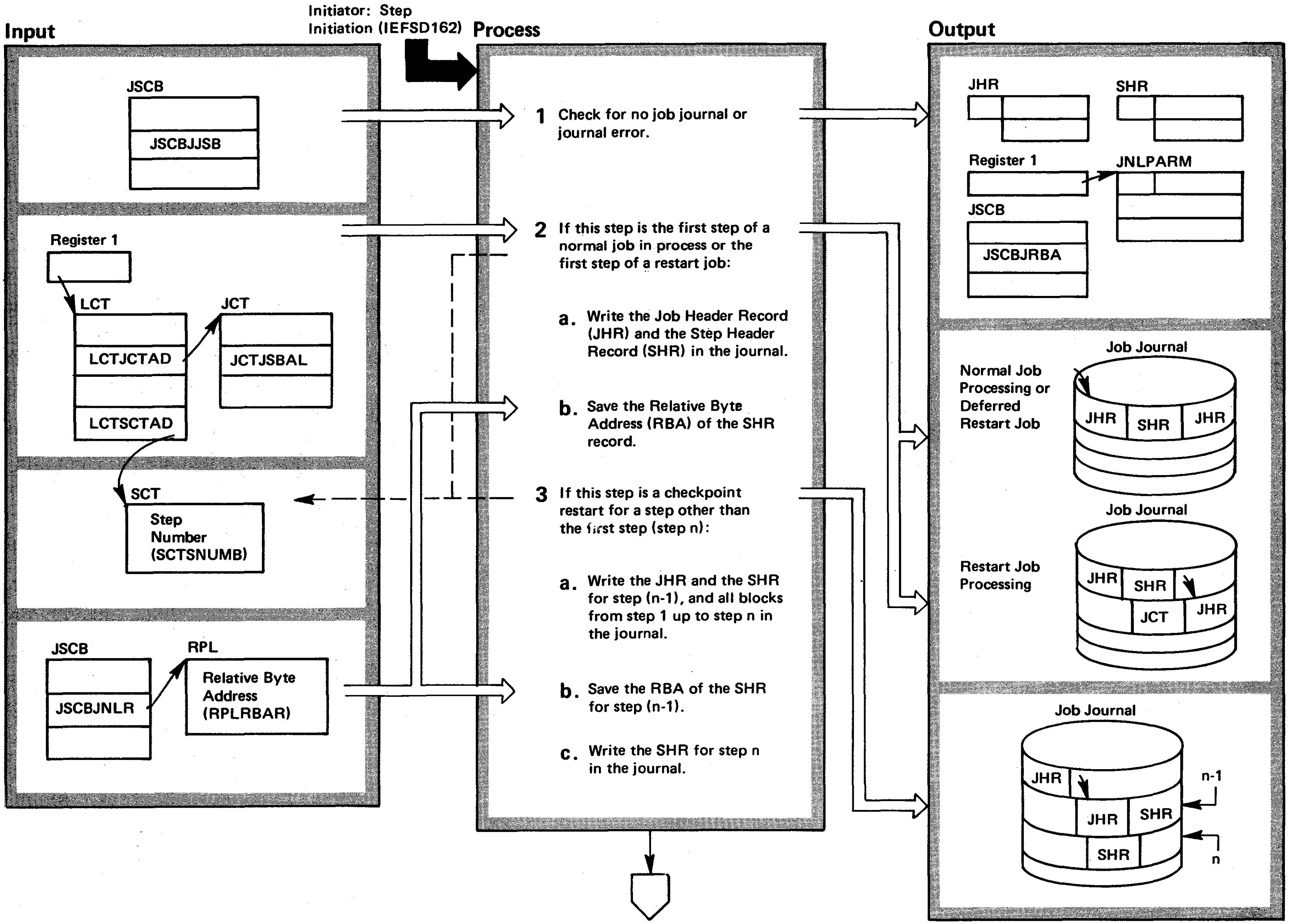
6 Pass information to interpreter.

* If request is other than 'Write' or 'Write Assign', omit steps 3 and 5.

Diagram 17-11. Restart Interface Processing (IEFXB602) (Part 2 of 2)

Extended Description	Module	Label
This routine builds a virtual address table (VAT) to be used by the journal merge routine during SWA reconstruction processing.		
1 The VAT is an 800-byte table. The JSCB pointer to the VAT is constructed.	IEFXB602	VATBUILD
2 If either a 'write' or a 'write/assign' function is requested, the routine determines the number of entries to be made in the VAT after the SWA manager performs its function.		
3 The routine uses the external parameter area (EPA) to get the SWA virtual address (used for the initial VATNVA field in the VAT) and the block ID if one exists.		
4 The routine uses the IEFQMREQ macro instruction to give control to module IEFQB550. The operation field, QMPOP, indicates whether the function is a 'write,' a 'write/assign,' an 'assign,' or a 'read' operation. The EPA updating occurs only for a 'write/assign' or an 'assign' operation.		
5 The relative block number (RBN) is placed in the VAT for each entry, and the block ID field of the VAT is filled in if not already there.		
6 The routine returns control to the interpreter. The output to the interpreter is the same as the input from the interpreter but with additional information that was filled in by the SWA manager routine.		

Diagram 17-12. Building Step Header Record for Job Journal (IEFXB604) (Part 1 of 4)



1 Check for no job journal or journal error.

2 If this step is the first step of a normal job in process or the first step of a restart job:

a. Write the Job Header Record (JHR) and the Step Header Record (SHR) in the journal.

b. Save the Relative Byte Address (RBA) of the SHR record.

3 If this step is a checkpoint restart for a step other than the first step (step n):

a. Write the JHR and the SHR for step (n-1), and all blocks from step 1 up to step n in the journal.

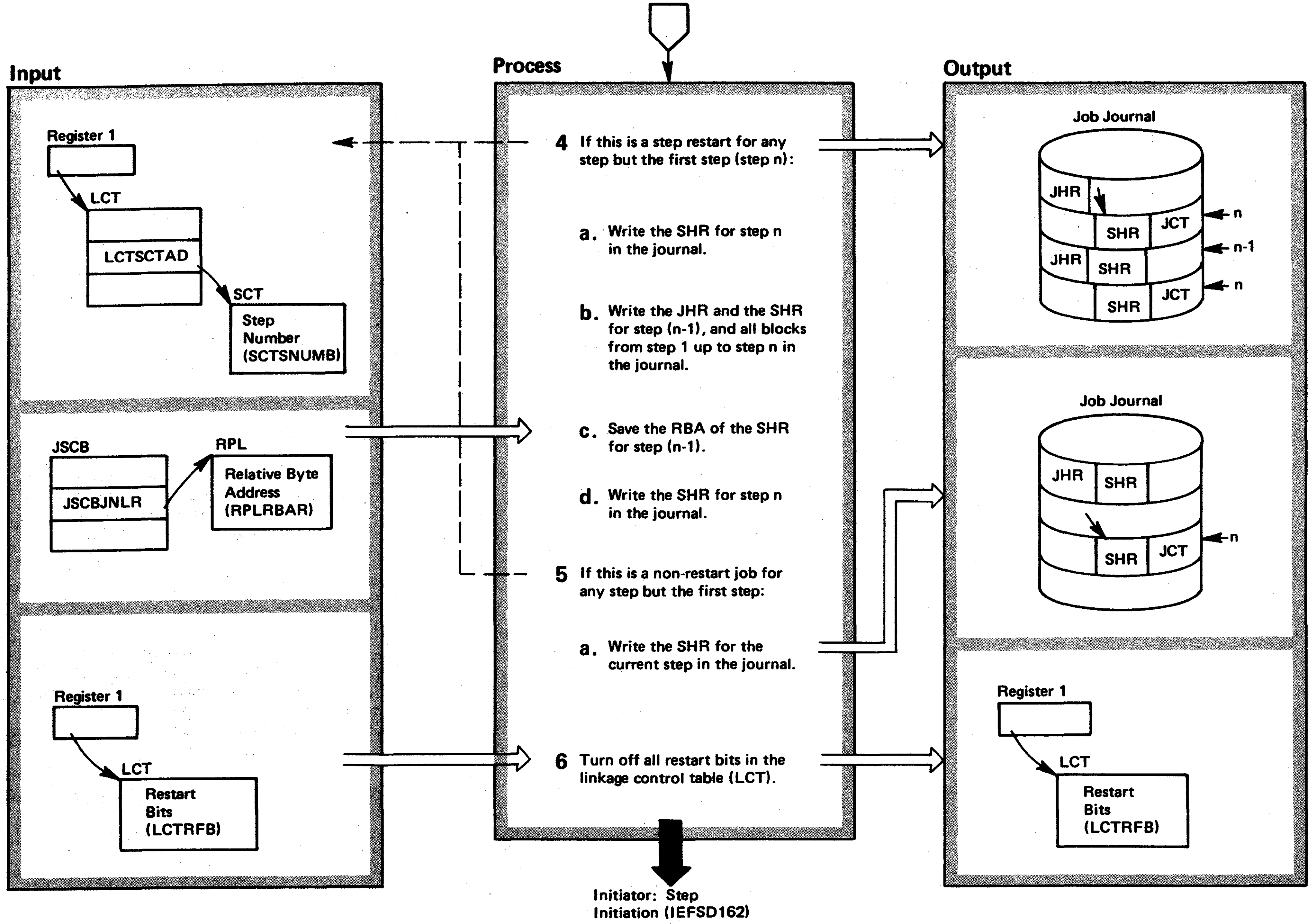
b. Save the RBA of the SHR for step (n-1).

c. Write the SHR for step n in the journal.

Diagram 17-12. Building Step Header Record for Job Journal (IEFXB604) (Part 2 of 4)

Extended Description	Module	Label
1 Check the JSCBJNLF and JSCBJNLE bits in the JSCBJJSB field to determine whether there is no job journal or there is a journal error. Change job state in the JSCB to "in allocation."	IEFXB604	
2 If the failing step is the first step of the job or if it is the first step of a non-restart job, write the JHR and the SHR in the journal. Set JCTJSBAL to indicate that the job state is "in allocation", and write the job control table (JCT) in the journal for all jobs except automatic checkpoint restart jobs, to record the "in allocation" status.		
3 If the failing step (step n) is any step but the first step of an automatic checkpoint job, write the JHR and the SHR for step n-1 in the journal, and all the control blocks of all previous steps up to but not including the failing step. This information must be saved to permit a possible subsequent restart. Finally, write the SHR for step n in the journal.		

Diagram 17-12. Building Step Header Record for Job Journal (IEFXB604) (Part 3 of 4)



4 If this is a step restart for any step but the first step (step n):

a. Write the SHR for step n in the journal.

b. Write the JHR and the SHR for step (n-1), and all blocks from step 1 up to step n in the journal.

c. Save the RBA of the SHR for step (n-1).

d. Write the SHR for step n in the journal.

5 If this is a non-restart job for any step but the first step:

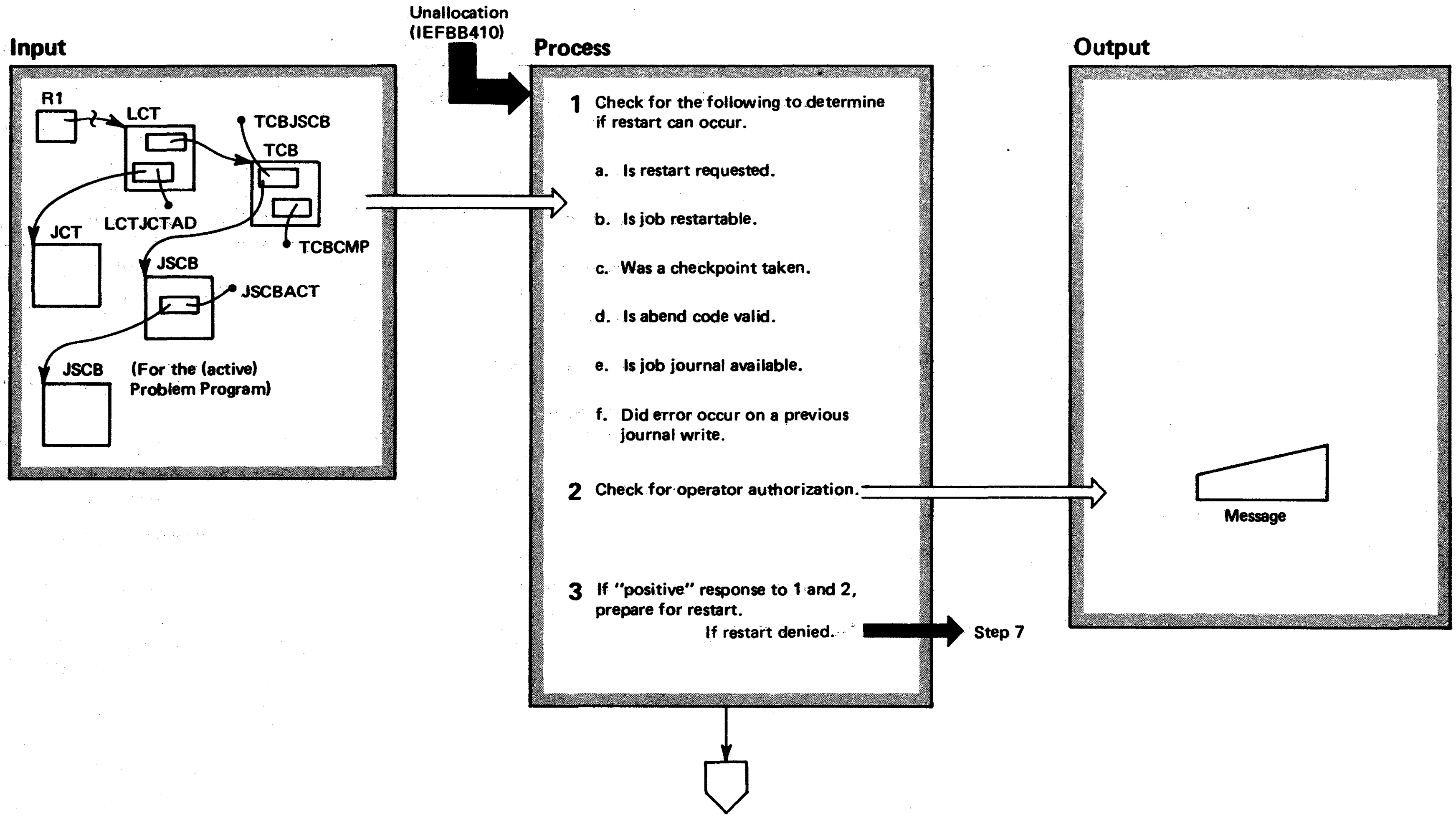
a. Write the SHR for the current step in the journal.

6 Turn off all restart bits in the linkage control table (LCT).

Diagram 17-12. Building Step Header Record for Job Journal (IEFXB604) (Part 4 of 4)

Extended Description	Module	Label
4 If the failing step is any step but the first step of a step restart job, write the SHR for step n, and then write the JHR and the SHR for step n-1 in the journal. Write all the control blocks for steps 1 thru n-1, and lastly write the SHR for step n in the journal once again. As in step 3, this information is saved to permit a possible restart.	IEFXB604	
5 If the failing step is any step but the first step of a non-restart job, write the SHR of the current step in the journal. Again, this information is saved to permit a possible restart.		
6 Turn off all restart bits (LCTRFBSM, LCTRFBCR, and LCTRFBDC) before exiting.		

Diagram 17-13. Preparing an Abended Job Step for Restart (IEFRPREP) (Part 1 of 4)



1 Check for the following to determine if restart can occur.

a. Is restart requested.

b. Is job restartable.

c. Was a checkpoint taken.

d. Is abend code valid.

e. Is job journal available.

f. Did error occur on a previous journal write.

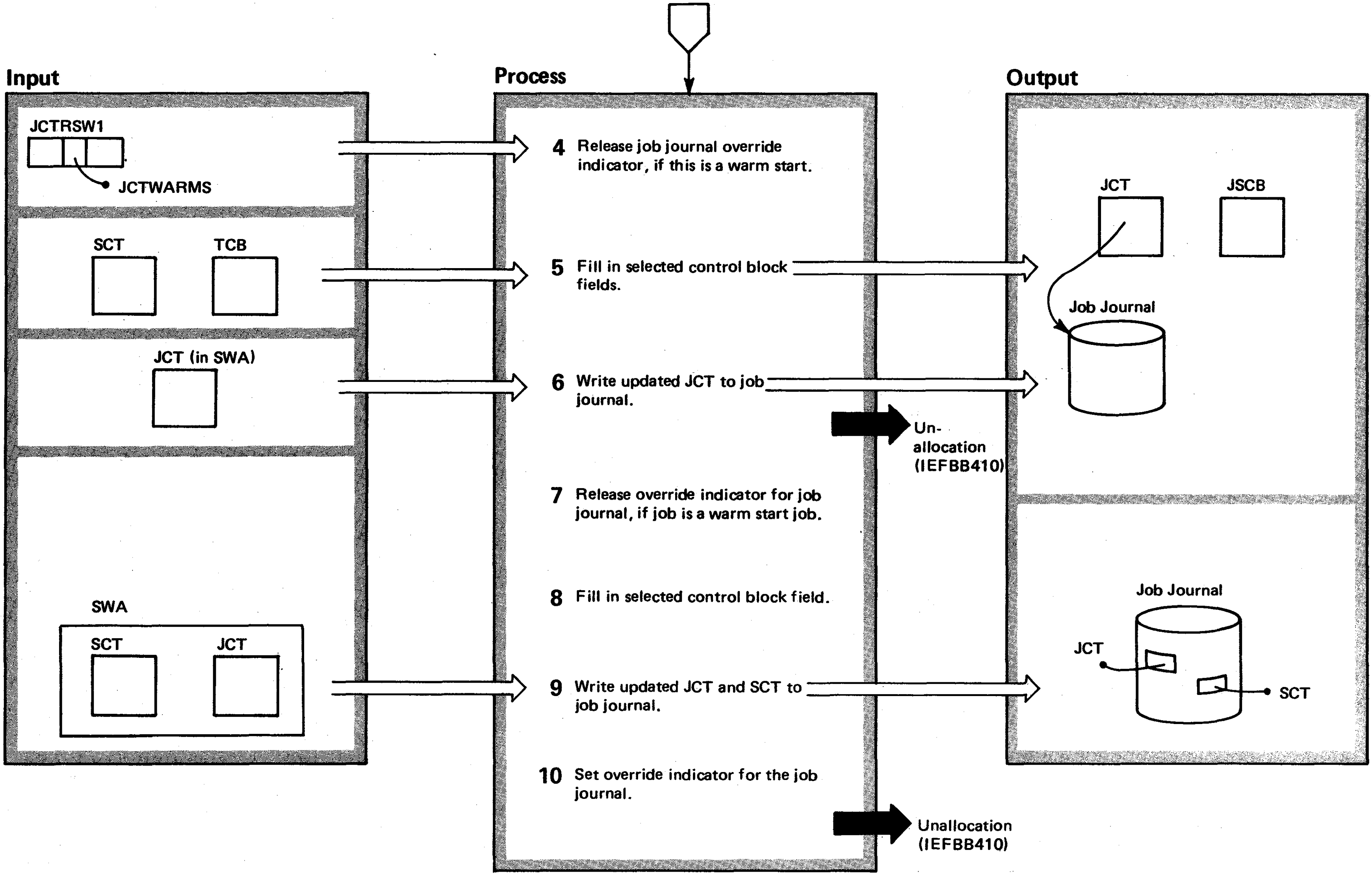
2 Check for operator authorization.

3 If "positive" response to 1 and 2, prepare for restart.
If restart denied.

Diagram 17-13. Preparing an Abended Job Step for Restart (IEFRPREP) (Part 2 of 4)

Extended Description	Module	Label
This routine determines if an abended (abnormally terminated) job step task can be restarted. If it can, the routine prepares the task for a restart.		
1 a) Test JCTNORST.	IEFRPREP	IEFPREP
b) The job cannot restart if, after a checkpoint was taken, dynamic allocation routines have scratched a dynamically allocated data set that is used by the job.		
c) Test JCTCKFT.		
d) Check TCBCMP against the IEFYRCDS table of eligible restart ABEND codes. This table is contained in a SYSGEN module.		JNL03
e) The job journal must have been specified as a SYSGEN option or at IPL time. Test JSCBJNLF.		JNL02
f) Test JSCBJNLE. The routine IEFXB500 will have set this flag if an error occurred during a previous writing of information to the job journal.		JNL01
2 Routine issues a WTOR macro instruction for the operator to give decision regarding a restart.		RP130
3 For any negative response in steps 1 and 2, go to step 7.		

Diagram 17-13. Preparing an Abended Job Step for Restart (IEFRPREP) (Part 3 of 4)



4 Release job journal override indicator, if this is a warm start.

5 Fill in selected control block fields.

6 Write updated JCT to job journal.

7 Release override indicator for job journal, if job is a warm start job.

8 Fill in selected control block field.

9 Write updated JCT and SCT to job journal.

10 Set override indicator for the job journal.

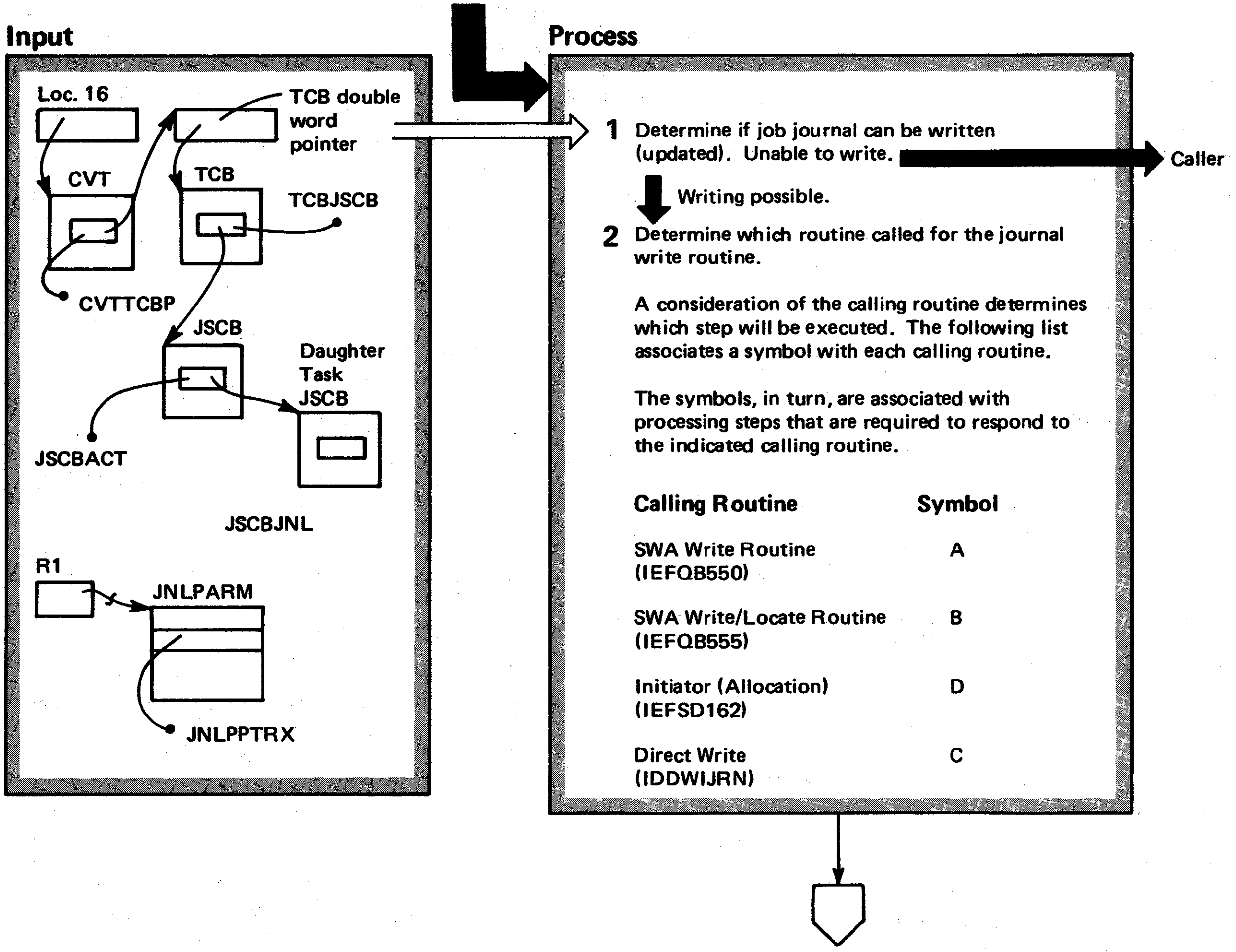
Diagram 17-13. Preparing an Abended Job Step for Restart (IEFRPREP) (Part 4 of 4)

Extended Description	Module	Label
4 The override indicator in the JSCB is set off to allow writing on the journal if the job is a warm start job. Otherwise, the indicator is already off at entry time.	IEFRPREP	
5 Field indicators affected: ● JCTACODE (same information as TCBCMP). ● JCTCKPTR (if checkpoint restart). ● JCTSTEPR (if step restart). ● JCTSCT (with value from SCTSNUMB). Based on operator reply, either hold job or re-enqueue job for immediate restart. Save restart step's SCT pointer (in field JCTSSTR). ● JCTJSBEX and JSCBJSBX (same information; that is, the job is executing). ● JCTJSBTM and JSCBJSBT (same information; that is, the job is terminating).		
6 This prohibits writing on the job journal (see step 4).	IEFXB500	
7 Set override bit as in step 4 to allow updating of job journal.	IEFRPREP	

Extended Description	Module	Label
8 Fields (indicators) affected: ● JCTCKPTR (see step 5) ● JCTSTEPR (see step 5) ● JCTRESTT (no-restart indicator) ● JCTACODE (see step 5) ● JCTABEND ● SCTONLYC (condition code, for use by allocation) ● SCTABEND		
9 The routine writes the JCT and SCT to the job journal.	IEFPREP IEFQB550 IEFXB500	CABEND2
10 The routine sets the override bit on to suppress further writing to the job journal until the job has restarted.	IEFRPREP	

Diagram 17-14. Writing Blocks to the Job Journal (IEFXB500) (Part 1 of 4)

- Initiator (IEFSD162),
- SWA Manager (IEFQB550 or IEFQB555),
- Step Header Create (IEFXB604), or Direct Write (IDDWIJRN)



1 Determine if job journal can be written (updated). Unable to write. → Caller

Writing possible.

2 Determine which routine called for the journal write routine.

A consideration of the calling routine determines which step will be executed. The following list associates a symbol with each calling routine.

The symbols, in turn, are associated with processing steps that are required to respond to the indicated calling routine.

Calling Routine	Symbol
SWA Write Routine (IEFQB550)	A
SWA Write/Locate Routine (IEFQB555)	B
Initiator (Allocation) (IEFSD162)	D
Direct Write (IDDWIJRN)	C

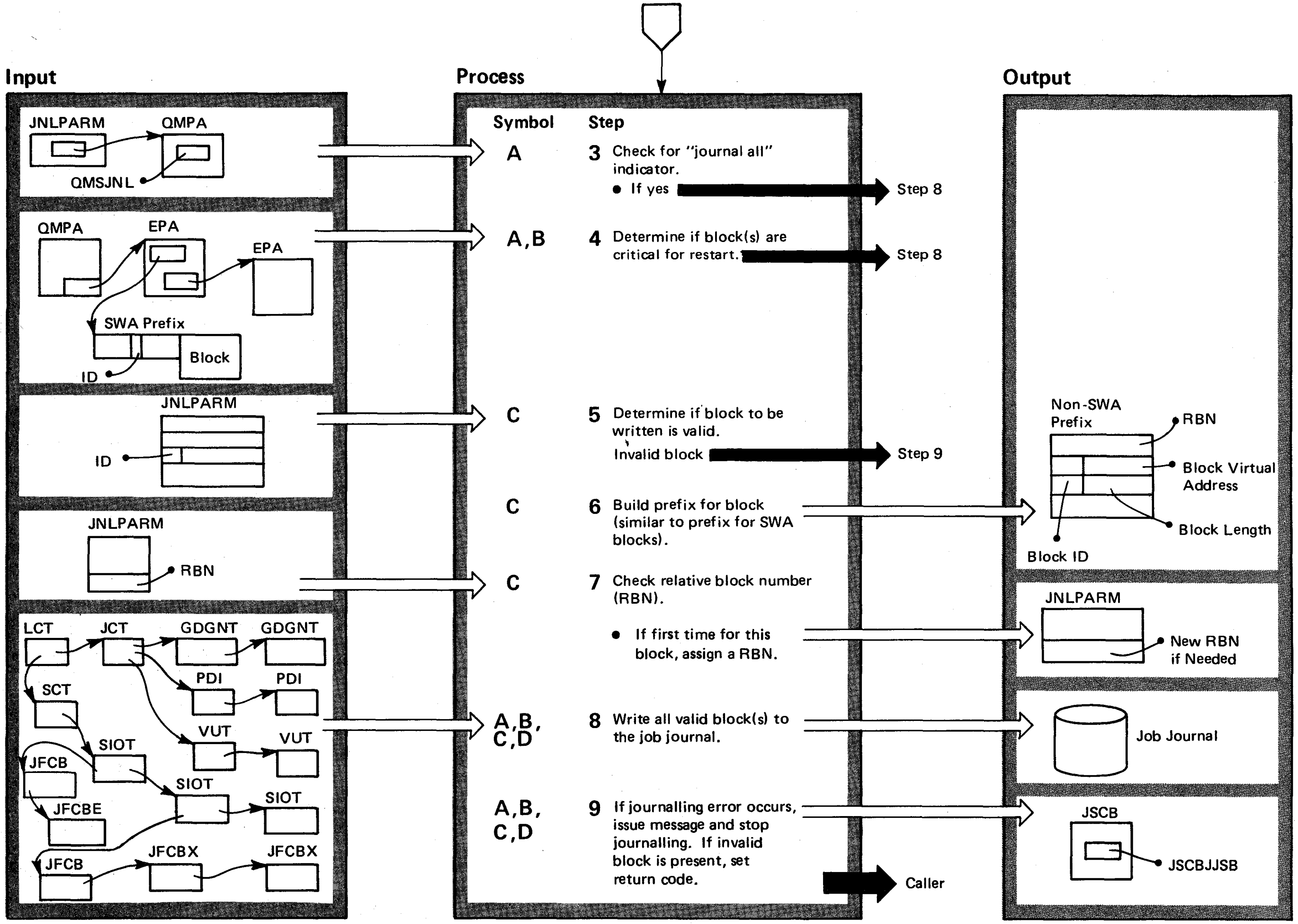
Diagram 17-14. Writing Blocks to the Job Journal (IEFXB500) (Part 2 of 4)

Extended Description	Module	Label
This routine writes (updates) critical control blocks to the job journal for restart or termination preparation processing.		
1 Flags (JSCBJNLN, JSCBJNLF, and JSCBJNLE) in the JSCB field JSCBJNL indicate if a journal override condition exists, if a job journal exists, or if an error exists on the journal (from a previous 'write' situation).	IEFXB500	IEFXB500
2 The first field, JNLPCALL, of the parameter list at JNLPARM contains the indicator tested at this point.		

The second word of the journal parameter list contains the value indicated below:

Calling Routine	*Value*
SWA Write	QMPA address
SWA Write/Locate	External Parameter Area (EPA) chain address
Initiator	Linkage Control Table (LCT) address
Direct Write (of non-SWA blocks)	Extension Block Address (In this case, a chain of control blocks can exist. Each block contains the address of the block to be journaled, its identifier, its length, and the RBN (relative block number).)

Diagram 17-14. Writing Blocks to the Job Journal (IEFXB500) (Part 3 of 4)



Symbol	Step
A	3 Check for "journal all" indicator. ● If yes → Step 8
A,B	4 Determine if block(s) are critical for restart. → Step 8
C	5 Determine if block to be written is valid. Invalid block → Step 9
C	6 Build prefix for block (similar to prefix for SWA blocks).
C	7 Check relative block number (RBN). ● If first time for this block, assign a RBN.
A,B, C,D	8 Write all valid block(s) to the job journal.
A,B, C,D	9 If journalling error occurs, issue message and stop journalling. If invalid block is present, set return code. → Caller

Diagram 17-14. Writing Blocks to the Job Journal (IEFXB500) (Part 4 of 4)

Extended Description	Module	Label
3 *SWA Write (and Move):*		
If this indicator (QMSJNL) is on, the routine journals the control block regardless of the job state (see step 4). Only the terminator modules use this indicator.	IEFXB500	SWAWRT
4 The journal-write module contains a list of all critical control blocks for the four main job states: • Interpreter state • Allocation state • Problem program execution state • Termination state This list (or template) also indicates critical non-SWA ('direct write') control blocks (e.g., step header record, VIO data sets such as data set page control table header and VIO data set control block).		SWAWRT SWAWRTL
The ID of each block that is scheduled to be journalled is found in the SWA prefix for the block. The ID is matched with the list in the template for the particular job state involved, to determine if the block is critical for restart.		
The format of the EPA appears below:		

For SWA locate:

↑ Block to be written or read	4
↑ SWA virtual address	3
Block ID	1
Length of block written or read	4
0 or ↑ next EPA	4

For SWA move:

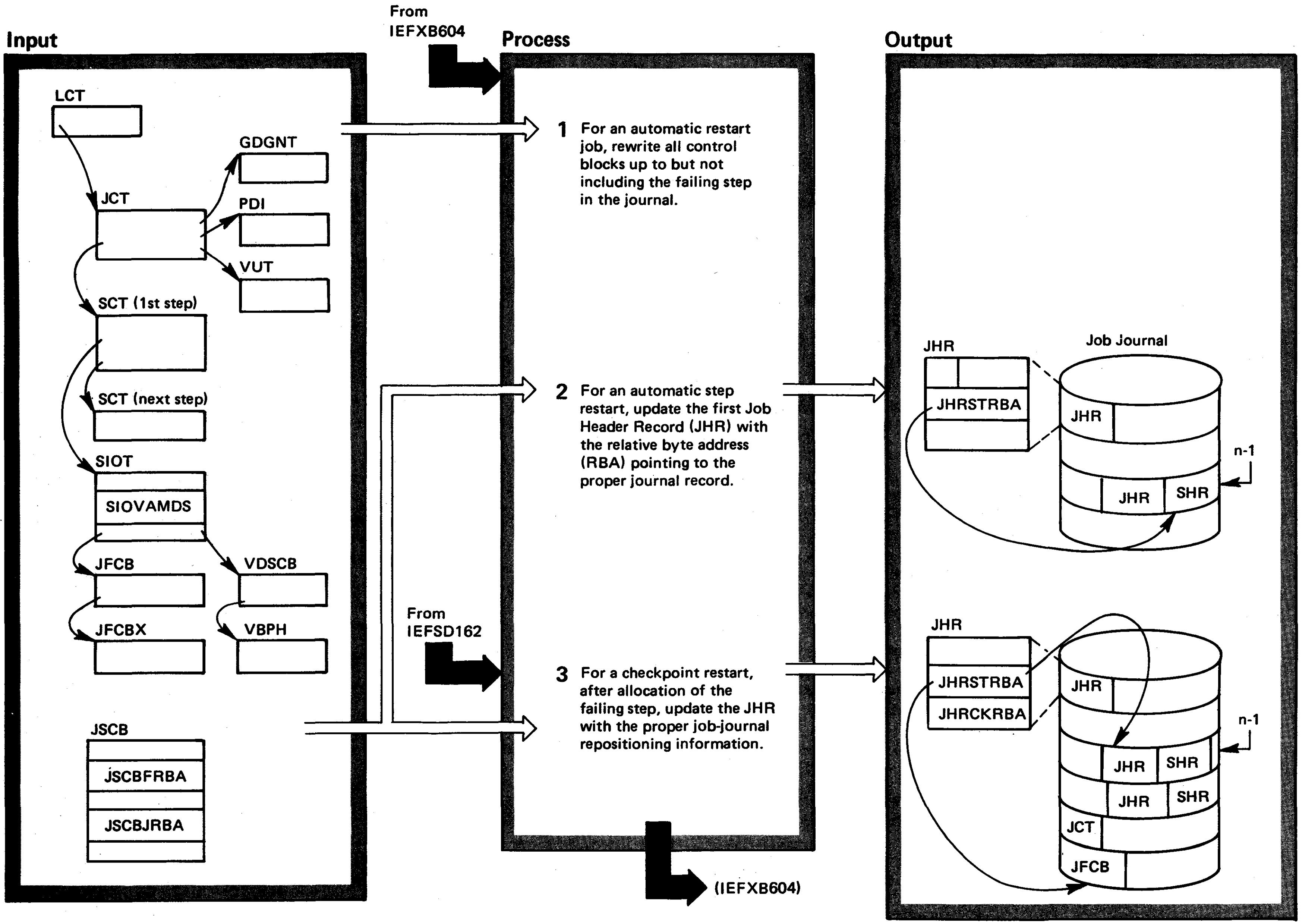
↑ Buffer to which block is read or from which block is written	4
↑ SWA virtual address from which block is read or to which block is written	3
Block ID	1

For SWA assign:

SWA virtual address (from SWA manager)	3
0	1
0	4

Extended Description	Module	Label
5 *Direct Write:*		
The parameter list contains the block ID. This ID is matched against the template as in step 4.		DRTWRT
6 The routine builds the prefix before it journals the block.		
7 For the first journalling of the block, the RBN (in the fourth word of the parameter list) is zero. This routine assigns a unique RBN that will be used if subsequent journalling of the block is required.		
8 Except when journalling control blocks at allocation-time, the order in which the blocks are journalled depends on the order in which they are updated on SWA. At allocation-time, the blocks are journalled in the following order: JCT, SCT (of current step), one or more GDGNT (generation data group name table), one or more PDI (passed data set information), one or more VUT (volume unload table), first SIOT plus JFCB plus JFCBX (one or more) or JFCBE, additional SIOT-JFCB-JFCBXs chains and SIOT-JFCB-JFCBE chains.	IEFXB500	RUNCHAIN
For the allocation-time journalling, the pointer chain beginning at the LCT gives block addresses.		
When writing to the journal, the routine uses the request parameter list (RPL) that was built by the SWA create routine (IEFXB600). It passes the RPL to the subsystem routine (JES2) as a parameter in the PUT macro instruction that results in a JES2 routine putting the block on the journal.		JOURNAL
9 Error indicator in the JSCB is set, a WTP macro instruction is used to issue a message, a return code is placed in the parameter list JNLPARM, and control returns to the caller.		ERRMSG

Diagram 17-15. Journal for Restarted Jobs (IEFXB500) (Part 1 of 2)



1 For an automatic restart job, rewrite all control blocks up to but not including the failing step in the journal.

2 For an automatic step restart, update the first Job Header Record (JHR) with the relative byte address (RBA) pointing to the proper journal record.

3 For a checkpoint restart, after allocation of the failing step, update the JHR with the proper job-journal repositioning information.

Diagram 17-15. Journal for Restarted Jobs (IEFXB500) (Part 2 of 2)

Extended Description	Module	Label
1 For an automatic step restart or a checkpoint restart, write all critical control blocks from step one up to but not including the failing step in the journal. Blocks are written as if they were all part of step n-1, where n is the failing step's number. Critical control blocks are: JCT, PDI, GDGNT, VUT, SCT, SIOT, JFCB, JFCBX, and JFCBE. VIO blocks are also written if SIOVAMDS is on.	IEFXB500	RUNCHAIN RUNSIOT
2 For an automatic step restart, a GET macro with update is issued, using the relative byte address (RBA) saved in JSCBFRBA by the merge routine. The job header record (JHR) is then updated by inserting an RBA which was saved in JSCBJRBA by IEFXB604. This RBA points to the SHR for step n-1 and will be used by the merge routine after restart.		JHRUPDT
3 For a checkpoint restart, after allocation of the failing step, update the JHR by inserting JSCBJRBA and the RBA returned from the last PUT macro. This RBA will be used at restart-time to reposition the journal data set.		JHRUPDT

Index

- ABDUMP initialization (See **OS/VS2 System Initialization Logic**)
- ABEND
  - in SWA manager move mode 3-265
- ABENDed jobstep, preparing for restart 3-516
- abnormal end of SMF writer function 3-460
- abnormal termination of log task 3-476
- ACB (access control block)
  - creation for pseudo access method 3-178
  - in converter/interpreter interface 3-178
  - in JFCB housekeeping control 3-318
  - in JLOCATE 3-333
  - in pseudo access method 3-184
  - in subsystem initiation message writer 3-186
  - in SWA create interface 3-216
- access control block (see ACB)
- access method, pseudo (see pseudo access method)
- account tables (see ACT)
- ACT (account tables)
  - in interpreter 3-254
  - in job deletion 3-208
  - in step deletion 3-208
- action queue, deferred, in SRM 3-28
- action/algorithm scheduling in SRM 3-23,3-23.2,3-23.3 **(VS2.03.807)**
- active subsystem
  - notification 3-172
  - requests 3-172
- addresses, virtual in SWA, updating 3-504
- affinity (see CPU affinity)
- affinity processor
  - function 3-304-3-305
- affinity removed
  - function 3-304, 3-298, 3-368
- affinity requests, allocating 3-300
- ALCWA (allocation work area)
  - in allocate request to unit 3-304, 3-306
  - in allocating offline devices 3-376
  - in allocation via algorithm 3-348
  - in common allocation cleanup 3-378
  - in common allocation control 3-280
  - in demand allocation 3-355
  - in fixed device control 3-294
  - in generic allocation control 3-338
  - in nonspecific volume allocation control 3-308
  - in recovery allocation 3-358
  - in specific volume allocation 3-298
- algorithm, allocating via 3-348
  - cover/reduce algorithm 3-349
  - interface to SRM 3-351
  - multi-unit requests 3-349
  - UCB candidates list 3-351
- algorithm tables
  - in allocate request to unit 3-304-3-306
  - in allocating offline devices 3-366
  - in allocation via algorithm 3-339
  - in common allocation cleanup 3-378-3-379
  - in demand allocation 3-355
  - in generic allocation control 3-342, 3-344
  - in multi-unit request processing 3-366
  - in nonspecific volume allocation control 3-312
  - in recovery allocation 3-358
  - in specific volume allocation 3-298
- algorithms, SRM 3-30
  - in periodic entry point scheduling 3-32
  - scheduling 3.23.2 **(VS2.03.807)**
- alias-named data sets, processing 3-332-3-333
- all active subsystem notification 3-172-3-173
- allocate catalog control
  - function 3-336
- allocate from groups picked by algorithm (see IEFAB478 object module)
- allocate function control (see IEFDB410 object module)
- allocate request to unit 3-302
- allocate via the algorithm
  - function 3-348
- allocate VIO data sets
  - function 3-280-3-281
- allocate within a generic
  - function 3-342, 3-344, 3-346
- allocating affinity requests
  - in allocating requests to units 3-304
  - in allocating requests to specify volumes 3-300
  - in allocation recovery 3-358
- allocating direct-access request 3-294
- allocating non-specific volume requests 3-308
- allocating permanently resident volume requests 3-294
- allocating reserved volume requests 3-294
- allocating system log 3-472
- allocating teleprocessing requests 3-286
- allocating a unit
  - building a VM&V request block 3-302
  - unloading a volume 3-302
- allocating volumes and units to requests 3-280
- allocation common ESTAE exit routine (IEFAB4ED) **(VS2.03.804)**
  - error processing 3-291-3-413 **(VS2.03.804)**
- allocation environment, current, providing information about 3-422
- allocation message routine
  - function 3-380-3-381
- allocation queue manager (see IEFAB4FA object module)
- allocation queue manager request block (see AQMRB)
- allocation/unallocation 3-269
  - affinity request 3-300
  - catalog search 3-334
  - common allocation clean-up 3-378
  - common control (see also common allocation control) 3-280
  - common control for unallocation 3-430
  - common unalloaction functions 3-402
  - DD function control 3-322
  - ddname allocation
    - function 3-412, 3-428
  - demand allocation 3-355
  - device, offline, allocation of 3-366
  - disposition processing 3-440
  - dynamic allocation control 3-414
  - dynamic environment, current, providing information about 3-423
  - dynamic information retrieval 3-422
  - dynamic unallocation control 3-416
  - fixed device control 3-294
  - function map, building (in initiator/unallocation interface) 3-404
  - generic devie type control 3-317
  - installation routine verification (in SVC 99 control) 3-412
  - interface to initiator
    - allocation 3-396
    - unallocation 3-402
  - introduction to allocation/unallocation 3-269
  - ISAM request error checking (in common allocation cleanup) 3-380
  - JES2 notified of unallocation of data set and associated resources 3-438-3-439
  - JFCB housekeeping control 3-314
  - JLOCATE, functions of 3-334
  - major functions of allocation/unallocation 3-271
  - mount equalization for MSS volumes 3-291, 3-350, 3-370
  - MSS interface to obtain preferred UCB list to update UCB candidate list 3-371, 3-377
  - offline device allocation 3-366
  - overview of allocation/unallocation 3-269
  - passed data set information, obtaining 3-334
  - processing job step (SVC 99 control) 3-412
  - reattempted allocation, criteria for 3-378
  - recovery 3-358
  - remove in-use attribute 3-424

  remove in-use processing 3-424
  requests to unit 3-302
  retry 3-378
  SRM interface
    in common allocation clean-up 3-382
    in non-specific volume allocation 3-312
  step allocation control
    function 3-398
  step initiator
    in initiator/allocation interface 3-396
    in step initiation 3-200
  SVC 99 control 3-412
  termination error, processing 3-382
  unit unallocation 3-444
  via algorithm 3-348
  volume list (in disposition processing, IEFAB4A2) 3-440
  volume mount and verify (VM&V)
    interface 3-386
    processing 3-390
allocation work area (see ALCWA)
ALTCPREC SYSEVENT code (33)
  processing in SRM SYSEVENT code processor 3-18
alternate disposition message routine
  function 3-443
alternation of SWA subpool 3-267
AMWA (access method work area)
  in converter/interpreter interface 3-178
  in pseudo access method 3-184
  in subsystem initiation 3-176
analyzer, MF/1 syntax 3-86
APF (see authorized program facility)
APG (automatic priority group)
  changing priorities in 3-62
  in step initiation 3-205
AQMRB (allocation queue manager request block)
  in generic allocation control 3-338
ASCB (address space control block)
  in CPU management (SRM) 3-62
  in dynamic allocation control 3-414
  in SMF cross-memory post error exit 3-460
  in SRM interface 3-6
  in SRM service routine (IRARMSRV) 3-9.2 **(VS2.03.807)**
  in step initiation 3-200
  in storage management (IRARMSTM) 3-46 **(VS2.03.807)**
  in storage management (SRM) 3-46
  in SVC 99 control 3-412
  in swappable user evaluation (IRARMWM2) 3-70 **(VS2.03.807)**
  in swap-in
    control 3-40
  in swap-out
    control 3-42
  in SYSEVENT processing in SRM SYSEVENT code processor 3-11
  in user evaluation (IRARMCVL) 3-43.4 **(VS2.03.807)**
ASCB priority
  in step initiation 3-200
ASID (address space identifier)
  in job unallocation 3-410
ASM (see auxiliary storage manager)
ASMCNSTS SYSEVENT code (27)
  processing in SRM SYSEVENT code processor 3-17
ASMVT
  in interval measurement gathering routine for paging 3-122
  in resource monitor periodic monitoring (IRARMRM1) 3-66 **(VS2.03.807)**
  in storage management (IRARMSTM) 3-46 **(VS2.03.807)**
  in storage management (SRM) 3-48
ASSIGN
  processing 3-264
ASSIGN/LOCATE processing 3-266
ASSIGN/START processing 3-264
assignment of CPU task affinity
  function 3-201, 3-199
ASXB (address space extension block)
  in SMF cross memory post error exit 3-460
asynchronous exits (see exit asynchronous)
ATCA
  in allocation/volume mount and verify (VM&V) interface 3-388
  in volume mount and verify (VM&V) 3-394
attributes, user (see VAPS)
automatic checkpoint/restart
  processing 3-498
  SWA create interface 3-216
automatic priority group (see APG)
automatic step restart 3-500
auxiliary page shortage 3-48
auxiliary storage manager I/O request area (see AIA)
available queue element (see AQE)
AVQLOW SYSEVENT code (23)
  processing in SRM storage management (IRARMSTM) 3-49
  processing in SRM SYSEVENT code processor 3-16
AVQOK SYSEVENT code (24)
  processing in SRM storage management (IRARMSTM) 3-49
  processing in SRM SYSEVENT code processor 3-16
AVR (automatic volume recognition)
  in generic allocation
    function 3-340, 3-341

BASEA (see MSRDA)
batch allocation
  in common allocation control 3-280
BRINGIN SYSEVENT (44)
  processing in SRM SYSEVENT processor 3-21
broadcast data set (see SYS1.BRODCAST)
build eligible devices list (EDL)
  function 3-282-3-283
building step header record for job journal 3-512

CAT (channel availablity table)
  in MF/1 channel sampling module 3-140
catalog, allocating (see SVC 99) 3-337
catalog, implied by data set names 3-334
catalog, master 3-335
catalog, private, searching 3-334
catalog processing 3-204
catalog searching 3-334
catalog unallocation control
  function 3-336, 3-318, 3-432
cataloged procedures 3-232
CCT
  in CPU load balancing swap analysis 3-66
  in CPU management (IRARMCPM) 3-62 **(VS2.03.807)**
  in CPU management (SRM) 3-62
  in resource monitor periodic monitoring (IRARMRM1) 3-66 **(VS2.03.807)**
  in storage management (SRM) 3-48
change ddname/JES3 exit (IEFDB4FB)
  function 3-418-3-419
channel, logical
  analysis of I/O load 3-56
  imbalance 3-54
  in I/O mangement 3-54
channel, measurement
  MF/1 initialization for 3-100
channel availability table (see CAT)
channel data collected by MF/1
  interval 3-130
  sampling 3-140
  second CPU 3-142
checkpoint data set 3-486
checkpoint/restart 3-202, 3-483
  ABENDed job step, preparing for restart 3-516
  automatic
    in SWA create interface 3-216
    processing 3-498
  data set descriptor records processing 3-486, 3-483
  deferred 3-216
  dynamic allocation interface 3-486
  header record 3-489

  - step initiation 3-202
  - job journal writing 3-520
- CIB (command input block)
  - in measurement facility control 3-80
- CIB (command input buffer)
  - in job initiation 3-196
- clean-up processing
  - in common allocation 3-378
  - in volume mount & verify (VM&V) 3-394
- clock, TOD (see TOD clock)
- coefficients, resource (see resource factor coefficient)
- collect data for MF/1 (IRARMWR3) 3-73.8 **(VS2.03.807)**
- command, reconfiguration (see reconfiguration commands)
- command processing
  - in the input stream 3-230
- commands
  - in the input stream 3-230
  - WRITELOG START 3-466
- comment or continuation statement processing 3-226
- common allocation clean-up
  - called by 3-378
  - common allocation parameter list, building 3-378
  - functions 3-378
  - requests excluded (see also requests, allocation) 3-378
- common allocation control
  - called by 3-280
  - fixed device control 3-290
  - function 3-280
  - function map 3-430
  - generic allocation control, use with 3-288
  - order of allocation 3-280
  - parameter list, function map 3-280
  - recovery, occasions for use 3-288
  - serialization of 3-282
  - waiting for devices 3-280
- common allocation parameter list
  - building 3-378
- common request router
  - processing
    - function 3-172
  - request block construction 3-412
- common unallocation functions 3-430
- comparator, clock (see clock comparator)
- COMWA (converter/interpreter common work area)
  - converter use of in
    - identifying verbs or JCL statements 3-226
    - initialization 3-224
    - processing commands in the input stream 3-230
    - processing in-stream and cataloged procedures 3-232
    - termination 3-242
  - interpreter use of in initialization 3-246
- concatenation, dynamic
  - function 3-418
- CONFIGCH SYSEVENT code (29)
  - processing in SRM SYSEVENT code processor 3-17
- continuation statement processing 3-226-3-229
- control, common allocation (see common allocation control)
- control blocks (see data areas)
- conversion of bit mask
  - function 3-200-3-201
- converter (see also interpreter)
  - command verb validation routine
    - function 3-230, 3-228
  - comment or continuation validation routine
    - function 3-226
  - converting statements to internal text 3-236-3-239
  - entering defaults into internal text 3-240-3-241
  - error messages
    - processing by subsystem initiation message writer 3-186-3-187
  - get routine
    - function 3-226
  - identifying verbs on JCL statements 3-226-3-229
  - initialization
    - function 3-224-3-225
  - instream procedure routines
    - function 3-232
  - pre-scan routine
    - function 3-228
  - processing commands in the input stream 3-230-3-23
  - processing in-stream and cataloged procedures 3-232-3-233
  - processing symbolic parameters 3-234-3-235
  - pseudo-access method 3-182-3-185
  - scan routine
    - function 3-236, 3-240, 3-234, 3-226
  - SWA manager interface routine
    - function 3-233
  - symbolic parameter routine
    - function 3-234
  - termination routine
    - function 3-242-3-243
  - test and store utility routine
    - function 3-252
  - use by master subsystem 3-178-3-181
  - verb identifier routine
    - function 3-226, 3-228, 3-232, 3-238
- converter/interpreter
  - interface 3-178-3-181
  - operator message module
    - function 3-258-3-259
- converting an allocation in dynamic allocation control 3-414
- COPYDMDT SYSEVENT code **(VS2.03.807)**
  - code processor 3-11 **(VS2.03.807)**
  - processing in SRM SYSEVENT 3-22 **(VS2.03.807)**
- corequisite publications iv (preface)
- count table
  - in allocation via algorithm 3-350
  - in common allocation control 3-280
  - in demand allocation 3-355
  - in fixed device control 3-294
  - in nonspecific volume allocation control 3-312
  - in specific volume allocation 3-300
- cover/reduce algorithm
  - function 3-366, 3-348, 3-374, 3-280
- CPU activity initialization in MF/1 3-96
- CPU affinity
  - in job initiation 3-199
  - in step initiation 3-201
- CPU load balancing swap analysis 3-66
- CPU management in SRM 3-62
- CPU measurements in MF/1
  - in interval MG routine 3-118-3-121
- CPU utilization **(VS2.03.807)**
  - in CPU management (IRARMCPM) 3-63,3-64 **(VS2.03.807)**
- CRI (catalog return information)
  - in DD function control 3-322-3-330
  - in JLOCATE 3-334
- cross-memory posting of SMF error exit -460
- CSCB (command scheduling control blo
  - in allocation/initiator interface 3-
  - in initiator/allocation interface 398
  - in job initiation 3-196
  - in step initiation 3-200
  - in SWA create interface 3-6
- CSD (common system data are
  - in common allocation cont 3-282
  - in CPU management (IRA MCPM) 3-62 **(VS2.03.807)**
  - in MF/1 channel initializ on 3-100
- current allocation environ t, providing information about 3-422 or table)
- CVT (communication interface 3-396
  - in allocation/init mount and verify (VM&V)
  - in allocation/v
    - interface p header record for the job journal
  - in building
    - 3-512 equest router 3-172
  - in comm name assignment 3-188
  - in dat /allocation interface 3-396
  - in in or/unallocation interface 3-404
  - in val measurement gathering routine for workload
  - i 6
  - erging job journal to SWA 3-492
  - MF/1 channel initialization 3-100
  - MF/1 CPU activity initialization 3-96

n MF/1 device initialization 3-104
in MF/1 paging activity initialization 3-96
in MF/1 second CPU test channel sampling module 3-142
in MF/1 termination processor 3-110
in STAE exit processing for SMF 3-458
in subsystem determination 3-174
in subsystem interface 3-164
in switching SMF data sets 3-454
in unallocation/initiator interface 3-404
in volume mount and verify (VM&V) 3-394
in writing blocks to the job journal 3-520
in writing SMF records 3-450

DADSM
  allocation interface 3-304
  parameter list in allocate request to unit 3-304
  VM&V interface 3-386
DAIRFAIL (IKJEFFI8) failure in dynamic allocation 3-486 **(VS2.03.810)**
Data Area section 7-1855
  data control (IRBMFDTA) in MF/1 3-106
data set assignment 3-188
data set descriptor record processor (see also checkpoint/restart)
  in SWA create interface
    function 3-486, 3-216, 3-217
data set enqueue parameter list building
  function 3-198-3-199
data set name (see also DSN resolution)
  in data set tree processing 3-198-3-199
data set name assignment
  function 3-188
data set name resolution
  function 3-324
data set tree structure processing in job and step initiation
  function 3-198-3-199
data sets, releasing
  in common unallocation control 3-432-3-433
data sets, SMF 3-454
DCB (data control block)
  in converter/interpreter interface 3-178
  in data set descriptor records processing 3-486
  in switching SMF data sets 3-454
  in writing SMF records 3-450
  resolution in DD function control
    function 3-330-3-331
DD function control (IEFAB454)
  DCB resolution 3-331
  DISP resolution 3-332-3-333
  GDG (generation data group) processing 3-324-3-325
  processing
    function -322
  volume/unit resolution 3-328-3-331
DD preparation
  function 3-314-3-15
DD processing control
  function 3-324, 3-31
ddname allocation 3-428, -412
ddname and relative position number
  informing the JES3 subsystem 3-418-3-419
ddname search routine
  function 3-416, 3-418, 3-
DEB (data extent block)
  deconcatenation, dynamic
  step initiation, in 3-204
deconcatenation routine
  function 3-420
defaults, converter, entering into JCL i nal text 3-240
deferred action processor (IRARMCEN RM 3-28
deferred action queue **(VS2.03.807)**
  in deferred action process 3-28 **(VS2.0**
deferred restart job determination (in SWA
  interface) 3-217
deferring algorithms in SRM 3-23 **(VS2.03.80**
DELETE subroutine
  in SWA manager move mode 3-265
DELETE/LOCATE function in SWA manager loca de
  3-266-3-267

demand allocation
  processing
    function 3-355
  use with generic allocation 3-407
demand requests
  definition 3-355
  operator replies 3-375
  processing 3-375, 3-377
  volunit entry, updating 3-372, 3-373
DEQ macro instruction (see ENQ/DEQ/RESERVE routine)
determination of subsystem name 3-175
determine device requirements
  function 3-282-3-283
determining device requests
  for request not yet allocated 3-289
DEVALLOC SYSEVENT code (28)
  processing in SRM SYSEVENT code processor 3-17
device allocation/unallocation (see allocation/unallocation)
device data collected by MF/1 3-145
device end post handler
  function 3-394
device groups no longer needed, determining 3-372
device groups that must remain serialized 3-373
device sampling in MF/1
  initialization 3-104
  processing 3-145
device selections from JES3 3-284-3-285
devices, generic (see generic allocation control)
devices, waiting for
  in common allocation control 3-289
direct access data set (see DADSM)
direct access label read
  function 3-340, 3-394
direct read
  in pseudo access method 3-182
direct write
  in pseudo access method 3-182
DISP (disposition) information (see also disposition processing)
    completing in JFCB R IN SIOT 3-322
DISP (disposition) resolution (see also disposition processing)
  associated with commands in the input stream 3-231
  in DD function control 3-333
dispatching
  priority, changing
    in CPU management 3-62
disposition message routine
  function 3-443
disposition processing control
  function 3-440
disposition processing in IEFAB4A2 (see also DISP resolution, DISP information) 3-440
  in common unallocation control (IEFAB4A0) 3-430
  in DD function control (IEFAB454)
    function 3-332-3-333
DMDT (domain descriptor table) **(VS2.03.807)**
  in resource monitor MPL adjustment processing (IRARMRM2) 3-67.0 **(VS2.03.807)**
  in resource monitor periodic monitoring (IRARMRM1) 3-66 **(VS2.03.807)**
  in swap analysis (IRARMCAP) 3-36 **(VS2.03.807)**
DOM count in VM&V (volume mount and verify) tables 3-391
DOM (delete operator message) ID entries
  in allocation/volume mount and verify (VM&V) interface 3-388
  in volume mount and verify (VM&V) 3-392-3-393
domains **(VS2.03.807)**
  definition/description 3-3 **(VS2.03.807)**
  of work indicated in IPS 3-3 **(VS2.03.807)**
DONTSWAP SYSEVENT code (code 41)
  exception to authorization 3-5
  in SYSEVENT processor 3-20
  in workload manager 3-71
DSAB (data set association block)
  in allocate request to unit 3-302
  in allocating offline devices (IEFAB486) 3-368
  in allocation via algorithm 3-348

  in step initiation 3-202
  job journal writing 3-520
CIB (command input block)
  in measurement facility control 3-80
CIB (command input buffer)
  in job initiation 3-196
clean-up processing
  in common allocation 3-378
  in volume mount & verify (VM&V) 3-394
clock, TOD (see TOD clock)
coefficients, resource (see resource factor coefficient)
collect data for MF/1 (IRARMWR3) 3-73.8 **(VS2.03.807)**
command, reconfiguration (see reconfiguration commands)
command processing
  in the input stream 3-230
commands
  in the input stream 3-230
  WRITELOG START 3-466
comment or continuation statement processing 3-226
common allocation clean-up
  called by 3-378
  common allocation parameter list, building 3-378
  functions 3-378
  requests excluded (see also requests, allocation) 3-378
common allocation control
  called by 3-280
  fixed device control 3-290
  function 3-280
  function map 3-430
  generic allocation control, use with 3-288
  order of allocation 3-280
  parameter list, function map 3-280
  recovery, occasions for use 3-288
  serialization of 3-282
  waiting for devices 3-280
common allocation parameter list
  building 3-378
common request router
  processing
    function 3-172
  request block construction 3-412
common unallocation functions 3-430
comparator, clock (see clock comparator)
COMWA (converter/interpreter common work area)
  converter use of in
    identifying verbs or JCL statements 3-226
    initialization 3-224
    processing commands in the input stream 3-230
    processing in-stream and cataloged procedures 3-232
    termination 3-242
  interpreter use of in initialization 3-246
concatenation, dynamic
  function 3-418
CONFIGCH SYSEVENT code (29)
  processing in SRM SYSEVENT code processor 3-17
continuation statement processing 3-226-3-229
control, common allocation (see common allocation control)
control blocks (see data areas)
conversion of bit mask
  function 3-200-3-201
converter (see also interpreter)
  command verb validation routine
    function 3-230, 3-228
  comment or continuation validation routine
    function 3-226
  converting statements to internal text 3-236-3-239
  entering defaults into internal text 3-240-3-241
  error messages
    processing by subsystem initiation message writer 3-186-3-187
  get routine
    function 3-226
  identifying verbs on JCL statements 3-226-3-229
  initialization
    function 3-224-3-225
  instream procedure routines
    function 3-232
  pre-scan routine
    function 3-228
  processing commands in the input stream 3-230-3-231
  processing in-stream and cataloged procedures 3-232-3-233
  processing symbolic parameters 3-234-3-235
  pseudo-access method 3-182-3-185
  scan routine
    function 3-236, 3-240, 3-234, 3-226
  SWA manager interface routine
    function 3-233
  symbolic parameter routine
    function 3-234
  termination routine
    function 3-242-3-243
  test and store utility routine
    function 3-252
  use by master subsystem 3-178-3-181
  verb identifier routine
    function 3-226, 3-228, 3-232, 3-238
converter/interpreter
  interface 3-178-3-181
  operator message module
    function 3-258-3-259
converting an allocation in dynamic allocation control 3-414
COPYDMDT SYSEVENT code **(VS2.03.807)**
  code processor 3-11 **(VS2.03.807)**
  processing in SRM SYSEVENT 3-22 **(VS2.03.807)**
corequisite publications iv (preface)
count table
  in allocation via algorithm 3-350
  in common allocation control 3-280
  in demand allocation 3-355
  in fixed device control 3-294
  in nonspecific volume allocation control 3-312
  in specific volume allocation 3-300
cover/reduce algorithm
  function 3-366, 3-348, 3-374, 3-280
CPU activity initialization in MF/1 3-96
CPU affinity
  in job initiation 3-199
  in step initiation 3-201
CPU load balancing swap analysis 3-66
CPU management in SRM 3-62
CPU measurements in MF/1
  in interval MG routine 3-118-3-121
CPU utilization **(VS2.03.807)**
  in CPU management (IRARMCPM) 3-63,3-64 **(VS2.03.807)**
CRI (catalog return information)
  in DD function control 3-322-3-330
  in JLOCATE 3-334
cross-memory posting of SMF error exit 3-460
CSCB (command scheduling control block)
  in allocation/initiator interface 3-398
  in initiator/allocation interface 3-398
  in job initiation 3-196
  in step initiation 3-200
  in SWA create interface 3-216
CSD (common system data area)
  in common allocation control 3-282
  in CPU management (IRARMCPM) 3-62 **(VS2.03.807)**
  in MF/1 channel initialization 3-100
current allocation environment, providing information about 3-422
CVT (communication vector table)
  in allocation/initiator interface 3-396
  in allocation/volume mount and verify (VM&V) interface 3-388
  in building a step header record for the job journal 3-512
  in common request router 3-172
  in data set name assignment 3-188
  in initiator/allocation interface 3-396
  in initiator/unallocation interface 3-404
  in interval measurement gathering routine for workload 3-126
  in merging job journal to SWA 3-492
  in MF/1 channel initialization 3-100
  in MF/1 CPU activity initialization 3-96

    in MF/1 device initialization   3-104
    in MF/1 paging activity initialization   3-96
    in MF/1 second CPU test channel sampling module
      3-142
    in MF/1 termination processor   3-110
    in STAE exit processing for SMF   3-458
    in subsystem determination   3-174
    in subsystem interface   3-164
    in switching SMF data sets   3-454
    in unallocation/initiator interface   3-404
    in volume mount and verify (VM&V)   3-394
    in writing blocks to the job journal   3-520
    in writing SMF records   3-450

DADSM
    allocation interface   3-304
    parameter list in allocate request to unit   3-304
    VM&V interface   3-386
DAIRFAIL (IKJEFFI8) failure in dynamic allocation
  3-486 **(VS2.03.810)**
Data Area section   7-1855
    data control (IRBMFDTA) in MF/1   3-106
data set assignment   3-188
data set descriptor record processor (see also
  checkpoint/restart)
    in SWA create interface
        function   3-486, 3-216, 3-217
data set enqueue parameter list building
    function   3-198-3-199
data set name (see also DSN resolution)
    in data set tree processing   3-198-3-199
data set name assignment
    function   3-188
data set name resolution
    function   3-324
data set tree structure processing in job and step initiation
    function   3-198-3-199
data sets, releasing
    in common unallocation control   3-432-3-433
data sets, SMF   3-454
DCB (data control block)
    in converter/interpreter interface   3-178
    in data set descriptor records processing   3-486
    in switching SMF data sets   3-454
    in writing SMF records   3-450
    resolution in DD function control
        function   3-330-3-331
DD function control (IEFAB454)
    DCB resolution   3-331
    DISP resolution   3-332-3-333
    GDG (generation data group) processing   3-324-3-325
    processing
        function   3-322
    volume/unit resolution   3-328-3-331
DD preparation
    function   3-314-3-315
DD processing control
    function   3-324, 3-314
ddname allocation   3-428, 3-412
ddname and relative position number
    informing the JES3 subsystem   3-418-3-419
ddname search routine
    function   3-416, 3-418, 3-420
DEB (data extent block)
    deconcatenation, dynamic   3-420
    step initiation, in   3-204
deconcatenation routine
    function   3-420
defaults, converter, entering into JCL internal text   3-240
deferred action processor (IRARMCEN) in SRM   3-28
deferred action queue **(VS2.03.807)**
    in deferred action process   3-28 **(VS2.03.807)**
deferred restart job determination (in SWA create
  interface)   3-217
deferring algorithms in SRM   3-23 **(VS2.03.807)**
DELETE subroutine
    in SWA manager move mode   3-265
DELETE/LOCATE function in SWA manager locate mode
  3-266-3-267

demand allocation
    processing
        function   3-355
    use with generic allocation   3-407
demand requests
    definition   3-355
    operator replies   3-375
    processing   3-375, 3-377
    volunit entry, updating   3-372, 3-373
DEQ macro instruction (see ENQ/DEQ/RESERVE
  routine)
determination of subsystem name   3-175
determine device requirements
    function   3-282-3-283
determining device requests
    for request not yet allocated   3-289
DEVALLOC SYSEVENT code (28)
    processing in SRM SYSEVENT code processor   3-17
device allocation/unallocation (see allocation/unallocation)
device data collected by MF/1   3-145
device end post handler
    function   3-394
device groups no longer needed, determining   3-372
device groups that must remain serialized   3-373
device sampling in MF/1
    initialization   3-104
    processing   3-145
device selections from JES3   3-284-3-285
devices, generic (see generic allocation control)
devices, waiting for
    in common allocation control   3-289
direct access data set (see DADSM)
direct access label read
    function   3-340, 3-394
direct read
    in pseudo access method   3-182
direct write
    in pseudo access method   3-182
DISP (disposition) information (see also disposition
  processing)
        completing in JFCB R IN SIOT   3-322
DISP (disposition) resolution (see also disposition
  processing)
    associated with commands in the input stream   3-231
    in DD function control   3-333
dispatching
    priority, changing
        in CPU management   3-62
disposition message routine
    function   3-443
disposition processing control
    function   3-440
disposition processing in IEFAB4A2 (see also DISP
  resolution, DISP information)   3-440
    in common unallocation control (IEFAB4A0)   3-430
    in DD function control (IEFAB454)
        function   3-332-3-333
DMDT (domain descriptor table) **(VS2.03.807)**
    in resource monitor MPL adjustment processing
      (IRARMRM2)   3-67.0 **(VS2.03.807)**
    in resource monitor periodic monitoring (IRARMRM1)
      3-66 **(VS2.03.807)**
    in swap analysis (IRARMCAP)   3-36 **(VS2.03.807)**
DOM count in VM&V (volume mount and verify) tables
  3-391
DOM (delete operator message) ID entries
    in allocation/volume mount and verify (VM&V)
      interface   3-388
    in volume mount and verify (VM&V)   3-392-3-393
domains **(VS2.03.807)**
    definition/description   3-3 **(VS2.03.807)**
    of work indicated in IPS   3-3 **(VS2.03.807)**
DONTSWAP SYSEVENT code (code 41)
    exception to authorization   3-5
    in SYSEVENT processor   3-20
    in workload manager   3-71
DSAB (data set association block)
    in allocate request to unit   3-302
    in allocating offline devices (IEFAB486)   3-368
    in allocation via algorithm   3-348

  in allocation/initiator interface 3-396
  in common allocation cleanup 3-378
  in common allocation control 3-280
  in common unallocation control 3-434
  in ddname allocation 3-428
  in demand allocation 3-355
  in dynamic allocation control 3-414
  in dynamic concatenation 3-418
  in dynamic deconcatenation 3-420
  in dynamic information retrieval 3-422
  in dynamic unallocation control 3-416
  in fixed device control (IEFAB430) 3-294
  in generic allocation control (IEFAB471) 3-342
  in initiator/allocation interface 3-396
  in initiator/unallocation interface 3-402
  in recovery allocation (IEFAB485) 3-360
  in remove in-use attribute routine (IEFDB480) 3-424
  in SVC 99 control (IEFDB400) 3-412
  in unit unallocation processing (IEFAB4A4) 3-446
DSAB entry checks made (in ddname allocation) 3-428
DSAB QDB (data set association block queue descriptor block)
  in allocation/initiator interface 3-396
  in common allocation cleanup 3-380
  in ddname allocation 3-428
  in dynamic allocation control 3-414
  in dynamic concatenation 3-418
  in dynamic deconcatenation 3-420
  in dynamic information retrieval 3-422
  in dynamic unallocation control 3-416
  in initiator/allocation interface 3-396
  in initiator/unallocation interface 3-402
  in nonspecific volume allocation control 3-310
  in remove in-use attribute routine 3-424
  in SVC 99 control (IEFDB400) 3-412
  in unallocation/initiator interface 3-402
  in unit unallocation processing 3-446
DSCB (data set control block)
  in switching SMF data sets 3-454
DSDR (data set descriptor record)
  checkpoint/restart, processing of 3-483
DSENQT (data set enqueue table)
  in DD function control (IEFAB454) 3-332
  in interpreter
    creating and chaining tables 3-254
    initialization 3-246
  in step initiation 3-200
DSN resolution in data set tree processing 3-198-3-199
dsname search routine
  function 3-416
DSNT (data set name table)
  in DD function control 3-326
DTMVT (measurement vector table for trace and report data, see also INMVT, MFMVT, STMVT, TMMVT)
  in MFDATA SVC mainline 3-114
  in MF/1 report generator control (IRBMFRGM) 3-148
DWWIN
  in interval measurement gathering routine for workload 3-126
  in MF/1 workload initialization 3-98
dynamic allocation
  convert routine
    function 3-414
  DAIRFAIL processing 3-486 **(VS2.03.810)**
  ESTAE exit
    function 3-413
  function validity checker
    function 3-414
  processing 3-414
  SVC 99 control (IEFDB400) 3-412
dynamic allocation request for unit and volume, processing 3-280
dynamic concatenation
  criteria for 3-418
  processing 3-418
dynamic deconcatenation
  criteria for 3-420
  processing 3-420
dynamic information retrieval
  function 3-422
dynamic support system (see DSS)
dynamic unallocation 3-416

ECB (event control block)
  in converter/interpreter interface 3-180
  in swap-out control 3-43
ECCDB
  in MF/1 channel interval measurement gathering routine 3-130
  in MF/1 channel initialization 3-100
  in MF/1 channel sampling module 3-140
  in MF/1 second CPU test channel sampling module 3-142
ECCED
  in MF/1 channel interval measurement gathering routine 3-130
  in MF/1 channel initialization 3-100
  in MF/1 channel sampling module 3-140
ECCPE
  in MF/1 channel initialization 3-100
  in MF/1 channel sampling module 3-140
EDDCD
  in interval measurement gathering routine for devices 3-134
  in MF/1 device initialization 3-104
EDDDB
  in interval measurement gathering routine for devices 3-134
  in MF/1 device initialization 3-104
EDDED
  in interval measurement gathering routine for devices 3-134
  in MF/1 device initialization 3-104
  in MF/1 device sampling module 3-144
EDL (eligible device list)
  building 3-122, 3-283
  contents 3-283
  in allocating offline devices 3-366
  in allocation via algorithm 3-348
  in common allocation cleanup 3-382
  in common allocation control 3-282
  in demand allocation 3-355
  in fixed device control 3-294
  in generic allocation control 3-338
  in nonspecific volume allocation control 3-312
  in recovery allocation 3-358
  in specific volume allocation 3-298
EDT
  in allocate request to unit 3-302
  in allocating offline device 3-366
  in common allocation control 3-280
  in generic allocation control 3-338
  in JFCB housekeeping control 3-316
  in recovery allocation 3-360
eligible units, determining
  in specific volume allocation control 3-298-3-299
eliminate ineligible groups and generics
  function 3-348, 3-362, 3-366
end of task (see EOT)
ENQ/DEQ routine for allocation
  function 3-386, 3-357, 3-312
ENQ macro instruction (see ENQ/DEQ/RESERVE routine)
ENQRLSE SYSEVENT code (21)
  processing in SRM SYSEVENT code processor 3-16
enqueue parameter list, use of in job initiation 3-198-3-199
enqueueing on volume serial number 3-312
entry point scheduling in SRM, periodic 3-32
entry point summary for SRM **(VS2.03.807)**
  control function (IRARMCTL) 3-43.6 **(VS2.03.807)**
  CPU management (IRARMCPM) 3-65.0 **(VS2.03.807)**
  functional recovery routine (IRARMERR) 3-9.13 **(VS2.03.807)**
  interface function (IRARMINT) 3-9.0 **(VS2.03.807)**
  I/O management (IRARMIOM) 3-61.0 **(VS2.03.807)**
  MF/1 interface (IRARMWAR) 3-73.10 **(VS2.03.807)**
  resource monitor (IRARMRMR) 3-67.2 **(VS2.03.807)**
  service routine (IRARMSRV) 3-9.13 **(VS2.03.807)**

  storage management (IRARMSTM) 3-51.2 **(VS2.03.807)**
  sysevent processor (IRARMEVT) 3-22.6 **(VS2.03.807)**
  workload manager (IRARMWLM) 3-73.4 **(VS2.03.807)**
environment current allocation, providing information about (IEFDB470) 3-422
EPA (external parameter area)
  format 3-523
  in allocation/initiator interface 3-396
  in dynamic allocation control 3-414
  in dynamic unallocation control 3-416
  in initiator/allocation interface 3-396
  in JFCB housekeeping control 3-318
  in remove in-use attribute routine 3-426
  in SWA manager locate mode 3-266
  in SWA manager move mode 3-264
EPAL (external parameter area locate mode, see EPA)
EPAM (external parameter area move mode, see EPA)
EPFA
  in full analysis (IRARMCAS) 3-34
error codes
  set in dynamic concatenation (IEFDB450) 3-418
error messages
  processing by subsystem initiation message writer 3-186-3-187
error processing (see also error recovery ESTAE processing)
  in allocation via algorithm 3-351
  in allocation recovery 3-365
  in common allocation clean-up 3-378
  in common allocation control 3-307
  in DD function control 3-333
  in demand allocation 3-355
  in fixed device allocation control 3-297
  in generic allocation control 3-341
  in initiator/allocation interface 3-399
  in JFCB housekeeping control 3-317, 3-319
  in JLOCATE (IEFAB469) 3-337, 3-335
  in job journal merge 3-508
  in non-specific volume allocation control 3-313, 3-377
  in offline/allocated device allocation 3-369
  in specific volume allocation control 3-301
  in subsystem initiation
    IEFJCNTL 3-177
    IEFJJOBS 3-177
  in volume mount and verify 3-395
error recursion (see recursion processing of errors)
error, syntax, detecting in converter (IEFVFA) 3-234
ESTAE
  for SWA create interface 3-217
  in converter initialization 3-224
  in interpreter initialization 3-246
  in job initiation 3-196
event-driven MF/1 functions
  in MFROUTER 3-138
  in MF/1 termination processor (IRBMFTMA) 3-110
exchange swap 3-23.0,3-36 **(VS2.03.807)**
exclusive control (see XCTL routine)
exclusive data set attribute
  handling in initiator 3-199
EXEC statement
  in interpreter 3-257
exit, attention (see attention exit)
exit handling (see EXIT routine)
exit, initiator 3-200-3-201
express swap-in 3-23.0,3-36 **(VS2.03.807)**
external parameter area (see EPA)
external parameter area locate mode (see EPA)
external parameter area move mode (see EPA)

faults (see page faults)
fetch (see program fetch)
FETCHLIB 3-204
five functional groups in SRM 3-3 **(VS2.03.807)**
fixed device control (IEFAB430)
  count fields updated 3-294
  direct access UCB use 3-294
  processing
    function 3-294
  use with common allocation control 3-283
  use with nonspecific volume allocation control 3-297
frame (see page frame)
FRR (see functional recovery routine)
full analysis (see system resources manager)
function codes
  in subsystem interface 3-161
  in SWA manager move mode 3-264
  in SWA manager locate mode 3-266
  used by JES2 and JES3 3-161
functional recovery routine (see also termination conditions)
  for SRM 3-9, 3-7

GDG (generation data group) processing
  function 3-323-3-325
GDGALL requests, processing 3-325
GDGNT (generation data group name table)
  in DD function control 3-324
  in JLOCATE 3-334
generation data group (see GDG)
generic allocation control (IEFAB471)
  AVR 3-341
  function 3-338
generic processing, build tables for
  function 3-338-3-339
generic table build processing 3-338
GET
  in pseudo access method 3-184
group ID list
  in allocation via algorithm 3-348
group lock/unlock ESTAE exit (IEFAB4E7) **(VS2.03.804)**
  function 3-411 **(VS2.03.804)**
group lock/unlock interface (IEFAB4EC) **(VS2.03.804)**
  function 3-411 **(VS2.03.804)**
GWT
  in step initiation 3-200

header record for job journal, step, building 3-512
HIPO (see Method-of-Operation section)
housekeeping (see JFCB housekeeping)
HSKPWA (JFCB housekeeping work area)
  in DD function control 3-322
  in JFCB housekeeping control 3-314
  in JLOCATE 3-334

ICBME object module
  function 3-350-3-351, 3-377
ICT
  in I/O load balancing swap analysis (SRM) 3-56
  in I/O management (SRM) 3-54
  in storage management (SRM) 3-48
IDACAT11 object module
  function in JLOCATE 3-337
IEAVPFTE object module
IEEMB803 object module
  function 3-466, 3-470, 3-474, 3-472
IEEMB804 object module
  function 3-480
IEEMB806 object module
  function 3-476
IEEMB807 object module
  function 3-472, 3-468
IEEMB825 object module
  function 3-458
IEEMB827 object module
  function 3-460
IEEMB829 object module
  function 3-450-3-451, 3-456, 3-452, 3-454
IEEMB830 object module
  function 3-452, 3-450-3-451
IEEMSER (see MSRDA)
IEFAB4A0 object module
  function 3-430
  function map for 3-430
IEFAB4A2 object module
  function 3-440
IEFAB4A4 object module
  function 3-444

IEFAB4A6 object module
  function 3-396-3-397
IEFAB4A8 object module
  function 3-436-3-437
IEFAB4B0 object module
  function 3-443
IEFAB4B2 object module
  function 3-443
IEFAB4EB object module
  function 3-334-3-335
IEFAB4EC object module **(VS2.03.804)**
  function 3-411 **(VS2.03.804)**
IEFAB4ED object module **(VS2.03.804)**
  function 3-291-3-413 **(VS2.03.804)**
IEFAB4EE object module
  function 3-380-3-381
IEFAB4EF object module
  function 3-336-3-337, 3-314-3-315
IEFAB4E0 object module
  function 3-340-3-341
IEFAB4E7 object module **(VS2.03.804)**
  function 3-411 **(VS2.03.804)**
IEFAB4FA object module
  function 3-341, 3-346, 3-388, 3-290, 3-378, 3-435, 3-372
IEFAB4FC object module
  function 3-396, 3-398, 3-418, 3-436
IEFAB4FD object module
  function 3-380, 3-398, 3-400
IEFAB4FE object module
  function 3-396, 3-412
IEFAB4F0 object module
  function 3-386, 3-357, 3-312
IEFAB4F2 object module
  function 3-368-3-369
IEFAB4F4 object module
  function 3-336, 3-318, 3-432
IEFAB4F5 object module
  function 3-336
IEFAB4F7 object module
  function 3-316, 3-412, 3-418, 3-400, 3-322
IEFAB4F8 object module
  function 3-340, 3-394
IEFAB4F9 object module
  function 3-340, 3-418
IEFAB421 object module
  function 3-280
IEFAB422 object module
  function 3-282-3-283
IEFAB423 object module
  function 3-282-3-283
IEFAB424 object module
  function 3-282-3-283
IEFAB425 object module
  function 3-286-3-287
IEFAB426 object module
  function 3-282-3-283
IEFAB428 object module
  function 3-280-3-281, 3-302-3-303
IEFAB430 object module
  function 3-294
IEFAB431 object module
  function 3-280-3-281
IEFAB432 object module
  function 3-304-3-305
IEFAB433 object module
  function 3-298
IEFAB434 object module
  function 3-302
IEFAB435 object module
  function 3-302
IEFAB436 object module
  function 3-308
IEFAB438 object module
  function 3-282-3-283
IEFAB440 object module
  function 3-350, 3-310
IEFAB441 object module
  function 3-368, 3-302
IEFAB442 object module
  function 3-304, 3-298, 3-368
IEFAB451 object module
  function 3-314
IEFAB452 object module
  function 3-324, 3-314
IEFAB453 object module
  function 3-314-3-315
IEFAB454 object module
  function 3-322
IEFAB455 object module
  function 3-334
IEFAB456 object module
  function 3-324
IEFAB457 object module
  function 3-326
IEFAB458 object module
  function 3-330-3-331
IEFAB459 object module
  function 3-332-3-333
IEFAB461 object module
  function 3-323-3-325
IEFAB463 object module
  function 3-326, 3-328
IEFAB464 object module
  function 3-328, 3-330
IEFAB466 object module
  function 3-326, 3-334
IEFAB469 object module
  function 3-334
IEFAB470 object module
  function 3-316-3-317
IEFAB471 object module
  function 3-338
IEFAB472 object module
  function 3-338-3-339
IEFAB473 object module
  function 3-340, 3-341
IEFAB474 object module
  function 3-348, 3-366
IEFAB475 object module
  function 3-342, 3-344, 3-346
IEFAB476 object module
  function 3-348
IEFAB477 object module
  function 3-360, 3-348, 3-368
IEFAB478 object module
  function 3-370, 3-350, 3-290
IEFAB479 object module
  function 3-355
IEFAB48A object module
  function 3-374-3-375
IEFAB480 object module
  function 3-366, 3-348, 3-374, 3-280
IEFAB481 object module
  function 3-348, 3-362, 3-366
IEFAB485 object module
  function 3-358
IEFAB486 object module
  function 3-366
IEFAB487 object module
  function 3-374, 3-364
IEFAB488 object module
  function 3-374, 3-376
IEFAB489 object module
  function 3-377
IEFAB49A object module
  function 3-394, 3-388
IEFAB49B object module
  function 3-394
IEFAB490 object module
  function 3-378
IEFAB491 object module
  function 3-280, 3-366
IEFAB492 object module
  function 3-386
IEFAB493 object module
  function 3-390
IEFAB494 object module
  function 3-390
IEFAB495 object module
  function 3-392

IEFAB496 object module
  function 3-394
IEFAB498 object module
  function 3-390, 3-388, 3-394
IEFAB499 object module
  function 3-392
IEFAB820 object module
  function 3-206
IEFBB401 object module
  function 3-396
IEFBB402 object module
  function 3-396
IEFBB404 object module
  function 3-398
IEFBB410 object module
  function 3-402, 3-404
IEFBB412 object module
  function 3-406, 3-414
IEFBB414 object module
  function 3-404-3-405
IEFBB416 object module
  function 3-410
IEFDB4A0 object module
  function 3-416
IEFDB4A1 object module
  function 3-416, 3-426
IEFDB4FA object module
  function 3-416
IEFDB4FB object module
  function 3-418-3-419
IEFDB4FC object module
  function 3-416, 3-418, 3-420
IEFDB4FF object module
  function 3-424, 3-420, 3-422, 3-428, 3-418, 3-416
IEFDB4F9 object module
  function 3-418
IEFDB400 object module
  function 3-412
IEFDB402 object module
  function 3-413
IEFDB410 object module
  function 3-414, 3-412
IEFDB411 object module
  function 3-414
IEFDB412 object module
  function 3-414
IEFDB413 object module
  function 3-414
IEFDB450 object module
  function 3-418
IEFDB460 object module
  function 3-420
IEFDB470 object module
  function 3-422
IEFDB480 object module
  function 3-424
IEFDB481 object module
  function 3-416, 3-426
IEFDB490 object module
  function 3-412
IEFDSLST object module
  function 3-198-3-199
IEFDSTBL object module
  function 3-198-3-199
IEFIB600 object module
  function 3-216
IEFIB605 object module
  function 3-216
IEFIB645 object module
  function 3-216-3-217
IEFICPUA object module
  function 3-201, 3-199
IEFIIC object module
  function 3-196
IEFIMASK object module
  function 3-200-3-201
IEFJACTL object module
  function 3-182, 3-184
IEFJCDLT object module
  function 3-176-3-177
IEFJCNTL object module
  function 3-177, 3-179
IEFJDIRD object module
  function 3-182-3-183
IEFJDSNA object module
  function 3-188
IEFJDWRT object module
  function 3-182-3-183
IEFJJCLS object module
  function 3-176-3-177
IEFJJOBS object module
  function 3-176
IEFJJTRM object module
  function 3-190
IEFJRASP object module
  function 3-172
IEFJREAD object module
  function 3-184-3-185
IEFJSDTN object module
  function 3-174
IEFJSREQ object module
  function 3-161-3-167
IEFJWRTE object module
  function 3-184-3-185
IEFJWTOM object module
  function 3-186
IEFNB903 object module
  function 3-246
IEFPARAM (initiation parameter list)
  in step initiation 3-202
IEFQB550 object module
  function 3-264
IEFQB555 object module
  function 3-266
IEFQB580 object module
  function 3-264
IEFQB585 object module
  function 3-264
IEFRPREP object module
  function 3-516
IEFSD101 object module
  function 3-200-3-201
IEFSD102 object module
  function 3-200-3-201
IEFSD160 object module
  function 3-196
IEFSD161 object module
  function 3-196, 3-198
IEFSD162 object module
  function 3-200
IEFSD164 object module
  function 3-208
IEFSD166 object module
  function 3-210
IEFSD263 object module
  function 3-206
IEFSMFIE object module
  function 3-200-3-201
IEFVDA object module
  function 3-248-3-249
IEFVEA object module
  function 3-248-3-249
IEFVFA object module
  function 3-236, 3-238, 3-240, 3-234, 3-226
IEFVFB object module
  function 3-234
IEFVGK object module
  function 3-250
IEFVGT object module
  function 3-252
IEFVHA object module
  function 3-226
IEFVHC object module
  function 3-226
IEFVHCB object module
  function 3-226, 3-228, 3-232, 3-238
IEFVHE object module
  function 3-248
IEFVHEB object module
  function 3-228

- IEFVHF object module
  - function 3-242-3-243
- IEFVHH object module
  - function 3-256
- IEFVHM object module
  - function 3-230, 3-228
- IEFVHN object module
  - function 3-258
- IEFVHQ object module
  - function 3-233
- IEFVHR object module
  - function 3-258-3-259
- IEFVH1 object module
  - function 3-224-3-225
- IEFVINA object module
  - function 3-232
- IEFVINB object module
  - function 3-232
- IEFVINC object module
  - function 3-232
- IEFVJA object module
  - function 3-248
- IEFXB500 object module
  - function 3-521, 3-202, 3-524, 3-520, 3-512, 3-518
- IEFXB590 object module
  - function 3-525
- IEFXB601 object module
  - function 3-492, 3-496, 3-498-3-509, 3-216, 3-494
- IEFXB602 object module
  - function 3-510-3-511
- IEFXB604 object module
  - function 3-512, 3-514, 3-202
- IEFXB609 object module
  - function 3-486, 3-216
- IEFXB610 object module
  - function 3-487
- IEL (initiator entrance list)
  - in job initiation 3-196
- IGX00013 object module
  - function 3-82, 3-80, 3-91
- IGX00014 object module
  - function 3-114
- IKJEFFI8 object module 3-486 (**VS2.03.810**)
- IMCB (SRM user I/O measurement control table)
  - in I/O management (SRM) 3-54
  - in SYSVENT processing in SRM SYSEVENT code processor 3-13
- INCOA (common option area for input source options, see also MFCOA, STCOA, TMCOA)
  - in MF/1 input merge control 3-84
  - in MF/1 list option module 3-88
  - in MF/1 syntax analyzer 3-86
- INITATT SYSEVENT code (10)
  - in work load manager 3-73
  - processing in SRM SYSEVENT code processor 3-13
- INITDET (SYSEVENT code 11)
  - in I/O management 3-54-3-55
  - in workload manager 3-70-3-71
  - processing in SRM SYSEVENT code processor 3-13
- initialize for MF/1 (IRARMWR1) 3-73.6 (**VS2.03.807**)
- initiation
  - of the master scheduler 3-176-3-177
    - data set name assignment 3-188-3-189
  - of a subsystem 3-176-3-177
    - data set name assignment 3-188-3-189
  - step 3-200
    - notify subsystem of step initiation 3-202
- initiator attach module
  - function 3-206
- initiator builder of completion code interface 3-458
- initiator control initialization
  - function 3-196
- initiator data set enqueue
  - function 3-200-3-201
- initiator device allocation interface routine
  - function 3-200
- initiator interface control and interface to allocate catalog
  - function 3-196
- initiator job initiation 3-196
- initiator job select routine
  - function 3-196, 3-198
- initiator recovery processing 3-212
- initiator SMF exit 3-200-3-201
- initiator/terminator
  - processing 3-193
  - SWA subpool for 3-267
- initiator/unallocation interface
  - functions 3-402
  - when called 3-402
- input stream (see converter)
- INMVT (measurement vector table for temporary input options, see also DTMVT, MFMVT, STMVT, TMMVT)
  - in MF/1 input merge control 3-84
  - in MF/1 syntax analyzer 3-86
- input to MF/1, analyzing 3-86
- input options for MF/1 (see options, MF/1)
- IN queue for SRM (**VS2.03.807**)
  - definition 3-23 (**VS2.03.807**)
  - in CPU management (IRARMCPM) 3-62 (**VS2.03.807**)
  - in resource monitor periodic monitoring (IRARMRM1) 3-67 (**VS2.03.807**)
  - in select user for swap-out (IRARMCPO) 3-43.2 (**VS2.03.807**)
  - in storage management (IRARMSTM) 3-46 (**VS2.03.807**)
  - in swap analysis (IRARMCAP) 3-36 (**VS2.03.807**)
  - in swappable user evaluation (IRARMWM2) 3-70 (**VS2.03.807**)
- installation performance specifications (see IPS values)
- in-stream procedures (see JCL statements)
  - director entry build, directory search, processing 3-232-3-233
- instructions (see also macro instructions)
- integrity (see data set integrity processing)
- interface, subsystem 3-159
- internal text
  - converting JCL to 3-236
  - data set, getting JCL statement from 3-249
  - entering defaults into 3-240
- interpreter (see also converter/interpreter)
  - creating and chaining tables 3-252
  - enqueue routine
    - function 3-256
  - error messages
    - processing by subsystem initiation message writer 3-186-3-187
  - ESTAE, setting up 3-246
  - EXEC statement processor
    - function 3-248-3-249
  - get and route routine
    - function 3-248
  - get key/positional utility routine
    - function 3-250
  - initialization 3-246
    - function 3-246
  - interface to SWA create 3-216
  - job statement processor
    - function 3-248
  - parameter values, analyzing 3-248
  - pseudo access method 3-182-3-185
  - termination
    - function 3-258
  - use by master subsystem 3-178-3-181
  - writing tables into SWA 3-256
- interval, MF/1
  - in MFDATA SVC mainline 3-114
  - timed (in MFSTART mainline) 3-82
- interval-driven MF/1 routines
  - for data (IRBMFDTA) 3-106
  - for CPU (IRBMFDCP) 3-118
  - for devices 3-134
  - for paging (IRBMFDPP) 3-126
  - for workload (IRBMFDWP) 3-126
  - initialization of in MF1MAINL 3-90
  - termination processor (IRBMFTMA) 3-110
- I/O load balancing in SRM
  - swap analysis (IRARMIL2) 3-56
  - user I/O monitoring (IRARMIL0) 3-58
- I/O management in SRM (IRARMIOM) 3-54

IOS UCB LUT (I/O supervisor unit control block logical unit table)
  in allocating offline devices 3-374
  in common allocation control 3-282
  in fixed device control 3-294
  in generic allocation control 3-340
  in JFCB housekeeping control 3-316
  in job unallocation 3-410
  in recovery allocation 3-358
IRARMCAP
  function 3-36
IRARMCAP entry point in IRARMCTL **(VS2.03.807)**
  function 3-36,3-43.6 **(VS2.03.807)**
IRARMCAS
  function 3-34
IRARMCAT
  function 3-26
IRARMCEL
  function 3-30
IRARMCEL entry point in IRARMCTL **(VS2.03.807)**
  function 3-30,3-43.6 **(VS2.03.807)**
IRARMCEN
  function 3-28
IRARMCEN entry point in IRARMCTL **(VS2.03.807)**
  function 3-28,3-43.6 **(VS2.03.807)**
IRARMCET
  function 3-32
IRARMCET entry point in IRARMCTL **(VS2.03.807)**
  function 3-32,3-43.6 **(VS2.03.807)**
IRARMCL2
  function 3-66
IRARMCPI entry point in IRARMCTL **(VS2.03.807)**
  function 3-43.0,3-43.6 **(VS2.03.807)**
IRARMCPM object module
  function 3-62
IRARMCPM object module **(VS2.03.807)**
  entry point summary 3-65.0 **(VS2.03.807)**
  function 3-62 **(VS2.03.807)**
IRARMCPO entry point in IRARMCTL **(VS2.03.807)**
  function 3-43.2,3-43.7 **(VS2.03.807)**
IRARMCSI
  function 3-40
IRARMCSI entry point in IRARMCTL **(VS2.03.807)**
  function 3-40,3-43.6 **(VS2.03.807)**
IRARMCSO
  function 3-42
IRARMCSO entry point in IRARMCTL **(VS2.03.807)**
  function 3-42,3-43.6 **(VS2.03.807)**
IRARMCTL object module
  function 3-24
IRARMCTL object module **(VS2.03.807)**
  entry point summary 3-43.6 **(VS2.03.807)**
  function 3-24,3-43.6 **(VS2.03.807)**
IRARMCVL entry point in IRARMCTL **(VS2.03.807)**
  function 3-43.4,3-43.7 **(VS2.03.807)**
IRARMERR object module
  function 3-9
IRARMERR object module **(VS2.03.807)**
  entry point summary 3-9.13 **(VS2.03.807)**
  function 3-5 **(VS2.03.807)**
IRARMEVT object module
  function 3-12
  overview of functions 3-11
IRARMEVT object module **(VS2.03.807)**
  entry point summary 3-22.6 **(VS2.03.807)**
  function 3-12 **(VS2.03.807)**
IRARMHIT entry point in IRARMWLM **(VS2.03.807)**
  function 3-73.2,3-73.4 **(VS2.03.807)**
IRARMIL0 entry point in IRARMIOM **(VS2.03.807)**
  function 3-58,3-61.0 **(VS2.03.807)**
IRARMIL2
  function 3-56
IRARMINT object module
  function 3-6
IRARMINT object module **(VS2.03.807)**
  entry point summary 3-9.0 **(VS2.03.807)**
  function 3-6 **(VS2.03.807)**
IRARMIOM object module
  function 3-54
IRARMIOM object module **(VS2.03.807)**
  entry point summary 3-61.0 **(VS2.03.807)**
  function 3-54 **(VS2.03.807)**
IRARMIPS entry point in IRARMEVT **(VS2.03.807)**
  entry point description 3-22.6 **(VS2.03.807)**
IRARMI04 entry point in IRARMSRV **(VS2.03.807)**
  function 3-9.6,3-9.13 **(VS2.03.807)**
IRARMI05 entry point in IRARMSRV **(VS2.03.807)**
  function 3-9.8,3-9.13 **(VS2.03.807)**
IRARMMS2
  function 3-52
IRARMRMR object module **(VS2.03.807)**
  entry point summary 3-67.2 **(VS2.03.807)**
  function 3-45.1 **(VS2.03.807)**
IRARMRM1 entry point in IRARMRMR **(VS2.03.807)**
  function 3-66,3-67.2 **(VS2.03.807)**
IRARMRM2 entry point in IRARMRMR **(VS2.03.807)**
  function 3-67.0,3-67.2 **(VS2.03.807)**
IRARMSET object module
  function 3-32
IRARMSRV object module
  function 3-48, 3-32, 3-42, 3-40, 3-6
IRARMSRV object module **(VS2.03.807)**
  entry point summary 3-9.13 **(VS2.03.807)**
  function 3-9.2 **(VS2.03.807)**
IRARMSTM object module
  function 3-46
IRARMSTM object module **(VS2.03.807)**
  entry point summary 3-51.2 **(VS2.03.807)**
  function 3-46 **(VS2.03.807)**
IRARMWAR object module
  function 3-70
IRARMWAR object module **(VS2.03.807)**
  entry point summary 3-73.10 **(VS2.03.807)**
  function 3-69 **(VS2.03.807)**
IRARMWLM object module
  function 3-70
IRARMWM2 entry point in IRARMWLM **(VS2.03.807)**
  function 3-70,3-73.4 **(VS2.03.807)**
IRARMWM3 entry point in IRARMWLM **(VS2.03.807)**
  function 3-73.0,3-73.4 **(VS2.03.807)**
IRARMWR1 entry point in IRARMWAR **(VS2.03.807)**
  function 3-73.6,3-73.10 **(VS2.03.807)**
IRARMWR3 entry point in IRARMWAR **(VS2.03.807)**
  function 3-73.8,3-73.10 **(VS2.03.807)**
IRBMFALL object module
  function 3-80-3-81
IRBMFANL object module
  function 3-86
IRBMFCNV object module
  function 3-147, 3-151
IRBMFDCP object module
  function 3-118
IRBMFDDP object module
  function 3-134
IRBMFDEA object module
  function 3-106-3-107
IRBMFDHP object module
  function 3-130
IRBMFDPP object module
  function 3-122
IRBMFDTA object module
  function 3-106
IRBMFDWP object module
  function 3-126
IRBMFECH object module
  function 3-140
IRBMFEDV object module
  function 3-144
IRBMFEVT object module
  function 3-138
IRBMFICP object module
  function 3-96
IRBMFIDV object module
  function 3-104
IRBMFIHA object module
  function 3-100
IRBMFINP object module
  function 3-84
  use of 3-80, 3-86
IRBMFIOI object module

  - function 3-95, 3-111
- IRBMFIPG object module
  - function 3-96
- IRBMFIPP
  - function 3-96
- IRBMFIWK object module
  - function 3-98
- IRBMFMFC object module
  - function 3-80
- IRBMFMPR object module
  - function 3-112
- IRBMFRCR object module
  - function 3-150
- IRBMFRDR object module
  - function 3-150
- IRBMFRGM object module
  - function 3-146
- IRBMFRHR object module
  - function 3-150
- IRBMFRPR object module
  - function 3-150
- IRBMFRWR object module
  - function 3-150
- IRBMFSAR object module
  - function 3-147
- IRBMFSDE object module
  - function 3-82-3-83
- IRBMFTCH object module
  - function 3-142
- IRBMFTMA object module
  - function 3-110
- IRBMFTRM object module
  - function 3-111
- ISV (internal service value) **(VS2.03.807)**
  - in individual user evaluation (IRARMWM3) 3-73.0 **(VS2.03.807)**
  - in swappable user evaluation (IRARMWM2) 3-70 **(VS2.03.807)**
- item, defined in MF/1 syntax analyzer 3-86

- JCL data set reading from 3-226
- JCL statement (see also converter)
  - comment, checking for 3-226
  - continuation, checking for 3-226
  - converting statements to internal text 3-236
  - DD processing 3-229
  - EXEC processing 3-229
  - identifying verbs on 3-226-3-229
  - merging from JCL data set and procedure library 3-229
  - null processing 3-229
- JCL to JCLS conversion
  - function 3-176-3-177
- JCLS to SWA conversion
  - function 3-177, 3-179
- JCT (job control table)
  - in ABENDed job restart preparation 3-516
  - in allocation/initiator interface 3-396
  - in automatic checkpoint/restart 3-498
  - in building a step header record for the job journal 3-512
  - in DD function control 3-322
  - in dynamic allocation control 3-414
  - in dynamic unallocation 3-416
  - in initiator/allocation interface 3-396
  - in initiator/unallocation interface 3-402
  - in interpreter
    - creating and chaining tables 3-252
    - writing tables into SWA 3-256
  - in JLOCATE 3-334
  - in job deletion 3-208
  - in job initiation 3-196
  - in job unallocation 3-410
  - in merge cleanup 3-502
  - in step continue processing 3-494
  - in step deletion 3-208
  - in step initiation 3-200
  - in SVC 99 control 3-412
  - in unallocation/initiator interface 3-402
- JCTX (job control table extension) **(VS2.03.804)**
  - SWA manager id 3-261 **(VS2.03.804)**
  - writing tables into SWA 3-256,3-257 **(VS2.03.804)**
- JES exit in converting JCL statements to internal text 3-238-3-239
- JESCT (job entry subsystem control table)
  - in common request router 3-172
  - in data set name assignment 3-188
  - in subsystem determination 3-174
  - in subsystem interface 3-159
- JES2/3
  - function codes 3-161
  - notified of step initiation 3-202
  - system interface 3-171-3-191
- JES3
  - flags used by common unallocation control 3-430
  - interface routine
    - function 3-282-3-283
  - multi-unit, nonspecific volume requests, checking 3-382
- JFCB (job file control block)
  - DCB information in, completion 3-322
  - DISP information in, completing 3-322
  - in allocate request to unit 3-302
  - in allocating offline devices 3-366
  - in allocation via algorithm 3-348
  - in allocation/initiator interface 3-396
  - in allocation/volume mount and verfiy (VM&V) interface 3-386
  - in common allocation cleanup 3-378
  - in common allocation control 3-280
  - in common unallocation 3-432
  - in data set descriptor records, processing 3-486
  - in data set name assignment 3-188
  - in DD function control 3-322
  - in demand allocation 3-355
  - in disposition processing 3-440
  - in dynamic allocation control 3-414
  - in dynamic information retrieval 3-422
  - in fixed device control 3-294
  - in generic allocation control 3-338
  - in initiator/allocation interface 3-396
  - in initiator/unallocation interface 3-402
  - in interpreter 3-252
  - in JFCB housekeeping control 3-314
  - in JLOCATE 3-334
  - in job unallocation 3-410
  - in nonspecific volume allocation control 3-308
  - in recvery allocation 3-358
  - in SVC 99 control 3-414
  - in switching SMF data sets 3-454
  - in unallocation/initiator interface 3-402
  - in volume mount and verify (VM&V)/allocation interface 3-386
  - in writing SMF records 3-450
- JFCB housekeeping
  - functions 3-314
  - STEPCAT request processing 3-314
- JFCBE (job file control block extension for 3800 printer) **(VS2.03.810)**
  - in checkpoint/restart 3-483, 3-499, 3-501, 3-522, 3-523, 3-525 **(VS2.03.810)**
  - in interpreter 3-245, 3-249, 3-255 **(VS2.03.810)**
- JFCBX (job file control block extension)
  - in allocation/initiator interface 3-396
  - in checkpoint/restart 3-483 **(VS2.03.810)**
  - in common allocation control 3-280
  - in disposition processing 3-440
  - in dynamic allocation control 3-414
  - in initiator/allocation interface 3-396
  - in initiator/unallocation interface 3-402
  - in interpreter 3-245 **(VS2.03.810)**
  - in SVC 99 control 3-412
  - in unallocation/initiator interface 3-402
- JLOCATE (IEFAB469) processing 3-334
- JMR (job management record)
  - in converter initialization 3-224
  - in converter termination 3-242
  - in interpreter, initialization 3-246
  - in interpreter, writing tables into SWA 3-256
  - in SWA create interface 3-216

JNLPARM
  in writing blocks to the job journal 3-520
job account table, use in step and job deletion (initiator) 3-208
job control language (see JCL)
job entry subsystem (JES)
  interface 3-159
job initiation 3-196
job journal
  building step header record for 3-512
  changes for VS2 Release 3 3-483
  errors during reconstruction 3-508
  in step initiation 3-202-3-203
  journal for restarted jobs 3-524
  journal merge error processing 3-508
  journal merge reading 3-506
  merge cleanup 3-502-3-503
  merging job step entries onto the SWA 3-494, 3-492
  overview 3-483
  step header record, building 3-512
  writing
    blocks to 3-520-3-523
    in preparing ABENDed job for restart 3-518-3-519
JOB statement, checking 3-249
JOBCAT DD statement, processing 3-249
JOBLIB
  in interpreter 3-249
  in step initiation 3-204
job delete routine
  function 3-210
job scheduler
  overview 3-153
job select routine
  function 3-196, 3-198
job status indicators
  in SWA create interface 3-216-3-217
  setting in JSCB and JCT by step initiator 3-203
job step, ABENDed, preparing for restart 3-516
job step allocation (see step allocation)
job termination
  for a subsystem 3-190
job time limit, calculating in step initiator 3-202
job unallocation
  functions 3-410
  interface with initiator 3-402
  parameter list for initiator interface 3-402
JOBSELCT SYSEVENT code (8)
  in workload manager 3-71
  processing in SRM SYSEVENT code processor 3-13
JOBTERM SYSEVENT code (9)
  in workload manager 3-71
  processing in SRM SYSEVENT code processor 3-13
journal (see job journal)
journal merge
  loading journal merge routine 3-246
  processing 3-492
journal merge routine
  function 3-492, 3-496, 3-498-3-509, 3-216, 3-494
journal write routine
  function 3-520, 3-202, 3-524, 3-512, 3-518
JSCB (job step control block)
  in ABENDed job restart preparation 3-516
  in allocation/initiator interface 3-396
  in automatic step restart 3-500
  in building a step header record for the job journal 3-512
  in common allocation cleanup 3-378
  in common allocation control 3-280
  in data set descriptor records processing 3-486, 3-492
  in ddname allocation control 3-428
  in dynamic allocation 3-414
  in dynamic concatenation 3-418
  in dynamic deconcatenation 3-420
  in dynamic information retrieval 3-422
  in dynamic unallocation control 3-416
  in initiator/allocation interface 3-396
  in initiator/unallocation interface 3-402
  in JFCB housekeeping control 3-314
  in JLOCATE 3-334
  in job initiation 3-196
  in journal merge error processing 3-508
  in log initialization 3-466
  in merge cleanup 3-502
  in merging job journal to SWA 3-492
  in nonspecific volume allocation control 3-308
  in remove in-use control routine 3-424
  in remove in-use attribute 3-424
  in restart interface processing 3-510
  in subsystem interface 3-159
  in SVC 99 control 3-412
  in SWA create interface 3-216
  in SWA manager locate mode 3-266
  in system restart processing 3-496
  in unallocation/initiator interface 3-402
  in updating virtual addresses in SWA 3-504
  in writing blocks to the job journal 3-520
  in writing SMF records 3-450

keyword processing
  in converting JCL statements to internal text 3-236
  in parameter value analysis 3-248-3-249

LCA (log control area)
  in log initialization 3-466
  in terminating the system log 3-470
  in writing data on the system log 3-480
LCCA (logical communications configuration area)
  in CPU load balancing swap analysis 3-66
  in CPU management (SRM) 3-62
  in interval measurement gathering routine for CPU 3-118
  in MFDATA mainline 3-114
LCH
  in I/O load balancing swap analysis 3-56
  in I/O management (SRM) 3-54
LCT (linkage control table)
  in ABENDed job restart preparation 3-516
  in allocation/initiator interface 3-396
  in building a step header record for the job journal 3-512
  in converter/interpreter interface 3-178
  in initiator/allocation interface 3-396
  in initiator/unallocation interface 3-402
  in job initiation 3-196
  in job deletion 3-208
  in step continue processing 3-494
  in step deletion 3-208
  in step initiation 3-200
  in subsystem initiation 3-176
  in SWA create interface 3-216
  in SWA manager locate mode 3-266
  in unallocation/initiator interface 3-402
link pack area (see LPA)
listing MF/1 options 3-88
load balancing in SRM swap analysis
  CPU 3-62
  I/O 3-56
locating specific volume request 3-298
lock manager (see SETLOCK)
lock, SRM 3-25
locking services/considerations
  in SRM 3-5
log data set (see system log)
log hardcopy (see hardcopy of system log)
log, system (see system log)
log task abnormal termination, processing
  function 3-476
logical reconfiguration (see reconfiguration commands)
long wait processing in the SRM 3-49

main storage occupancy analysis (IRARMMS2) 3-52
mainline, initialization MF1 3-90
major name
  in job initiation 3-198-3-199
master catalog, searching in JLOCATE (IEFAB469) 3-334-3-335
master JCL

  conversion to SWA control blocks in
    converter/interpreter interface   3-178
  in data set name assignment   3-188-3-189
  in sybsystem initiation   3-176-3-177
master subsystem
  common request router   3-172
  converter/interpreter interface   3-178
  in data set name assignment   3-188
  in subsystem determination   3-174
  in subsystem initiation   3-176
  in subsystem initiation message writer   3-186
  in subsystem job termination   3-190
  interface overview   3-159
  pseudo access method   3-182
MCT
  in main storage occupancy analysis   3-52
  in resource monitor MPL adjustment processing
    (IRARMRM2)   3-67.0 **(VS2.03.807)**
  in resource monitor periodic monitoring (IRARMRM1)
    3-66 **(VS2.03.807)**
  in storage management (IRARMSTM)   3-46
    (VS2.03.807)
  in storage management (SRM)   3-46
  in swap-in control   3-40
  in timer action analysis   3-26
MEL (merge entrance list)
  in automatic checkpoint/restart   3-498
  in automatic step restart   3-500
  in merging job journal to SWA   3-492
  in step continue processing   3-494
  in SWA create interface   3-216
MEMCREAT SYSEVENT code (6)
  in SRM interface   3-7
  processing in SRM SYSEVENT code processor   3-12
MEMDEL SYSEVENT code (7)
  processing in SRM SYSEVENT code processor   3-12
merge of MF/1 options (see also options, MF/1)
  in MF/1 control   3-80
merge cleanup for restart or step continue processing
  3-502
merging job step entries in job journal   3-494
message processor for MF/1 (IRBMFMPR)   3-112
MF/1
  binary to channel conversion routine
    function   3-147
  channel event data sampling module
    function   3-140
  channel interval measurement gathering routine
    function   3-130
  channel measurements
    initialization
      function   3-100
    sampling   3-140, 3-142
  channel report generator
    function   3-150
  CPU measurement
    initialization
      function   3-96
    gathering
      function   3-118
    interval   3-118-3-121
  CPU report generator
    function   3-150
  data control routine
    function   3-106
  data control ESTAE recovery routine
    function   3-106-3-107
  device event data sampling module
    function   3-144
  device interval measurement gathering routine
    function   3-134
  device measurements
    initialization routine
      function   3-104
    interval   3-134
    sampling   3-144
  device report generator
    function   3-150
  dynamic allocation
    function   3-80-3-81
  event driven measurement routines
    calling from MFROUTER   3-139
  flowchart, inter-module   6-1577
    function   3-111
  general resource resource release routine
    function   3-111
  initialization (mainline)   3-90
  initialization
    for channel measurement   3-100
    for CPU activity   3-96
    for device measurement   3-104
    for paging activity measurement   3-96
    for workload measurement   3-98
  input merge control
    function   3-84
  IOS initialization/termination routine
    function   3-95, 3-111
  list option module   3-88
  mainline initialization   3-90
  measurement facility control module
    function   3-80
  merge of options   3-84, 3-80
  message processor
    function   3-112
  MFC (measurement facility control) module   3-80
  MFDATA SVC mainline   3-114
  MFROUTER SVC processor
    function   3-138
  MFSTART mainline processor
    ESTAE routine
      function   3-82-3-83
  options, listing   3-88
  options, merging   3-84, 3-80
  overview of MF/1   3-75
  paging activity initialization   3-96
  paging measurements
    initialization
      function   3-96, 3-92
    interval measurement gathering routine
      function   3-122
  paging report generator
    function   3-150
  recovery routine, ESTAE   3-83
  report generation
    control ESTAE routine
      function   3-147
    control module
      function   3-146
    modules for CPU, paging, workload, channels, and
  devices   3-150
  SARG function   3-77, 3-75
  second CPU test channel sampling module
    function   3-142
  SMF-related records
    for channels   3-133
    for CPU   3-119
    for devices   3-135
    for paging   3-123
  START parameters, processing of   3-81
  stopping MF/1   3-81
  syntax analyzer
    function   3-86
  SYSEVENT code (WKLDINIT) issued   3-99
  termination
    in measurement facility control   3-81
    processing
      function   3-110
  visual table of contents   3-79
  workload measurement
    initialization   3-98
    interval measurement gathering routine
      function   3-126
    report generator
      function   3-150
MFC (measurement facility control), IRBMFMFC in MF/1
  3-80
MFCOA (measurement facilities common options area, see
  also INCOA, STCOA, and TMCOA)
  in input merge control   3-84
  in mainline initialization of MF/1   3-90

  in measurement facility control 3-80
  in MFSTART mainline 3-82
  in MF/1 data control 3-106
  in MF/1 report generator control 3-146
MFDATA SVC routine (MF/1)
  function 3-114
  processing 3-114
MFIMAINL
  function 3-90
MFLISTOP
  function 3-88
MFMVT (measurement vector table for problem state options, see also DTMVT, INMVT, STMVT, TMMVT)
  in input merge control 3-84
  in mainline initialization of MF/1 3-90
  in measurement facility control 3-80
  in MFSTART mainline 3-82
  in MF/1 report generator control 3-146
MFPCT (problem control table)
  in measurement facility control 3-80
  in MF/1 data control 3-106
MFPMA (problem measurement area, see also TMPMA)
  in input merge control 3-84
  in mainline initialization of MF/1 3-90
  in MF/1 reprot generator control 3-146
MFROUTER service routine (IRBMFEVT) 3-138
  MF/1 channel initialization 3-100
MFSEL (subtask elements table)
  in MF/1 report generator control 3-146
MFSTART mainline (MF/1)
  function 3-82, 3-80, 3-91
minor name (rname)
  in job initiation enqueue parameter list 3-198-3-199
mount control blocks
  building and contents
    function 3-392-3-393
mount equalization for MSS volumes 3-291, 3-350, 3-370
mount failure for MSS volume 3-387
mount message, building, issuing; and verifying volumes 3-392-3-395
mounting a volume (see volume mount & verify)
MP (see multi-processor system)
MSRDA or BASEA (master scheduler resident data area)
  in allocation/initiator interface 3-398
  in common allocation control 3-280
  in initiator/allocation interface 3-398
  in log initialization 3-466
  in log task abnormal termination 3-478
  in terminating the system log 3-470
  in writing data on the system log 3-480
MSS
  mount equalization 3-291, 3-350, 3-370
  no AVR processing 3-341
MUG (multi-unit generic)
  ensuring each request is allocated to a single generic 3-366
  not successfully handled by algorithm 3-367
multiprogramming level **(VS2.03.807)**
  definition/description 3-3 **(VS2.03.807)**
  management to 3-3,3-23 **(VS2.03.807)**
  in resource monitor periodic monitoring 3-66 **(VS2.03.807)**
  in resource monitor MPL adjustment processing 3-67.0
  sampling and adjusting 3-66,3-67.0 **(VS2.03.807)**
multi-unit generic (see MUG)
multi-unit/multi-generic requests processing
  function 3-348, 3-366
multi-unit requests
  tape data sets, processing 3-380
  unsuccessful processing 3-379
  within a generic 3-317
multiple device type determination
  function 3-326, 3-328
multiple request for the same unit
  processing in fixed device control 3-299
MVCA
  in allocation/volume mount and verify (VM&V) interface 3-388
  in volume mount and verify (VM&V) 3-392
MVCA chain processor
  function 3-390, 3-388, 3-394
MVCAX
  in allocation/volume mount and verify (VM&V) interface 3-394

NEL (interpreter entrance list)
  creation by master subsystem 3-178
  in converter
    initialization 3-224
    processing commands in the input stream 3-230
    termination 3-242
  in SWA create interface 3-216
new address space (see address space)
NEWIPS SYSEVENT code (32)
  in workload manager 3-71
  processing in SRM SYSEVENT code processor 3-18
NIOWAIT SYSEVENT code (3)
  in SRM storage management 3-47
  processing in SRM SYSEVENT code processor 3-12
non-cancellable property as indicated in PPT 3-201
nonshareable device allocation 3-359
nonspecific volume allocation
  allocation recovery 3-358
  processing (IEFAB436) 3-308
  types of requests
    public volumes 3-308, 3-356
    storage volumes 3-308
  use with fixed device control in allocating to permanently resident or reserved volumes 3-296-3-297
  use with generic allocation in allocating to private volume or public volume 3-344-3-345
non-swapable property as indicated in PPT 3-200
normal dynamic allocation control
  function 3-414
"not ready" devices (in recovery allocation) 3-364-3-365
notification
  to active subsystem (function codes) 3-161

occupancy analysis of main storage in SRM 3-52
offline/allocated device allocation 3-366
  operator interface 3-374
  processing
    function 3-366
offline allocation requests 3-366
offlines/allocateds, processing
  function 3-374-3-375
OKSWAP SYSEVENT code (42)
  in SRM interface 3-7
  in workload manager 3-71
  processing in SRM SYSEVENT code processor 3-20
open checkpoint data set routine
  function 3-487
OPEN processing 3-204
Operation (see Method of Operation Section)
operator cancelled jobs
  processing in common allocation cleanup 3-382
operator console (see console)
options for MF/1
  in channel initialization 3-100
  in CPU initialization 3-96
  in device initialization 3-104
  in the MFDATA SVC Mainline 3-114
  in paging initialization 3-96
  in workload initialization 3-98
  listing 3-88
  merging on input 3-84
  validity checking 3-86
Organization (see Program Organization Section)
OUCB (system resources manager use control block)
  in control swap-in (IRARMCSI) 3-40 **(VS2.03.807)**
  in CPU management (IRARMCPM) 3-64 **(VS2.03.807)**
  in CPU load balancing swap analysis 3-66
  in deferred action processing 3-28
  in individual user evaluation (IRARMWM3) 3-73.0 **(VS2.03.807)**
  in partial analysis 3-36
  in SRM interface 3-9

  in storage management (IRARMSTM)   3-46 (**VS2.03.807**)
  in swappable user evaluation (IRARMWM2)   3-70 (**VS2.03.807**)
  in swap-in control   3-40
  in swap-out control   3-42
  in timer action analysis   3-26
  in user evaluation (IRARMCVL)   3-43.4 (**VS2.03.807**)
  in user ready processing (IRARMHIT)   3-73.2 (**VS2.03.807**)
  in workload management   3-70
OUT queue for SRM (**VS2.03.807**)
  definition   3-23.0 (**VS2.03.807**)
  in resource monitor periodic monitoring (IRARMRM1) 3-67 (**VS2.03.807**)
  in select user for swap-in (IRARMCPI)   3-43.0 (**VS2.03.807**)
  in swap analysis (IRARMCAP)   3-36 (**VS2.03.807**)
  in swappable user evaluation (IRARMWM2)   3-70 (**VS2.03.807**)
  in user ready processing (IRARMHIT)   3-73.2 (**VS2.03.807**)
OUXB (system resources manager user extension block)
  in control swap-in (IRARMCSI)   3-40 (**VS2.03.807**)
  in CPU management (SRM)   3-62
  in individual user evaluation (IRARMWM3)   3-73.0 (**VS2.03.807**)
  in I/O management (IRARMIOM)   3-54 (**VS2.03.807**)
  in storage management (IRARMSTM)   3-46 (**VS2.03.807**)
  in SRM interface   3-9
  in Swappable User evaluation (IRARMWMZ)   3-70
  in user evaluation (IRARMCVL)   3-43.4 (**VS2.03.807**)
  in user ready processing (IRARMHIT)   3-73.2 (**VS2.03.807**)
  in control swap-in   3-41
  in SYSEVENT processing in SRM SYSEVENT code processor   3-12
  in workload management   3-70
override processing in interpreter
  in creating and chaining tables   3-254
  in writing tables into SWA   3-257

packaging of SRM   3-3.2 (**VS2.03.807**)
page free request (see PGFREE)
page load (see PGLOAD)
page stealing   3-46-3-47
paging measurements for MF/1
  initialization   3-97
parameter value analysis in interpreter   3-248
parse (see IKJPARSE)
parse of MF/1 syntax   3-86
passed data set information scan
  function   3-334
path, device (see device path)
PCCA (physical communications configuration area)
  in interval measurement gathering routine for CPU 3-118
  in MF/1 channel initialization   3-101
  in MF/1 channel sampling module   3-140
PCCB (private catalog control block)
  in JFCB housekeeping control   3-314
  in JLOCATE   3-336
  in step initiation   3-204
PCCB routine
  function   3-336-3-337, 3-314-3-315
PDI (passed data set information)
  in DD function control   3-322
  in JLOCATE   3-334
  in job unallocation   3-410
  searching   3-334
PDI read and chain
  function   3-334-3-335
performance group descriptor (see WPGD) (**VS2.03.807**)
performance group period change
  by workload manager   3-70, 3-72
performance objective
  use by workload manager   3-71
permanently resident volumes
  allocating request for   3-294
PFK (see program function key)
pool (see quick cell)
posting SMF
  error exit   3-460
PPT (program properties table)
  in step initiation   3-200
  scan
    function   3-200-3-201
primary job entry subsystem initialization   3-176
"privileged" property (as indicated in program properties table)   3-200-3-201
PRLIST   3-294
PROC statement   3-233
procedure, cataloged, processing   3-232
procedure, in-stream, processing   3-232
process job condition codes
  function   3-406, 3-414
process TP requests
  function   3-286-3-287
processors, command (see command processing)
PROCSTEP (procedure step)   3-226
program properties table
  function   3-200-3-201
programmer, writing to (see WTP)
prompting exit (see pre-prompt exit, LOGON)
PSA (prefixed save area)
  in MF/1 channel sampling module   3-140
  in MF/1 second CPU test channel sampling module 3-142
PSCB (protected step control block)
  in dynamic allocation control   3-414
pseudo access method in subsystem initiation
  control
    function   3-182, 3-184
  direct read and write
    function   3-182-3-183
  sequential read and write
    function   3-184-3-185
PSLIST (public storage list)
  in nonspecific volume allocation control   3-308
public volume allocation requests
  in allocating nonspecific volume requests   3-308
  in demand allocation   3-356
  processing in fixed device control   3-294
PVT (page vector table)
  in interval measurement gathering routine for paging 3-122
  in main storage occupancy analysis   3-52
  in resource monitor MPL adjustment processing (IRARMRM2)   3-67.0 (**VS2.03.807**)
  in storage management (IRARMSTM)   3-46 (**VS2.03.807**)
  in swap-in control   3-40

QDB (queue descriptor block)
  in dynamic allocation   3-414
  in dynamic unallocation   3-416
  in SVC 99 control   3-412
QMNGRIO macro interface handler
  function   3-264
QMPA (queue management parameter area)
  in converter
    initialization   3-225
    processing in-stream and cataloged procedures 3-232
  in converter/interpreter interface   3-178
  in job deletion   3-210-3-211
  in job initiation   3-196
  in restart interface processing   3-510
  in step deletion   3-210-3-211
  in SWA create interface   3-216
  in SWA manager locate mode   3-266
  in SWA manager move mode   3-264
  in writing blocks to the job journal   3-520
QSCEFL SYSEVENT code (18)
  processing in SRM SYSEVENT code processor   3-15
QSCECMP SYSEVENT code (13)
  in SRM CPU load balancing swap analysis   3-67

  in SRM CPU management 3-63
  in SRM SYSEVENT code processor 3-14
  in SRM workload manager 3-71
QSCEST SYSEVENT code (12)
  in SRM I/O management (IRARMIOM) 3-54-3-55
  in SRM SYSEVENT code processor 3-14
queue manager processing
  in SWA manager move mode 3-264
quiesce processing
  in SRM swap-out control 3-42

RACF accessor environment **(VS2.03.804)**
  deleting 3-193,3-197 **(VS2.03.804)**
  initializing 3-216,3-217 **(VS2.03.804)**
  writing JCTX into SWA 3-256,3-257 **(VS2.03.804)**
RCT **(VS2.03.807)**
  in resource monitor MPL adjustment processing (IRARMRM2) 3-67.0 **(VS2.03.807)**
  in resource monitor periodic monitoring (IRARMRM1) 3-66 **(VS2.03.807)**
read
  in pseudo access method 3-182
READ macro instruction
  in SWA manager move mode 3-264-3-265
READ/LOCATE commands 3-266
real frame (see page frame)
real page shortage in SRM 3-52-3-53
real timer interval requests 3-508
recording, error (see error recording)
recovery allocation (IEFAB485) (see also allocation)
  conditions that cause execution of 3-289
  interface with operator
    function 3-374, 3-364
  online devices
    function 3-377
  processing
    function 3-358
  type requests that need recovery or retry 3-305
recovery, error (see error recovery ESTAI)
recovery, FRR (see functional recovery routine)
recovery reply options processor
  function 3-374, 3-376
recovery routine (see also functional recovery routine)
  for MF/1 3-83
REGION parameter 3-201
release data set
  function 3-396-3-397
remove in-use attribute routine (IEFDB480)
  functions 3-424
remove in-use control routine
  function 3-424
remove in-use processor
  function 3-416, 3-426
report generators, MF/1, calling
  in data control 3-109
  in report generation control 3-146-3-147
REQSERVC sysevent code, processing 3-19
REQSVDAT SYSEVENT code **(VS2.03.807)**
  processing in SRM SYSEVENT code processor 3-22.5 **(VS2.03.807)**
request router, common 3-172
request subsystem services function codes 3-161
requests, allocation
  not satisfied
    processing in common allocation clean-up (IEFAB490) 3-378
    retry criteria 3-379
  satisfied
    processing in common allocation clean-up 3-378
    multi-unit tape data sets, processing 3-380
    volume mount & verify interface 3-380
requests, region (see region requests)
reserved volume allocation requests
  processing in fixed device control (IEFAB430) 3-294
RESETPG SYSEVENT code (31)
  processing in SRM SYSEVENT code processor 3-17
  use by workload manager 3-71
resource factor coefficient, use of (RFC)
  in SRM CPU management 3-65
  in SRM I/O management 3-55
resources manager (see system resources manager)
resource monitor MPL adjustment **(VS2.03.807)**
  processing (IRARMRM2) 3-67.0,3-67.3 **(VS2.03.807)**
resource monitor periodic monitoring **(VS2.03.807)**
  (IRARMRM1) 3-66,3-67.3 **(VS2.03.807)**
restart (see also checkpoint/restart, DSS)
  automatic step 3-500
  interface routine
    function 3-510-3-511
  system
    processing 3-496
restart preparation routine
  function 3-516
restarting (see restart)
retry
  overview 3-273
  processing in common allocation clean-up 3-379
rewinding requests, processing in volume mount and verify 3-390
RLCT
  in I/O load balancing swap analysis 3-56
  in I/O management (SRM) 3-54
RMCA
  in CPU load balancing swap analysis 3-66
  in CPU management (SRM) 3-62
  in I/O load balancing swap analysis 3-56
  in I/O management (SRM) 3-54
  in partial analysis 3-36
  in workload management 3-72
RMCT (system resources manager control table)
  in algorithm request 3-30
  in CPU load balancing swap analysis 3-66
  in deferred action processing 3-28
  in full analysis 3-34
  in I/O load balancing swap analysis 3-56
  in I/O management (SRM) 3-54
  in main storage occupancy analysis 3-52
  in partial analysis 3-36
  in periodic entry point scheduling 3-32
  in select user for swap-in (IRARMCPI) 3-43.0 **(VS2.03.807)**
  in select user for swap-out (IRARMCPO) 3-43.2 **(VS2.03.807)**
  in SRM control 3-23
  in SRM interface 3-5
  in SRM service routine (IRARMSRV) 3-9.8 **(VS2.03.807)**
  in storage management (IRARMSTM) 3-46 **(VS2.03.807)**
  in swap analysis (IRARMCAP) 3-36 **(VS2.03.807)**
  in swappable user evaluation (IRARMWM2) 3-70 **(VS2.03.807)**
  in swap-in control 3-40
  in timer action analysis 3-26
  in user ready processing (IRARMHIT) 3-73.2 **(VS2.03.807)**
  in workload management 3-69
RMEP
  in periodic entry point scheduling 3-32
  in storage management (IRARMSTM) 3-46 **(VS2.03.807)**
  used in processing actions/algorithms 3-23.2,3-23.3 **(VS2.03.807)**
RMPT
  in CPU management (SRM) 3-62
  in partial analysis 3-36
  in periodic entry point scheduling 3-32
route requests to active subsystems 3-172
RPL (request parameter list)
  in log writer processing 3-474
  in pseudo access method 3-182
  in subsystem initiation message writer 3-186
RPL/ACB interface
  in subsystem initiation message writer 3-186
  in pseudo access method 3-182
RRPA
  in collect data for MF/1 (IRARMWR3) 3-78.8 **(VS2.03.807)**
  in full analysis 3-34

  in user evaluation (IRARMCVL) 3-43.4 (**VS2.03.807**)
RSM (see real storage manager)
RSMCNSTS SYSEVENT code (22)
  processing in SRM SYSEVENT code processor 3-16
RSTORCMP SYSEVENT code (19)
  processing in SRM SYSEVENT code processor 3-15
  use by SRM workload manager 3-71
RTB (response/throughput bias) (**VS2.03.807**)
  in user evaluation (IRARMCVL) 3-43.4 (**VS2.03.807**)
R/TM (see recovery termination)
RV (recommendation value) (**VS2.03.807**)
  in CPU management (IRARMCPM) 3-63 (**VS2.03.807**)
  in select user for swap-in 3-43.0 (**VS2.03.807**)
  in select user for swap-out 3-43.2 (**VS2.03.807**)
  in swap analysis 3-37 (**VS2.03.807**)
  in user evaluation 3-43.5 (**VS2.03.807**)

scan dictionary
  in converting statements to internal text 3-236
scheduler (see job scheduler)
scheduling, SRM periodic entry point IRARMCET 3-32
scratch requests 3-358
screen image buffer (see SIB)
SCT (step control table)
  in ABENDed job restart preparation 3-518
  in allocation/initiator interface 3-396
  in data set descriptor records, processing 3-486
  in DD function control 3-322
  in dynamic allocation 3-414
  in dynamic information retrieval 3-422
  in dynamic unallocation control 3-416
  in initiator/allocation interface 3-396
  in interpreter
    creating and chaining tables 3-252
    writing tables into SWA 3-256
  in JFCB housekeeping control 3-314
  in job deletion 3-208
  in step continue processing 3-494
  in step deletion 3-208
  in step initiation 3-200
  in SVC 99 control 3-412
searching for volser in UCBs (in job unallocation) 3-411
SECHT
  in SRM interface 3-5
second CPU test channel sampling module (IRBMFTCH), function 3-142
second level interrupt handler (see SLIH)
security environment (RACF) (**VS2.03.804**)
  deleting 3-193,3-197 (**VS2.03.804**)
  initializing 3-216,3-217 (**VS2.03.804**)
  writing JCTX into SWA 3-256,3-257 (**VS2.03.804**)
select user for swap-in (IRARMCPI) 3-43.0 (**VS2.03.807**)
select user for swap-in (IRARMCPO) 3-43.2 (**VS2.03.807**)
SETDMN SYSEVENT code (**VS2.03.807**)
  processing in SRM SYSEVENT code processor 3-20 (**VS2.03.807**)
setting domains 3-20 (**VS2.03.807**)
sequential read
  in pseudo access method 3-182
sequential write
  in pseudo access method 3-182
serialization
  in common allocation control 3-282
service rate
  explanation of use by SRM workload manager 3-69
shared data set attributes, replacing in job initiation 3-199
SIB (screen image buffer)
signal processor (see SIGP instruction)
single line message (see WTO)
SIOT (step I/O table)
  completing DCB information in 3-329
  completing DISP information in 3-333
  copying unit information into 3-331
  in allocate request to unit 3-302
  in allocating offline devices 3-366
  in allocation/initiator interface 3-396
  in allocation/volume mount and verify (VM&V) interface 3-386
  in allocation via algorithm 3-348
  in common allocation cleanup 3-378
  in common allocation control 3-280
  in common unallocation 3-432
  in data set descriptor records processing 3-486
  in DD function control 3-322
  in demand allocation 3-355
  in disposition processing 3-440
  in dynamic allocation control 3-414
  in dynamic concatenation 3-418
  in dynamic information retrieval 3-422
  in dynamic deconcatenation 3-420
  in dynamic unallocation control 3-416
  in fixed device control 3-294
  in generic allocation control 3-338
  in initiator/allocation interface 3-396
  in initiator/unallocation interface 3-402
  in interpreter 3-252
  in JFCB housekeeping control 3-314
  in JLOCATE 3-334
  in job unallocation 3-410
  in nonspecific volume allocation control 3-308
  in recovery allocation 3-358
  in remove in-use processor 3-424
  in specific volume allocation 3-298
  in SVC 99 control 3-412
  in unallocation/initiator interface 3-402
  in unit unallocation processing 3-444
  selecting in non-specific allocation control 3-308
SMCA (system management control area)
  in SMF cross-memory post error exit 3-460
  in STAE exit processing for SMF 3-458
  in switching SMF data sets 3-454, 3-456
  in volume mount and verify (VM&V)/allocation interface 3-386
  in writing SMF records 3-450
SMF (System Measurement Facility)
  cross-memory post error exit routine
    function 3-460
  in converter
    initialization 3-224
  dynamic dd routine
    function 3-418
  exit in step initiation 3-200
  initialization exit support module
    function 3-200-3-201
  interface to interpreter 3-256
  record manager
    function 3-452, 3-450-3-451
  record writing
    function 3-450-3-451, 3-456, 3-452, 3-454
  records, writing
    in MF/1 routine for channels 3-131
    in MF/1 routine for CPU 3-119
    in MF/1 routine for devices 3-135
    in MF/1 routine for paging 3-123
    in MF/1 routine for workload 3-127-3-129
    in step initiation 3-200
  STAE exit processing
    function 3-458
  SYSEVENT REQPGDAT (39) issued to obtain paging data 3-20
  TCTIOT construction interface
    function 3-206
SMF data set
  in MFDATA mainline 3-115
  opening of 3-450
  opening of alternate 3-454
  splitting 3-454, 3-453
  switching 3-454, 3-452, 2-303
SMF records
  count of, updating 3-456
  splitting 3-452
space, address (see address space)
special protect key 3-200
specific volume allocation control
  processing
    function 3-298
  use with fixed device control 3-295
  use with generic allocation 3-344
SQA

- in MF/1 device initialization 3-46

SQALOW SYSEVENT code (25)
- processing in SRM SYSEVENT code processor 3-16
- in SRM storage management 3-50-3-51

SRM (see also system resources manager)
- algorithm processor 3-23, 3-23.2, 3-23.3, 3-24 (**VS2.03.807**)
- collect data for MF/1 3-73.8 (**VS2.03.807**)
- control algorithm 3-23
- control swapin routine 3-40
- control swapout routine 3-42
- CPU load balancing swap analysis 3-66
- CPU management routines 3-62
- deferred action processor 3-28
- FRR 3-9
- full analysis retry function 3-34
- individual user evaluation 3-73.0 (**VS2.03.807**)
- initialize for MF/1 3-73.6 (**VS2.03.807**)
- interface module 3-6
- I/O load balancing swap analysis routine 3-56
- I/O management routines 3-54
- I/O load balancing user I/O monitoring 3-58 (**VS2.03.807**)
- main storage occupancy routine 3-52
- module/entry point cross reference 3-3.2, 3-3.3 (**VS2.03.807**)
- nonresident set to new IPS routine 3-32
- obtain/free SQA storage 3-9.6 (**VS2.03.807**)
- partial analysis routine 3-36
- periodic entry point scheduling routine 3-32
- processing algorithms and actions 3-23, 3-23.2, 3-23.3 (**VS2.03.807**)
- requeue SRM TQE 3-9.8 (**VS2.03.807**)
- resource monitor MPL adjustment processing 3-67.0 (**VS2.03.807**)
- resource use algorithms, overview 3-45
- resource monitor periodic monitoring 3-66 (**VS2.03.807**)
- RMEP algorithm and action invocation flags 3-23.3 (**VS2.03.807**)
- select user for swap-in 3-43.0 (**VS2.03.807**)
- select user for swap-out 3-43.2 (**VS2.03.807**)
- service routine 3-9.2 (**VS2.03.807**)
- storage management routines 3-46
- supervisor service request routine 3-48, 3-32, 3-42, 3-40
- swap analysis 3-36 (**VS2.03.807**)
- swappable user evaluation 3-70 (**VS2.03.807**)
- sysevent processor 3-11
- sysevent routers and processors 3-12
- timer action analysis 3-26
- user evaluation 3-43.4 (**VS2.03.807**)
- user ready processing 3-73.2 (**VS2.03.807**)
- uses ASM and RSM 3-3 (**VS2.03.807**)
- workload activity recording routine 3-70
- workload management function, overview 3-69
- workload manager algorithm module 3-70

SSCVT (subsystem communications vector table)
- in common request router 3-172
- in subsystem determination 3-174
- in subsystem interface 3-159

SSIB (subsystem identification block)
- in common request router 3-172
- in data set name assignment 3-188
- in job initiation 3-196
- in log initialization 3-466
- in subsystem determination 3-174
- in subsystem initiation 3-176
- in subsystem interface 3-159
- in switching log data sets 3-472
- in terminating the system log 3-470

SSOB (subsystem options block)
- in common request router 3-172
- in converter/interpreter interface 3-178
- in data set name assignment 3-188
- in job initiation 3-196
- in log initialization 3-466
- in subsystem determination 3-174
- in subsystem initiation 3-176
- in subsystem interface 3-159
- in subsystem job termination 3-190

SSVT (subsystem vector table)
- in subsystem interface 3-159

stack, FRR (see FRR stack)

STAE (set task asynchronous exit)
- for SMF 3-458

START command (see also START/LOGON/MOUNT overview)
- parameters used by MF/1 START command 3-84, 3-80

statement (see JCL statement)

status, console (see console status)

STC (started task control)
- SWA subpool for 3-267

step allocation processing in initiator/allocation interface 3-396

step continue processing (IEFXB601) 3-494

step delete routine
- function 3-208

step header reocrd, for job journal, building
- function 3-512, 3-514, 3-202

step initiation 3-200

step, preparing for allocation 3-396

step restart
- automatic, job journal processing for 3-500
- reconstructing SWA for 3-216

step time processing if job step is canceled 3-207

step unallocation 3-402

STEPCAT requests 3-314

STEPL (STAE exit parameter list)
- in job initiation 3-196

STCOA (common option area for supervisor state options, see also INCOA, MPCOA, TMCOA)
- in mainline initialization of MF/1 3-90

STGST (global supervisor table)
- in mainline initialization of MF/1 3-90
- in MF/1 workload initialization 3-98

STMMV
- in MFROUTER processor 3-138
- in MF/1 channel initialization 3-100
- in MF/1 device initialization 3-104

STMVT (measurement vector table for supervisor state options, see also DTMVT, INMVT, MFMVT, TMMVT)
- in mainline initialization of MF/1 3-90

STOP command
- MF/1 enabling use of 3-80

STOP command for MF/1
- in data control routine 3-108, 3-106
- in measurement facility control 3-80

STOP processing 3-196-3-197

STOP MONITOR command

storage deletion routine
- function 3-176-3-177

storage volume allocation requests
- in fixed device allocation (IEFAB430) 3-295
- in nonspecific volume allocation control 3-309

storage management (see real storage manager, virtual storage management, system resources manager)

STPRT
- in MF/1 channel initialization 3-100
- in MF/1 CPU initialization 3-96
- in MF/1 device initialization 3-104
- in MF/1 paging initialization 3-96
- in MF/1 workload initialization 3-98

stream, input (see converter)

structure, examining alternatives in MF/1 syntax analyzer 3-86

STRVT (resource vector table)
- in mainline initialization of MF/1 3-90

STSCT (supervisor control table)
- in mainline intialization of MF/1 3-90
- in MF/1 termination processor 3-110

STSGT
- in channel interval measurement gathering routine 3-130
- in interval measurement gathering routine for CPU 3-118
- in interval measurement gathering routine for devices 3-134
- in interval measurement gathering routine for paging 3-122

  in interval measurement gathering routine for workload 3-126
  in MFDATA mainline 3-114
  in MFROUTER processor 3-138
  in MF/1 channel initialization 3-100
  in MF/1 CPU initialization 3-96
  in MF/1 device initialization 3-104
  in MF/1 paging initialization 3-96
  in MF/1 termination processor 3-110
STSMA (supervisor measurement table)
  in channel interval measurement gathering routine 3-130
  in interval measurement gathering routine for CPU 3-118
  in interval measurement gathering routine for devices 3-134
  in interval measurement gathering routine for paging 3-122
  in mainline initialization of MF/1 3-90
  in MFDATA mainline 3-114
  in MF/1 channel initialization 3-100
  in MF/1 CPU initialization 3-96
  in MF/1 device initialization 3-104
  in MF/1 paging initialization 3-96
  in MF/1 workload initialization 3-98
subsystem allocation requests, processing in common allocation control 3-281
subsystem determination
  function 3-174
subsystem initiation 3-176
  in converter/interpreter interface 3-178
  in data set name assignment 3-188
  in pseudo access method 3-182
  message writer
    function 3-186
  processing
    function 3-176
subsystem/initiator SWA interface
  function 3-216
subsystem interface
  function codes 3-161
  introduction 3-159
subsystem job termination
  function 3-190
subsytem name, determination of 638
subsystem, routing request to 3-172
supervisor state, putting initiator task into 3-197
SVC interruptions (see supervisor interruptions handler)
SVC 34
  commands in the input stream, processing 3-230
SVC 99 control (IEFDB400) 3-412
  in JLOCATE 3-337
SVC 109 (see extended SVC routing)
SVC 116 (see extended SVC routing)
SVC 122 (see extended SVC routing)
SVCIH (see supervisor interruption handler)
SWA (scheduler work area)
  block length 3-265
  in automatic checkpoint/restart 3-498
  in automatic step restart 3-500
  in converter/interpreter interface 3-180
  in data set descriptor records processing 3-486
  in job initiation 3-196
  in merge cleanup 3-502
  in SWA manager locate mode 3-266
  in SWA manager move mode 3-264
  in system restart processing 3-496
  interface to in-stream and cataloged procedures 3-233
  interpreter writing tables into SWA 3-256
  merging from job journal 3-492
  virtual address in SWA, updating 3-504
SWA conversion from JCLS
  function 3-177, 3-179
SWA create interface
  function 3-216-3-217
SWA manager
  function code 3-264
  interface to in-stream and cataloged procedure processing 3-233
  interface to interpreter 3-256
  interface to job step allocation 3-397
  interface to job unallocation 3-411
  interface module
    function 3-264
  locate mode
    function 3-266
  move mode
    function 3-264
  SWA merge processing 3-492, 3-503
SWA prefix
  in SWA manager move mode 3-265
  in SWA manager locate mode 3-267
SWA reader routine
  function 3-396, 3-412
SWA reconstruction processing
  in journal merge error processing 3-509
  in restart processing 3-511
SWA subpool
  alternation 3-267
SWA virtual address
  in SWA manager move mode 3-265
  in SWA manager locate mode 3-267
swap (**VS2.03.807**)
  in SRM exchange swap 3-23.0 (**VS2.03.807**)
swap analysis 3-36 (**VS2.03.807**)
swap analysis in SRM
  CPU 3-62
  in partial analysis routine (IRARMCAP) 3-36
  I/O load 3-56
"swap package", definition of 3-37
swap-in (**VS2.03.807**)
  in SRM express swap-in 3-23.0 (**VS2.03.807**)
  in SRM select use for swap-in (IRARMCPI) 3-43.0 (**VS2.03.807**)
  in SRM unilateral swap-in 3-23.0, 3-40 (**VS2.03.807**)
swap-in, address space
  in SRM control swap-in (IRARMCSI) 3-40
  in SRM I/O management routine (IRARMIOM) 3-54, 3-57
  storage evaluation for in SRM partial analysis 3-36
swap-in, timer dependent, in SRM timer action analysis (IRARMCAT) 3-26
swap-out (**VS2.03.807**)
  in SRM select use for swap-out (IRARMCPO) 3-43.2 (**VS2.03.807**)
  in SRM unilateral swap-out 3-23.0, 3-42 (**VS2.03.807**)
swap-out, address space
  in SRM control swap-out 3-42
  in SRM I/O load balancing swap analysis 3-56
  in SRM partial analysis 3-36
  timer dependent, in SRM timer action analysis (IRARMCAT) 3-26
SWINFL SYSEVENT code (17)
  processing in SRM SYSEVENT code processor 3-15
switching SMF data sets 3-454, 3-452, 2-303
switching system log data sets (IEEMB803) 3-472
SWOUTCMP SYSEVENT code (15), processing in SRM SYSEVENT code processor 3-15
SWPINST SYSEVENT code (**VS2.03.807**)
  processing in SRM SYSEVENT code processor 3-16 (**VS2.03.807**)
symbolic parameters, processing in the converter 3-234
syntax analyzer of MF/1 input (IRBMFANL) 3-86
syntax checker for allocation
  function 3-424, 3-420, 3-422, 3-428, 3-418, 3-416
syntax errors in JCL symbolic parameters, scanning for in converter 3-235
SYQSCCMP SYSEVENT code (36)
  processing in SRM SYSEVENT code processor 3-18
SYQSCST SYSEVENT code (35)
  processing in SRM SYSEVENT code processor 3-18
SYSCHK DD statement processing in converter 3-229
SYSEVENT codes
  MF/1 related 3-98
  processing 3-12-3-22
  in SRM CPU management 3-63
  in SRM CPU swap load balancing analysis 3-67
  in SRM interface 3-5-3-9
  in SRM workload manager 3-71
  tracing 3-6

- SYSEVENT List 3-11 (**VS2.03.807**)
- SYSOUT data set name
  - assigning for subsystem initiation 3-188
- System Activities Measurement Facility (see MF/1)
- system log
  - allocating new 3-472
  - ESTAE processor
    - function 3-476
  - initialization 3-466
  - initializing log data set 3-466
  - message module
    - function 3-472, 3-468
  - switching 3-472
  - terminating 3-470
    - abnormally 3-476
  - unallocating
    - current 3-472
    - during termination 3-470
  - writer processing 3-474
  - writing 3-480, 2-306
- system log data set (see system log)
- system log initialization
  - writer module
    - function 3-466, 3-470, 3-474, 3-472
- System Measurement Facility (see SMF)
- system message interface routine
  - function 3-380, 3-398, 3-400
- system parameter library (see SYS1.PARMLIB)
- system reconfiguration (see reconfiguration commands)
- system resources manager (SRM) (see also workload manager) 3-3
  - algorithms
    - in periodic entry point scheduling 3-32
    - resource use algorithms, introduction to 3-45
  - algorithm request routine 3-30
  - allocated UCB list, use of 3-310-3-311
  - analysis routines
    - full 3-34
    - partial 3-36
  - automatic priority group (APG) management 3-45.1 (**VS2.03.807**)
  - auxiliary slot shortage prevention 3-45 (**VS2.03.807**)
  - consists of five functional groups (**VS2.03.807**)
    - control function 3-3 (**VS2.03.807**)
    - interface function 3-3 (**VS2.03.807**)
    - resource use algorithm 3-3 (**VS2.03.807**)
    - sysevent processor 3-3 (**VS2.03.807**)
    - workload manager 3-3 (**VS2.03.807**)
  - control routine 3-23,3-25
  - control swap-in routine 3-40
  - control swap-out routine 3-42
  - CPU management 3-62
  - CPU load balancing swap analysis 3-66
  - deferred action processor 3-28
  - domain MPL adjustment routine 3-67.0 (**VS2.03.807**)
  - ENQ/DEQ algorithm 3-9.8 (**VS2.03.807**)
  - error processing 3-9
  - functional recovery routine 3-9, 3-7
  - how packaged 3-3.2 (**VS2.03.807**)
  - interface, general 3-5
  - interface
    - to UCB cnadidates list in offline/allocated device allocation 3-371
  - interface routine (IRARMINT) 3-6, 3-5
    - in non-specific volume allocation control 3-312-3-313
    - with allocation in common allocation cleanup 3-382
  - introduction, general 3-3
  - I/O load balancing swap analysis 3-56
  - I/O management routine 3-54
  - lock, obtaining and releasing in full analysis routine 3-34
  - locking considerations for SYSEVENTs 3-5
  - main storage occupancy analysis 3-52
  - page replacement 3-45 (**VS2.03.807**)
  - pageable real storage shortage prevention 3-45 (**VS2.03.807**)
  - periodic entry point scheduling routine 3-32
  - PSLIST use 3-311
  - real page shortage prevention 3-45 (**VS2.03.807**)
  - resource monitor 3-45.1 (**VS2.03.807**)
  - resource use algorithms, introduction to 3-45
  - SQA shortage prevention 3-45 (**VS2.03.807**)
  - SRB analysis processing 3-34
  - SRB scheduling 3-23 (**VS2.03.807**)
  - storage management routine 3-46
  - swap evaluation 3-37
  - swap-in control routine 3-40
  - swap-out control 3-42
  - SYSEVENT code
    - locking considerations 3-5
    - MF/1 related (WKLDINIT) 3-99
    - processing 3-11-3-22
  - time-driven queue
    - defintion of 3-3
  - timer action analysis 3-26
  - TQE
    - in periodic entry point scheduling 3-33
  - visual contents for HIPO diagrams 3-4
  - visual table of contents 3-4 (**VS2.03.807**)
  - workload manager
    - introduction to 3-69
    - routine 3-70
- system restart
  - processing to cause restart at the next step 3-494
- system, stopping (see stopping)
- system trace (see trace, system)
- system trace termination (see trace termination)
- SYS1.LINKLIB, obtaining master JCL from in subsystem initiation 3-176
- SYS1.PROCLIB
  - not opened successfully in subsystem initiation 3-177

- tape label read
  - function 3-340, 3-418
- tape unloading (**VS2.03.804**)
  - in unit allocation 3-445,3-446 (**VS2.03.804**)
- TCB (task control block)
  - in allocation/initiator interface 3-396
  - in ABENDed job restart preparation 3-516
  - in common unallocation 3-430
  - in dynamic allocation control 3-414
  - in initiator/allocation interface 3-396
  - in initiator/unallocation interface 3-402
  - in job deletion 3-208
  - in log initialization 3-466
  - in merging job journal to SWA 3-492
  - in MFDATA mainline 3-114
  - in remove in-use attribute 3-424
  - in step deletion 3-208
  - in step initiation 3-200
  - in subsystem interface 3-159
  - in SVC 99 control 3-412
  - in unallocation/initiator interface 3-402
  - in writing blocks to the job journal 3-520
- TCT (timing control table)
  - in initiator/allocation interface 3-396
- TCTIOT (timing control table I/O table)
  - in allocate request to unit 3-302
  - in dynamic allocation control 3-414
  - in step initiation 3-200
- teleprocessing
  - allocation of device requests 3-286
- terminal recognizer, calling in MF/1 syntax analyzer 3-86
- terinating allocation error processing (common allocation cleanup) 3-378
- termination, abnormal, log task 3-476
- termination, system activities measurement 3-110
- terminator (see initiator/terminator)
- TERMWAIT SYSEVENT code (2)
  - processing in SRM SYSEVNET code processor 3-12
  - use by workload manager 3-69
- text EXEC statement condition codes
  - function 3-500
- test if device is ready
  - function 3-340-3-341
- text, internal (see converter, internal text)
- text keys, in ddname allocation 3-429
- TGETTPUT SYSEVENT code (34)

  processing in SRM SYSEVENT code processor 3-18
  use by workload manager 3-71
time dependent swap-in processing 3-26
time driven queue (in SRM), definition of 3-3
time limit, step 3-202
timer action analysis in SRM 3-26
timer second level interrupt handler (see timer SLIH)
TIMEREXP SYSEVENT code (1)
  processing in SRM SYSEVENT code processor 3-12
  in periodic entry point scheduling 3-32-3-33
timing control in step initiation 3-202
TIOT (task input/output table)
  in allocating offline devices 3-366
  in allocation via algorithm 3-348
  in common allocation cleanup 3-378
  in common allocation control 3-280
  in ddname allocation control 3-428
  in demand allocation 3-355
  in dynamic allocation control 3-414
  in dynamic concatenation 3-418
  in dynamic deconcatenation 3-420
  in dynamic unallocation control 3-416
  in dynamic information retrieval 3-422
  in generic allocation control 3-338
  in nonspecific volume allocation control 3-308
  in recovery allocation 3-358
  in remove in-use processor 3-426
  in specific volume allocation 3-298
  in step initiation 3-200
TIOT, expandable, build, update, rebuild
  function 3-280-3-281, 3-302-3-303
TIOT manager control routine
  function 3-396, 3-398, 3-418, 3-436
TPCA (see TPC)
TQE (timer queue element)
  in MFROUTER processor 3-139
  in requeue SRM TQE (IRARMI05) 3-9.8 (**VS2.03.807**)
  in SRM periodic entry point scheduling 3-33
TSB (terminal status block)
  in dynamic allocation control 3-414
TSO LOGON (see LOGON)

UCB (unit control block)
  in allocate request to unit 3-302
  in allocating offline devices 3-366
  in allocation/initiator interface 3-396
  in allocation/volume mount and verify (VM&V) interface 3-386
  in allocation via algorithm 3-348
  in common allocation cleanup 3-378
  in common allocation control 3-280
  in common unallocation 3-430
  in demand allocation 3-355
  in dynamic allocation control 3-414
  in dynamic unallocation control 3-416
  in fixed device control 3-294
  in generic allocation control 3-338
  in initiator/allocation interface 3-396
  in initiator/unallocation interface 3-402
  in job unallocation 3-410
  in MF/1 device initialization 3-104
  in MF/1 device sampling module 3-144
  in nonspecific volume allocation control 3-308
  in recovery allocation 3-358
  in specific volume allocation 3-298
  in unallocation/initiator interface 3-402
  in volume mount and verify (VM&V)/allocation interface 3-386
UCB candidate list in offline/allocated device allocation building, and interfacing with SRM 3-371
UCB list in nonspecific volume allocation control
  building
    function 3-350, 3-310
  interfacing with SRM 3-312
  releasing 3-312
UCB update routne
  use in allocate request to unit 3-303
UCM (unit control module)
  in communications task overview 3-418
  in unit unallocation processing 3-444
UCME (unit control module entry)
  in communications task overview 3-418
UIC (in referenced internal count) (**VS2.03.807**)
  in resource monitor periodic monitoring 3-67.1 (**VS2.03.807**)
  in SRM service routine 3-9.3 (**VS2.03.807**)
  in storage management (IRARMSTM) 3-46 (**VS2.03.807**)
unallocate requests to be rearranged
  function 3-360, 3-348, 3-368
unallocated volunit entries, processing 3-371
unallocating current system log 3-472
unallocation (see also allocation/unallocation)
  control, common
    function 3-430
  dynamic unallocation control and processor
    function 3-416, 3-426
  job unallocation
    function 3-410
  step unallocation control
    function 3-404-3-405
unallocation deserialization (**VS2.03.804**)
  in common allocation cleanup 3-381 (**VS2.03.804**)
  in job unallocation 3-411 (**VS2.03.804**)
unallocation/initiator interface
  function 3-402, 3-404
unilateral swap-out/swap-in 3-23.0, 3-36 (**VS2.03.807**)
unit affinity (see allocating affinity requests)
unit, allocating request to (see allocating requests to units)
unit allocation
  in offline/allocated device allocation 3-369
unit, eligible
  in specific volume allocation control 3-299
unit information
  copying in dd function control 3-323
  retrieving in dd function control 3-323
  in JFCB housekeeping 3-316
unit name conversion
  function 3-316-3-317
unit unallocation (IEFAB4A4)
  direct access device processing 3-446
  in common unallocation 3-434
  non-direct access device processing 3-446
  tape processing 3-446
units code
  in workload manager 3-71
unload requests, processing in volume mount and verify 3-390
  for an MSS volume 3-391
unmounted volumes, processing in demand allocation 3-357
update algorithm tables
  function 3-368-3-369
update DDR count routine
  function 3-282-3-283
update UCB routine
  function 3-302
user evaluation (IRARMCVL) 3-43.4 (**VS2.03.807**)
user ready processing (IRARMHIT) 3-73.2 (**VS2.03.807**)
user, swapping (see swap-in, swap-out)
USERRDY SYSEVENT code (4)
  processing in SRM SYSEVENT code processor 3-12
  use by workload manager 3-71

validity checking unallocated device or data set requests that need a specific volume 3-368
values, IPS (see IPS values)
VAT (virual address table)
  in automatic step restart 3-500
  in journal merge error processing 3-508
  in merge cleanup 3-502
  in merging job journal to SWA 3-492
  in restart interface processing 3-510
  in system restart processing 3-496
  in updating virtual addresses in SWA 3-504
verbs, converter identifying on JCL statement 3-226
verify control routine
  function 3-394

VERIFYPG SYSEVENT code (30)
  processing in SRM SYSEVENT code processor 3-17
VIO allocation requests
  processing in common allocation control 3-280
VIO eligible requests, processing in dd function control 3-328
virtual address in SWA, updating 3-504
VM&V count tables, contents 3-391
VM&V request block
  building in allocate request to unit 3-302
  in allocation/volume mount and verify (VM&V) interface 3-386
  in common allocation cleanup 3-378
  in volume mount and verify (VM&V) 3-390
VOLSER searching in job unallocation 3-411
volume allocation control, nonspecific
  function 3-308
volume demounting for an MSS volume 3-391
volume, determining if it is on an eligible unit 3-298
volume, enqueueing on in nonspecific volume allocation control 3-312
volume information
  copying and retrieving in dd function control 3-327
volume list
  building
    in common unallocation control 3-434
    in disposition processing 3-440
  obtaining new 3-336
volume mount and verify (VM&V)
  conditions under which a volume cannot be used 3-395
  control
    function 3-390
  determining functions to be performed 3-390
  DOMR and cleanup routine
    function 3-394, 3-388
  functions 3-390
  interface with allocation
    function 3-386
  interface to common allocation cleanup 3-395
  MSS mount request, handling 3-392
  routines used 3-390
  WTO/WTOR format routine
    function 3-392
volume, releasing
  function 3-434, 3-436-3-437
volume serial number (see VOLSER)
volume, specific allocation (see specific volume allocation control)
volume/unit resolution
  function 3-326
volume validity checker
  function 3-368, 3-302
volume, requests for
  permanently resident or reserved
    processing in fixed device allocation 3-294
  specific 3-298
volume unload control (see IEFAB494 object module)
volume unload for an MSS volume 3-391
volumes, verifying 3-394-3-395
volunit affinity processing
  function 3-282-3-283
volunit table
  eligible entries, locating in nonspecific volume allocation control 3-308
  in allocate request to unit 3-302
  in allocating offline devices 3-366
  in allocation/volume mount and verify (VM&V) interface 3-386
  in allocation via algorithm 3-348
  in common allocation cleanup 3-378
  in common allocation control 3-280
  in demand allocation 3-356
  in fixed device control 3-294
  in generic allocation control 3-338
  in nonspecific volume allocation control 3-308
  in recovery allocation 3-358
  in specific volume allocation 3-298
  in volume mount and verify (VM&V)/allocation interface 3-386
  obtaining space for 3-282-3-283
VSM (see virtual storage management)
VUT (volume unit table)
  in job unallocation 3-410
  in unit unallocation processing 3-444

wait holding resources
  function 3-280, 3-366
wait queue (deferred action queue) in SRM 3-28
WAIT queue for SRM **(VS2.03.807)**
  definition 3-23.0 **(VS2.03.807)**
  in user ready processing (IRARMHIT) 3-73.2 **(VS2.03.807)**
WAMT
  in collect data for MF/1 (IRARMWR3) 3-73.8 **(VS2.03.807)**
  in initialize for MF/1 (IRARMWR1) 3-73.6 **(VS2.03.807)**
  in interval measurements gathering routine for workload 3-126
  is SYSEVENT processing in SRM SYSEVENT code processor 3-11
warmstart
  in SWA create interface 3-217
  locating JFCBs and JFCBXs in initiator /unallocation interface 3-403
WKLDCOLL SYSEVENT code (46)
  processing in SRM SYSEVENT code processor 3-21
  use by workload manger 3-71
WKLDTERM SYSEVENT code (47)
  processing in SRM SYSEVENT code processor 3-21
WLLDINIT SYSEVENT code (45)
  processing in SRM SYSEVENT code processor 3-21
  use by workload manager 3-71
WMST (workload manager specification table)
  in individual user evaluation (IRARMWM3) 3-73.0 **(VS2.03.807)**
  in swappable user evaluation (IRARMWM2) 3-70 **(VS2.03.807)**
  in user evaluation (IRARMCVL) 3-43.4 **(VS2.03.807)**
  in workload management 3-69
  initialize for MF/1 (IRARMWR1) 3-73.6 **(VS2.03.807)**
work masks (group masks) in device allocation
  definition 3-272
  use of 3-339
workload level, introduction 3-69
workload manager
  function 3-69-3-73
workload measurement in MF/1
  initialization 3-98
  interval routine (RBMFDWP) 3-126
WPGD (performance group descriptor) **(VS2.03.807)**
  in individual user evaluation (IRARMWM3) 3-73.0 **(VS2.03.807)**
  in swappable user evaluation (IRARMWM2) 3-70 **(VS2.03.807)**
  in user evaluation (IRARMCVL) 3-43.4 **(VS2.03.807)**
  in user ready processing (IRARMHIT) 3-73.2 **(VS2.03.807)**
WPGDT
  in workload mangement 3-69
wrap-around CPU values in MF/1, adjusting for 3-119
write
  in pseudo access method 3-184
WRITE/ASSIGN function
  in SWA manager move mode 3-264-3-265
WRITE/LOCATE function
  in SWA manager locate mode 3-267
WRITE request
  in SWA manager move mode 3-264-3-265
WRITELOG command
  in switching system log data sets 3-473
write-to-log
  writing data to system log
    function 3-480
write-to-programmer (see WTP)
WTL (write to log) 3-480
WTO (write-to-operator) macro instruction
  in subsystem initiation message writer 3-186

3800 printing subsystem (**VS2.03.810**)
    in interpreter   3-245 (**VS2.03.810**)
    related JCL parameters   3-249 (**VS2.03.810**)
    (see also JFCBE and JFCBX) (**VS2.03.810**)

OS/VS2
System Logic Library
Volume 3
SY28-0763-0

READER'S COMMENT FORM

Your views about this publication may help improve its usefulness; this form will be sent to the author's department for appropriate action. Using this form to request system assistance or additional publications will delay response, however. *For more direct handling of such requests, please contact your IBM representative or the IBM Branch Office serving your locality.*

Possible topics for comment are:

Clarity Accuracy Completeness Organization Index Figures Examples Legibility

What is your occupation? ______________________________

Number of latest Technical Newsletter (if any) concerning this publication: ______________

Please indicate your name and address in the space below if you wish a reply.

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A.

Cut or Fold Along Line

Your comments, please . . .

This manual is part of a library that serves as a reference source for system analysts, programmers, and operators of IBM systems. Your comments on the other side of this form will be carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

Cut or Fold Along Line

Fold Fold

First Class
Permit 81
Poughkeepsie
New York

Business Reply Mail
No postage stamp necessary if mailed in the U.S.A.

Postage will be paid by:

International Business Machines Corporation
Department D58, Building 706-2
PO Box 390
Poughkeepsie, New York 12602

Fold Fold

IBM®

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
(U.S.A. only)

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
(International)