SY28-0765-0
File No. S370-36

Systems

OS/VS2
System Logic Library
Volume 5

VS2.03.805
VS2.03.807

IBM

This minor revision incorporates the following Selectable Units:

Supervisor Performance #1	VS2.03.805
Supervisor Performance #2	VS2.03.807

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 SY28-0717-0 incorporating changes released in the following Selectable Units Newsletters:

SN28-2688 (dated May 28, 1976)
SN28-2694 (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

Figures

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.

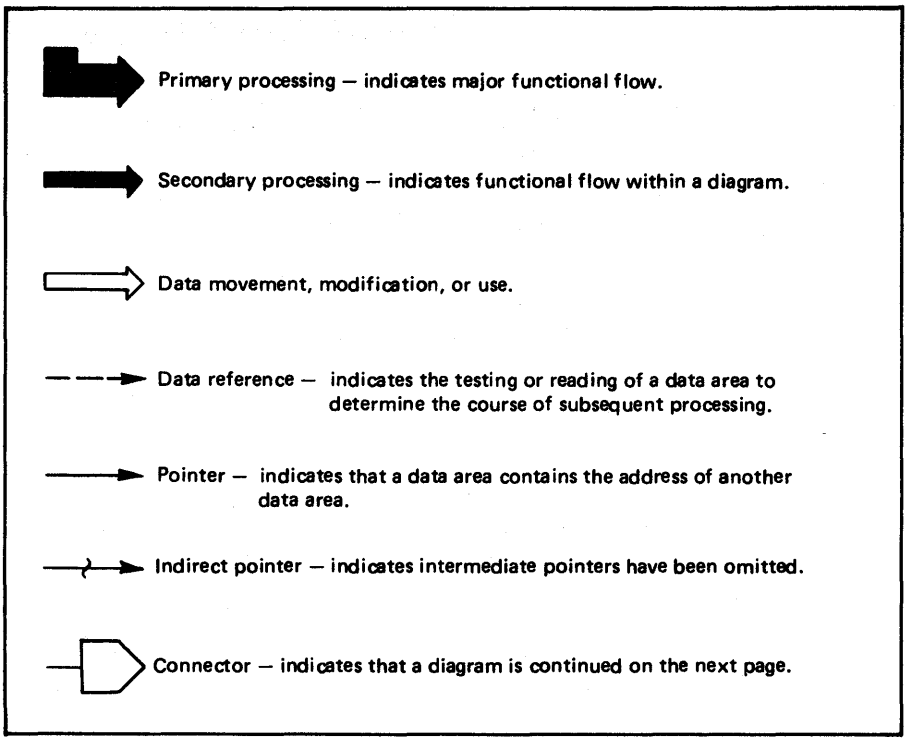


Figure 2-1. Key to Symbols Used in Method-of-Operation Diagrams

Real Storage Management (RSM) routines administer the use of real storage and direct the movement of virtual pages between auxiliary storage and real storage in page-size blocks. The routines make all addressable virtual storage in each address space appear as real storage to the user. Only virtual pages necessary for execution are kept in real storage; the remainder reside on auxiliary storage. RSM calls Auxiliary Storage Management (ASM) routines to perform the paging I/O necessary to transfer pages into and out of real storage. ASM also provides direct storage allocation and management for paging I/O space on auxiliary storage. The System Resources Manager provides guidance for RSM in the performance of some of these functions.

RSM assigns real storage frames on request from a pool of available frames (the available frame queue), associating virtual addresses with real storage addresses. Frames are repossessed on termination of use, when freed by a user, when a user is swapped out, or when needed to replenish the available frame queue. While a virtual page occupies a real frame, the page is considered pageable unless specified as fixed, either by the PGFIX routine, or by the system for its own use. RSM routines also allocate nonpageable (V=R) regions on request by those programs that cannot tolerate dynamic relocation. Such a region is allocated from a predefined area of real storage and is nonpageable. Programs in the V=R region do use dynamic address translation, although the addressing is on a one-to-one basis.

RSM routines determine the working set size for swap-in and swap-out functions. They maintain the necessary information to remove the virtual pages of an address space from real storage during swap-out and to re-establish them during swap-in. ASM provides the paging I/O for the swap function.

RSM also provides a set of service routines for use by the system:

- Table building for address translation
- Page fault processing
- Alteration of the pageable status of virtual pages
- Capability for paging in virtual pages before needed
- Capability for paging out selected pages
- Address translation from real to virtual addresses
- Varying real storage frames online or offline
- Virtual I/O (VIO) services
- Error recovery processing

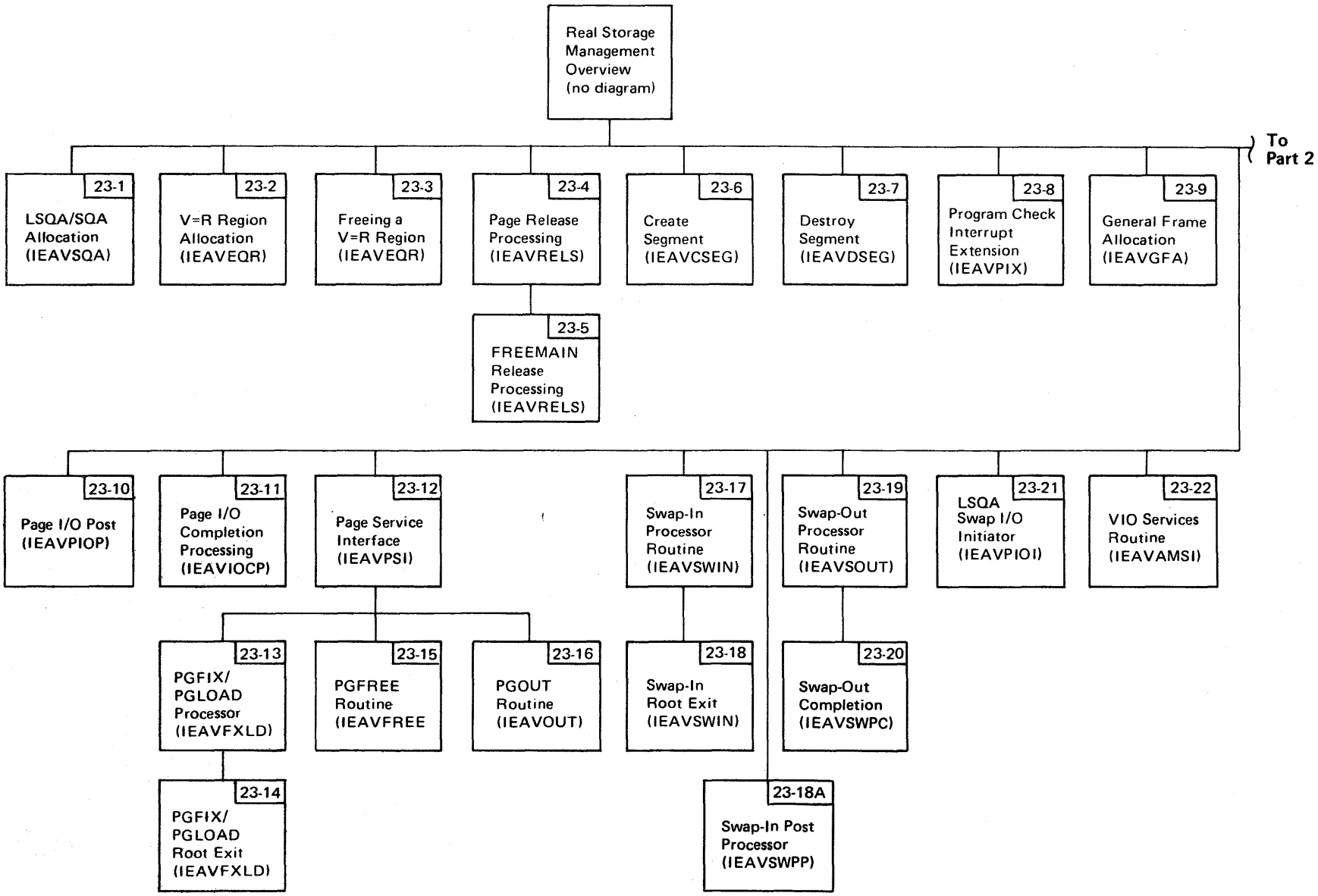


Figure 2-43. Real Storage Management Visual Contents (Part 1 of 2)

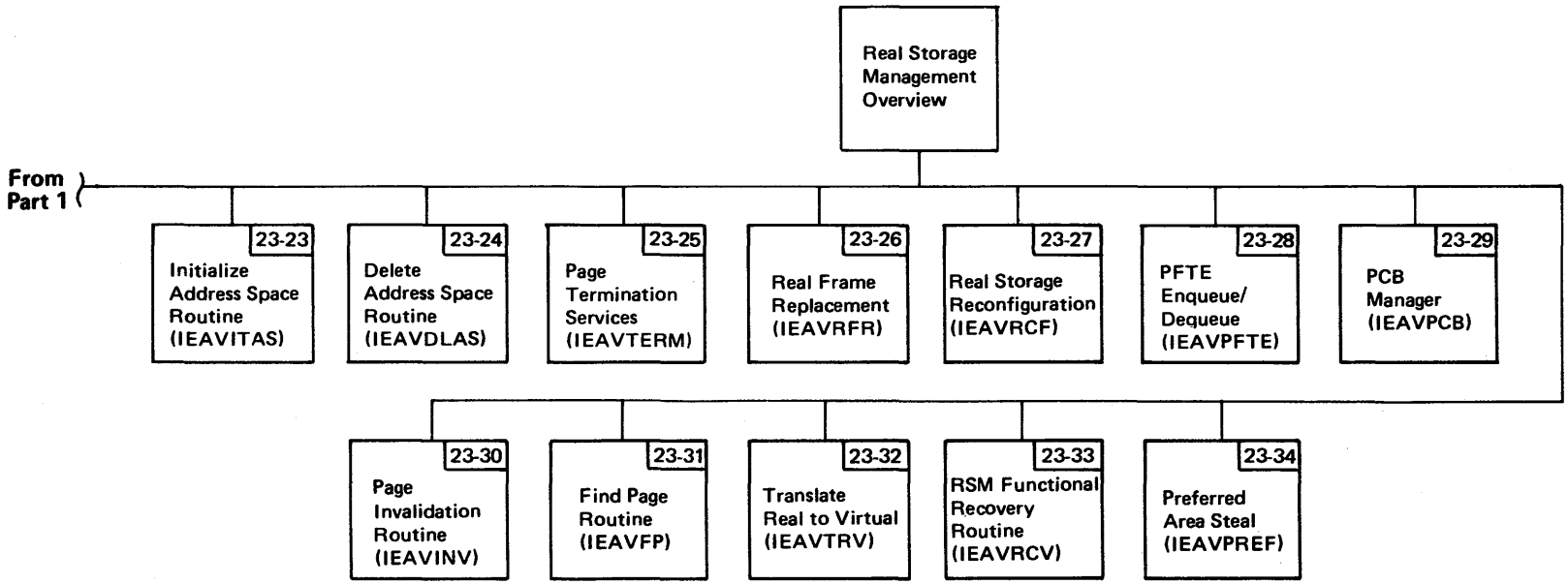


Figure 2-43. Real Storage Management Visual Contents (Part 2 of 2)

Diagram 23-1. LSQA/SQA Allocation (IEAVSQA) (Part 1 of 2)

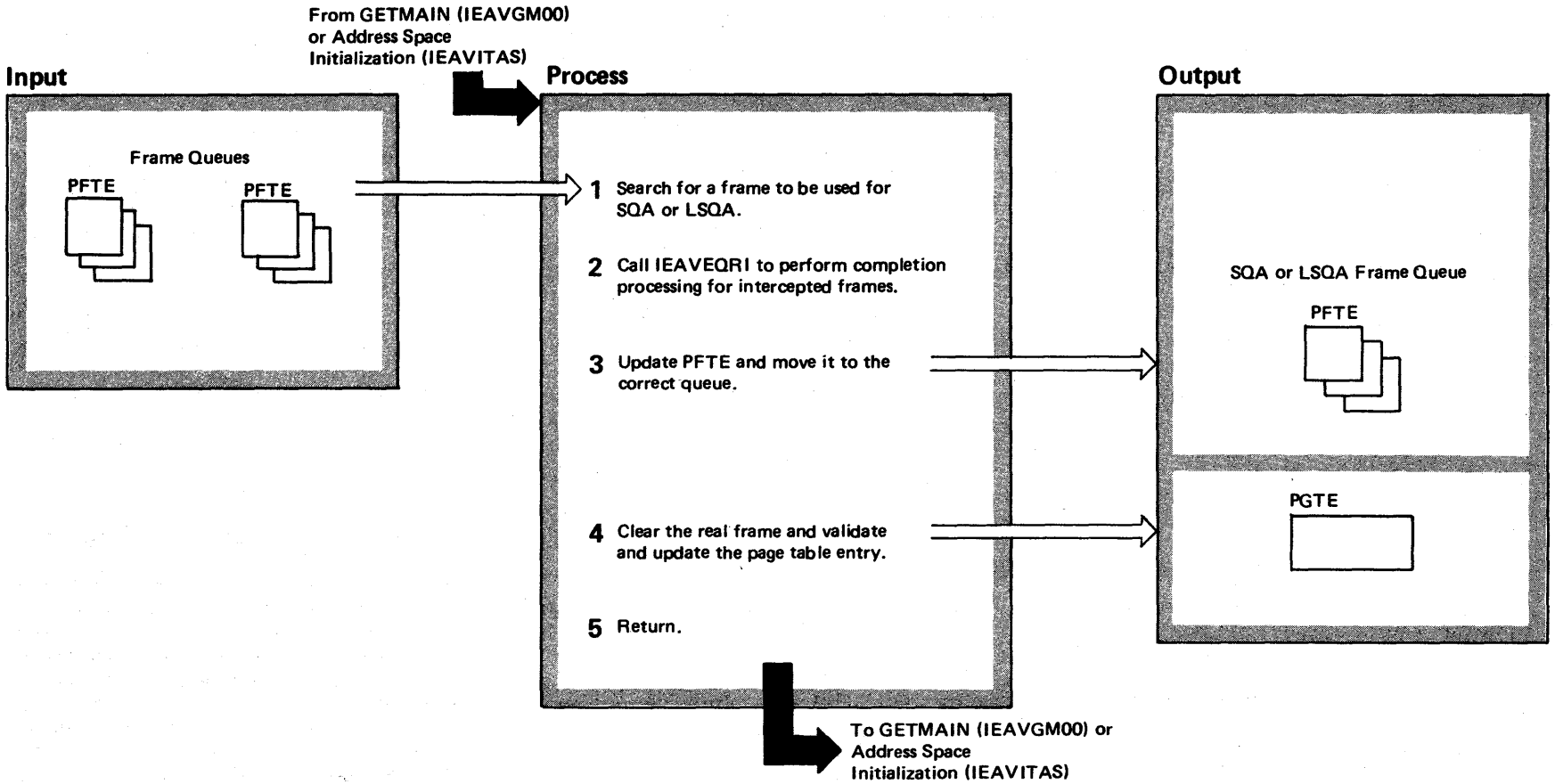


Diagram 23-1. LSQA/SQA Allocation (IEAVSQA) (Part 2 of 2)

Extended Description	Module	Label
SQA or LSQA Allocation (IEAVSQA) assigns real storage frames to those virtual pages that VSM or RSM's Address Space Initialization routine specifies to be SQA or LSQA pages. The caller holds the SALLOC lock and is in key 0, supervisor state.		
1 Satisfaction of the request is first attempted by accessing the AFQ (available frame queue) to find a frame outside the V=R area and also, if desirable, within the preferred area. If a PFTE (page frame table entry) for such a frame is found on this queue, it is dequeued and the request will be satisfied. If no preferred area frames exist on the AFQ, an attempt is made to steal a preferred area frame that holds a virtual page. Only unchanged, non-fixed frames for which no PCB exists are candidates for this stealing. Frames which are fixed, allocated to an active V=R region, offline, are changed, have paging I/O in progress, or contain a storage error are excluded. The Local and Common Frame queues are searched (in that order) for a frame meeting the steal criteria.	IEAVSQA	
The search stops as soon as a stealable frame in the preferred area is found. If none can be stolen, non-preferred area frames outside the V=R area on the AFQ become candidates, and one is stolen if it exists. If no such frames exist on the AFQ, but one was found on the local or common frame queue, it will be stolen and used for the request. If any such non-preferred, non-V=R frame is used, the physical storage unit containing the frame is converted from non-preferred to preferred storage.		
If no pageable area frames can be found, the V=R area frames on the AFQ become candidates and one is taken if any exist. If none can be found, the V=R area frames of the other queues become candidates and one will be taken. Frames that have been intercepted for a V=R region are skipped if a stealable, non-intercepted, V=R area frame exists on any of the queues. If no frame could be obtained and the request is for an LSQA page or unassigned frame (VSA=0), no further action is taken and a return code of 4 is passed to the caller.		
For SQA requests, the search moves to the SQA Reserve queue, where a certain number of frames are held, usually one. When a reserved frame is taken, the SQA Reserve Queue Deficit count is increased, telling the PFTE Enqueue routine that the next frame sent to the AFQ should be diverted to the SQA Reserve queue to replenish it. If the SQA Reserve queue is also empty, an out of real storage condition exists and return code 4 is given to the caller.		
2 If the selected frame was previously intercepted for a V=R region, the V=R Wait queue is scanned to locate the root PCB for the V=R region so that it can be marked as failed. Then the IEAVEQRI entry is called, passing the RBN of the intercepted PFTE to start the process that will lead V=R allocation to post the region ECB with code 16.		
3 If the input virtual storage address was 0, the VBN (virtual block number) in the PFTE of the selected frame is set to zero, as are all the PFTE flags except PFTVR (V=R area) and the Intercept flags. The PFTLSQA flag is also set. The PFTE is dequeued and the RBN (real block number) of the frame is placed in register one before returning to the caller with a code of zero.		
If the input virtual storage address is not zero, the frame is to be assigned to the page corresponding to the VSA. First the PFTE is moved to the LSQA or SQA frame queue, depending on whether the VSA is in the private or common area address range. The frame counts of the sending and receiving queues are adjusted where necessary. The VBN of the page is placed in the PFTE and either the current ASID or x'FFFF' (for SQA pages) placed in the PFTASID field. The PFTLSQA is turned on and all other flags except the PFTVR and intercept flags turned off. The system fix counters are also incremented.		
4 The PGTE for the page is updated with the real address of the block, the GETMAIN bit is set to one, and the invalid bit is set to zero. The XPTE protect key field is set to zero (for LSQA only). The real storage key is also set to zero. Finally, the entire page is cleared to zeros.		

Diagram 23-2. V=R Region Allocation (IEAVEQR) (Part 1 of 4)

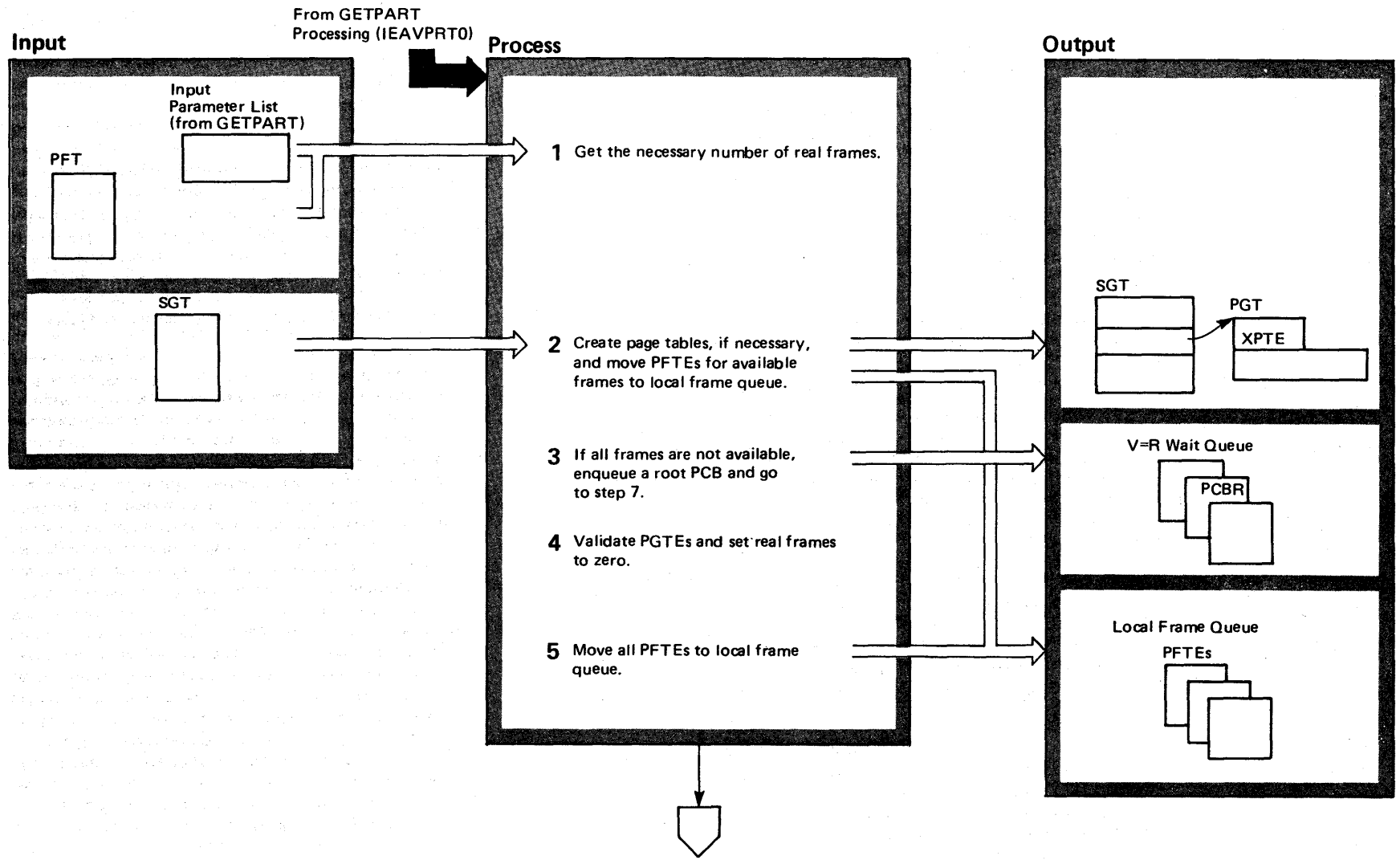


Diagram 23-2. V=R Region Allocation (IEAVEQR) (Part 2 of 4)

Extended Description	Module	Label
V=R Region Allocation (IEAVEQR) allocates contiguous regions of real storage for V=R requests.		
1 Upon receiving control, V=R allocation attempts to locate the proper number of contiguous real frames necessary for the region request. This is done by indexing through the PFT, starting with the PFTE that corresponds to the VSM-supplied starting address, and selecting frames for use. Frames need not be on the AFQ to be included in the region. Frames not available will be marked as intercepted for V=R allocation; they will be picked up later, as they become available. If an intercepted frame's page is in the current address space, it is paged out, thereby freeing up the frame for V=R. If an SQA, LSQA, long-fixed V=R allocated, offline, or intercepted page frame is encountered, the search is terminated and any frames already assigned to the region are restored to their previous status. If VSM indicated the region must start at the specified address, the allocation process is terminated and VSM is informed that allocation has failed with a return code of 16. If this requirement was not specified, the search is restarted with the first page frame following the unusable frame. This process continues until the region is allocated or the V=R area has been exhausted. If no region can be allocated anywhere in the V=R area, VSM is informed via return code 16 that allocation has failed.	IEAVEQR	IEAVEQR
2 Once a region has been allocated, the page tables are created where necessary. Then the status of each frame is determined. All frames selected from the AFQ are moved to the local frame queue and the system fix counters (RSMCNTFX and PVTCNTFX) are incremented. Also, fields in the PFTE are updated to reflect the new owner. Frames not on the AFQ have the PFTVRINT flag set. The starting address of the region is placed in the start address field of the input list.		
3 If all frames are not immediately obtained, a root PCB is placed on the V=R Wait queue pointed to by the PVT. Into the root is placed the count of the number of allocated but unavailable (intercepted) frames, the ASCB address, the address range of the selected region, and the input ECB address. Return code four is placed in register 15 and IEAVEQR exits.		
4 All PFTEs are moved to the local frame queue.		
5 The root PCB can be freed by putting it on the available queue.		

Diagram 23-2. V=R Region Allocation (IEAVEQR) (Part 3 of 4)

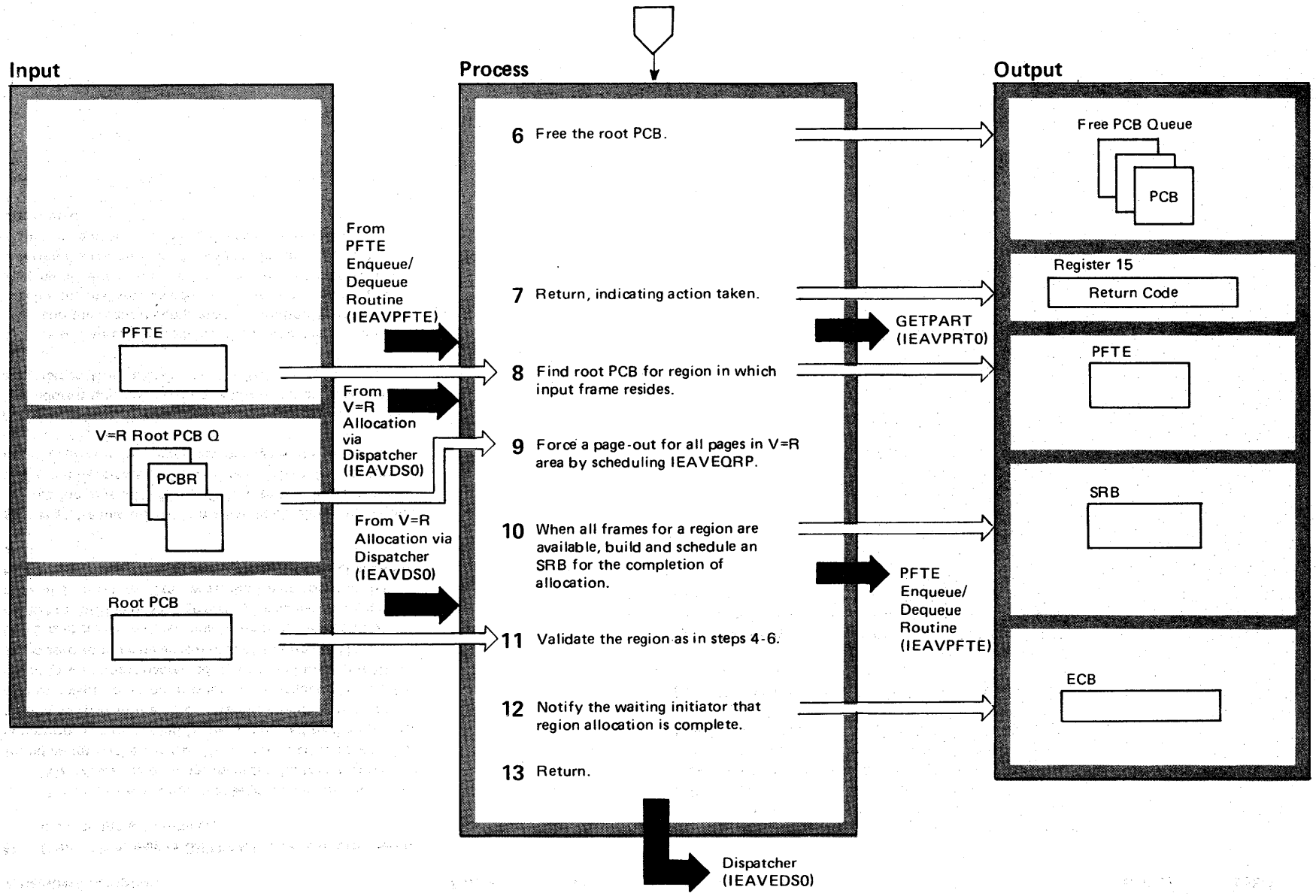


Diagram 23-2. V=R Region Allocation (IEAVEQR) (Part 4 of 4)

Extended Description	Module	Label
6 Control is passed to GETPART, with a return code to indicate the action taken.		
7 IEAVEQRI scans the V=R Wait queue looking for a root PCB whose region range includes the frame. If none is found an internal error ABEND is generated to record the condition. The unwanted frame is returned to the caller by leaving its RBN in the input parameter field. If the frame is part of a waiting region, it is left in its dequeued state, the count in the root is decreased by one, and the input RBN set to zero to indicate acceptance of the frame. Next the PFTE is checked to determine if the frame is also intercepted for offline or storage error processing. If so, a POST code of 16 is set up. Otherwise, the frame count in the PCBR is decreased by one and then the value is checked for zero.	IEAVEQRI	
8 If all frames are not immediately obtained, V=R Allocation schedules IEAVEQRP into the first address space that has pages in the V=R area. When that routine has issued page-outs for all addressable pages, it searches the PFTEs for another ASID to be cleared. If another ASID's pages are in the V=R area, IEAVEQRP schedules itself into that address space. If no other address spaces have V=R area frames, IEAVEQRP frees the SRB.		
9 If all frames are available, the region can be validated and the request completed. Validation consists of placing the real storage addresses into the proper PGTEs, turning off the invalid flag, turning on the GETMAIN flag, clearing the region to zeros, and setting all keys to 0. A return code of 0 is passed to the caller when this is completed.		
10 When the frame count in the root PCB becomes zero, a POST code of 0 is indicated. (GETCELL was called early in V=R allocation for an SRB area in order to schedule a POST in the caller's address space.) The completion routine is then scheduled with a POST code of 0 or 16. In all cases, Intercept returns to its caller.		
11 Get the local and SALLOC locks. Then follow the same procedure as in steps 4-6.		
12 The caller's ECB is posted with code 0 or 16. The root PCB and the SRB are freed before exiting; also the PFTFPCB bit is set.		

Diagram 23-3. Freeing a V=R Region (IEAVEQR) (Part 1 of 2)

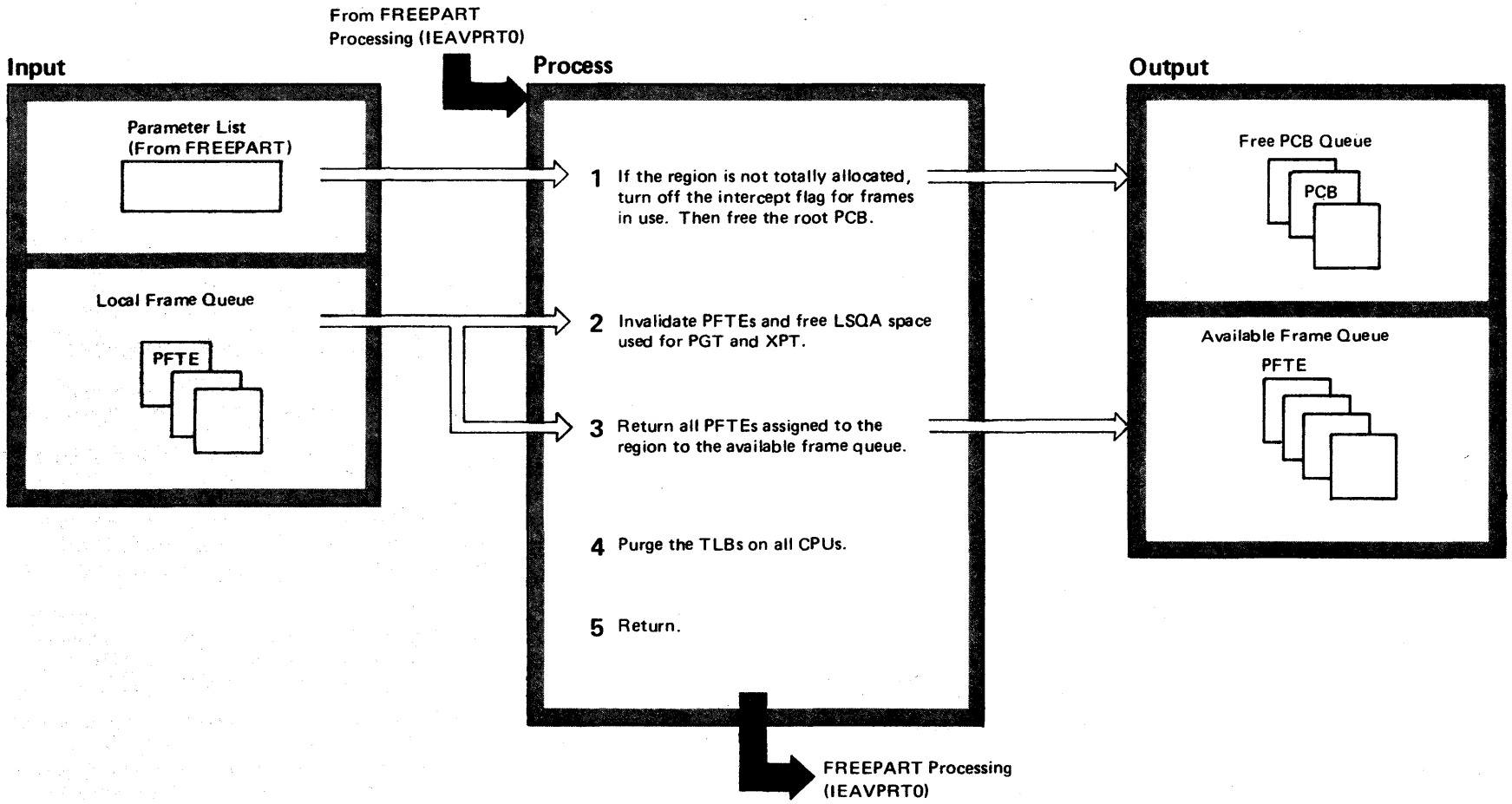


Diagram 23-3. Freeing a V=R Region (IEAVEQR) (Part 2 of 2)

Extended Description	Module	Label
The Free V=R Region routine (IEAVEQR) returns V=R allocated frames to the available queue for reuse by the system.		
1 If the PCBRINT flag in the page control block is set to one, the completion routine was scheduled to post the ECB; therefore, the completion routine needs the PCBR. In that case, the PCBR is not dequeued. Free sets PCBRPB so that when the completion routine does get control, it will check the bit and dequeue the PCBR. On the other hand if the PCBRFPCB bit is on in the Free routine, the completion routine has already run. Complete sets the bit to notify Free to free the PCBR in addition to freeing the V=R region.	IEAVEQR	IEAVEQRF
2 The PFTE for each frame that is part of the region is located and its V=R allocated flag turned off. Frames that are intercepted for V=R have the intercept flag turned off as well.		
3 Any frames already on the Local Frame Queue plus any unqueued frames are returned to the AFQ. Page tables that contain only V=R region pages are disconnected and freed. For each frame that was found on the local frame queue, the system fix counters (RSMCNTFX and PVTCNTFX) are decremented by one.		
4 Page invalidation is called to purge the TLBs on all CPUs in the system.		
5 Control is returned to FREEPART.		

Diagram 23-4. Page Release Processing (IEAVRELS) (Part 1 of 2)

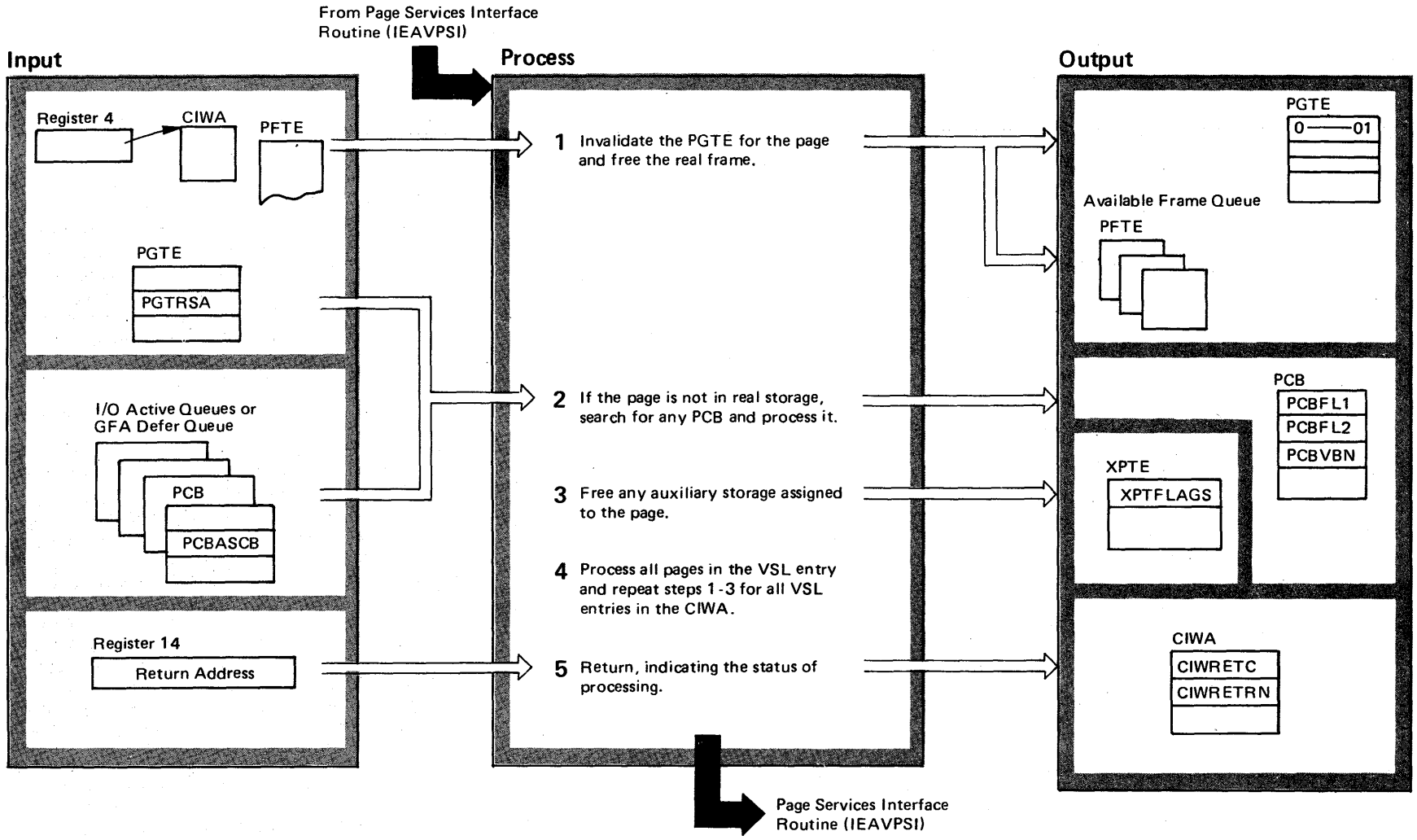


Diagram 23-4. Page Release Processing (IEAVRELS) (Part 2 of 2)

Extended Description	Module	Label
The PGRLSE routine (IEAVRELS) gets control from PSI to free one or more pages from real and auxiliary storage.		
1 PGRLSE performs initial checking on the VSL entry and the addresses contained in the VSL entry. If an invalid address is found, PGRLSE passes, in the CIWA, a return code of 4 to PSI.	IEAVRELS	IEAVRELS
PGRLSE obtains the PGTE and XPTE addresses from the Find Page routine. If the page is in real storage, PGRLSE checks the PFTE. If the frame is V=R allocated, fixed, or located in SQA or LSQA, PGRLSE terminates processing the page. Otherwise, PGRLSE calls Page Invalidation to invalidate the PGTE and then calls PFTE Enqueue/Dequeue to free the PFTE.		
2 If the page is not in real storage, PGRLSE first checks to see if the page is assigned by GETMAIN; if not, PGRLSE puts a return code of 4 in the CIWA and returns control. Then PGRLSE searches the following queues for PCBs associated with the virtual page being processed: GFA Defer Queue, Common I/O Active Queue, and Local I/O Active Queues. If any PCBs with non-zero fix counts are found, PGRLSE terminates processing of the page, leaving the PCBs on their queue.		
If PGRLSE finds a PCB on the GFA Defer Queue, it purges the PCB. If the PCB is for a page fault and is in SRB mode, PGRLSE calls the Reset routine of the PCIH to reactivate the suspended SRB. PGRLSE puts other page fault PCBs on the I/O active queue and requests I/O completion processing. If the PCB has a root PCB, the PCB count in the root PCB is decreased by one and the PCB is scheduled for I/O completion processing. For a PCB not for a page fault and without a root PCB, PGRLSE returns the PCB to the free queue.		
If PGRLSE finds PCBs on the I/O active queues, it purges them by setting to zero the virtual block number and by setting the free-real-storage flag to one.		
3 If the save auxiliary storage flag is not set to one in the XPTE, PGRLSE calls ASM to free the auxiliary slot assigned to the virtual page. PGRLSE sets the auxiliary storage assigned flag in the XPTE to zero. Then it returns control to PSI.		
4 PGRLSE processes all pages in the VSL entry and, if no errors occur, continues with the next VSL entry until complete.		
5 PGRLSE returns control to PSI, putting a return code in the CIWA.		

Diagram 23-5. FREEMAIN Release Processing (IEAVRELS) (Part 1 of 2)

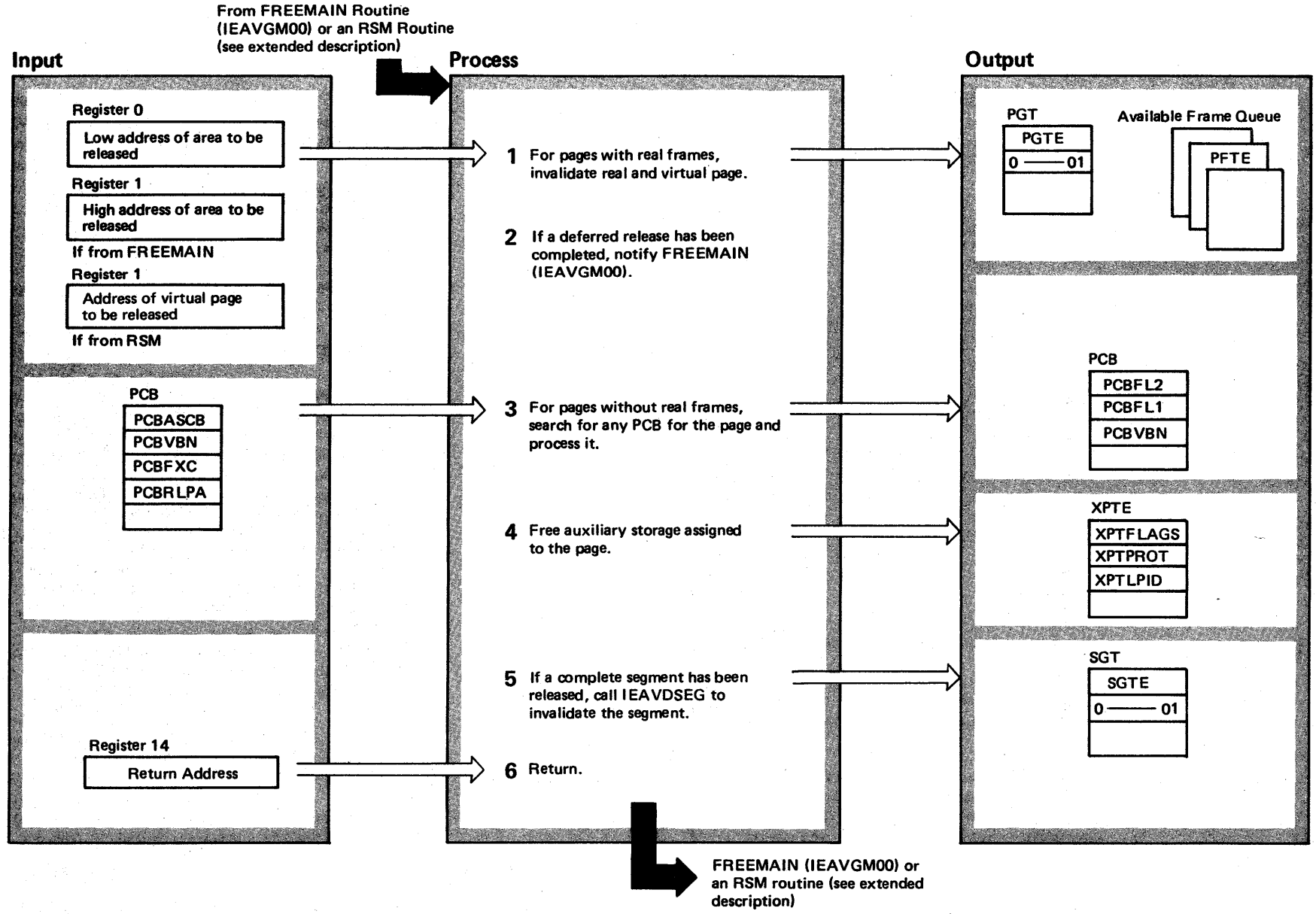


Diagram 23-5. FREEMAIN Release Processing (IEAVRELS) (Part 2 of 2)

Extended Description	Module	Label
Deferred and FREEMAIN Page Release Processing (a part of IEAVRELS) performs PGRLSE functions for two special cases: when an RSM routine frees a frame marked for deferred release, and when FREEMAIN frees a page. The RSM routines are: IEAVSOUT, IEAVTERM, IEAVFREE, IEAVIOCP, and IEAVPIOP.		
1 When entering at the IEAVRELV entry point, the caller holds the SALLOC lock and the local lock. When entering at the IEAVRELF entry point, the caller holds the SALLOC lock. Page Release uses the Find Page routine to get the PGTE and the XPTE addresses. If the page has a frame assigned, Page Release checks the PFTE. If the page is an SQA page or an LSQA page with a VBN matching the input VBN, Page Release moves the PFT to the available frame queue and then, using the Page Invalidation routine, invalidates the PGTE. The system fix counters (SQACNTFX, RSMCNTFX, and PVTCNTFX) are also decremented.	IEAVRELS	IEAVRELV IEAVRELF
2 If the deferred release flag is set in the PFTE and if the fix count is zero, Page Release notifies FREEMAIN that the virtual page may be used again.		
3 If the virtual page does not have a frame in real storage, Page Release checks for a PFTE with the deferred release flag set. If it finds one, it resets the flag and notifies FREEMAIN that the virtual page can be used again. If it does not find one, Page Release searches for PCBs for the virtual address and processes them according to the queue they are associated with. When all such PCBs are processed, Page Release sets the PGTE to zero.		
4 Page Release tests the XPTE. If the auxiliary-storage-assigned flag is set and the save-auxiliary-storage flag is not, Page Release calls ASM to free the auxiliary storage slot assigned to the logical page ID (LPID). Then Page Release resets the LPID generator value in the XPTE to zero and sets to zero all flags in the XPTE.		
5 If all PGT entries for a private area segment containing the input virtual address are set to zero, Page Release calls the Destroy Segment routine to invalidate the PGTEs and XPTEs and to prepare the table storage for FREEMAIN processing. Then Page Release frees the page table space.		
6 Page Release returns control to FREEMAIN or to the RSM routine that called it.		

Diagram 23-6. Create Segment (IEAVCSEG) (Part 1 of 2)

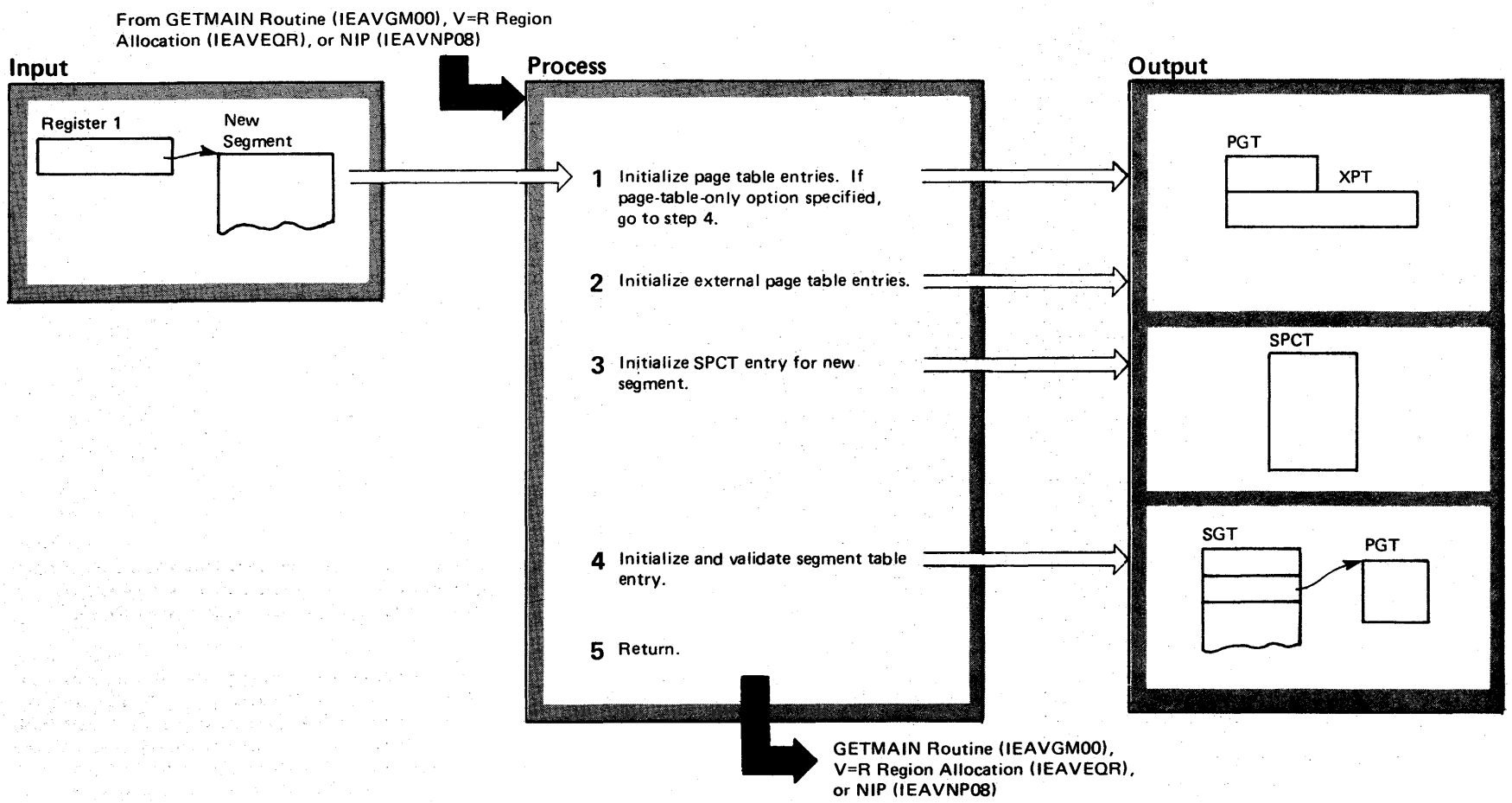


Diagram 23-6. Create Segment (IEAVCSEG) (Part 2 of 2)

Extended Description	Module	Label
The Create Segment routine (IEACSEG) is called by VSM functions, NIP, and the V=R Allocation routine to initialize the page tables for one or more newly created segments. The local lock must be held by the caller. V=R Allocation uses the IEAVCSGB entry point to avoid setting a pointer to the RSM FRR.		
1 When entered at IEAVCSEG, Create Segment sets a pointer to the RSM functional recovery routine IEAVRCV. It validity checks parameters passed. Then Create Segment initializes the page table for the new segment. It does this by setting 32 bytes of storage to zero, setting the real block number fields in each page table entry to zero, and setting the page table GETMAIN flag to zero and the invalid flag to one.	IEAVCSEG	IEAVCSEG IEAVCSGB
2 If the bypass XPT option was not selected, Create Segment establishes an external page table in the next 192 bytes of storage by setting each external page table entry to zeroes.		
3 If the XPT is created and the SPCT address is not zero, Create Segment sets the segment index, the virtual address of the PGT, and the segment entry count in the SPCT. If necessary, Create Segment enlarges the SPCT (under the SALLOC lock) by obtaining storage for the SPCT, under GETMAIN, moving the SPCT, updating the size and entry count fields in the SPCT, and freeing the old SPCT with FREEMAIN.		
4 Create Segment initializes the segment table entry for the new segment by setting the invalid flag to zero and inserting the page table length and the real storage address of the page table.		
5 Create Segment repeats the procedure for additional segments, and then returns to the caller.		

Diagram 23-7. Destroy Segment (IEAVDSEG) (Part 1 of 2)

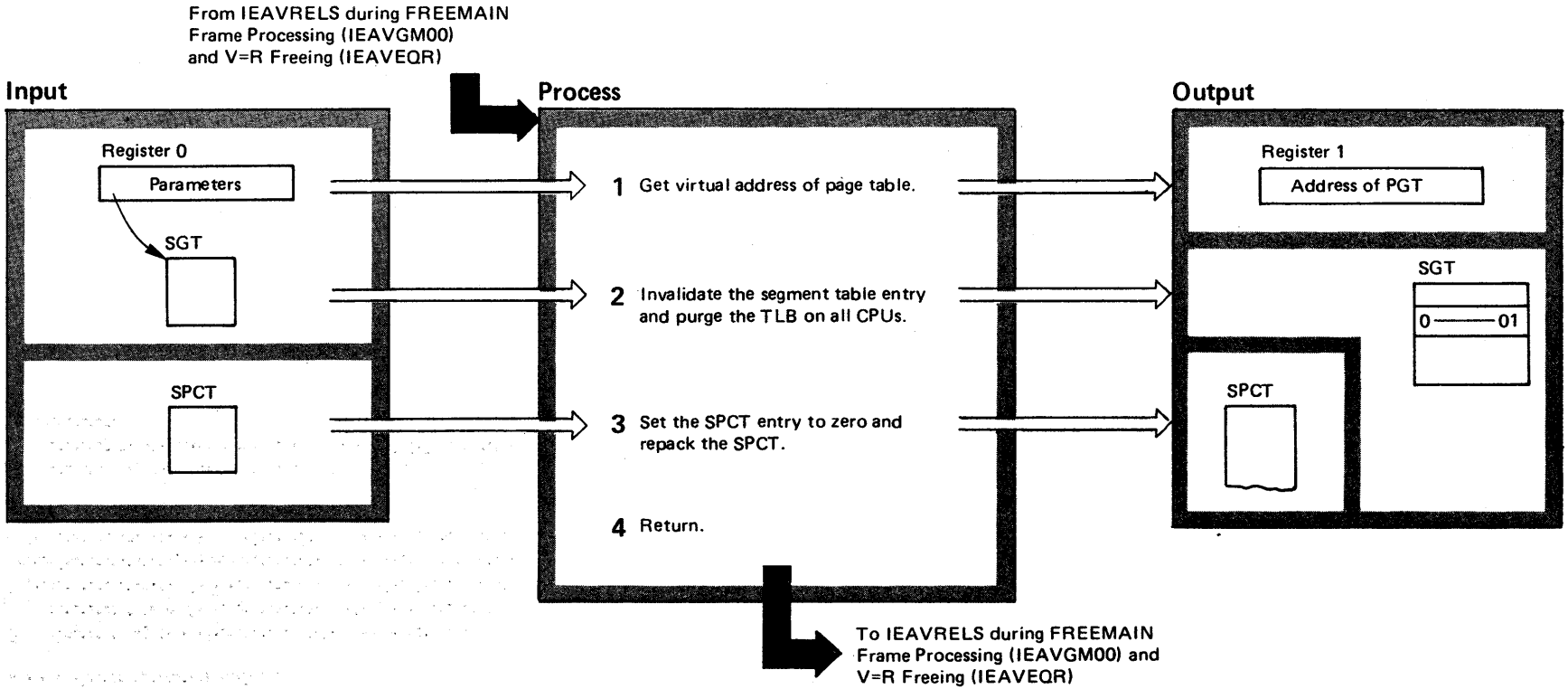


Diagram 23-7. Destroy Segment (IEAVDSEG) (Part 2 of 2)

Extended Description	Module	Label
The Destroy Segment routine (IEAVDSEG) invalidates control block entries for a virtual segment that is being deleted; it returns the address of the page table space to be freed with FREEMAIN by the caller. The caller must hold the SALLOC and local locks. Destroy Segment is called by V=R Allocation and by PGRLSE.		
1 Destroy Segment gets the virtual address of the page table by translating the real address obtained from the segment table entry.	IEAVDSEG	IEAVDSEG
2 Destroy Segment invalidates the segment table entry by setting the entry to zero and then setting the invalid flag to one. It then calls the Page Invalidation routine IEAVINV, passing a dummy PGTE address to invalidate the translation lookaside buffers.		
3 Destroy Segment checks the RSM Header for an SPCT address; if zero, Destroy Segment returns to the caller. If an address is given, Destroy Segment sets the SPCT entry matching the destroyed segment to zero, decreases the SPCT entry count, and repacks the last SPCT entry into the entry just set to zero.		
4 Destroy Segment returns to the caller with the virtual address of the page table, to be freed with FREEMAIN by the caller.		

Diagram 23-8. Program Check Interrupt Extension (IEAVPIX) (Part 1 of 2)

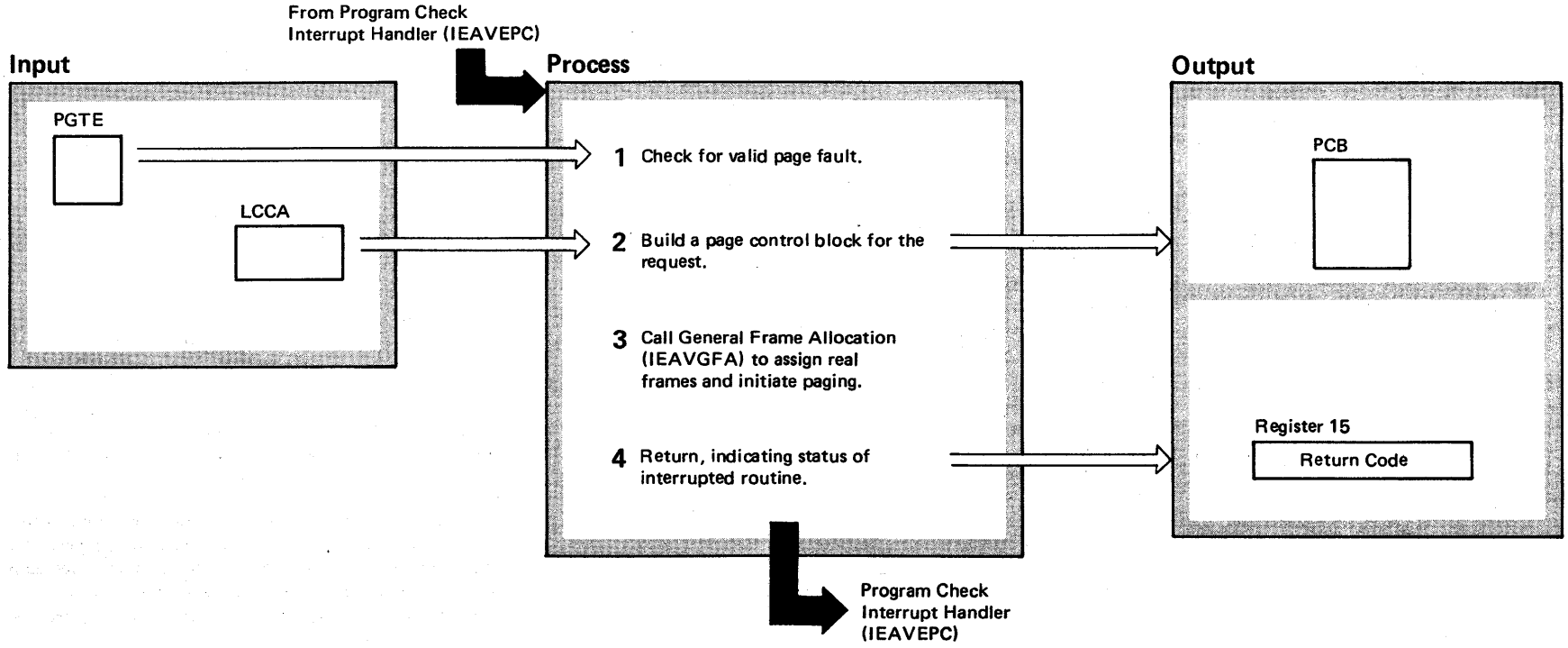


Diagram 23-8. Program Check Interrupt Extension (IEAVPIX) (Part 2 of 2)

Extended Description	Module	Label
The Program Check Interrupt Extension (IEAVPIX) services all page translation interrupts. It gets control from the PCIH on all page faults except those incurred by a routine holding a global lock, which should not have a page fault.		
1 PIX first acquires the storage allocation global lock (SALLOC) and sets up the RSM FRR. The page fault is checked for validity by checking the GETMAIN-assigned flag in the page table entry that corresponds to the virtual address for which the interruption occurred. If the flag is off or if no page tables exist for the virtual address, PIX returns to PCIH with a return code indicating the interrupt should be treated as a logical protection exception (0C4 ABEND). During this check, internal errors may be detected if the segment or page table is not correctly built or initialized, triggering special recovery processing. Return code 12 is given to PCIH to indicate a RSM error prevented page fault resolution. The page may also have been marked as valid in real storage because another CPU validated the page after the page fault occurred. For this case, no further processing would be required for the page fault and return code 4 is given.	IEAVPIX	IEAVPIX
2 A PCB is built and initialized for use by other RSM functions that must be employed to satisfy the page fault. General Frame Allocation (GFA) is then called with the PCB address passed as a parameter.		
3 General Frame Allocation attempts to assign a real frame to the virtual page. Upon completion of its function, it returns to PIX indicating the action taken. PIX interprets these return codes and issues the proper return code to the PCIH. The code indicates that either the interrupted routine may continue execution (no paging I/O was necessary to satisfy the page fault), or the interrupted routine's execution was suspended until paging I/O can be completed to satisfy the page fault.		

Diagram 23-9. General Frame Allocation (IEAVGFA) (Part 1 of 4)

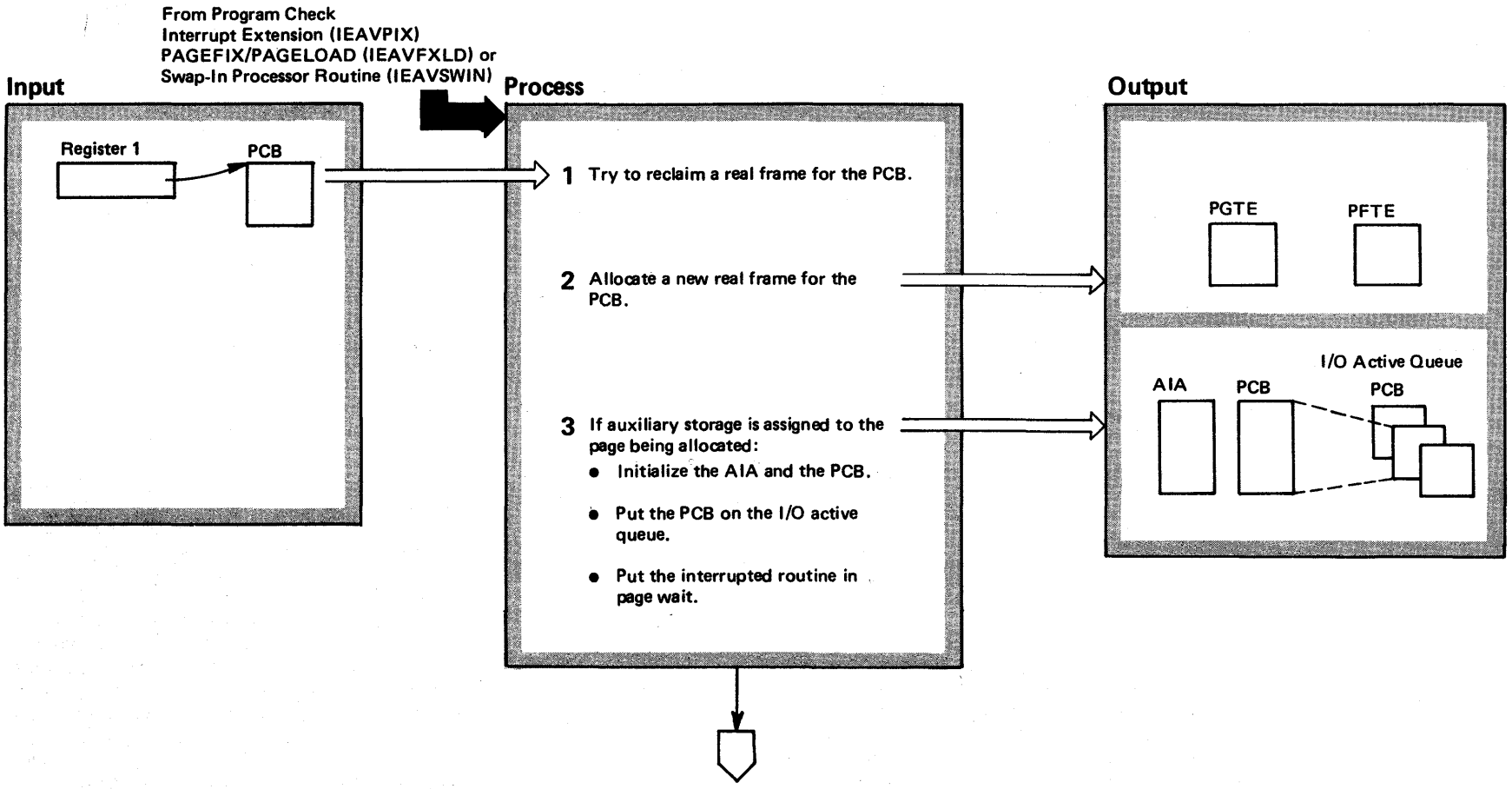


Diagram 23-9. General Frame Allocation (IEAVGFA) (Part 2 of 4)

Extended Description	Module	Label

General Frame Allocation (IEAVGFA) is called by Program Interrupt Extension, Page Fix, Page Load, and Swap-In to assign real frames to virtual pages. Each virtual page requiring a real frame is represented by a PCB (page control block).

1 If the PCB is not for the swap-in of a private area (LSQA or PGFIX) page, reclamation is attempted. — IEAVGFA

The last real storage frame the page occupied is located by looking at the real address field of the PGTE. The Page Frame Table Entry (PFTE) for the frame is examined to see if it still contains the page. If it does, the frame is used to satisfy the current request except as noted in the following paragraph. If input I/O is in progress for the frame, the current request is related to the existing request and the current requestor is suspended if he is satisfying a page fault. If output is in process, general frame allocation determines if the output operation has been marked non-reclaimable; that is, the PCB represents the output for a V=R or Vary Storage Intercepted frame. If it has been marked non-reclaimable, reclaim fails and the copy of the output page is duplicated in a new storage frame. Otherwise, the frame and the page are made immediately available by validating the PGTE and setting storage keys.

If the request is for a "long fix" of a page and reclamation would place the page in the V=R area or outside the preferred area for a non-swappable "long-fix" page, reclamation is suppressed. If necessary, the existing copy of the page is duplicated outside the V=R area or inside the preferred area and the frame is freed. Naturally, this can be done only if the page is not already fixed.

If reclamation is successful, the next input PCB is processed. Also, if the input PCB represents a fix request and the fix count in the PFTE is currently zero, the system fix counters are updated (incremented by one).

2 If reclamation fails or if it is not attempted, IEAVGFA tries to allocate a new real frame from the Available Frame Queue (AFQ).

If no special requirements exist, the first frame on the AFQ is assigned to the request. If the PCB represents a LSQA or PGFIX request, the system fix counters are incremented by one. Fix data is transferred from the PCB to the PFTE and allocation is complete. If no frames are available, the input PCB is marked 'defer' to indicate allocation failed.

Extended Description	Module	Label

There are special requirements associated with "long fix" and Stage I swap-ins. V=R area frames are never used for long-fix pages or Stage I pages of address spaces that may become non-swappable. Whenever possible, these types of pages are assigned to frames in a NIP-designated 'preferred area' so that they will be out of the way of most requests to vary storage.

However, if no 'preferred' frames are available, an attempt is made to steal a preferred area frame from some virtual page. Only unchanged, non-fixed frames for which no PCB exists are candidates for this stealing. Frames that are excluded are fixed, allocated to an active V=R region, offline, changed, have paging I/O in progress, or contain a storage error. The local and common frame queues are searched (in that order) for a frame meeting the steal criteria. The search stops as soon as a stealable frame in the preferred area is found. If no preferred area frame can be obtained, a non-preferred, non-V=R frame is used if available. If one such frame is found on the AFQ, the physical storage unit containing the frame is converted from non-preferred to preferred storage.

Stage I pages of swappable address spaces are treated similarly except that they can be placed in V=R area frames if no other frames are available. As in the simple case, if the page cannot be allocated, it is marked 'defer'.

If none are available or meet the allocation criteria, the input request is deferred by placing the PCB on the GFA Defer Queue and then continuing with the next input PCB.

If allocation is successful, it is determined if any other requests for the same page are presently deferred. If there are any, they are removed from the Defer queue and attached to the current request via the PCB relating mechanism, so they will be satisfied as well.

3 If no auxiliary storage copy of the page exists, an empty page is created by validating the page table entry (PGTE) for the page, setting the storage to zeroes, and setting the storage keys to the value specified in the external page table. If the PCB indicates a need for any follow-up processing, it is performed immediately where possible and scheduled for asynchronous processing if not. If no asynchronous processing is needed, the PCB is freed. In either case, the next input PCB is then begun.

Diagram 23-9. General Frame Allocation (IEAVGFA) (Part 3 of 4)

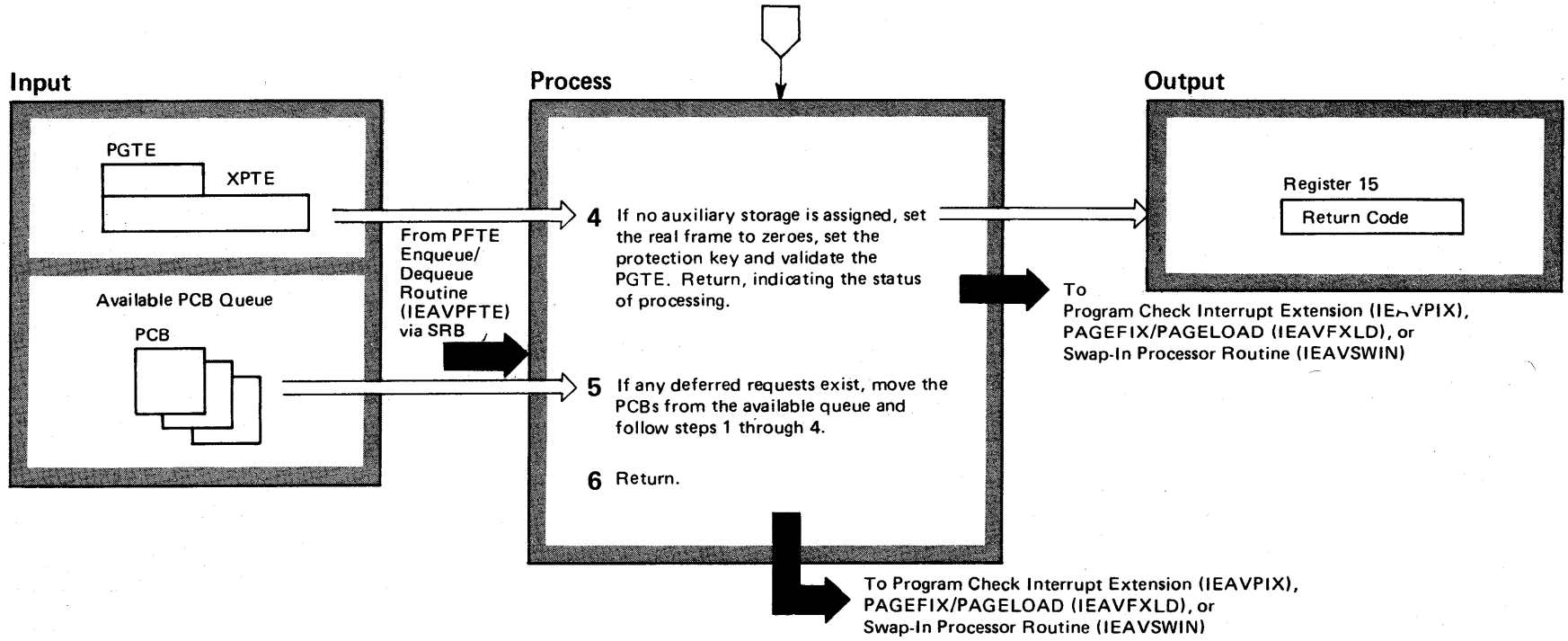


Diagram 23-9. General Frame Allocation (IEAVGFA) (Part 4 of 4)

Extended Description | **Module** | **Label**

4 If an auxiliary storage copy of the page does exist, the PCB is queued for page-in I/O and its AIA (ASM I/O Request Area) is placed on an internal I/O request queue. If the request involves a page fault, the execution of the faulting RB or SRB is stopped by a call to the Suspend routine of PCIH. A PCB flag is set to indicate that reset is required when the request is eventually satisfied. The next input PCB is then begun. When all PCBs are processed, IEAVGFA returns control to the caller.

5 When all input PCBs have been processed as above, the chain of ASM I/O request areas (AIAs) is passed to the Auxiliary Storage Manager (ASM) for satisfaction. If any input request was deferred, related to other I/O, or sent to ASM, a return code of 4 is given; otherwise the return code is 0.

6 IEAVGFA removes PCBs from the Defer Queue one at a time and passes them to the main portion of IEAVGFA for allocation processing. When all PCBs on the Defer Queue for this address space have been processed. IEAVGFA returns control to the caller.

Diagram 23-10. Page I/O Post (IEAVPIOP) (Part 1 of 2)

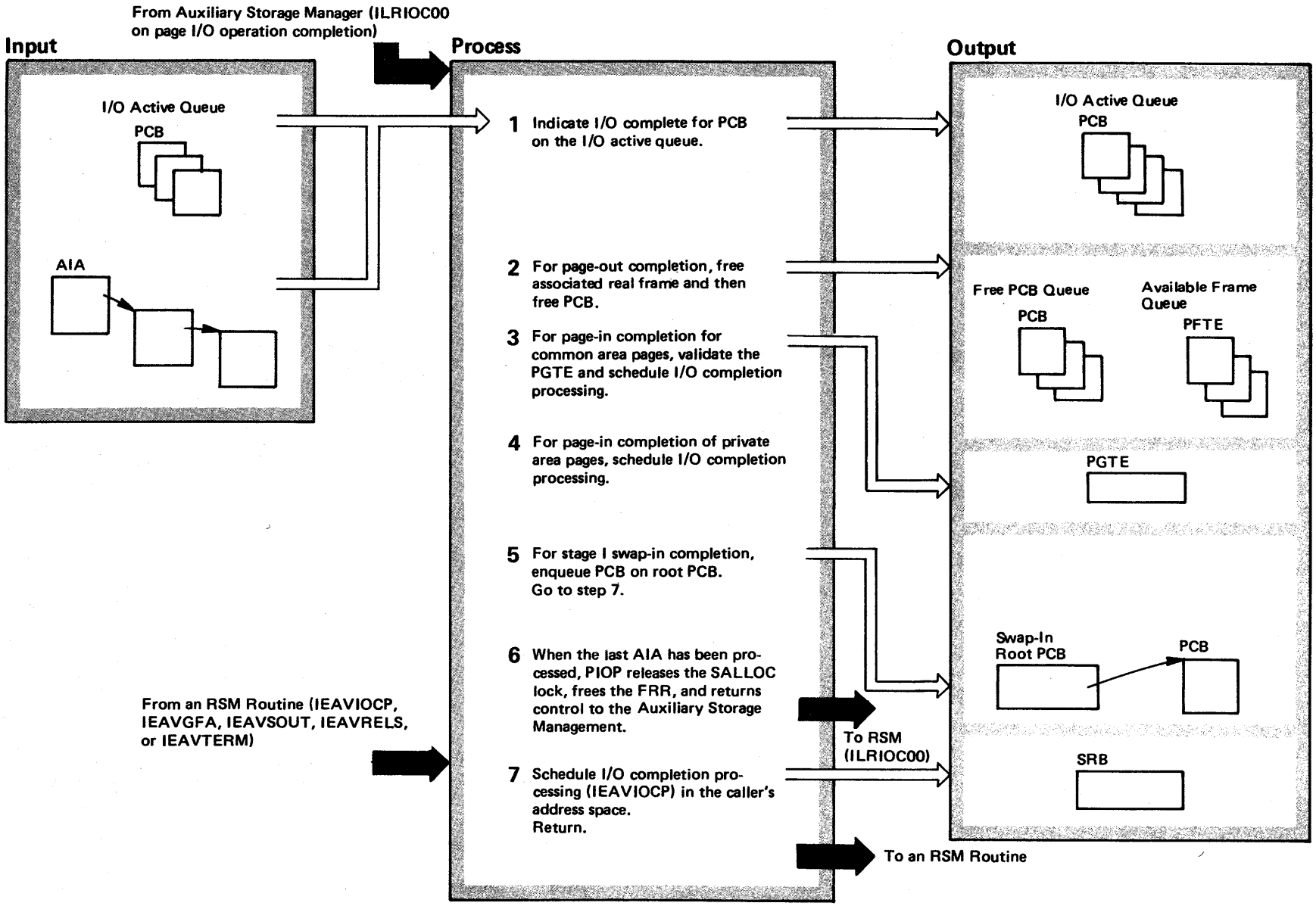


Diagram 23-10. Page I/O Post (IEAVPIOP) (Part 2 of 2)

Extended Description	Module	Label
Page I/O Post (IEAVPIOP) notifies waiting routines that an I/O operation has completed.		
1 After establishing FRR linkage and setting up the recovery communication area (RCA), PIOP gets the SALLOC lock. Then it checks the I/O completion information in the AIA that is passed as input. If an error is found, PIOP issues an abnormal termination request with a code of X'C0D'. PIOP moves the completion information from the AIA to the PCB and to all related PCBs. If an I/O error has occurred, PIOP sets I/O-completed and I/O error flags in each PCB.	IEAVPIOP	IEAVPIOP
2 PIOP processes the input depending upon whether the operation is a page-out, a normal page-in, or a stage I swap-in. For page-out PCBs, PIOP frees the PCB and the real frame unless an I/O error occurred. If an I/O error occurred, PIOP changes the page-out PCB to a page-in PCB and schedules the I/O completion processor to revalidate the page		
3 For common area page-in PCBs, PIOP schedules the I/O related PCBs by removing them from the related chain and either freeing them or, if IEAVRSET or a root exit routine is to be called, putting them on a common I/O active queue. PIOP determines from the free-real-storage flag whether to free or save the real so that any zero TCB fix PCBs are the last to be processed by the I/O completion processor. Then, if necessary, PIOP schedules I/O completion processing (IEAVIOCP).		
4 For private area page-in PCBs PIOP schedules the I/O completion processor to validate the PGTE and/or to call IEAVRSET or a root exit routine. The input PCB and any PCBs related to it remain unchanged. If an I/O error occurred, the I/O completion processor will not validate the PGTE and will call IEAVRSET with an error completion code.		
5 If the PCB in a stage I swap-in for a private and page and no I/O error occurred, PIOP removes the PCB from the I/O active queue and enqueues it to the swap-in root PCB. If an I/O error occurred, the PCB and the frame are freed and the root PCB fail flag is set. If the PCB is for the common area and no I/O error occurred, PIOP validates the PGTE. If an I/O error occurred, PIOP rearranges any related PCBs so that any zero TCB fix PCBs are the last to be processed by the I/O completion processor. If necessary, the I/O completion processor is scheduled. Whenever PIOP finds a PCB for a swap-in it decreases the root PCB count and, when that count goes to zero, calls the root exit routine. Whenever PIOP finds a PCB for a PGFIX or PGLOAD with an ECB, it decreases the root PCB count except when that count goes to zero. When that count goes to zero, PIOP makes sure that the I/O completion processor is scheduled to decrease the count and call the root exit routine.		
6 When the last AIA has been processed, PIOP releases the SALLOC lock, frees the FRR, and returns control to the Auxiliary Storage Manager.		
7 In a special scheduling routine of PIOP, called IEAVOPBR, PIOP examines the PCB and any related PCBs to determine how to schedule IEAVIOCP. If IEAVIOCP has not already been scheduled, PIOP gets an SRB, initializes it, and schedules it. Return to caller.	IEAVPIOP	IEAVOPBR

Diagram 23-11. Page I/O Completion Processing (IEAVIOCP) (Part 1 of 2)

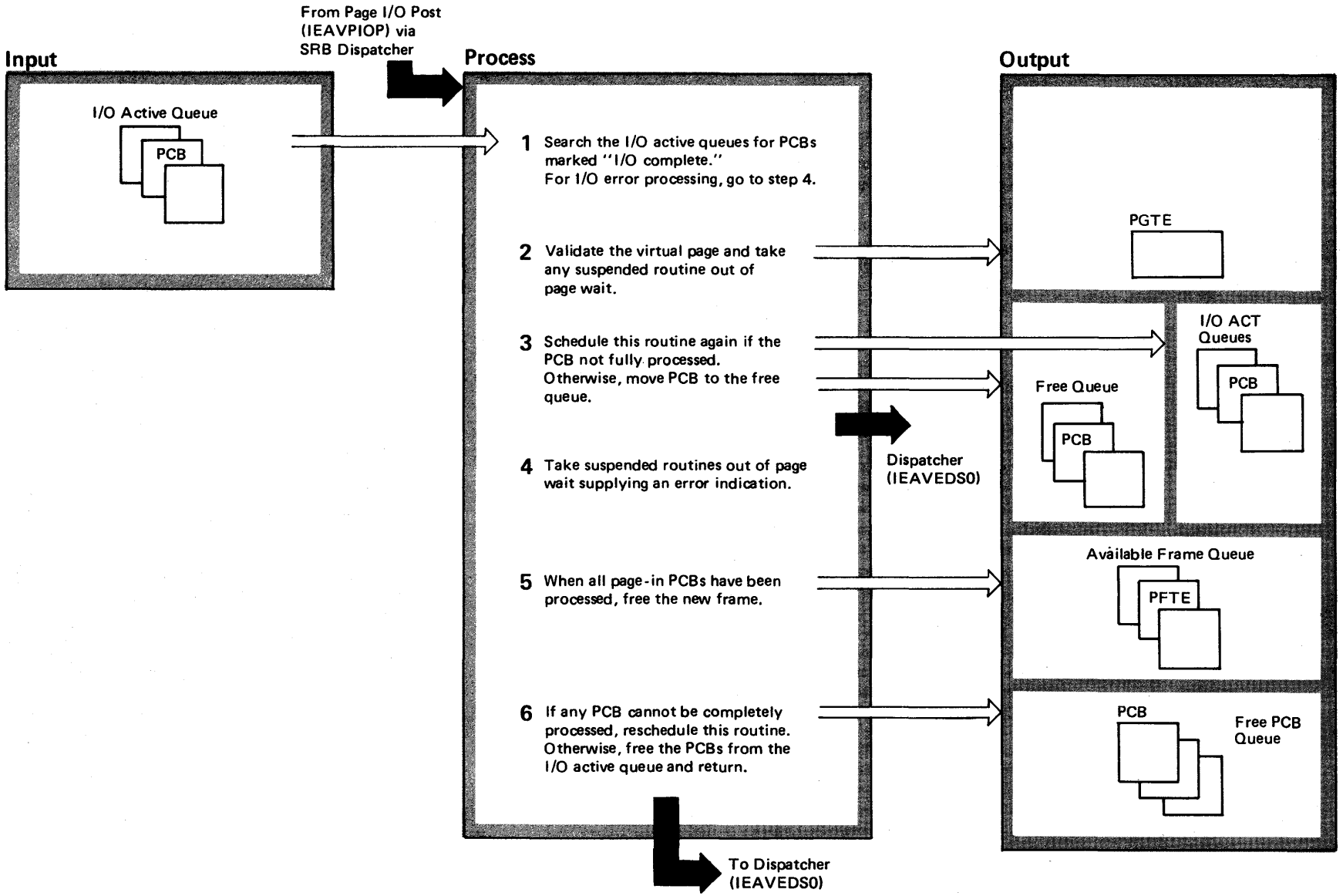


Diagram 23-11. Page I/O Completion Processing (IEAVIOCP) (Part 2 of 2)

Extended Description	Module	Label
The Page I/O Completion Processor (IEAVIOCP) processes all page-in I/O completion events.		
1 PIOCP establishes the FRR and gets the SALLOC lock. If requested, it also gets the local lock. Then PIOCP searches the local I/O active queue for the current address space, and searches the common I/O active queue for PCBs that have I/O-complete flags set.	IEAVIOCP	IEAVIOCP
2 PIOCP checks the PCBs found. If a stage 2 swap-in PCB is found that has not been intercepted, PIOCP sets the RBN in the PGTE. PIOCP processes all related PCBs by checking for one of three conditions: • The free-real storage flag is set. • The VBN is zero. • The virtual page represented by the PCB is already valid. If none of the conditions occur, PIOCP validates the page by setting the protection keys in the XPTE and setting to zero the page-invalid flag in the PGTE. Then PIOCP notifies the routine in the PCB that the I/O operation is complete. If the reset flag is on, PIOCP calls the reset routine of PCIH to release the routine that page-faulted. If the PCB has a root PCB, the PCB count in the root PCB is decreased by one. If the count becomes zero, PIOCP calls the root exit routine.		
3 As each PCB is processed, PIOCP frees it or enqueues it to an I/O active queue and calls a subroutine of IEAVPIOP to reschedule IEAVIOCP to complete processing. Finally, PIOCP frees the input PCB unless it is to be kept. Then PIOCP returns control to the dispatcher.		
4 For I/O errors, PIOCP notifies the routine specified in the PCB that the operation completed with an error. PIOCP also performs processing for swap-in and fix PCBs.		
5 When each PCB has been processed, PIOCP frees it or leaves it enqueued. When all PCBs are processed — and if all PCBs for the real frame have been freed — PIOCP frees the real frame. Note that during I/O error processing, when PFTFXCT is decremented to zero, the system fix counters are also decremented.		
6 If any PCBs cannot be freed, PIOCP calls IEAVPIOP to schedule IEAVIOCP again. Then PIOCP returns control to the dispatcher.		

Diagram 23-12. Page Services Interface (IEAVPSI) (Part 1 of 2)

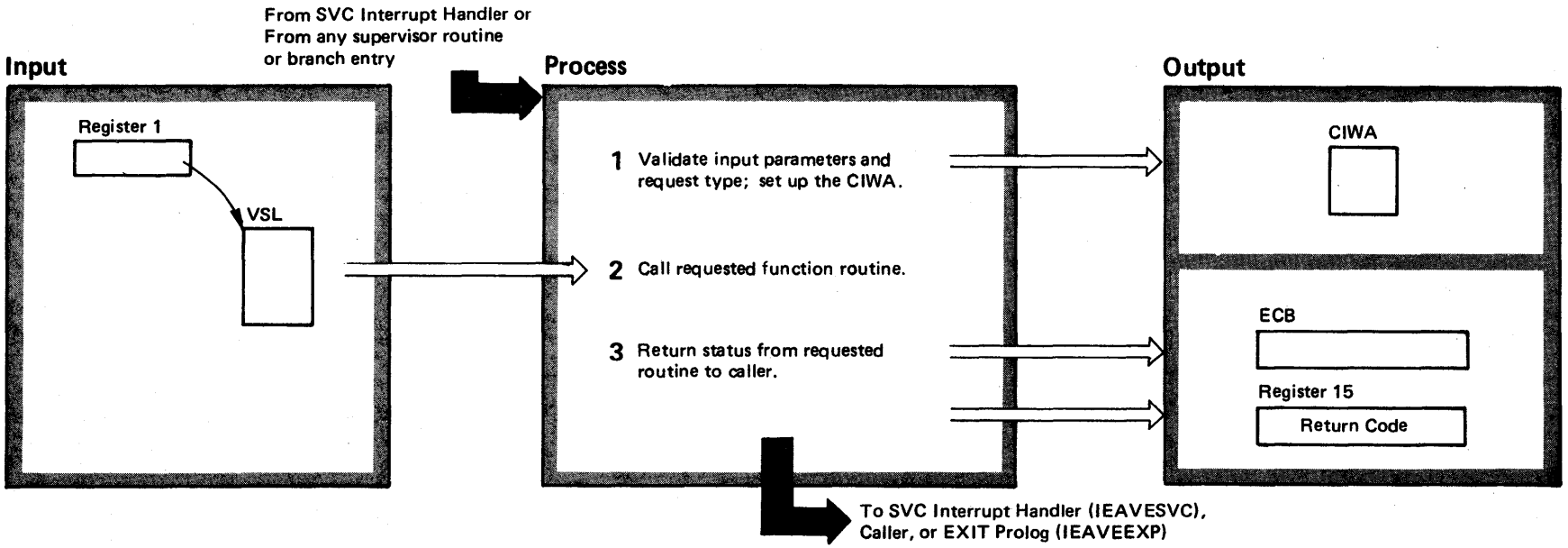


Diagram 23-12. Page Services Interface (IEAVPSI) (Part 2 of 2)

Extended Description	Module	Label
The Page Services Interface routine (IEAVPSI) processes all input requests for page service functions (PGFIX/PGLOAD, PGFREE, PGRLSE). Input is placed in a common internal work area (CIWA). All exit processing is done in the module also.		
1 When entered via an SVC 112, PSI gets the SALLOC lock and checks the requestor's authorization. If the requestor is not authorized, PSI returns a code of 4 in register 15. Otherwise, it sets up the virtual subarea list (VSL) entry in the CIWA and calls IEAVRELS. When entered via an SVC 113, PSI verifies that the ECB passed as input is in storage, gets the SALLOC lock, and checks the caller's authorization. PSI returns a code of 16 in register 15 if the caller is not authorized and the request is PGFIX or PGFREE or if the Real Address option is specified on a PGFIX or PGFREE request or for any other parameter error. If this validity check is successful then PSI sets up the CIWA with data from the VSL. When entered via a branch entry from a non-RSM routine for page services, PSI validates any ECB input and gets the SALLOC lock. If the Real Address option is specified for PGFIX or PGFREE, PSI returns a code of 16 in register 15 and issues an ABEND. If the validity check is successful, it sets up the CIWA and sets up and checks VSL entries.	IEAVPSI	IEAVPSI
When entered via a branch entry from an RSM routine, PSI checks any ECB input and constructs a VSL in the CIWA.		
2 PSI tests the operation and option bits in the CIWA for validity. Then it calls the page service functions requested by the operation flags in the CIWA. Any invalid bit combination results in PSI returning a code of 16 in register 15 and issuing an ABEND.		
3 When a return is made from the function routine, PSI examines the return code. If supplied, it posts the input ECB. If the return code is an error code, PSI requests abnormal termination with a completion code of x'171' and a reason code in register 15 for the requestor. If the caller was unauthorized, PSI sets an abnormal termination code of x'271'. If return code 8 is to be issued, indicating asynchronous completion of the request, PSI fixes the input ECB. Finally, PSI returns control to the caller or to the EXIT routine.		

Diagram 23-13. PGFIX/PGLOAD Processor (IEAVFXLD) (Part 1 of 2)

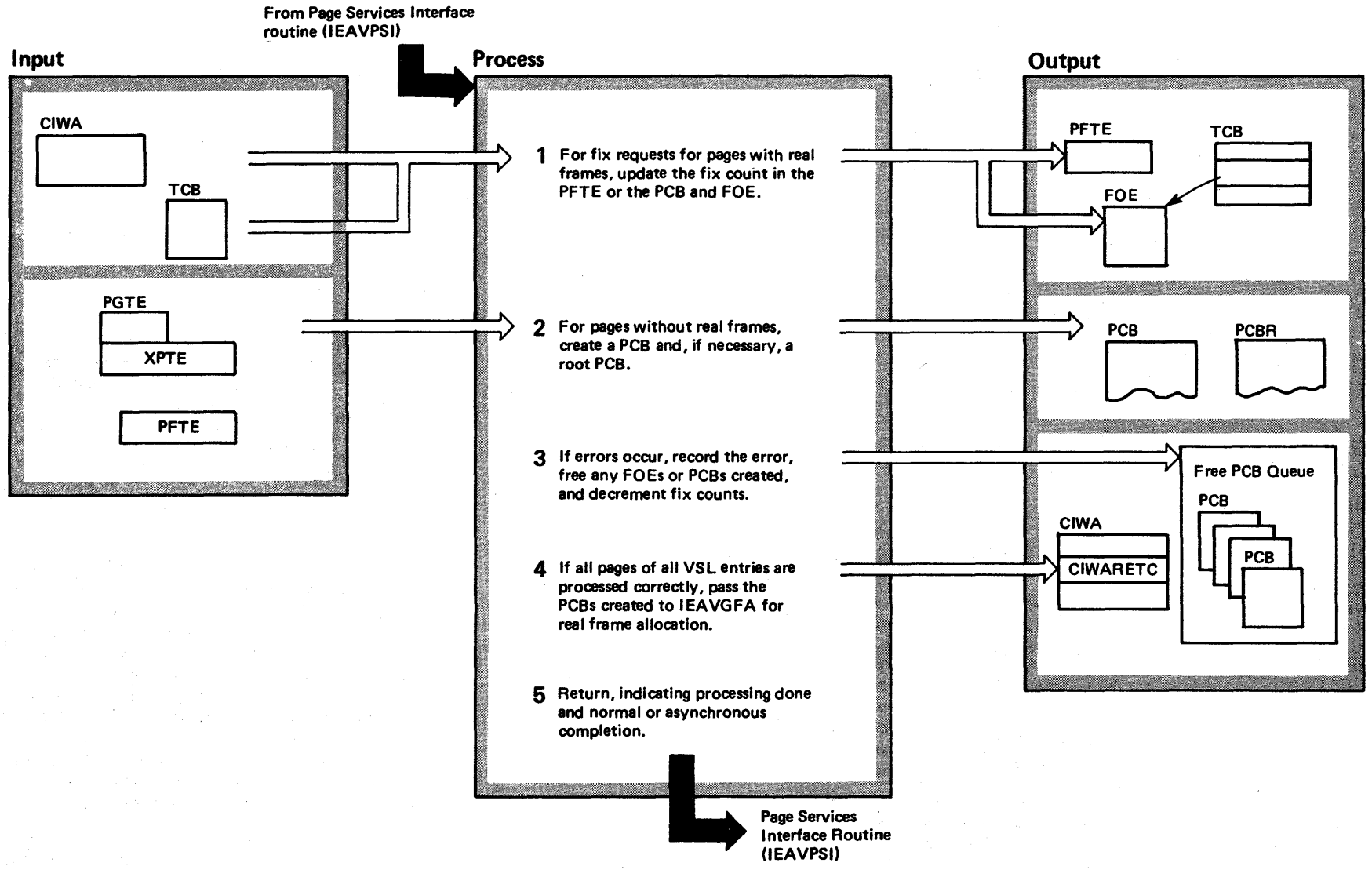


Diagram 23-13. PGFIX/PGLOAD Processor (IEAVFXLD) (Part 2 of 2)

Extended Description	Module	Label
The PGFIX/PGLOAD Processor routine (IEAVFXLD) handles requests for bringing virtual pages into real storage. The PGFIX processor also fixes the page in real storage.		
1 FXLD checks the pages to be sure they are GETMAIN-assigned and, for PGFIX, not VIO pages. Otherwise, FXLD returns a code of 4 to the exit processor. For pages with frames in real storage, FXLD fixes virtual pages not already fixed or re-fixed pages that are already fixed. It does this by increasing the fix count in the PFTE for the frame and by creating (or updating) a fix ownership element (FOE), which it enqueues to the fix ownership list (FOL) pointed to by the requestor's TCB. If the page is being requested for a long fix and is in a V=R area, FXLD creates a PCB and sets the long-fix flag to one, so the real frame can be moved out of the V=R area. If the page is not in a V=R area, FXLD sets the long fix flag in the PFTE to one. If the fix count in the PFTE is currently zero indicating that the frame is not already fixed, the system fix counters are updated (incremented by one).	IEAVFXLD	IEAVFXLD
2 When a virtual page is not in real storage, FXLD searches the internal PCB queue for a PCB for the page being processed. If a PCB is found, FXLD increases by one the fix count in the PCB and the FOE, if it is a PGFIX request. If a PCB is not found, FXLD creates one and initializes it. For a PGFIX request, FXLD also creates and initializes an FOE. If an ECB address is specified, FXLD checks for an existing root PCB. If none exists, FXLD creates one and initializes it. FXLD associates the regular PCB with the root PCB and increases the count of PCBs in the root PCB.		

Extended Description	Module	Label
3 If errors occur, FXLD puts each PCB it created and its associated root PCB on the available queue. For PGFIX requests, FXLD frees the FQE. If the return code in the CIWA is 4, it sets the error flag in the CIWA copy of the input virtual subarea list (VSL) entry; the CIWA copy of the VSL entry is copied over the user copy. Also, the CIWA return code is saved. FXLD calls IEAVFREE to free any virtual pages fixed before the error. Then FXLD returns control to IEAVPSI.		
4 If no errors have occurred, FXLD passes any PCBs created to IEAVGFA, which attempts to allocate real frames. If successful, IEAVGFA marks the PFTEs for the PGFIX requests.		
5 FXLD returns control to PSI, indicating processing is completed and specifying normal or asynchronous completion.		

Diagram 23-14. PGFIX/PGLOAD Root Exit (IEAVFXLD) (Part 1 of 2)

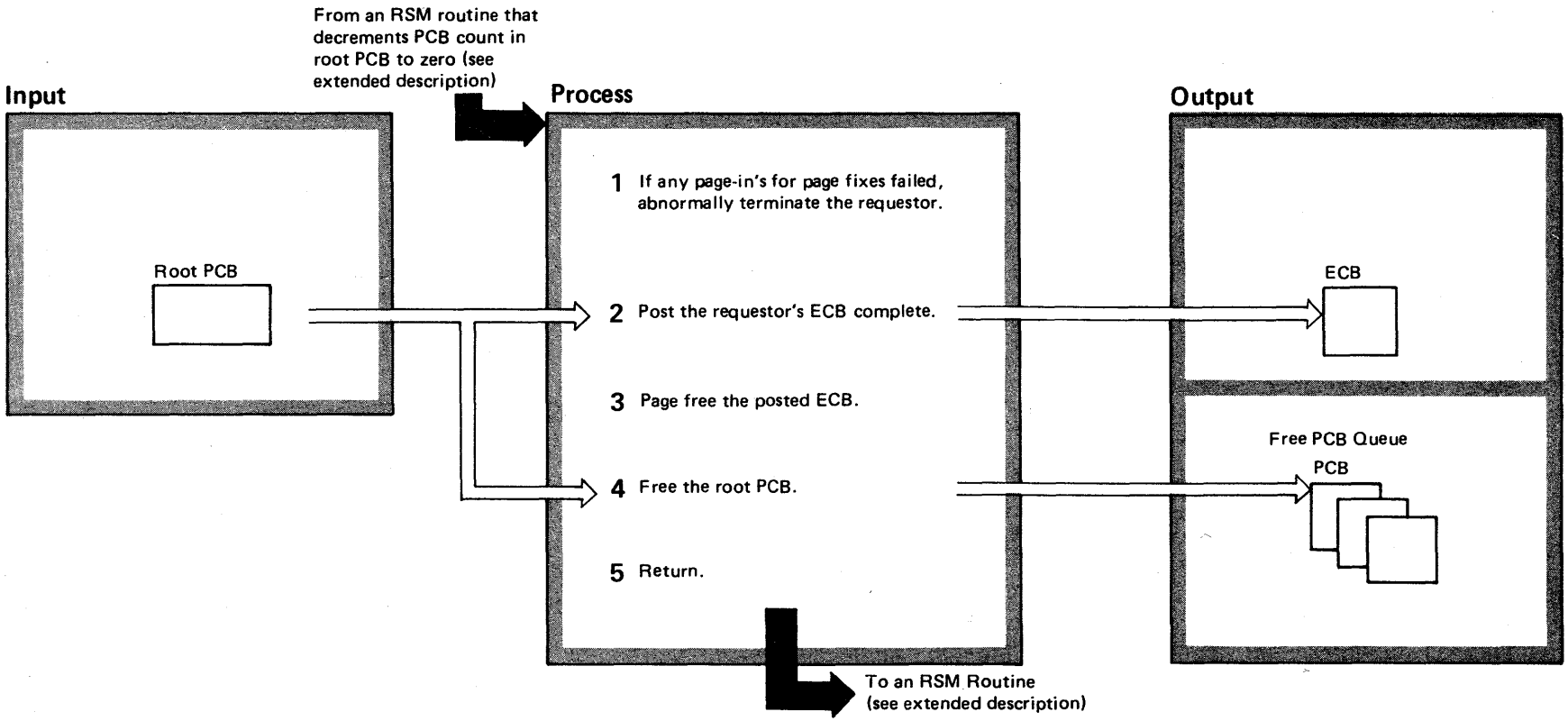


Diagram 23-14. PGFIX/PGLOAD Root Exit (IEAVFXLD) (Part 2 of 2)

Extended Description	Module	Label
The PGFIX/PGLOAD Root Exit (a part of IEAVFXLD) completes processing of a root page control block (PCB) when the PCB count has been decreased to zero. The local and SALLOC locks are held by the caller. RSM routines that use this exit are: IEAVPIOP, IEAVIOCP, IEAVSOUT, IEAVGFA, and IEAVTERM.		
1 If the intercept flag is set to one in the root PCB, go to step 3. If the intercept flag is set to zero, the FXLD Root Exit checks for an I/O error. If the request is for PGLOAD, FXLD Root Exit continues with normal processing. If the PGFIX request has an I/O error, the FXLD Root Exit schedules abnormal termination for the requestor, using the TCB address in the root PCB. If the TCB address is zero, FXLD Root Exit posts the requestor's ECB from the root PCB with an error POST code.	IEAVFXLD	IEAVFXL
2 If both the intercept flag and the I/O error flag are set to zero, the FXLD Root Exit posts the requestor's ECB with a zero POST code, indicating completion.		
3 If the free ECB flag is set to one in the root PCB, the FXLD Root Exit issues a PGFREE request through IEAVPSI.		
4 The FXLD Root Exit converts the root PCB to a regular PCB and returns it to the free queue.		
5 The FXLD Root Exit returns control to the calling routine.		

Diagram 23-15. PGFREE Routine (IEAVFREE) (Part 1 of 2)

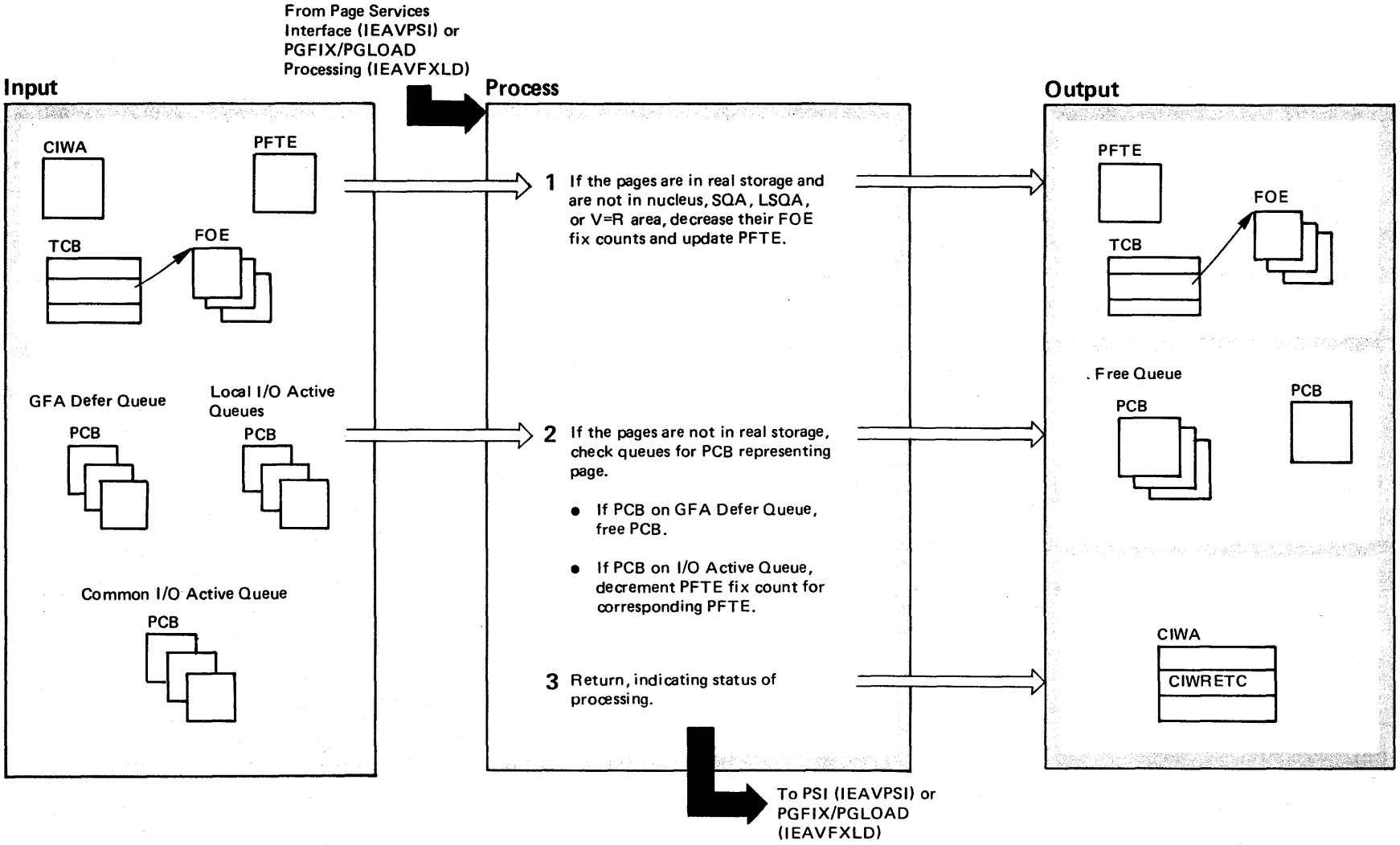


Diagram 23-15. PGFREE Routine (IEAVFREE) (Part 2 of 2)

Extended Description	Module	Label
The PGFREE routine (IEAVFREE) is called through Page Services Interface to free up a group of real pages previously fixed. When called by the PGFIX function, it also reverses a partially completed fix operation that is being abnormally terminated.		
1 PGFREE checks the status of the page being processed. If the page is not already in real storage and PGFIX is the caller, PGFREE returns control immediately. If PGFIX is not the caller and if the requestor supplies an ECB address, PGFREE performs purge processing as in step 2. Otherwise, PGFREE returns control. If the page is valid in real storage, PGFREE checks the page location. If the page resides in the nucleus, system queue area, local system queue area, or V=R area, then PGFREE does not process the page. Otherwise, IEAVFREE locates a fix ownership element (FOE), if one exists. If no FOE exists on the fix ownership list (FOL), PGFREE does no free processing. Otherwise, PGFREE decreases the fix count and frees the FOE if the count becomes zero. Then PGFREE decreases the fix count in the PFTE, unless PGFIX is the caller. If the PFTE fix count becomes zero and the page was long-fixed, the long fix flag is set to zero and the system fix counters are decremented by one; if a deferred release was specified, PGFREE calls the PGRLSE processor to perform deferred release processing.	IEAVFREE	IEAVFREE
2 If the requestor supplies an ECB address, PGFREE checks three queues for PCBs representing paging I/O for the current address space: General Frame Allocation (GFA) queue, the Common I/O Active Queue, and the Local I/O Active Queue. When it finds one, PGFREE checks for a root PCB and marks the root PCB intercepted, which prevents posting the ECB. If the root PCB has an FOE associated with it, PGFREE calls FOEDEL to find and remove all FOEs for the PGFIX request being purged. FOEDEL is called repeatedly until the PCB fix count is zero. PGFREE either frees the PCB from the GFA Defer Queue or decreases the PFTE fix count for the frames assigned to the virtual page on the I/O active queues. If the PFTFXCT is decremented to zero, the system fix counters are decremented by one. PGFREE checks for related PCBs as well, continuing until all three queues have been searched. **3** If an error is detected in the input data for list entry requests, PGFREE sets the error flag in the CIWA copy of the VSL entry and stores the whole VSL entry over the user-supplied copy. The CIWA return code of 4 is also saved. Then PGFREE returns to PSI for exit processing. If no errors occur, PGFREE passes the return code and output data to the caller, PGFIX or PSI, for exit processing.		

Diagram 23-16. PGOUT Routine (IEAVOUT) (Part 1 of 2)

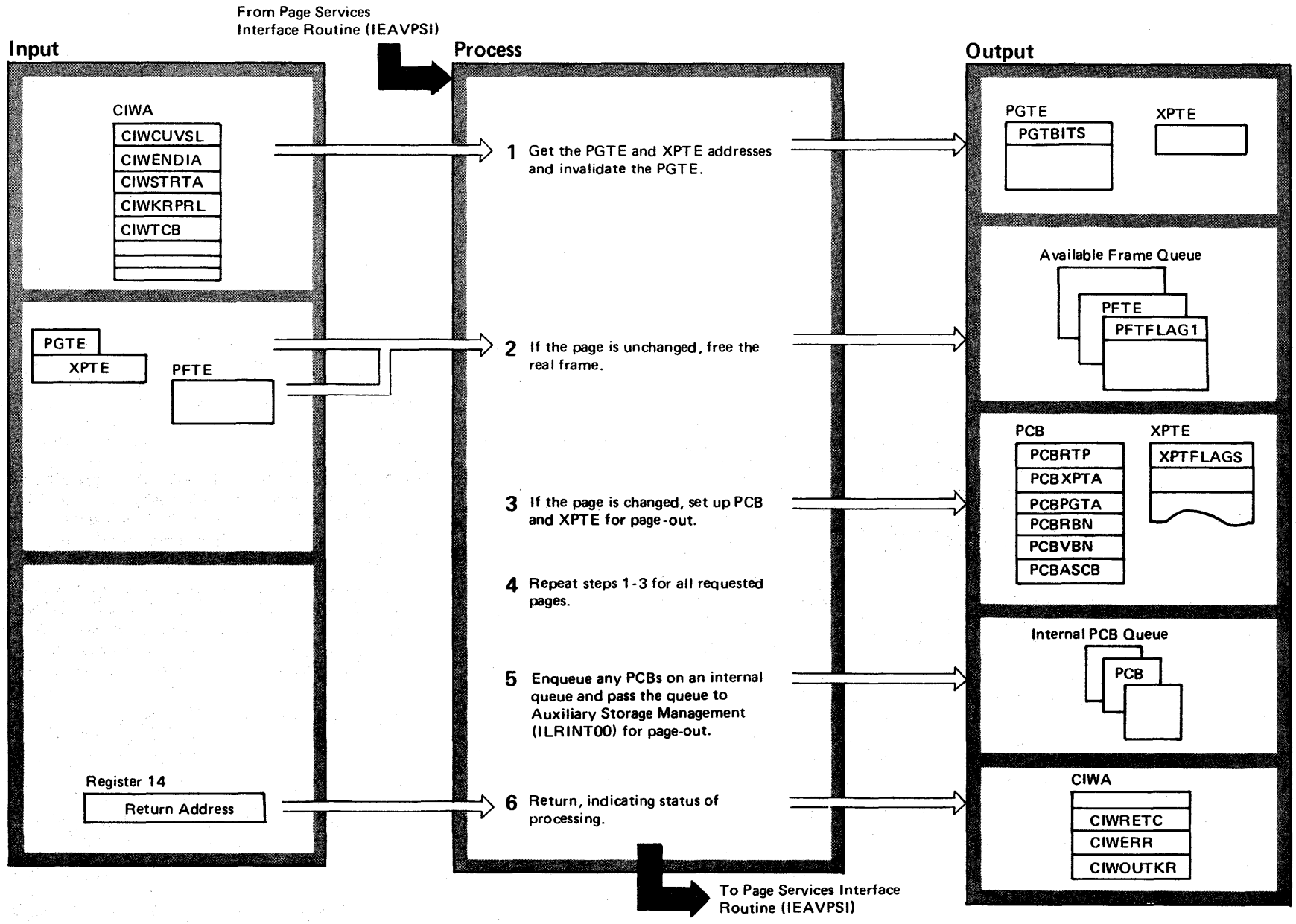


Diagram 23-16. PGOUT Routine (IEAVOUT) (Part 2 of 2)

Extended Description	Module	Label
The PGOUT routine (IEAVOUT) is called by the Page Services Interface routine to process a page-out for a selected virtual page.		
1 PGOUT processes each VSL entry in the CIWA. It checks and rounds the addresses to page boundaries; if an error is detected, PGOUT sets the CIWA return code to 4.	IEAVOUT	
For a page with a frame assigned in real storage, PGOUT invalidates the PGTE using the Page Invalidation routine. If the page resides in the nucleus, SQA, V=R space, LSQA, or quick start area, or if the page is unusable or fixed, no processing is done. If a PCB already exists, no processing is performed.		
2 If the page is unchanged, PGOUT returns the PFTE for the frame to the available frame queue. If the Keepreal option flag in the PCB or the internal Keepreal flag is set to one, PGOUT validates the PGTE and returns control.		
3 If the page has been changed, PGOUT builds a PCB and initializes fields in the PCB and XPTE.		
4 When the first VSL entry is complete, PGOUT checks the CIWA return code. For a zero return code, PGOUT gets the next VSL entry by using the Page Services Interface NEXTVSL subroutine; for a return code of 8, PGOUT performs exit processing; and for all other return codes, PGOUT performs error processing.		
5 PGOUT puts the created PCBs on an internal queue and, when all VSL entries have been processed, passes them to the Auxiliary Storage Manager by calling ILRPAGIO.		
6 If no errors have occurred, PGOUT returns control to PSI, putting the return code in the CIWA. If an invalid page address was detected, and the CIWA return code is 4, PGOUT sets the CIWA error flag to one and copies the CIWA copy of the VSL entry over the user copy. Then PGOUT returns to PSI.		

Diagram 23-17. Swap-In Processor Routine (IEAVSWIN) (Part 1 of 2)

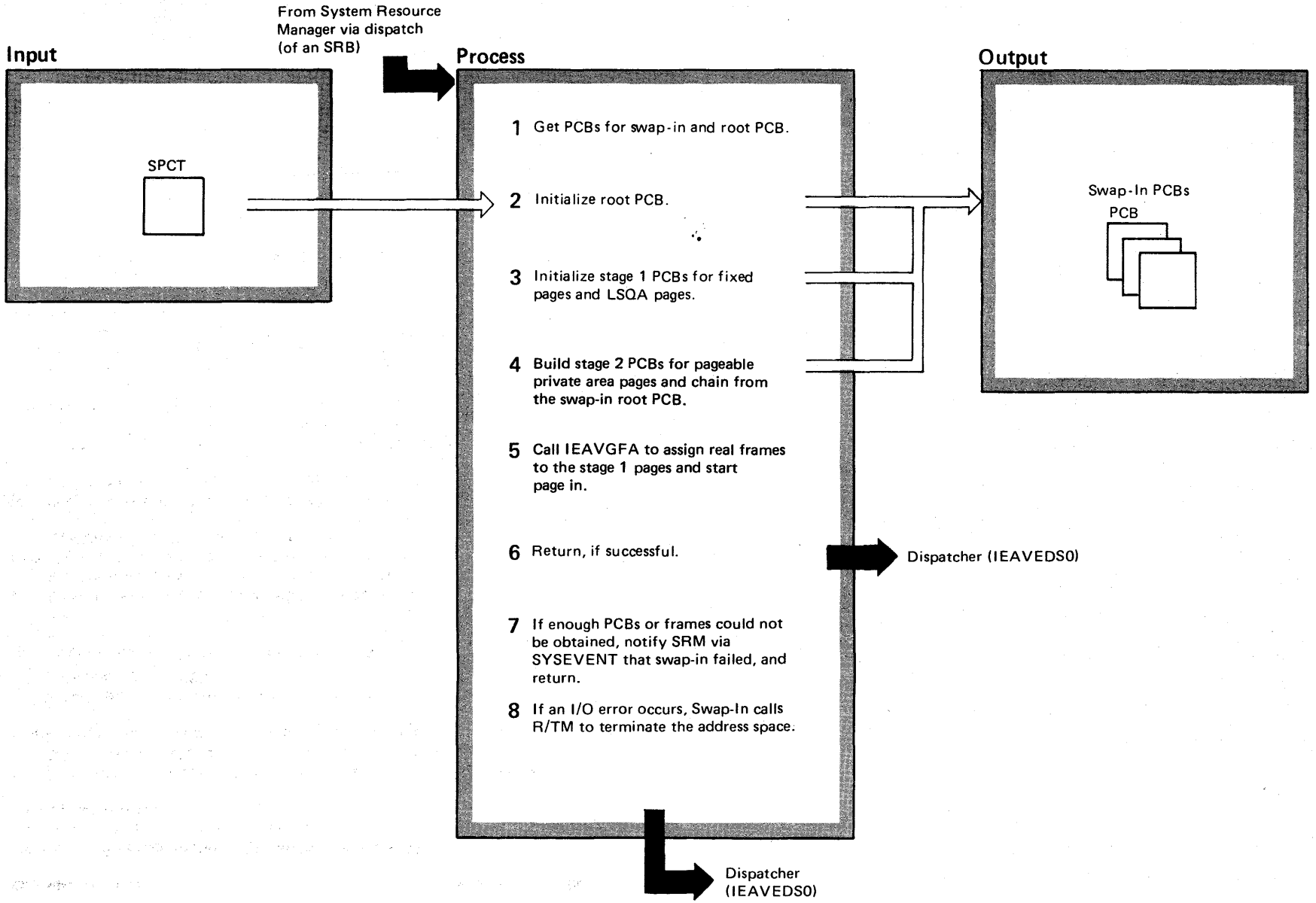


Diagram 23-17. Swap-In Processor Routine (IEAVSWIN) (Part 2 of 2)

Extended Description	Module	Label
The Swap-In Processor routine (IEAVSWIN) initializes I/O operations for an address to be swapped in (made active within an address space).		
1 After freeing the input SRB and establishing the FRR, Swap-In gets the SALLOC lock. If the swap-in request is valid, Swap-In gets enough PCBs for the swap-in operation.	IEAVSWIN	
2 Swap-In puts the root exit address and the ASCB address in the root PCB.		
3 Swap-In initializes PCBs for Stage I pages to be swapped in.		
4 Swap-In initializes Stage II page PCBs for swapping in and chains them from the swap-in root PCB. The Stage 2 PCBs will be passed to IEAVSWPP (an entry point in IEAVSWIN) in the IEAVSWPP SRB.		
5 Swap-In calls IEAVGFA to assign real frames and initiate the page-in process for the stage 1 pages.		
6 If the I/O successfully completes, Swap-In updates the count of swap-ins in the PVT, releases the SALLOC lock and the FRR, and returns control to the Dispatcher.		
7 If Swap-In cannot get enough PCBs to swap in the address space or if there are not enough real frames available, Swap-In issues a SYSEVENT to notify SRM that the swap-in failed.		
8 If an I/O error occurs, Swap-In calls R/TM (TYPE=MEMTERM) to terminate the address space.		

Diagram 23-18. Swap-In Root Exit (IEAVSWIN) (Part 1 of 2)

From Page I/O Post (IEAVPIOP), Page Termination Services (IEAVTERM), or General Frame Allocation (IEAVGFA) decreasing PCB count in root PCB to zero

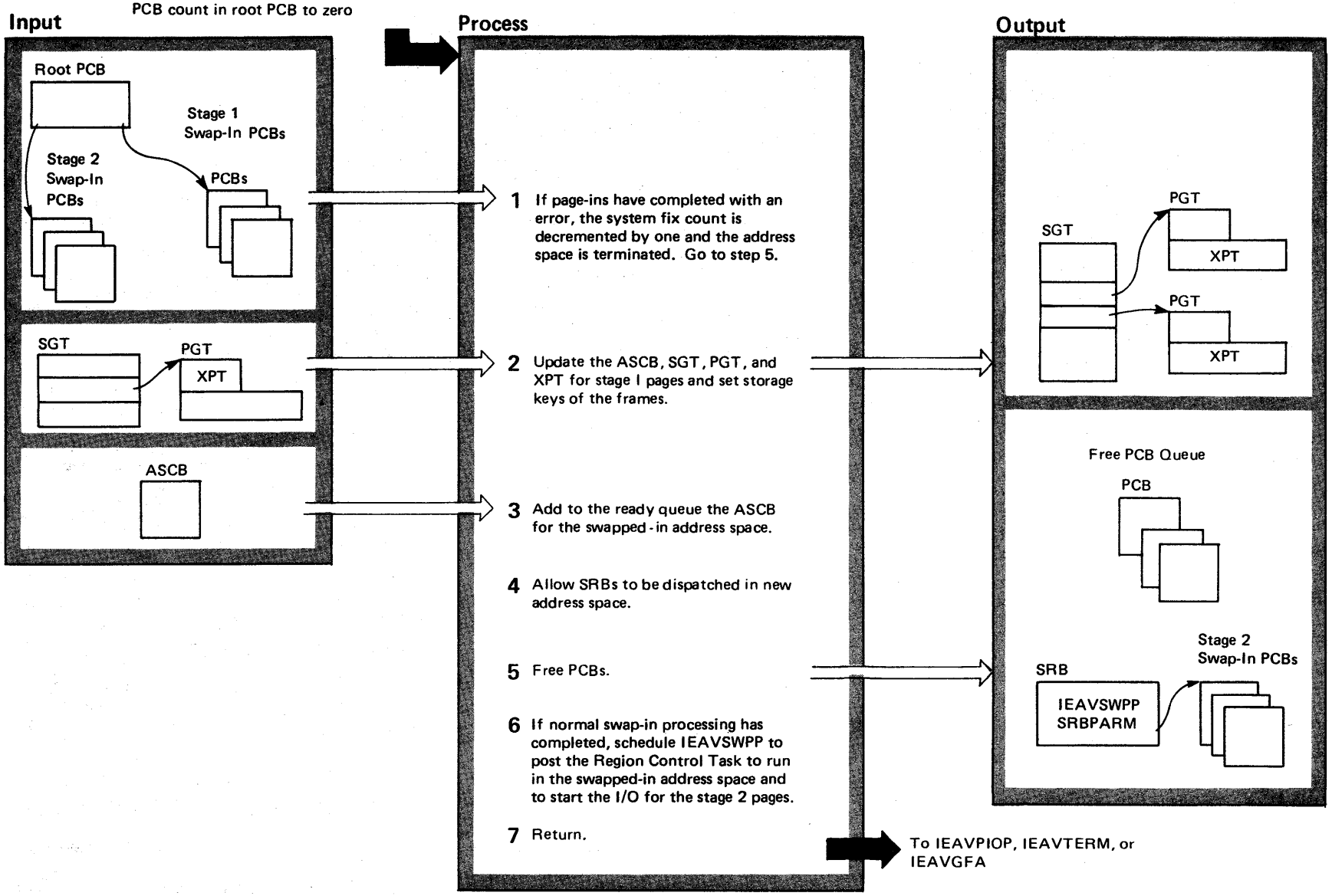


Diagram 23-18. Swap-In Root Exit (IEAVSWIN) (Part 2 of 2)

Extended Description	Module	Label
The Swap-In Root exit (part of IEAVSWIN) is called by Page I/O Post when I/O for Stage I pages has completed. The routine re-initializes the segment and page table entries, re-enqueues the ASCB, and makes the swapped-in address space dispatchable.		
1 If page-ins have completed in error, Swap-In Root Exit decreases the fix counts for the common area swap-in pages and completes error processing in step 5.	IEAVSWIN	IEAVSIRT
2 Swap-In Root Exit updates the ASCB, the PGT, and XPT with information about the swap-in Stage I pages. It then validates the PGT and XPT and sets the storage keys for the page frames.		
3 Swap-In Root Exit calls ASCBCHAP to add the ASCB to the ready queue.		
4 Swap-In Root Exit calls STATUS START to allow SRBs to be dispatched in a new address space.		
5 Swap-In Root Exit frees the root PCB and the chain of PCBs used for the swap-in.		
6 If the swap-in was successful, IEAVSIRT schedules an SRB routine, IEAVSWPP, to the swapped-in address space. This routine posts the RCT to begin restore processing and start the I/O for the stage 2 pages.		
7 Swap-In Root Exit returns control to the caller.		

Diagram 23-18A. Swap-In-Post Processor (IEAVSWPP) (Part 1 of 2)

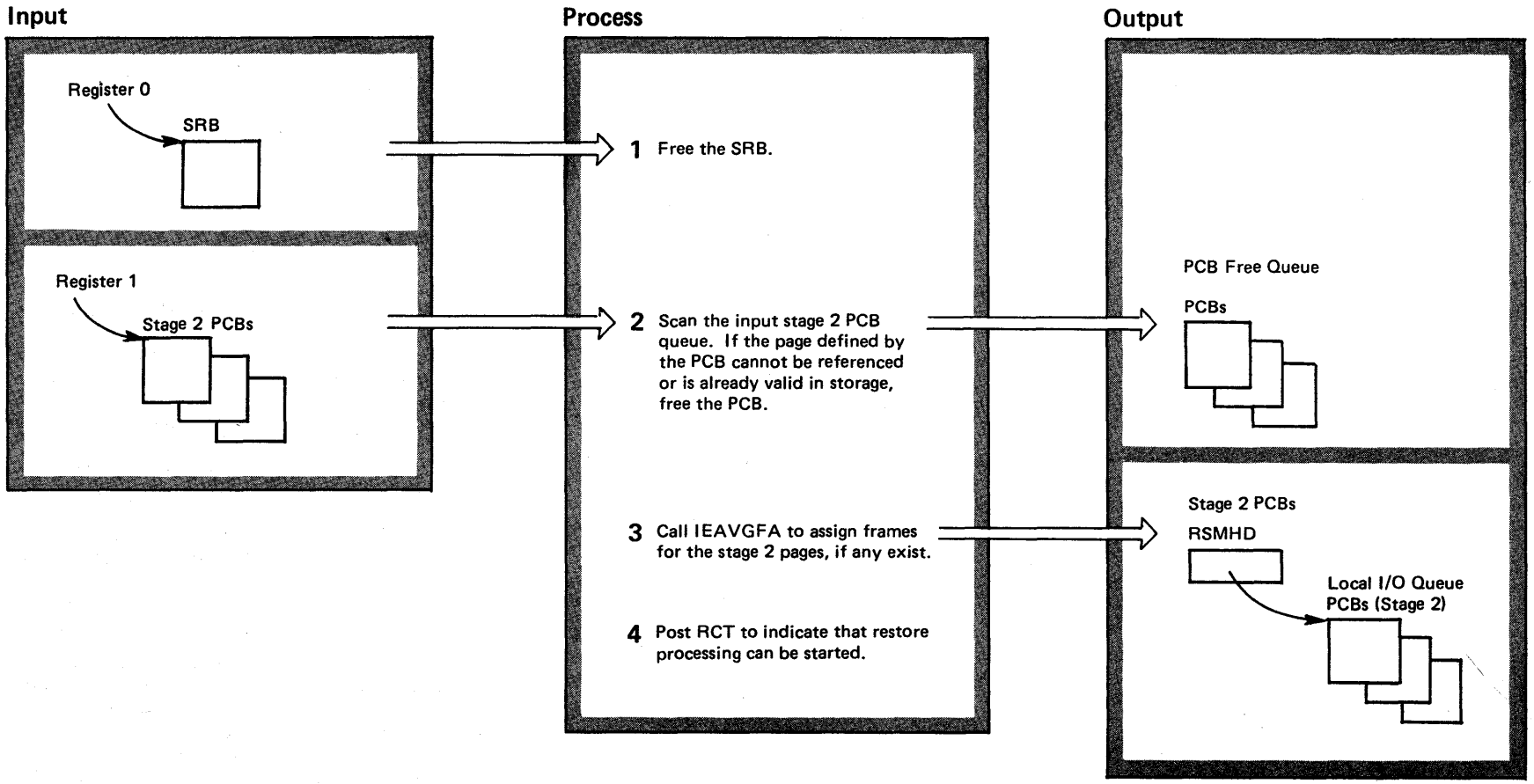


Diagram 23-18A. Swap-In-Post Processor (IEAVSWPP) (Part 2 of 2)

Extended Description	Module	Label
The Swap-Post processor (IEAVSWPP) initiates the I/O for the stage 2 pages and posts RCT when stage 1 swap-in is complete.	IEAVSWIN	IEAVSWPP

1 Free the input SRB.

2 For each PCB on the input stage 2 queue, call IEAVFP2 to obtain the PGTE/XPTE addresses for the page represented by the PCB. If the page was freed or the page is already in storage, free the PCB because the real frame assignment for this request is no longer required.

3 Call IEAVGFA to assign frames for the remaining stage 2 pages.

4 Post RCT to indicate that stage 1 swap-in is complete and that restore processing can now be started.

Diagram 23-19. Swap-Out Processor Routine (IEAVSOUT) (Part 1 of 4)

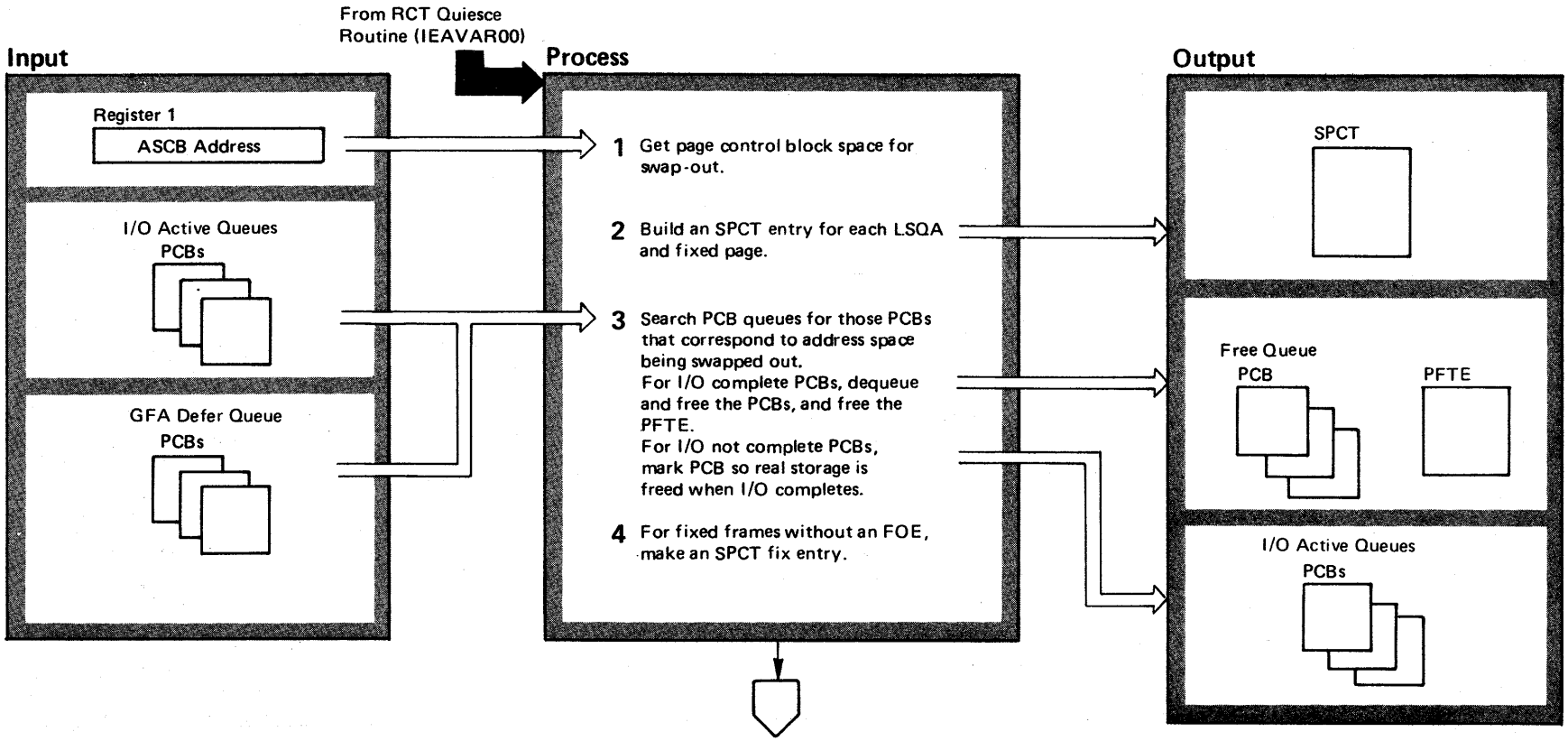


Diagram 23-19. Swap-Out Processor Routine (IEAVSOUT) (Part 2 of 4)

Extended Description	Module	Label
The Swap-Out Processor routine (IEAVSOUT) performs and initiates the process of logically disconnecting an address space, initiating the I/O operation for page-out to auxiliary storage, and saving in real storage the information required for a subsequent swap-in.		
1 Swap-Out calls STATUS to stop non-quiescable SRBs. Then it gets the SALLOC lock and sets the FRR. If the swap request is valid, swap-out calls IEAVPCB to get enough PCBs for all the frames in the address space plus one to be used as a swap-out parameter list. The list will contain a pointer to the LSQA PCBs, a pointer to the private area non-LSQA PCBs, an eight-byte parameter list passed to SRM on the swap-out complete sysevent, and an SRB used to schedule IEAVPIOI.	IEAVSOUT	
2 Swap-Out initializes the SPCT and then builds SPCT entries for each Stage I page (either LSQA or fixed).		
3 Swap-Out scans the Common I/O Queue for PCBs corresponding to the address space being swapped out. Swap-Out calls the Reset routine of PCIH and resets any fix indicators. If I/O is complete, Swap-Out frees the PCB. Then Swap-Out scans the I/O active queue and the GFA Defer Queue, processing PCBs in the same manner. If any root PCB count goes to zero, Swap-Out calls the root exit routine.		
4 If a fixed frame has no FOE, Swap-Out sets the fix count in the SPCT fix entry.		

Diagram 23-19. Swap-Out Processor Routine (IEAVSOUT) (Part 3 of 4)

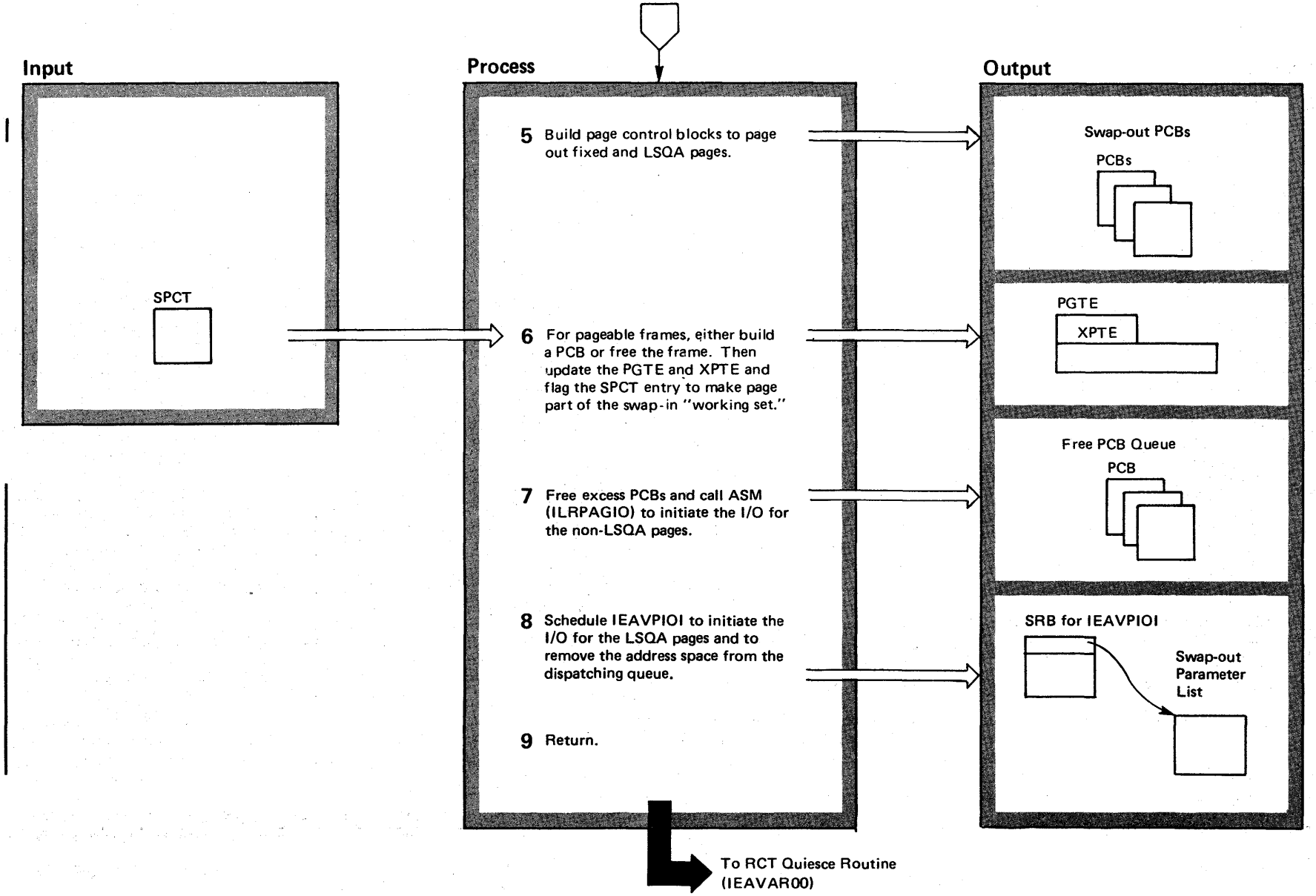


Diagram 23-19. Swap-Out Processor Routine (IEAVSOUT) (Part 4 of 4)

Extended Description | **Module** | **Label**

5 Swap-Out completes the initialization of swap-out PCBs for LSQA and fixed pages.

6 For pageable frames with no PCB defined, Swap-Out either frees the frame or creates a Swap-Out PCB. After updating the PGTE and XPTE for each page, Swap-Out marks the SPCT entry so that the page will be swapped in with the address space.

7 Swap-Out frees any PCBs not used and puts the swap-out PCBs on the local I/O active queue. Then swap-out invokes ASM at ILRPAGIO to initiate the I/O for the non-LSQA pages.

8 Swap-out schedules IEAVPIOI to start the I/O for the LSQA pages and to remove the address space from the dispatching queue. IEAVPIOI receives the swap-out parameter list containing a pointer to the LSQA PCBs.

9 Swap-Out frees the unused SPCT extensions, frees the unused SPCT extensions, frees the SALLOC lock and FRR, and returns control to RCT Quiesce with a return code in register 15.

Diagram 23-20. Swap-out Completion Routine (IEAVSWPC) (Part 1 of 2)

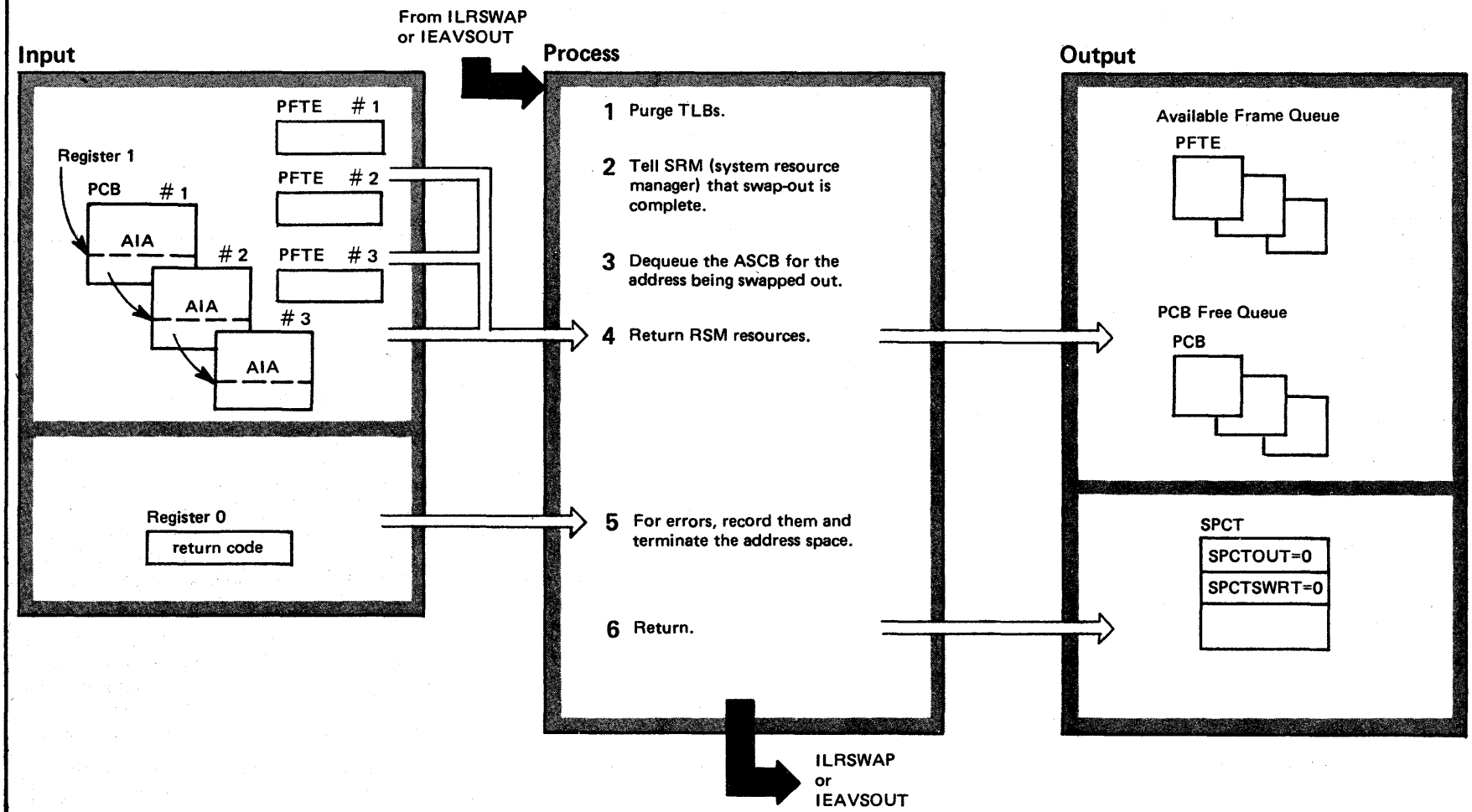


Diagram 23-20. Swap-out Completion Routine (IEAVSWPC) (Part 2 of 2)

Extended Description	Module	Label
The Swap-out completion routine (IEAVSWPC) handles completion processing for swap-outs. The input is the address of a chain of AIAs and a return code. IEAVSWPC is entered from the Swap-Out Processor (EAVSOUT) or from ILRSWAP. The SALLOC lock is held at entry.		
1 SWPC established the RSM FRR and calls IEAVINV to purge the translation lookaside buffers (TLBs).	IEAVSWPC	IEAVSWPC
2 If the I/O was successful, IEAVSWPC issues a SYSEVENT notifying the system resource manager (SRM) that the swap-out has been completed and passes status information to SRM about the swapped-out frames.		
3 For either a successful or unsuccessful swap-out, IEAVSWPC frees the area (PCB) containing the swap-out parameter list by calling the PCB manager (IEAVPCB).		
• The PCB defined flag is turned off in the PFTEs for the frames that were allocated to the swapped out pages. If the I/O was successful, IEAVSWPC calls PFTE enqueue/dequeue to free the frames and calls IEAVPCB to free the PCBs used for the swap-out. The system fix counters are decremented by 1 for each AIA passed as input.	IEAVSWPC	FREFMPCB
• If the I/O was unsuccessful, IEAVSWPC does not free the frames.	IEAVSWPC	FREEPCB
4 If a nonzero return code was given to IEAVSWPC, a COD abend is issued.		
The address space being swapped out is terminated via CALLRTM.	IEAVSWPC	MMTERM
5 IEAVSWPC removes the FRR and returns to the caller.	IEAVSWPC	DELTEFRR

Diagram 23-21. LSQA Swap I/O Initiator (IEAVPIOI) (Part 1 of 2)

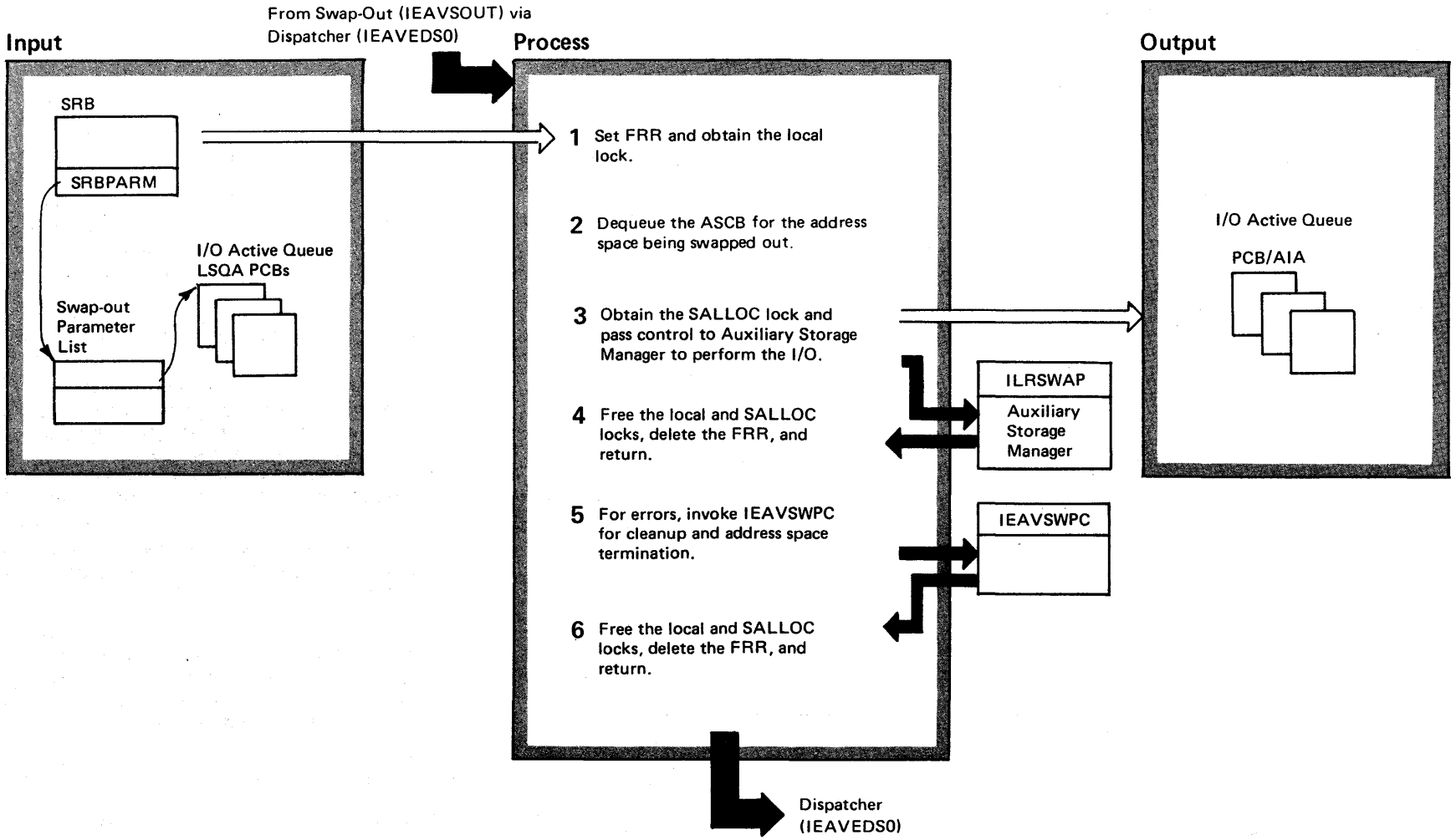


Diagram 23-21. Swap I/O Initiator (IEAVPIOI) (Part 2 of 2)

Extended Description	Module	Label
The Swap I/O Initiator (IEAVPIOI) starts the LSQA paging I/O for the address space being swapped out. The input is the address of the swap-out parameter list containing a pointer to the LSQA PCBs. IEAVPIOI passes the PCB/AIAs to ASM to start the swap-out I/O.		
1 PIOI establishes the RSM FRR, and gets the local lock.	IEAVPIOI	IEAVPIOI
2 PIOI calls ASCBCHAP to remove the address space from the dispatching queue.		
3 PIOI obtains the SALLOC lock and calls ASM (ILRSWAP) to perform the page-out I/O.		
4 If the I/O is successful, PIOI releasess the SALLOC and local locks, removes the FRR, and returns control to the Dispatcher.		
5 If an error occurs, or if the ASCBCHAP fails, IEAVPIOI calls IEAVSWPC for cleanup processing and for terminating the address space.		
6 PIOI releases the local and SALLOC locks, removes the FRR, and returns control to the dispatcher (IEAVEDS0).		

Diagram 23-22. VIO Services Routine (IEAVAMSI) (Part 1 of 4)

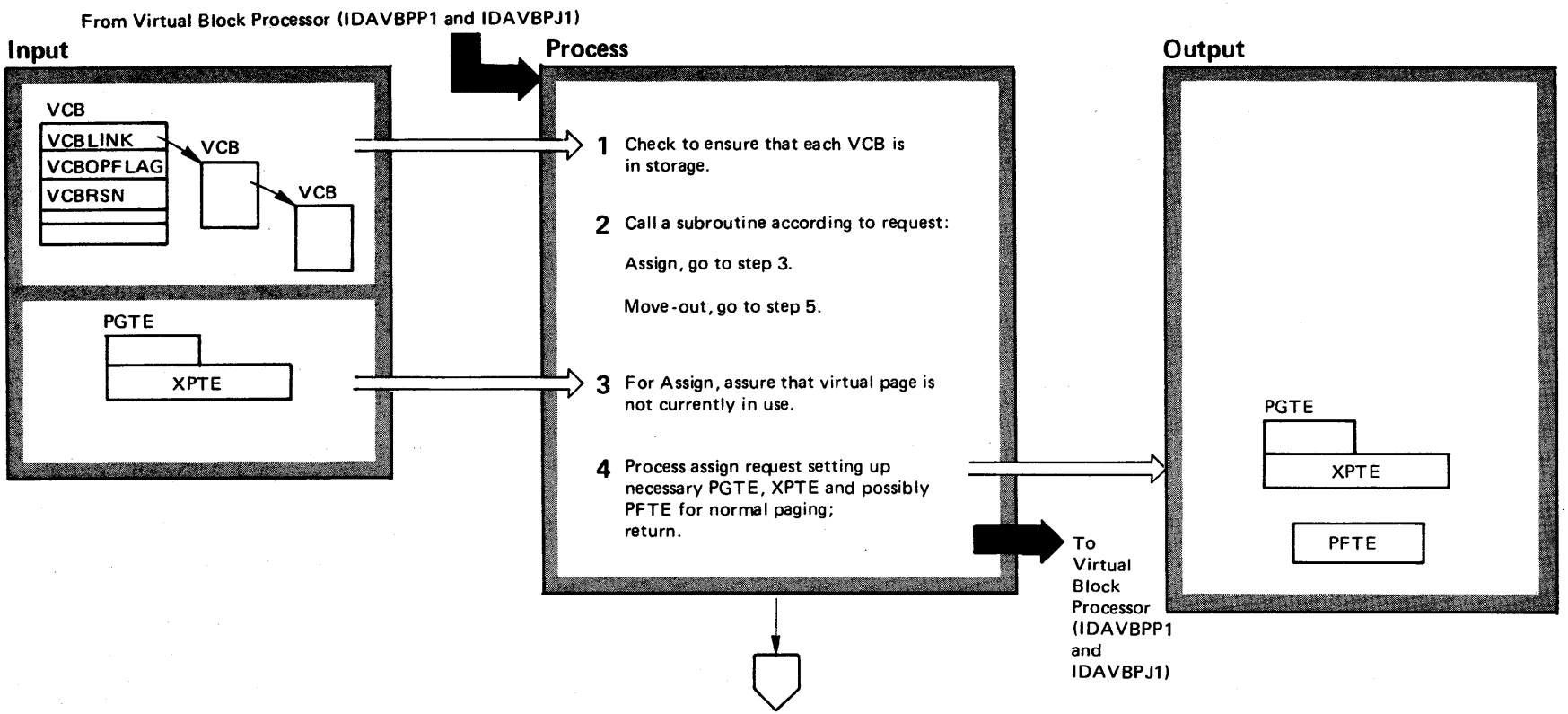


Diagram 23-22. VIO Services Routine (IEAVAMSI) (Part 2 of 4)

Extended Description	Module	Label
The VIO Services routine (IEAVAMSI) manipulates the page and external page tables; in some cases it also manipulates the page frame table for the VIO Processor when VIO data set pages are to be inserted or removed from the VIO buffer. One VCB (VIO Control Block) is supplied for each page to be processed.		
1 VIO Services obtains the global SALLOC lock and checks the input VCB to be sure that the real storage address specified is valid.	IEAVAMSI	
2 VIO Services checks the operation flags in the VCB for the operation to be performed.		
3 For an assign request, VIO Services checks for the following conditions: • GETMAIN-assigned flag and invalid flag in PGTE are set to one; • Real storage address in PGTE is zero; • Auxiliary-storage-assigned and the defer flags in the XPT are set to zero. If any of the preceding conditions are not met, VIO Services sets an error code in the VCB and returns to VBP with a code of 4 in register 15.		
4 If a null assignment is requested (LPID in VCB is zero), VIO Services sets the VIO flag to one in the XPTE and checks for further VCBs. Otherwise, if the RSN in the VCB is not zero, VIO Services gets the PFTE for the real frame that last contained the page. It checks to see whether the VIO flag is set to one and whether the data set ID matches the ID in the VCB. If so, the page has been reclaimed. If a PCB exists for the reclaimed PFTE, VIO Services updates the PCB to halt the page-out from freeing the real frame. It also updates the XPTE and the PGTE. Finally, VIO Services puts the virtual address of the VCB and the ASID in the PFTE.		

Diagram 23-22. VIO Services Routine (IEAVAMSI) (Part 3 of 4)

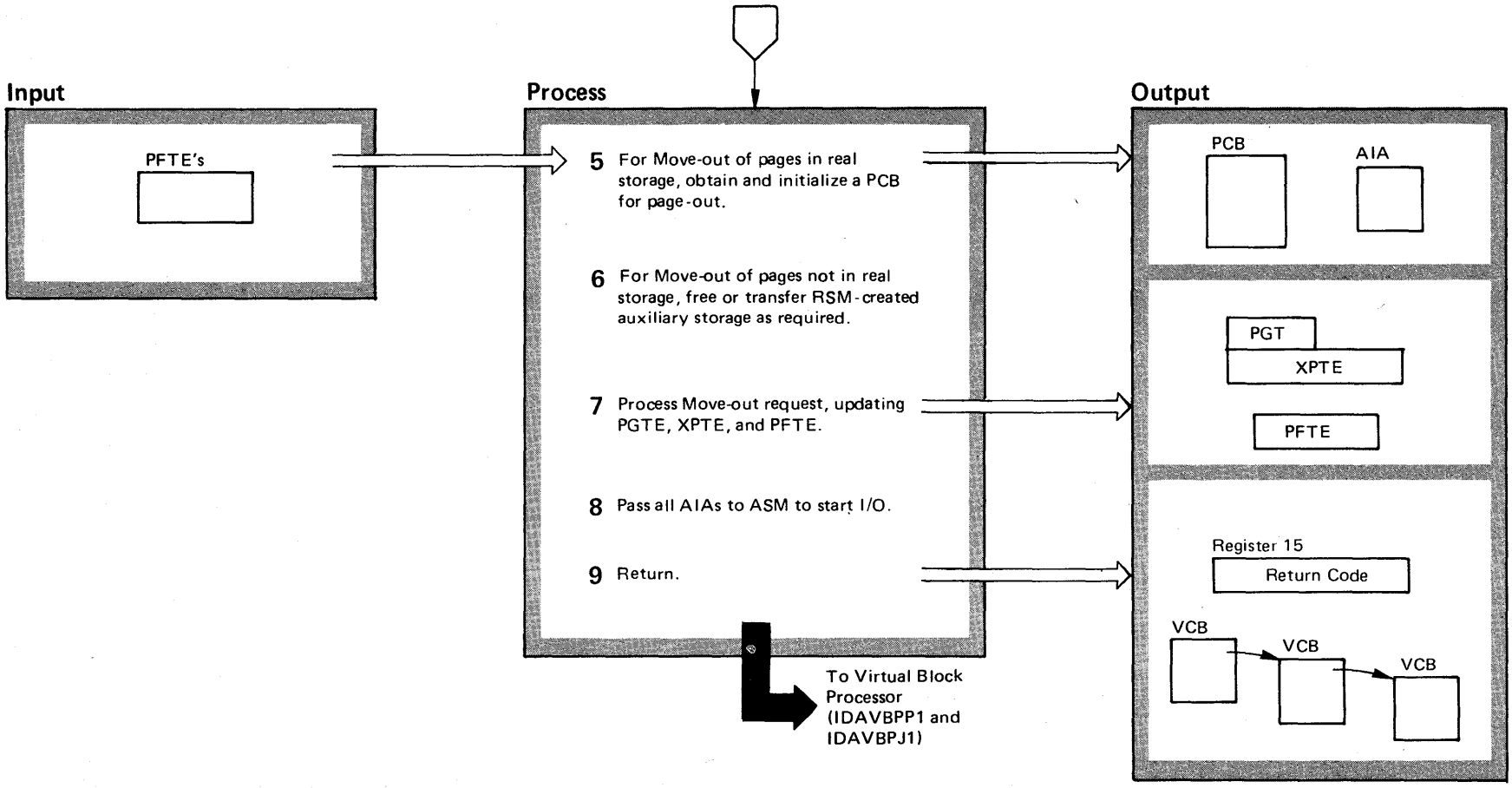


Diagram 23-22. VIO Services Routine (IEAVAMSI) (Part 4 of 4)

Extended Description	Module	Label
5 VIO Services returns to VBP, passing a return code in register 15.		
6 If the page is not in real storage, VIO Services transfers RSM-created auxiliary storage as required and sets the real storage address in the VCB to zero. If paging I/O is in process for the page, VIO Services quiesces page-in I/O and allows page-out I/O to complete normally. VIO Services processes all PCBs for the page according to the queue on which they reside. VIO Services releases all non-VIO auxiliary storage for the page and updates the VCB and the XPTE.		
7 VIO Services updates status flags in the XPTE, VCB, PGTE, and the PFTE to complete the Move-Out request according to the options specified in the VCB.		
8 When all VCBs have been processed, VIO Services passes any AIAs created to ASM (at ILRINT00) for page-out I/O processing. Then it returns to VBP, passing a return code in register 15.		

Diagram 23-23. Initialize Address Space Routine (IEAVITAS) (Part 1 of 2)

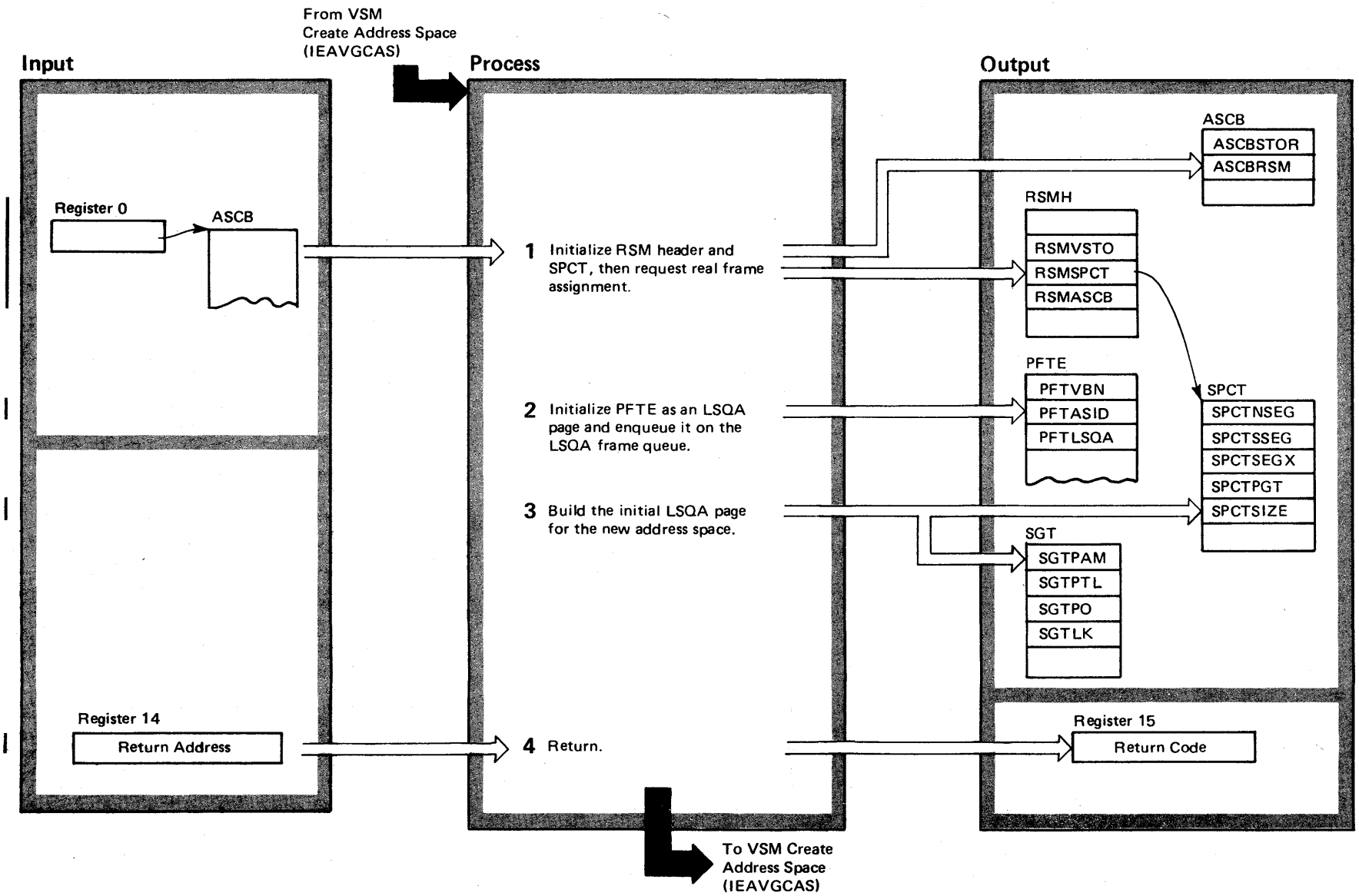


Diagram 23-23. Initialize Address Space Routine (IEAVITAS) (Part 2 of 2)

Extended Description	Module	Label
The Initialize Address Space routine (IEAVITAS) builds and initializes the RSM control blocks required to define an address space. The function runs in the Master Scheduler address space, is called in supervisor, key 0 state, and must run under a local lock.		
1 The Initialize routine sets up linkage with the RSM FRR IEAVRCV and acquires the SALLOC lock. Initialize obtains SQA space for the RSM Header (the ASM Header is part of the RSMHD) and the Swap Control Table (SPCT). If the GETMAIN fails, Initialize returns with a code of 4 in register 15. Then it calls ASM (ILRINT00) to assign a logical group number for the new address space. If none are available, Initialize returns to the caller with a code of 4 in register 15. It Initializes the RSM Header address in the ASCB and initializes RSM Header fields with the virtual addresses of the SGT, SPCT, and ASCB. Initialize then sets other RSM Header fields to zero. The ASM slot reserve routine (ILRSLTRV) is called to assign reserved slots for the address space. Next, the Initialization routine calls LSQA/SQA Allocation (IEAVSQA) to get a real frame. If the allocation fails, Initialize returns a code of 4 to the caller in register 15. If successful, Initialize initializes the segment table address in the ASCB.	IEAVITAS	
2 Initialize inserts into the page frame table entry the virtual block number of the page with the highest address in the new address space private area and the ASID of the new address space, and sets the LSQA flag to one. Then Initialize calls PFTE Enqueue/Dequeue (IEAVPFTE) to put the PFTE on the new address space's LSQA queue.		
3 Initialize sets the LSQA page to zero and clears the storage keys. It initializes the common area portions of the segment table and marks the private area portions invalid. Then Initialize sets up the SGTE for the private area containing the LSQA page. It initializes the page table last and all other pages invalid. It then initializes the external page table by putting the logical group number in each 12 byte entry. Initialize sets up fields in the SPCT for the active segment count, the segment entry count, the page table address, the segment ID, and the SPCT size. The local (RSMCNTFX) and global (PVTCNTFX) system fix counters are also updated.		
4 Initialize deletes linkage to the RSM FRR, frees the SALLOC lock, and returns to the caller.		

Error Processing

If an error occurs, the Initialize routine restores any successful set-up operations to their status before the error occurred. It frees any real frame obtained, releases the logical group number, and frees the SQA space before returning to the caller with a code of 4 in register 15.

Diagram 23-24. Delete Address Space Routine (IEAVDLAS) (Part 1 of 2)

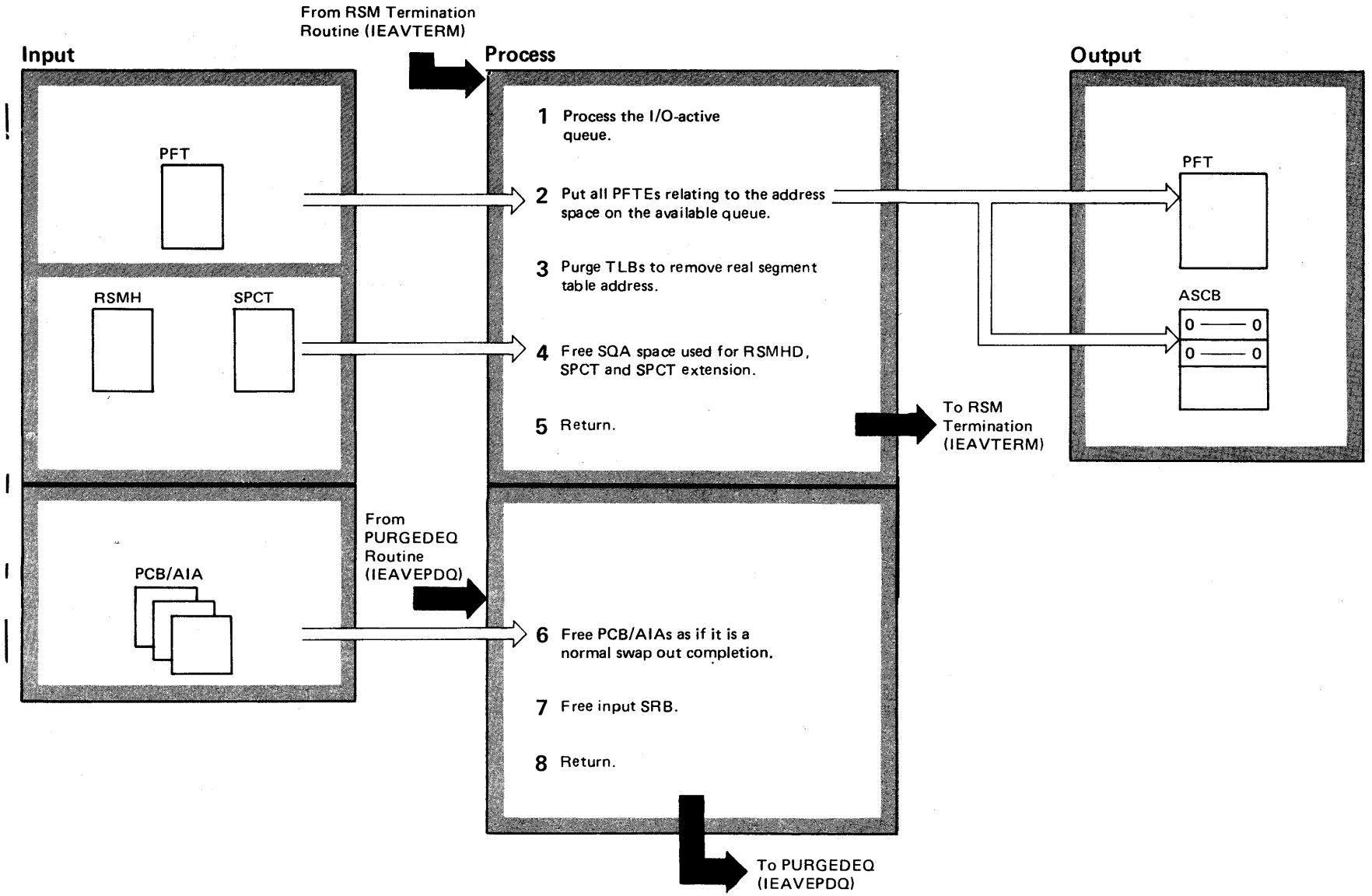


Diagram 23-24. Delete Address Space Routine (IEAVDLAS) (Part 2 of 2)

Extended Description	Module	Label
The Delete Address Space routine (IEAVDLAS) returns RSM resources associated with an address space being terminated. It runs in the Master Scheduler address space.		
1 Delete moves the local I/O-active queue for the address space to the Master local I/O-active queue.	IEAVDLAS	IEAVDLAS
2 Delete scans the local frame queue and calls IEAVPFTE to dequeue PFTEs on the queues, freeing them if no PCB has been defined. If no PCB is defined, the local (RSMCNTFX) and global (PVTCNTFX) fix counters are decremented for each LSQA and PGFIX frame. If any PCBs exist on the Local I/O Active Queue, Delete moves them to the Master Scheduler I/O Active Queue and changes their ASCB addresses to the Master Scheduler ASCB address. Then it sets to zero the RSM Header address and the real segment table address in the ASCB.		
3 Delete calls the Page Invalidate routine, IEAVINV, to purge all translation lookaside buffers.	IEAVINV	
4 Delete uses FREEMAIN to free the SQA space used for the RSM Header, the SPCT, and any SPCT extensions.	IEAVGM00	
5 Delete returns control to RSM Termination.		
6 If the SRB was scheduled to dispatch IEAVSWPP to start the stage 2 swap-out, Delete obtains the SALLOC lock and the PCB manager frees the string of PCB/AIAs addressed in the SRB. Then Delete releases the SALLOC lock.	IEAVDLAS	IEAVSRBP
7 Delete frees the SRB using the FREECELL routine.		
8 Delete returns control to the PURGEDEQ routine.		

Diagram 23-25. Page Termination Services Routine (IEAVTERM) (Part 1 of 2)

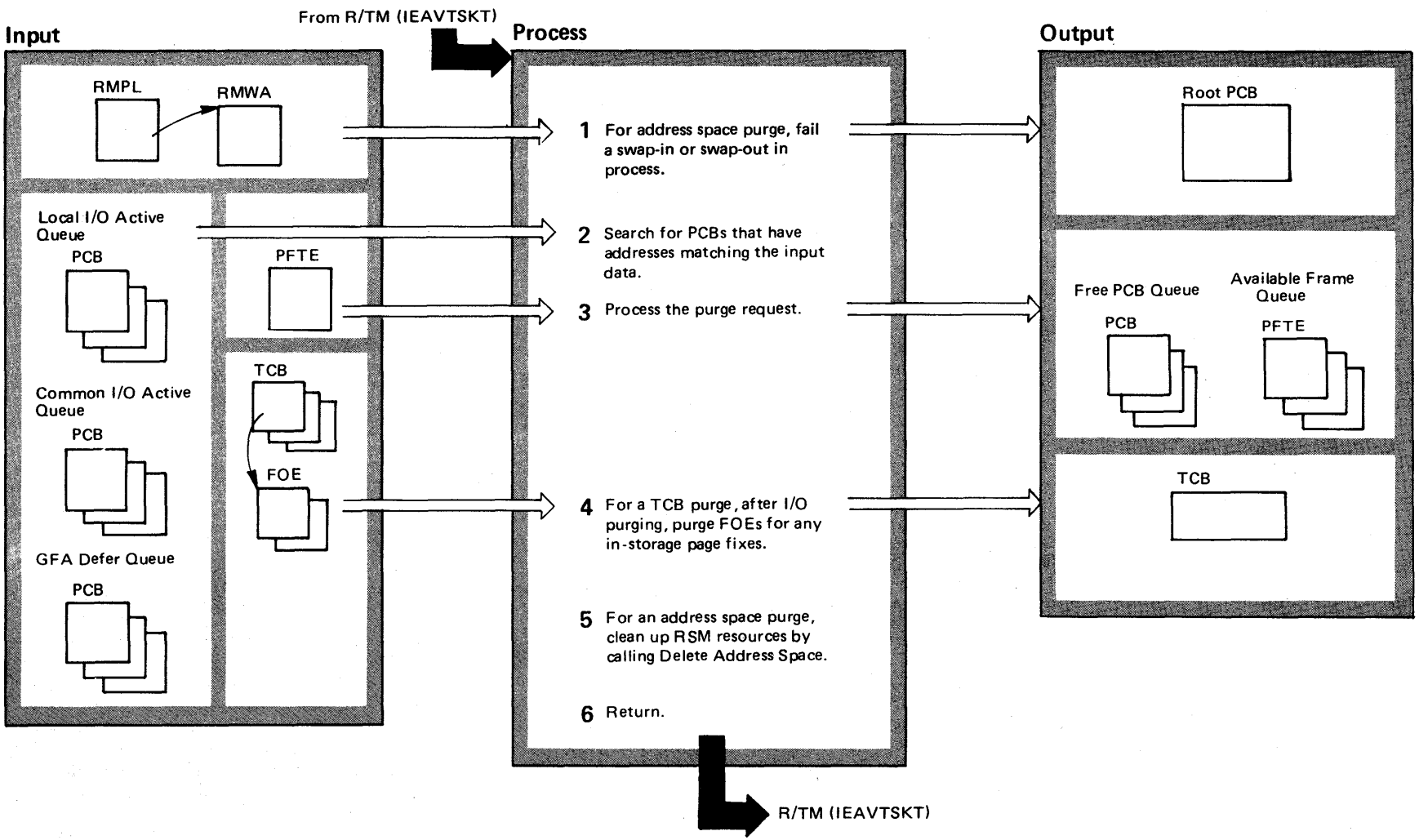


Diagram 23-24. Delete Address Space Routine (IEAVDLAS) (Part 2 of 2)

Extended Description	Module	Label
The Delete Address Space routine (IEAVDLAS) returns RSM resources associated with an address space being terminated. It runs in the Master Scheduler address space.		
1 Delete moves the local I/O-active queue for the address space to the Master local I/O-active queue.	IEAVDLAS	IEAVDLAS
2 Delete scans the local frame queue and calls IEAVPFTE to dequeue PFTEs on the queues, freeing them if no PCB has been defined. If no PCB is defined, the local (RSMCNTFX) and global (PVTCNTFX) fix counters are decremented for each LSQA and PGFIX frame. If any PCBs exist on the Local I/O Active Queue, Delete moves them to the Master Scheduler I/O Active Queue and changes their ASCB addresses to the Master Scheduler ASCB address. Then it sets to zero the RSM Header address and the real segment table address in the ASCB.		
3 Delete calls the Page Invalidate routine, IEAVINV, to purge all translation lookaside buffers.	IEAVINV	
4 Delete uses FREEMAIN to free the SQA space used for the RSM Header, the SPCT, and any SPCT extensions.	IEAVGM00	
5 Delete returns control to RSM Termination.		
6 If the SRB was scheduled to dispatch IEAVSWPP to start the stage 2 swap-out, Delete obtains the SALLOC lock and the PCB manager frees the string of PCB/AIAs addressed in the SRB. Then Delete releases the SALLOC lock.	IEAVDLAS	IEAVSRBP
7 Delete frees the SRB using the FREECELL routine.		
8 Delete returns control to the PURGEDEQ routine.		

Diagram 23-25. Page Termination Services Routine (IEAVTERM) (Part 1 of 2)

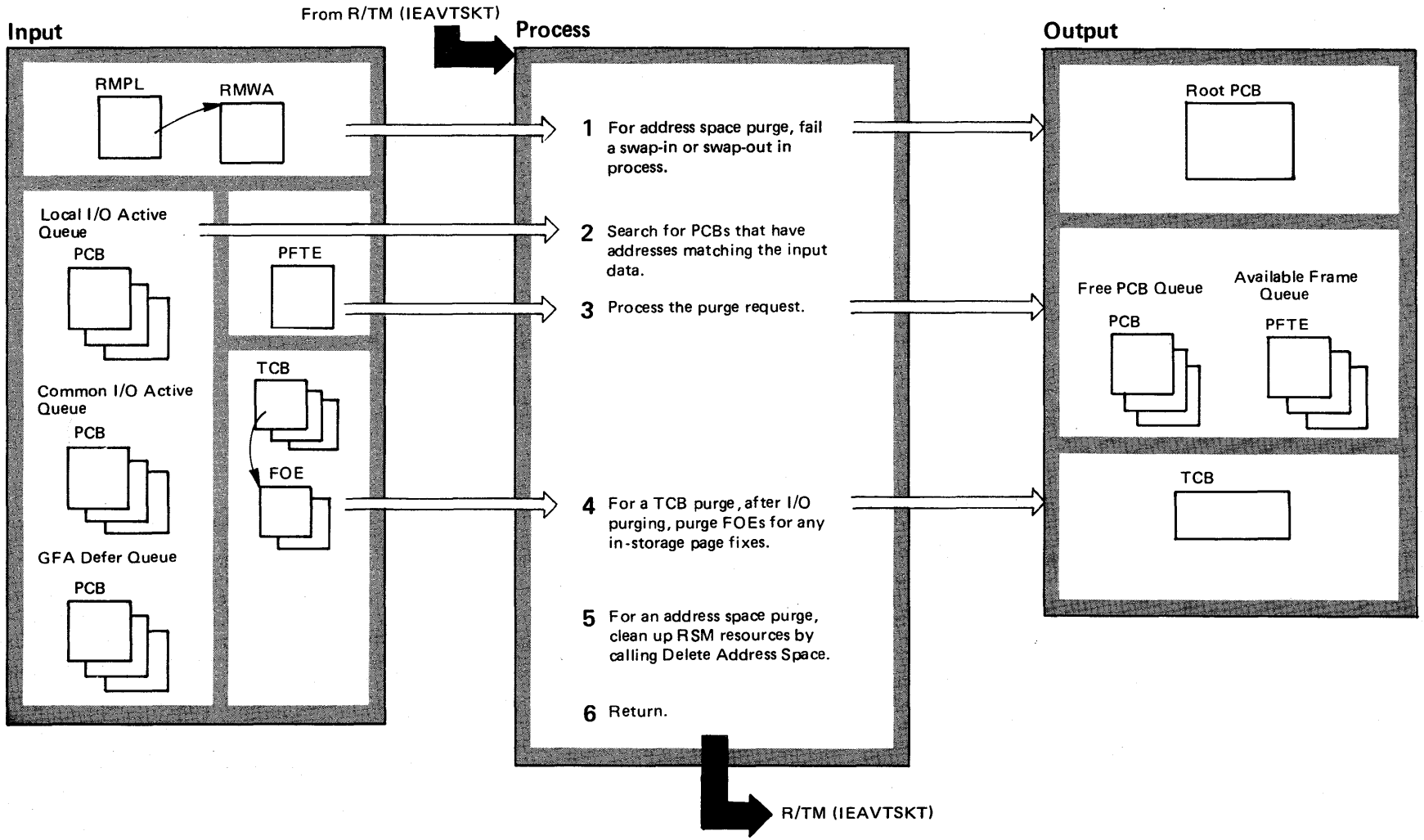


Diagram 23-25. Page Termination Services Routine (IEAVTERM) (Part 2 of 2)

Extended Description	Module	Label
The Page Termination Services (PTS) routine (IEAVTERM) is called by the Recovery/Termination Manager to quiesce paging I/O for an RB or TCB within a virtual address space or for an entire virtual address space. The routine may also free pages fixed by the TCB being quiesced.		
1 PTS gets the SALLOC lock, sets up the RSM FRR, and gets the local lock. PTS terminates any swapping operations. For a swap-in, the RSM-failed flag is turned on. For a swap-out, the SRM parameter list is freed. For an ASCB purge, PTS releases the local lock.	IEAVTERM	IEAVTERM
2 PTS searches for PCBs that have ASCB and TCB or RB addresses matching the input data. It searches the GFA Defer Queue and the I/O active queues. When a PCB is found, PTS processes it according to the queue it is on and the purge type.		
3 When the purge type is ASCB and the SRB mode flag in the PCB is set, PTS will reset the SRB routine if reset has been requested. If the purge type is RB and the RB address in the PCB matches the input RB address, PTS calls the Reset subroutine of PCIH to remove the specified routine from page wait. For all PCBs for which no I/O has started, PTS processes any root PCB and then frees the PCBs. For all PCBs for which I/O is active, PTS flags the PCB to cancel the I/O request. If the I/O is complete, PTS frees the PCB and the PFTE if there is no other requestor for the page. If the purge type is RB, PTS only frees one PCB. During I/O purge processing, if a PFTFXCT is decremented to zero, the system fix counters are decremented by one.		
4 For a TCB purge, PTS frees the FQE for a fix PCB, and, if requested, purges all in-storage fixes and FOEs. If the PFTFXCT is decremented to zero, the system fix counters are decremented by one.		
5 For an address space purge, PTS calls Delete Address Space to clean up the RSM resources and real frames.		
6 PTS returns to the Recovery/Termination Manager.		

Diagram 23-26. Real Frame Replacement (IEAVRFR) (Part 1 of 4)

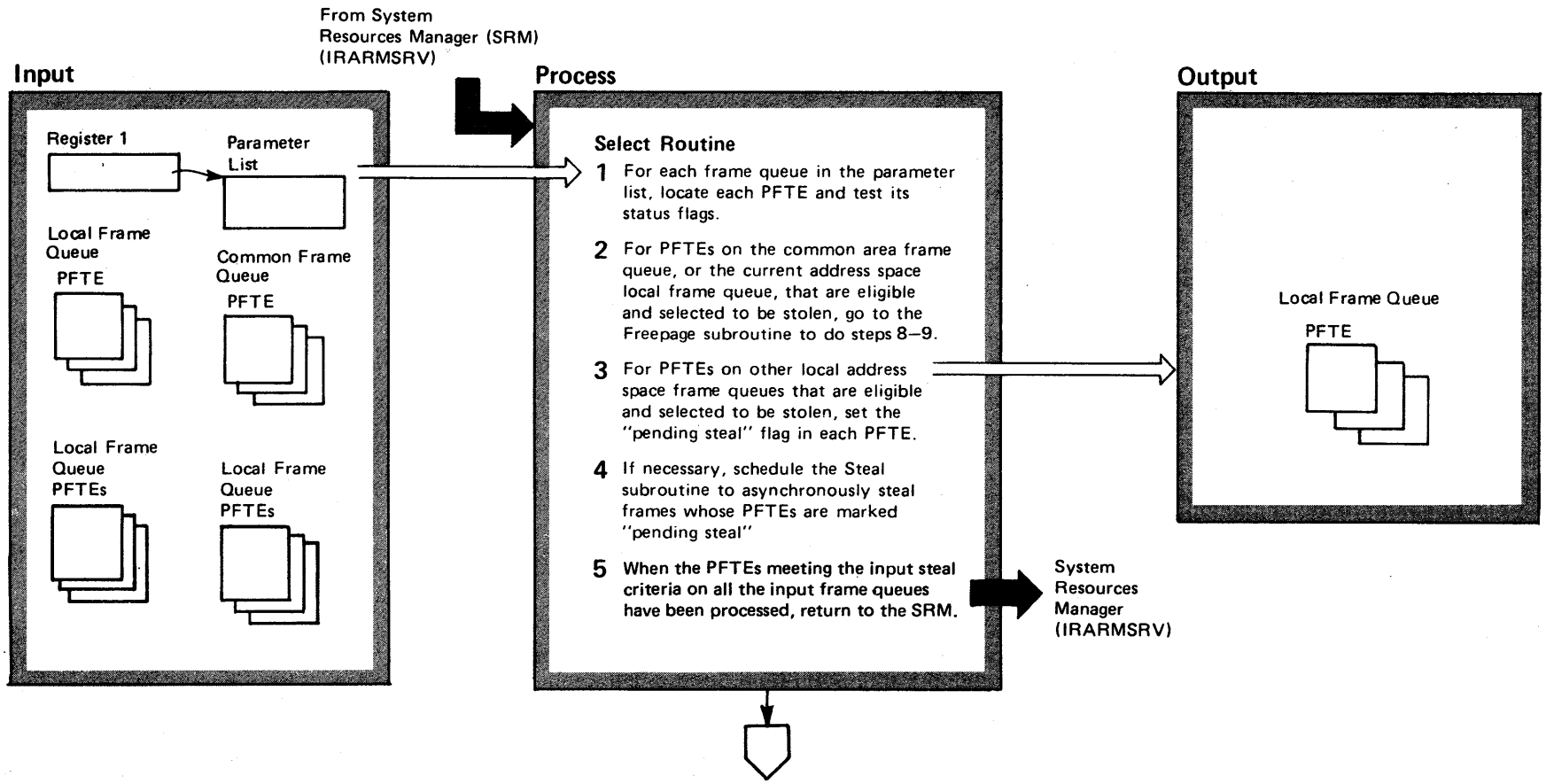


Diagram 23-26. Real Frame Replacement (IEAVRFR) (Part 2 of 4)

Extended Description	Module	Label
Real Frame Replacement (IEAVRFR) scans sets of real frames selected by the System Resources Manager (SRM) to determine if they are available for stealing. It also updates the unreferenced interval count (UIC), when requested. It returns to the SRM the count of stolen frames.	IEAVRFR	IEAVRFR
1-3 Real Frame Replacement (RFR) gets the SALLOC lock. Then, for each entry in the input parameter list, its Select routine accesses the local frame queue (LFQ) for the specific ASCB, or the common frame queue (CFQ) if the ASCB address is zero. The common frame queue contains entries for frames used by areas such as the PLPA, CSA, and MLPA. Frames represented on the local frame queues contain the user private area, excluding LSQA.		
RFR processes each PFTE on a queue, and its associated frame, in one or more of the following ways (as detailed in substeps a–f below):		
• Skips the frame and doesn't steal it. • Increases by one the count of stolen frames. • Resets the frame's usage history by zeroing its unreferenced interval count (UIC). • Increases by one the unreferenced interval count, when requested. • Flags the PFTE for a "pending steal". • Calls the Freepage subroutine (steps 8-9) to steal the frame.		
a) RFR determines that the PFTE is ineligible and doesn't steal the frame, if any of these conditions applies: • The frame is fixed (PFTE fix count is not zero). • The frame has outstanding I/O (PFTE "PCB-defined" flag is set). • The frame is part of a nonpageable region (V=R). • The frame is already flagged as 'pending steal'. • The frame contains a storage error.		
b) If the frame has been referenced (hardware reference bit is on), RFR resets the frame's usage history by zeroing the unreferenced interval count (UIC) in the PFTE. RFR then processes the next PFTE. (The UIC is a count of the number of intervals in which the frame's page has not been referenced.)		
c) When SRM requests that the UICs be updated, PFTUIC is incremented by one for each unreferenced PFTE in the requested queues.		
d) The count of stolen frames is increased by one. If the PFTE belongs to the local frame queues of an address space other than the current one, RFR flags the PFTE for a pending steal, then processes the next PFTE on the queue. (For processing of "pending steal" frames, see step 6.) If the frame has met the steal criteria (described above), RFR calls the Freepage subroutine to invalidate the page and steal the frame. (See steps 8–9.)		
e) When all the eligible frames have been stolen from the queue being examined, as specified in the input parameter list, RFR processes the next frame queue.		
4 The Steal routine is scheduled via an SRB to be run in the address space specified by the ASCB address, if a non-current local frame queue has PFTEs marked "pending steal". (These PFTEs were flagged in substep d, above.)		
5 The count of stolen frames is placed in the count field of the parameter list entry, for use by the SRM. After the entire parameter list has been processed, the Select routine releases the SALLOC lock and exits to the SRM.		

Diagram 23-26. Real Frame Replacement (IEAVRFR) (Part 3 of 4)

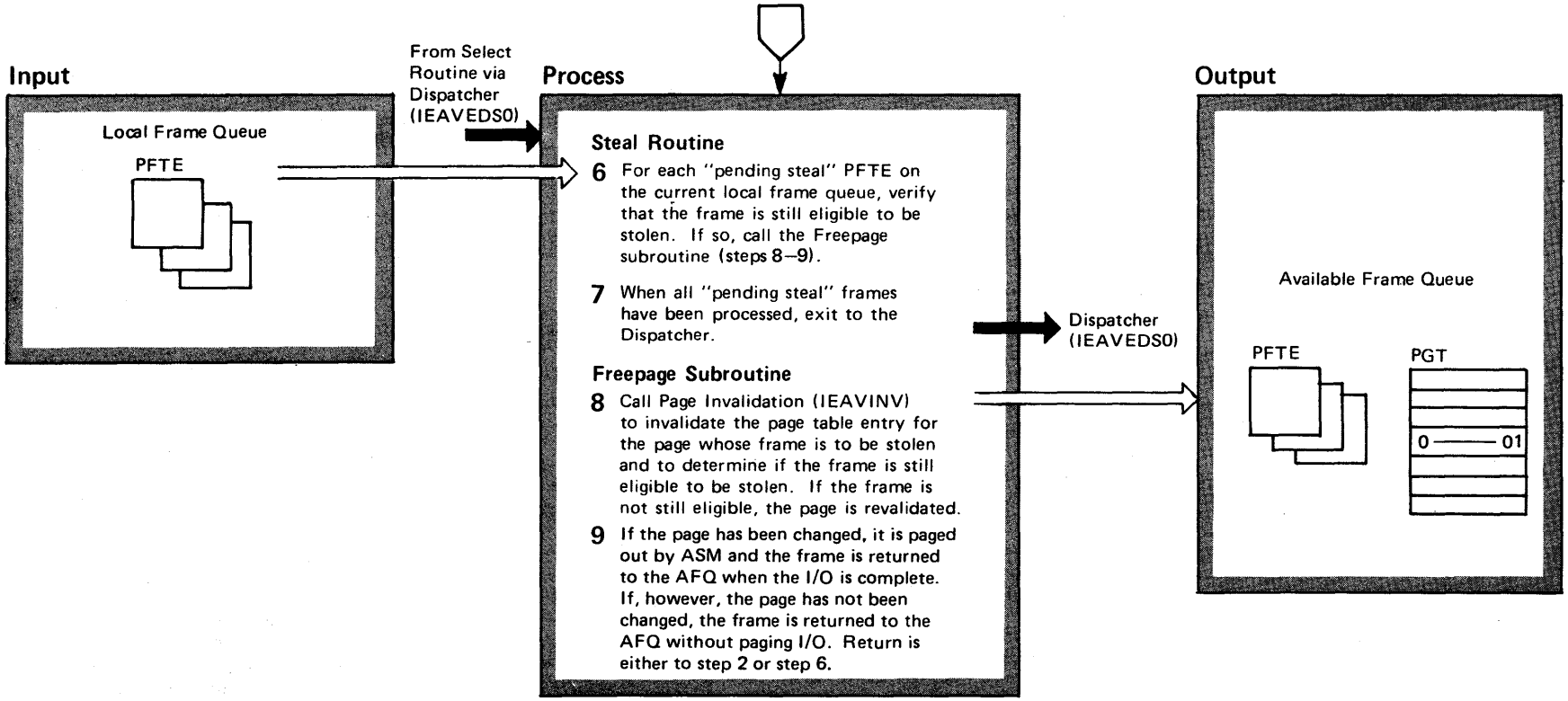


Diagram 23-26. Real Frame Replacement (IEAVRFR) (Part 4 of 4)

Extended Description	Module	Label
6 The Steal routine gets the SALLOC lock, frees the input SRB, then processes each PFTE marked "pending steal" on the local frame queue. It checks the "PCB defined" flag, the "storage error" flag, and the fix count in the PFTE. If any of these are set, the frame cannot be stolen. Steal turns off the steal indicators and gets the next PFTE. Otherwise, Steal calls the Freepage subroutine (steps 8–9).		IEAVRFRA
7 When it has processed all "pending steal" PFTEs, the Steal routine releases the SALLOC lock and returns, via the Dispatcher, to step 5.		
8 The Freepage subroutine invalidates the page by calling IEAVINV. Freepage tests the reference and change bits to ensure that no reference has taken place since the decision to steal the frame. If the page has been referenced, it is revalidated, and the PFTE steal indicators are reset.		FREEPAGE
9 If the page has been changed but not referenced, Freepage calls ASM to write out the page to a paging data set, and returns the frame's PFTE to the available frame queue (AFQ) when the I/O completes. If, however, the page has not been changed, Freepage returns the PFTE to the AFQ without any paging I/O.		

Diagram 23-27. Real Storage Reconfiguration Routine (IEAVRCF) (Part 1 of 4)

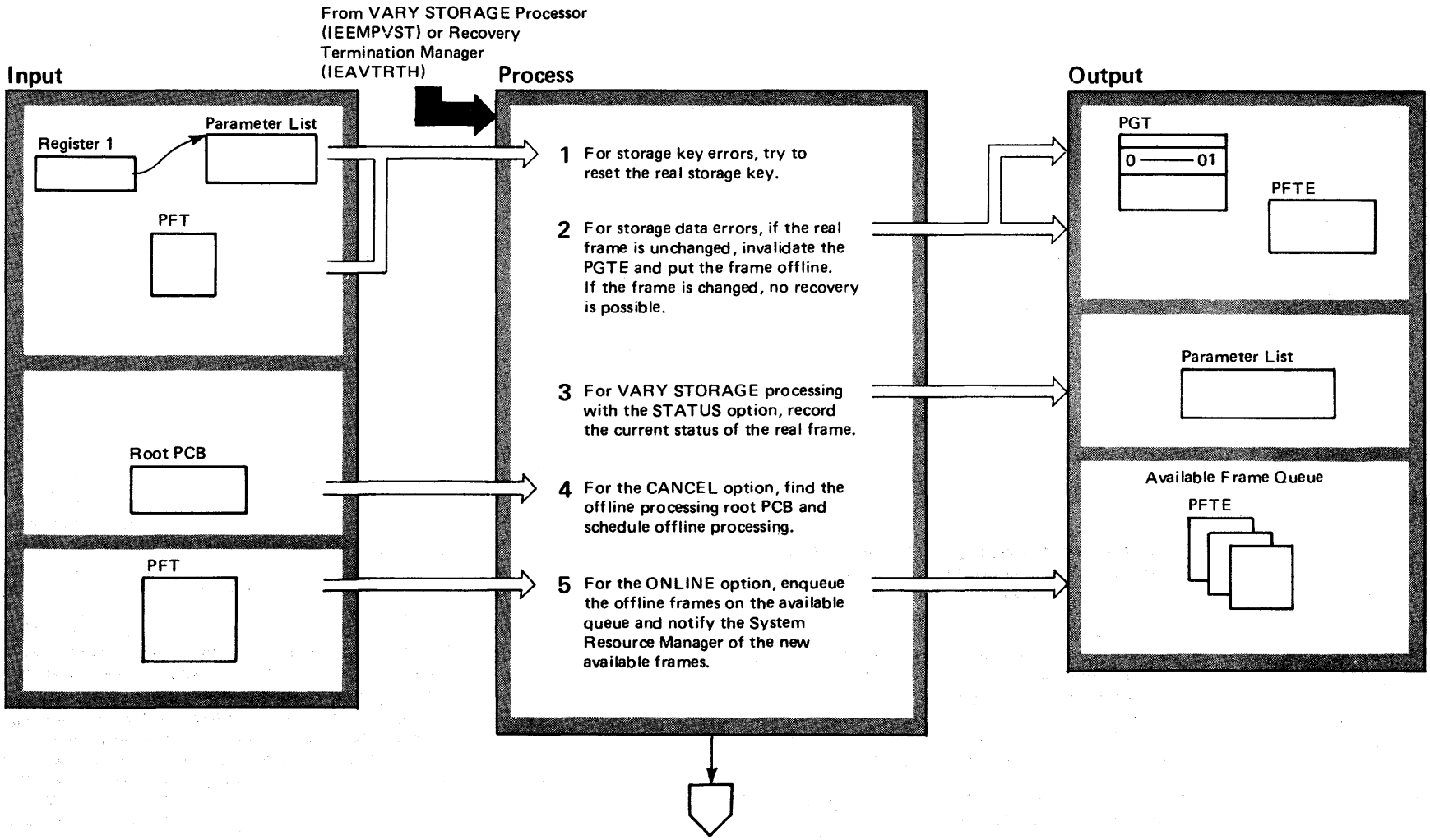


Diagram 23-27. Real Storage Reconfiguration Routine (IEAVRCF) (Part 2 of 4)

Extended Description	Module	Label
The Real Storage Reconfiguration (RSR) routine (IEAVRCF) adds to or subtracts from the real storage frames currently available for use by the system. When entered for cancel processing, RSR automatically follows with offline processing.		
1 After obtaining the SALLOC lock, RSR checks the option field in the parameter list for the option requested. If entered for a storage key error, RSR calls the Reset Storage Key routine to reset the key of any frame for which it has key information. It also sets the change flag in the frame. If the error is not recovered, RSR sets error flags in the PFTE and puts a return code of 8 in register 15.	IEAVRCF	
2 If entered for a storage data error, RSR sets error flags in the PFTE. If the frame is unchanged and pageable, RSR invalidates the PGTE and puts the PFTE on the available frame queue. If the frame contains changed, LSQA, or fixed data , RSR sets the pending-status indicator and sets a return code of 8 indicating no recovery is possible.		
3 If entered by VARY STORAGE for status processing, RSR records status information in the status list.		
4 If entered by VARY STORAGE for cancel processing, RSR searches for the offline-processing root PCB and dequeues it. RSR also posts the ECB with a code of 4.		
5 If entered for online processing, RSR sets the online flag in each PFTE and puts the PFTEs on the available queue. Then it notifies SRM of the new available frames.		

Diagram 23-27. Real Storage Reconfiguration Routine (IEAVRCF) (Part 3 of 4)

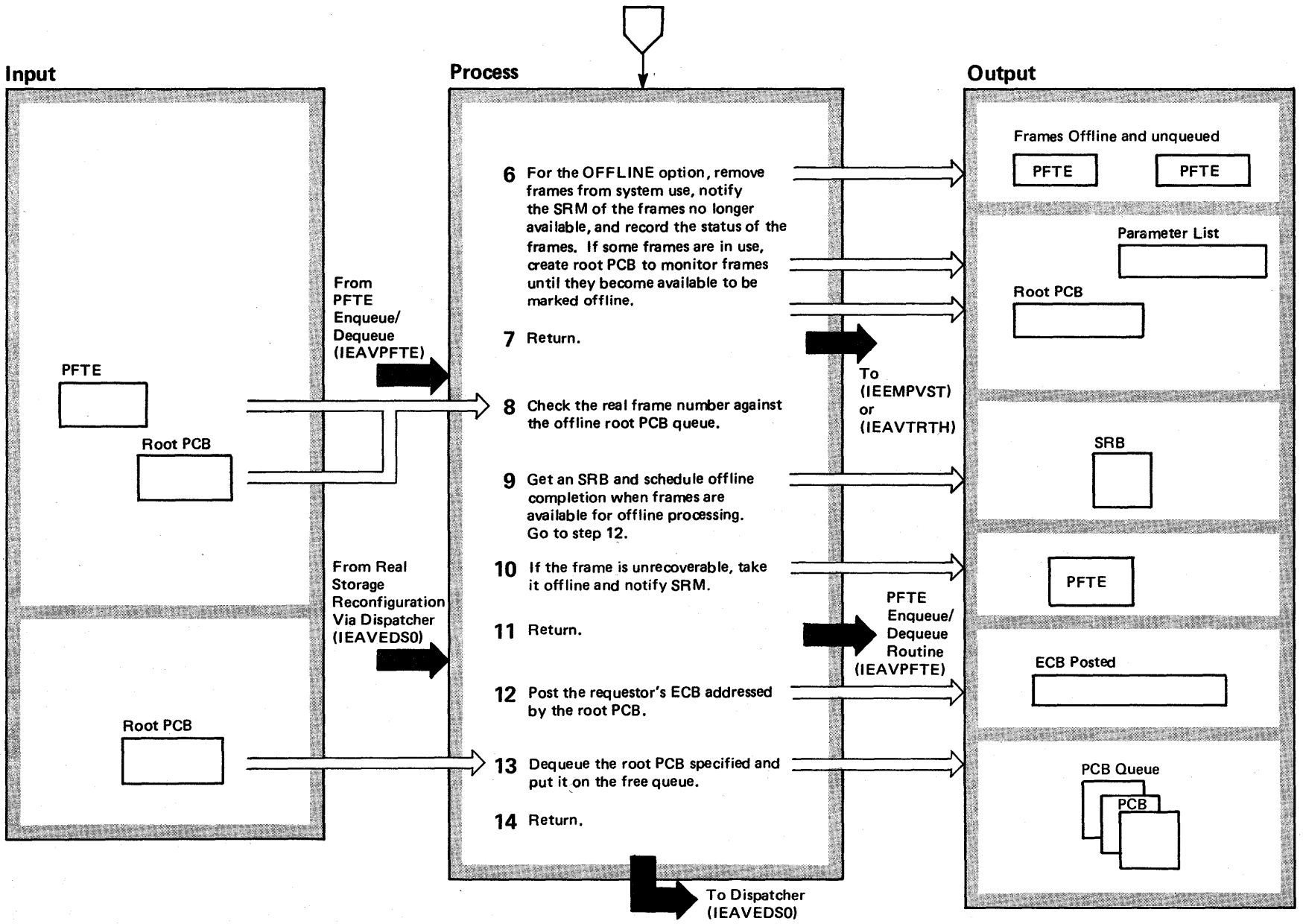


Diagram 23-27. Real Storage Reconfiguration Routine (IEAVRCF) (Part 4 of 4)

Extended Description	Module	Label
6 If entered for offline processing, RSR checks to see if the frame is in use. If it is not, RSR sets the offline flag in the PFTE and removes it from the queue it resides on. If the frame is in use, RSR sets the offline-intercept flag in the PFTE, sets condition indicators in the frame's status byte, and builds a root PCB to monitor the request.		
7 RSR returns control to the caller, passing a return code in register 15.		
8 RSR is entered from PFTE Enqueue/Dequeue when a frame with the PFTE offline-intercept flag set is sent to the available frame queue. RSR searches the offline wait queue for the corresponding root PCB. If the frame is accepted for offline processing, RSR resets the offline-intercept flag in the PFTE and decreases by one the PCB count in the root PCB.	IEAVRCF	IEARCFI
9 When the PCB count in the root PCB becomes zero, RSR schedules a POST of the requestor's ECB. (A GETCELL was done early in RSR for an SRB area for this purpose.)		
10 If the frame has a storage error, RSR removes the PFTE immediately, marks it offline, and notifies SRM of the decrease in available frames.		
11 When all offline-intercept frames are processed, RSR returns control to PFTE Enqueue/Dequeue.		
12 RSR offline completion is scheduled by the offline-intercept subroutine of RSR when the frame count in the root PCB becomes zero. RSR finds the corresponding root PCB and posts the ECB specified in the root PCB.	IEAVRCF	IEARCFC
13 RSR then dequeues the PCB and frees the quickcell used for the SRB.		
14 RSR returns control to the Dispatcher.		

Diagram 23-28. PFTE Enqueue/Dequeue Routine (IEAVPFTE) (Part 1 of 2)

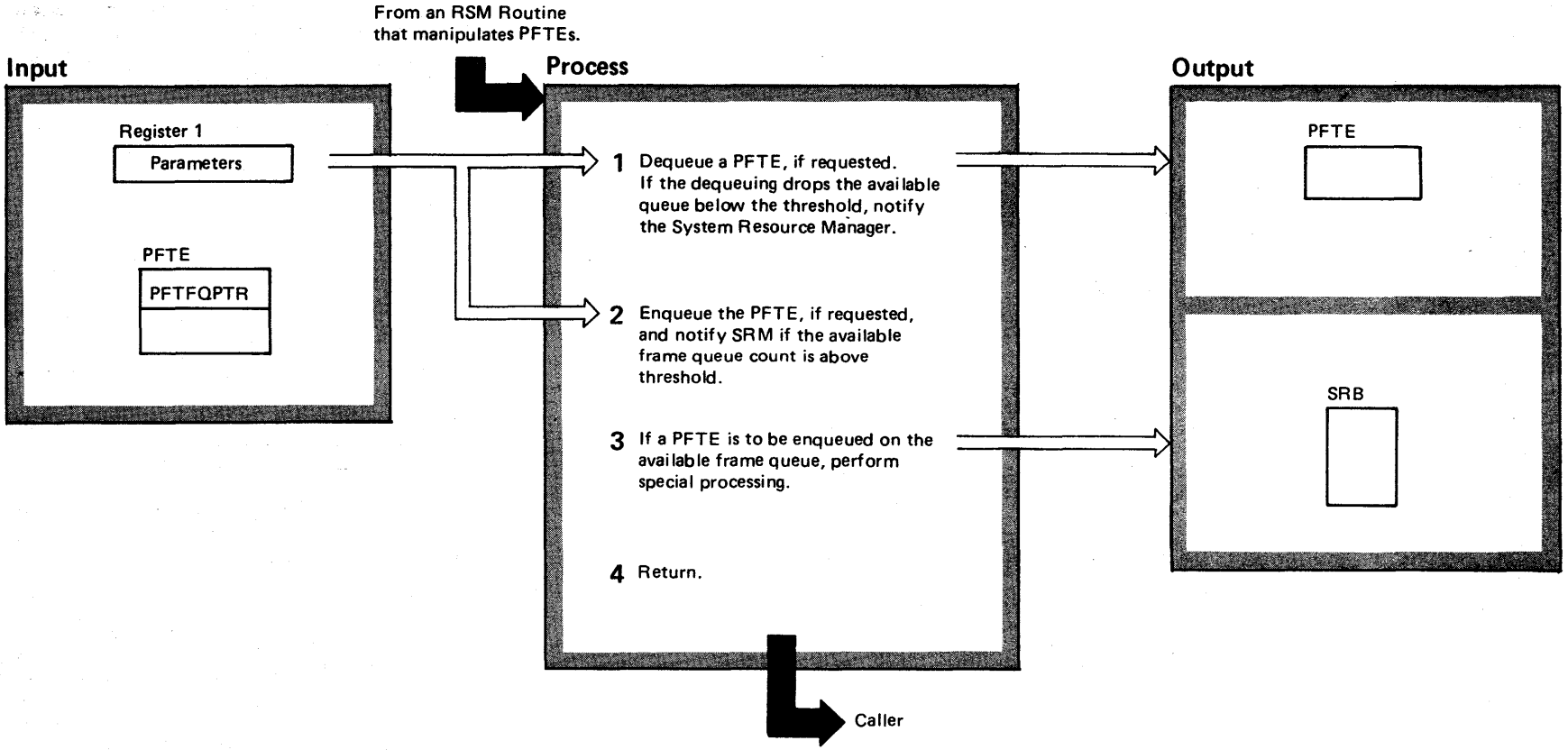


Diagram 23-28. PFTE Enqueue/Dequeue Routine (IEAVPFTE) (Part 2 of 2)

Extended Description	Module	Label
The PFTE Enqueue/Dequeue routine (IEAVPFTE) enqueues a PFTE (page frame table entry) at the end or the front of a specified queue, dequeues a PFTE from a specified queue, or moves a PFTE from one queue to another. The routine also intercepts PFTEs routed for the available frame queue (AFQ) and directs them to special queues or other RSM functions requiring the real storage frame represented by the PFTE. The caller holds the SALLOC lock.		
IEAVPFTE is responsible for increasing and decreasing the allocated frame count for each address space (ASCBFMCT) and the common area (PVTCFMCT). It will also compute the "page-seconds" information for an address space. Page-seconds are the total CPU time in milliseconds that each frame has used by an address space. Page-seconds are recomputed before each change of the local frame count; that is, ASCBFMCT is increased or decreased.		
1 The routine first checks for a dequeue request. If the PFTE is on a queue, the specified PFTE is dequeued and the QID field in the PFTE is set. If the PFTE was dequeued from the AFQ, special processing is done. The AFC (available frame count) in the PVT is decreased and the PFTONAVQ flag is turned off. If the AFQ is now below its safe threshold and a SYSEVENT has not been issued, one is issued to notify the System Resource Manager (SRM) of the low AFC. If the AFC is zero, a special SYSEVENT is issued to notify the SRM of the zero AFC.	IEAVPFTE	IEAVPFTE
When dequeuing the PFTE from a local frame queue, page-seconds are computed and ASCBFMCT is decremented by one. If the PFTE is dequeued from the common frame queue, PVTCFMCT is decremented.		
2 The routine tests to see if an enqueue operation is requested. If the PFTE is not to be enqueued, IEAVPFTE returns to the caller. If the TQID is the available frame queue ID, IEAVPFTE checks to see if the PFTE has been intercepted. If the V=R intercept flag is set, the V=R Intercept subroutine of V=R allocation (IEAVEQR) is called, passing the RBN of the PFTE. This subroutine returns either a zero RBN or the RBN of the PFTE passed by the intercept processor. If a zero RBN is returned, the frame is intercepted for V=R Allocation; IEAVPFTE returns control to the caller. If the returned RBN is not zero, or if the V=R intercept flag was not set, the PFTE offline intercept flag is tested. If the offline intercept flag is set, the PFTE is passed to Real Storage Reconfiguration (IEAVRCF). If the Reconfiguration routine returns a zero RBN, the frame has been intercepted for offline or is a bad page; IEAVPFTE returns control to the caller. If the Reconfiguration routine returns the input RBN, or if V=R Allocation returned the RBN and the PFTE offline intercept flag is not set, processing continues. This also occurs if no intercept flags are set in the PFTE. IEAVPFTE then checks to see if the SQA Reserve Queue requires frames. If so, the PFTE is diverted to the SQA Reserve Queue. If not intercepted, IEAVPFTE enqueues the PFTE on the AFQ. If the input TQID was not X'FF' and was not the AFQ ID, then IEAVPFTE puts the TQID in the PFTE and enqueues on it on the end of the specified queue. If the TQID is X'00' and the special "head of queue" flag is set, the PFTE is placed at the head of the AFQ. If the PFTE is to be enqueued to a local frame queue, page-seconds are computed and ASCBFMCT is incremented. If the PFTE is to be enqueued to the common frame queue, PVTCFMCT is incremented. Then IEAVPFTE returns to the caller.		
3 If the PFTE is to be queued on the Available Frame Queue (AFQ), IEAVPFTE sets to zero the storage keys on the real frame, enqueues the PFTE, and increases the available frame count (AFC) in the PVT. If a low-threshold violation is outstanding, IEAVPFTE checks to see if the new AFC is equal to or greater than the safe threshold. If it is, IEAVPFTE notifies the SRM that the AFC is sufficient. Next, IEAVPFTE tests the GFA defer queue. If there is a PCB for which defer processing has not been scheduled, and whose address space does not have defer processing scheduled, then IEAVPFTE schedules GFA defer processing with an SRB. Then IEAVPFTE returns to the caller.		

Diagram 23-29. PCB Manager (IEAVPCB) (Part 1 of 2)

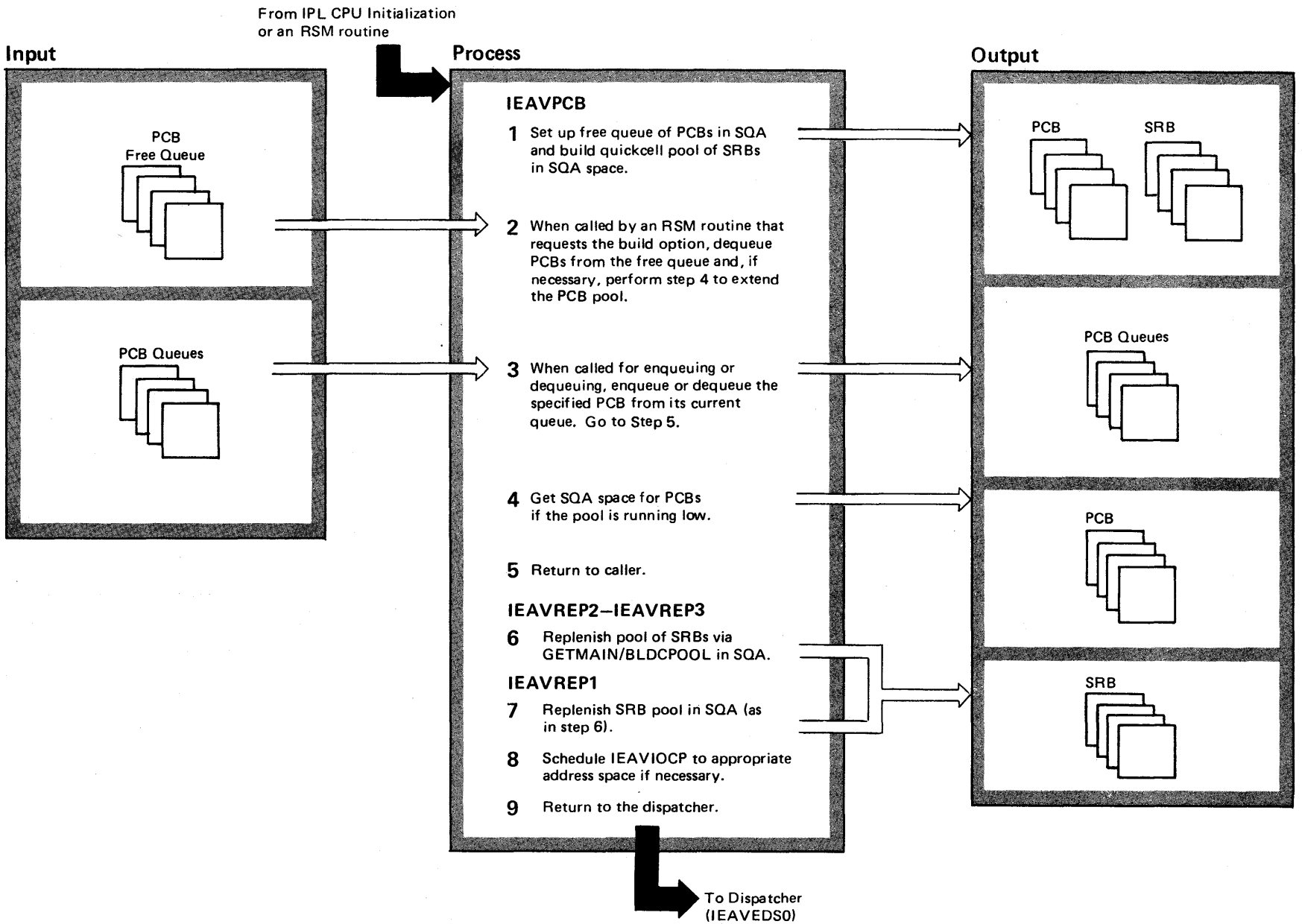


Diagram 23-29. PCB Manager (IEAVPCB) (Part 2 of 2)

Extended Description	Module	Label
The PCB Manager (IEAVPCB) obtains PCBs (page control blocks) from the PCB free queue, dequeues, enqueues, and moves PCBs. In addition, the routine attempts to maintain a minimum number of PCBs on the free queue by replenishing the queue. Entries IEAVREP1, IEAVREP2, and IEAVREP3 are used by RSM routines to replenish the SRB pool.		
1 IEAVPCB checks the input PCB address. If it is zero, the caller requests the build option. IEAVPCB checks for zero PCBs requested. If zero PCBs are requested and IEAVNIP0 is the caller, IEAVPCB builds a pool of PCBs in SQA. An internal routine, IEAVREP3, is invoked to build and initialize (BLDCPOOL) a pool of SRBs in SQA. Step 5 is then performed. If zero PCBs are requested and IEAVNIP0 is not the caller, a return code of 4 is passed to the caller. If it is not a zero PCB request, IEAVPCB checks to make sure there are enough PCBs on the free queue to satisfy the request. If there are not enough PCBs on the free queue, IEAVPCB expands the pool if possible, unless the GETMAIN-Inhibit flag is set. If the entire request cannot be satisfied, IEAVPCB returns control to the caller with a return code of 4.	IEAVPCB	IEAVPCB
If there are enough PCBs on the free queue to satisfy a request or if enough have now been obtained, the specified number of PCBs are all removed at one time; this prevents loss of the chain pointers for PCB requests greater than one. The number of PCBs dequeued is subtracted from the free queue depth. If the new depth value is below the free queue threshold, the GETMAIN inhibit bit is tested. If this flag is set, return is made to the caller. If the GETMAIN inhibit bit is not set, the PCB Replenish routine is called. In either case, return is made to the caller with a zero return code. For requests of more than one PCB, the PCBs are chained together using the standard chain pointers. All PCBs obtained for the caller will be set to zero, except the chain pointer fields and the queue number fields which are set to X'FF', and the AIAUSER1 field, which points to the PCB Address.	IEAVPCB	IEAVREP3
2 For a build request, IEAVPCB obtains SQA space for a calculated number of PCBs and for an equal number of SRBs.		
3 If the input PCB address is not zero, IEAVPCB performs dequeuing or enqueuing based on what the PQN and TQN parameters specify. If dequeued, the PCB queue number field is set to X'FF'. For enqueuing, the PCB is placed at the end of the specified queue. Then IEAVPCB returns control to the caller with a return code of 0.		
4 IEAVPCB clears the SQA space to zero and constructs the PCBs required. The PCBs are enqueued on the free PCB queue and the queue depth is updated.		
5 Return to the caller.		
6 Entry IEAVREP2 is branch entered by RSM routines requiring an SRB when the SRB pool has been depleted. IEAVREP2 will replenish the pool.		
7 Entry IEAVREP1 is entered via SRB scheduled by IEAVPIOP or IEAVREP1 itself to replenish the SRB pool.		
8 IEAVREP1 scans the RSMHD in each address space to determine if IEAVIOCP should be scheduled. When this is necessary, an SRB is obtained from the pool (via GETCELL) and IEAVREP1 is scheduled. If the GETCELL fails, IEAVREP1 schedules itself (using an SRB in the PVT) to replenish the pool.		

Diagram 23-30. Page Invalidation Routine (IEAVINV) (Part 1 of 2)

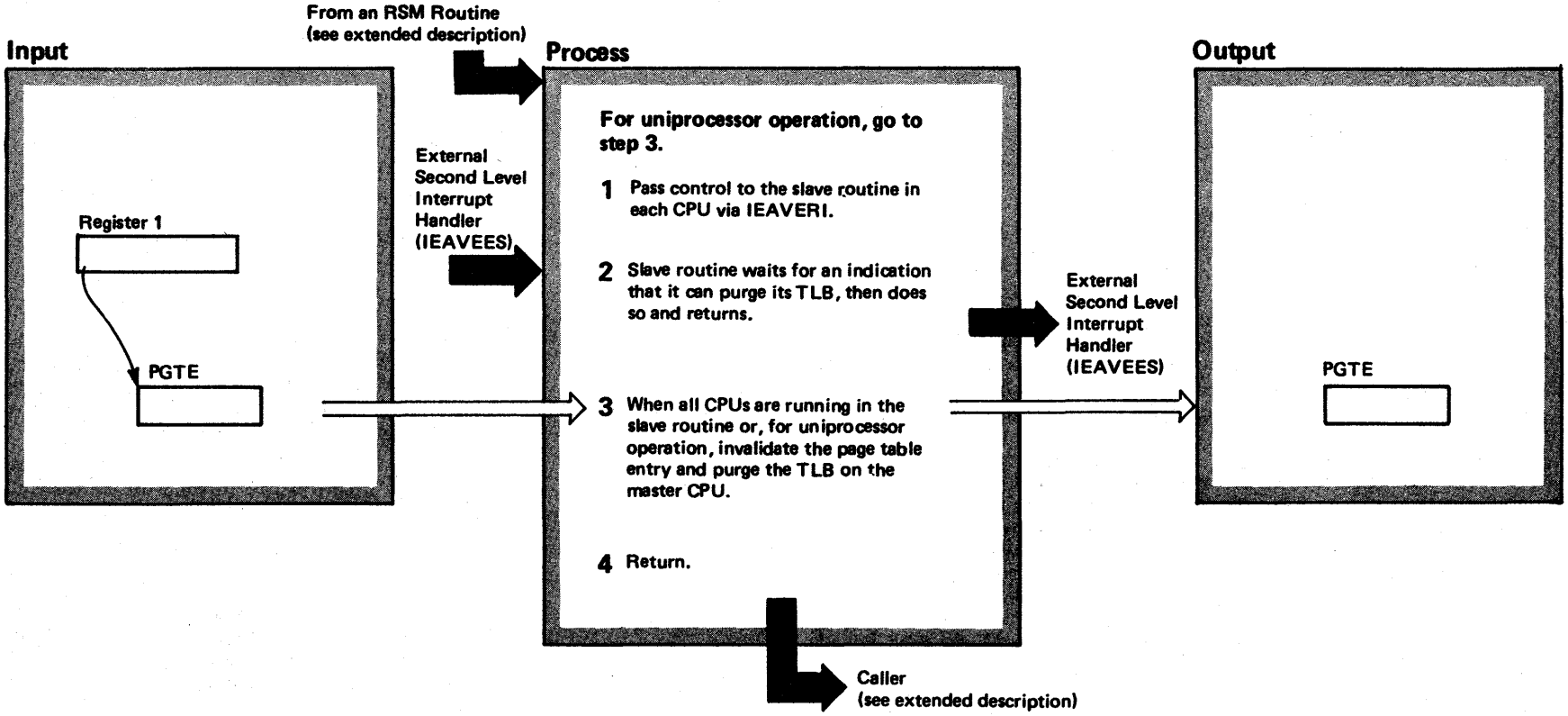


Diagram 23-30. Page Invalidation Routine (IEAVINV) (Part 2 of 2)

Extended Description	Module	Label
The Page Invalidation routine (IEAVINV) performs all necessary interprocessor synchronization, sets the page table entry invalid bit, and purges the translation lookaside buffer on every processing unit in the system. The main routine must be entered with the SALLOC lock. This routine is entered from the following RSM routines: IEAVRFR, IEAVRCF, IEAVOUT, IEAVPIOI, IEAVDLAS, and IEAVGFA.		
1 When executing in an MP environment, IEAVINV sets its internal indicator to zero and signals all other processors in the system to execute the slave subroutine.	IEAVINV	IEAVINV
2 The called (slave) subroutine of IEAVINV executing on the other processor sets the global spin indicator in the LCCA and waits for the internal indicator to be set to X'FF'. While waiting, the slave subroutine allows intermittent emergency signals and malfunction alert signals. When the slave subroutine finds X'FF' in the internal indicator, it purges its translation lookaside buffer, resets its global spin indicator, and returns.		IEAVINVA
3 When all other processors are in the slave subroutine, or when only one processor is online, IEAVINV sets the PGTE invalid bit to one, sets the internal indicator to X'FF', and purges its translation lookaside buffer. If the PGTE address is zero, no invalidation occurs but the other operations take place. Then IEAVINV returns control to the caller.		

Diagram 23-31. Find Page Routine (IEAVFP) (Part 1 of 2)

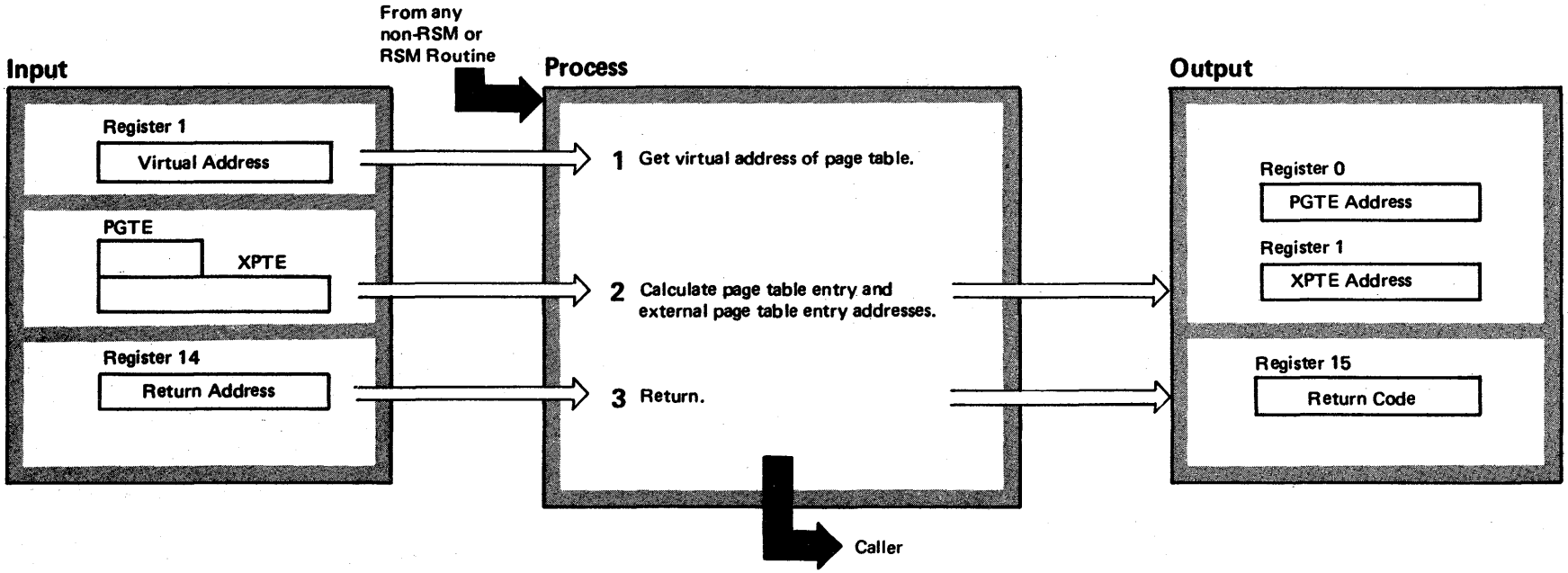


Diagram 23-31. Find Page Routine (IEAVFP) (Part 2 of 2)

Extended Description	Module	Label
The Find Page routine (IEAVFP) locates the page table entry (PGTE) and/or external page table entry (XPTE) corresponding to virtual address.		
1 Find Page gets the virtual address of the page table by translating the real address obtained from the segment table entry referenced by the virtual storage address. If the segment referenced is invalid, Find Page returns control to the caller with a return code of 4 in register 15.	IEAVFP	IEAVFP
2 Find Page calculates the PGTE and XPTE addresses from the virtual address of the page table and the page number obtained from the virtual address.		
3 When the calculation is complete, Find Page returns control to the caller.		

Diagram 23-32. Translate Real to Virtual Routine (IEAVTRV) (Part 1 of 2)

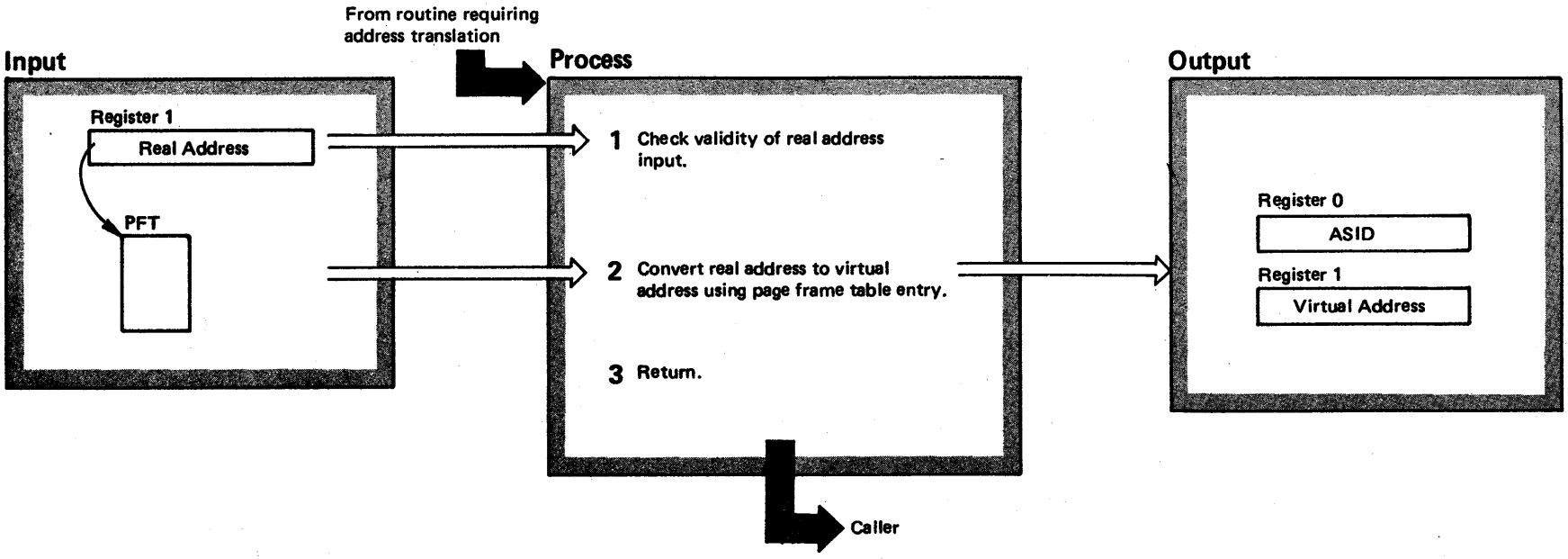


Diagram 23-32. Translate Real to Virtual Routine (IEAVTRV) (Part 2 of 2)

Extended Description	Module	Label
The Real to Virtual Translation routine (IEAVTRV) provides the virtual storage address and address space ID for an input real storage address. No locks are required.		
1 Translation of real to virtual checks the real storage address input. If the real address exceeds the boundaries of real storage, Translation returns to the caller with a return code of 4 in register 15.	IEAVTRV	IEAVTRV
2 If the real address is in the nucleus, Translation leaves the input address unchanged and sets register 0, the ASID, to X'FFFF' to indicate common area storage. If the address is not in the nucleus, Translation uses the input real address to find the page frame table entry; it then locates the virtual address and address space ID associated with the PFTE. If the frame is invalid, or on the available queue, or offline, or being used by VIO, Translation returns a code of 4 in register 15, signifying unsuccessful translation.		
3 Translation returns to the caller with a code of 0 in register 15 if translation is successful.		

Diagram 23-33. RSM Functional Recovery Routine (IEAVRCV) (Part 1 of 2)

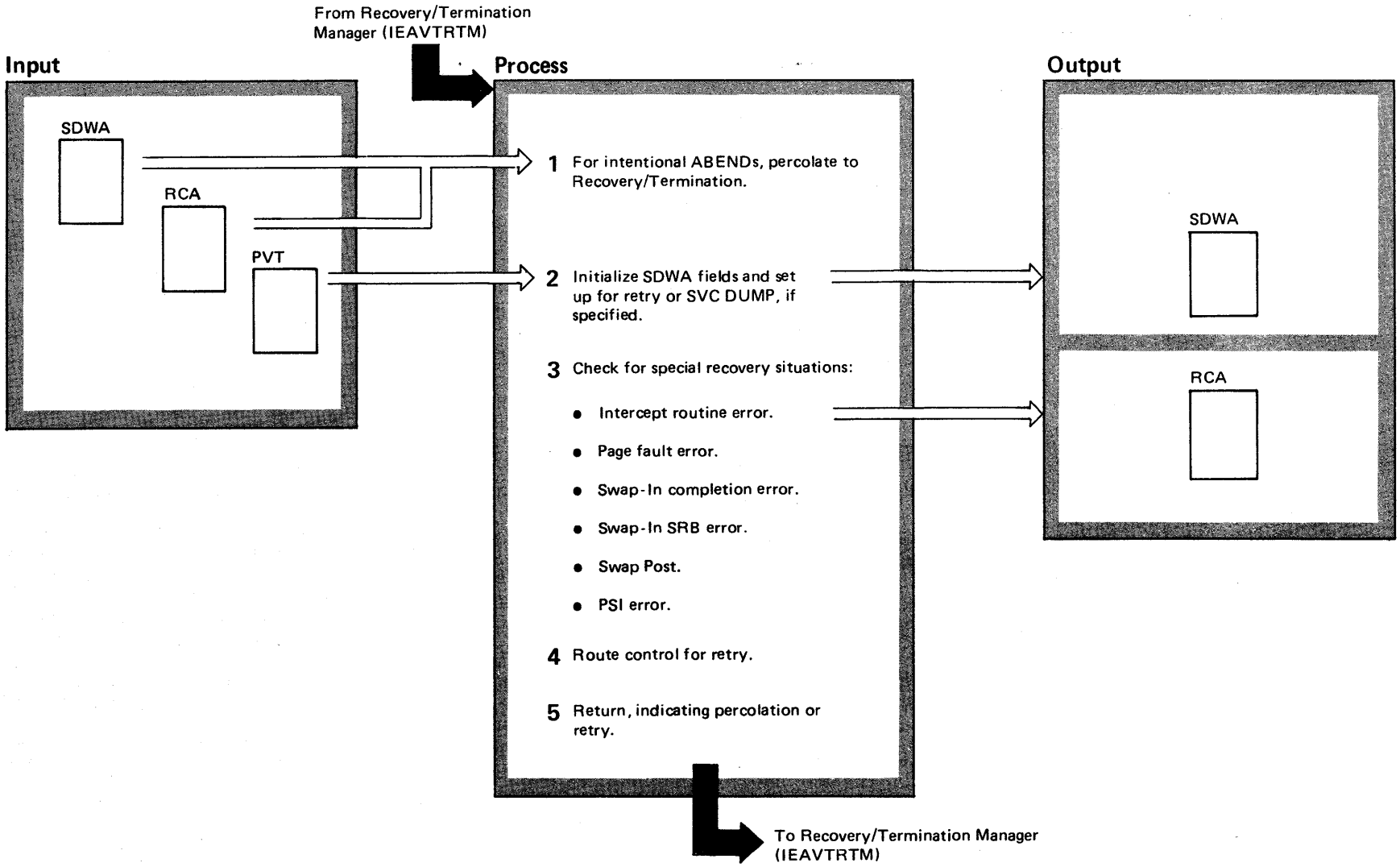


Diagram 23-33. RSM Functional Recovery Routine (IEAVRCV) (Part 2 of 2)

Extended Description	Module	Label
The RSM Functional Recovery Routine (IEAVRCV) provides three services: • The recording of software errors in RSM modules. • The clean-up of locks and deletion of the FRR for intentional ABEND situations. • The handling of unexpected errors by dumping, recording, releasing locks, and attempting to contain the effects of the error.		
1 After setting up a recovery FRR, the FRR checks the RCAABEND flag for an intentional ABEND situation. If one exists, the FRR releases any locks gotten by the RSM function, deletes the recovery FRR, indicates "continue with termination" and returns control to R/TM.	IEAVRCV	IEAVRCV
2 The FRR verifies the PVT pointer in the CVT. Next, the FRR checks the SDWA to see if the error was percolated. If so, the FRR does no recording. If not, the FRR sets fields in the SDWA to prepare for recording of the error. If the error was a non-percolated X'C0D', the FRR sets up for retry after calling for a dump. The FRR returns control to the point immediately following the X'C0D' ABEND that called it.		
3 If the V=R Intercept or Reconfiguration Intercept routine (IEAVEQRI or IEAVRCFI) was running, the FRR indicates retry at the point addressed by register 14. If RSM is providing second-level interruption handling for a page-fault program check, the FRR retries at an address within IEAVRCV, using RCAPARMI as a guide for processing: if RCAPARMI is zero, the FRR sets the return code to 12 in register 15; if it is not zero, the retry routine sets the return code to zero to allow any paging I/O in progress to complete. Then the retry routine frees the SALLOC lock, deletes the RSM FRR and passes control to the Program Check Interrupt Handler (IEAVEPC). If the Swap-in Root Exit is executing, the FRR abnormally terminates the swapped-in address space and attempts retry at the address saved from register 14. If the Swap-in SRB is executing, the FRR abnormally terminates the swapped-in address space and attempts to retry at the Dispatcher entry point used for SRB exits. If the Swap-In-Post is executing and the RCARETAD is non-zero, the FRR releases the SALLOC lock, if held, and attempts retry at the address specified in RCARETAD. If RCARETAD is zero, the FRR performs the same processing described in paragraph 5. If IEAVPSI has suffered a program check because of bad input parameters, the FRR changes the ABEND code to X'17' and indicates "continue with termination".		
4 If none of the special situations apply, the FRR checks to see if retry will be handled by internal RSM routines. If so, and if RCARETAD is non-zero, the FRR attempts retry at the address specified in RCARETAD.		
5 The FRR performs lock clean-up and deletion of its own FRR. Then it returns control to R/TM, passing a percolate or retry return code previously set.		

Diagram 23-34. RSM Preferred Area Steal (IEAVPREF) (Part 1 of 2)

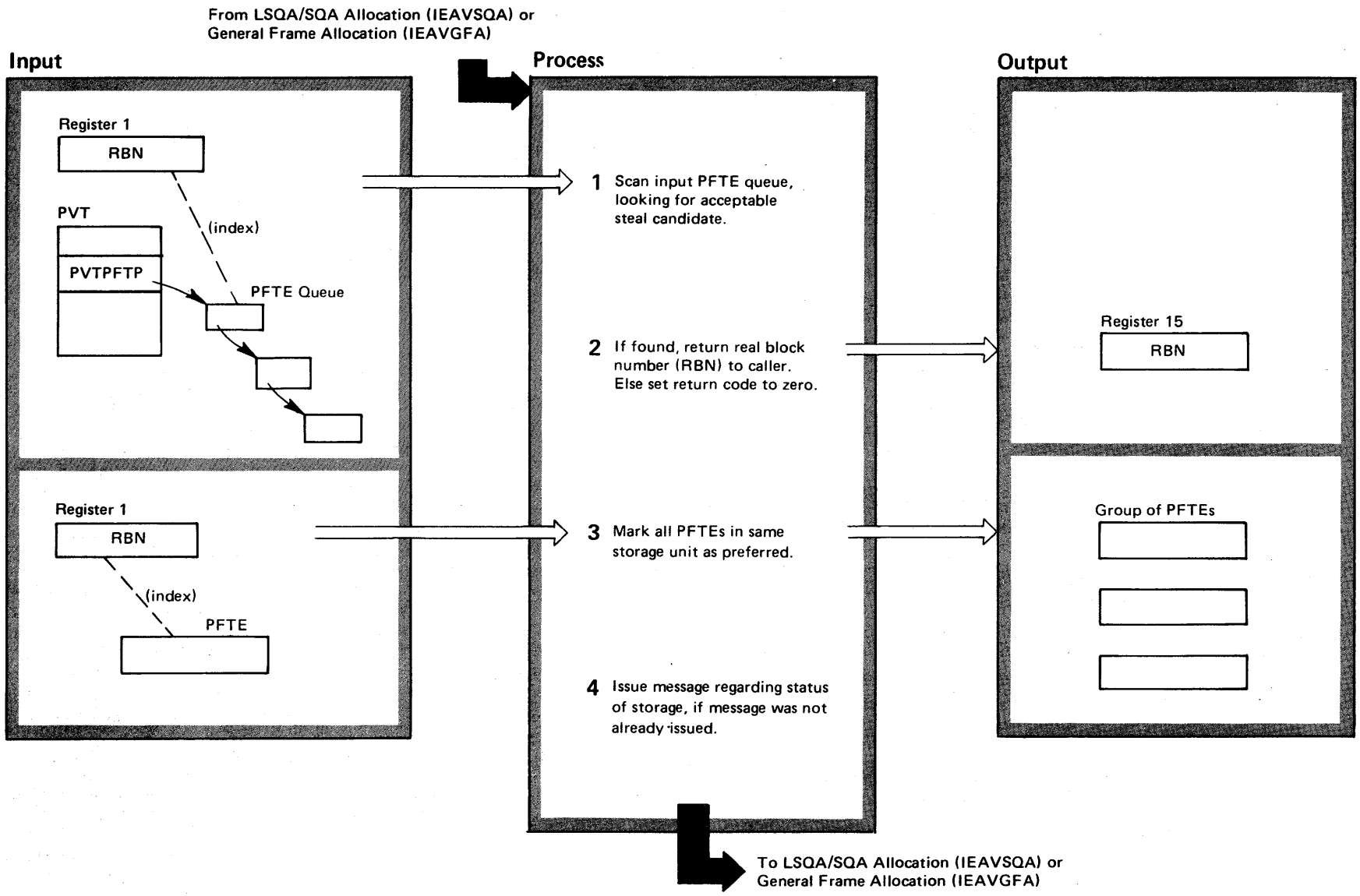


Diagram 23-34. RSM Preferred Area Steal (IEAVPREF) (Part 2 of 2)

Extended Description	Module	Label
The preferred area steal routine runs as a subroutine of IEAVGFA or IEAVSQA. Its purpose is to either scan an input frame queue to select a preferred area frame to steal or convert the status of a storage unit from non-preferred to preferred storage.		
1 If the call is to steal, the input RBN is the first RBN on the frame queue to be searched. Each frame is examined until one is found which meets the criteria for the steal.	IEAVPREF	IEAVPREF
2 If a suitable frame to be stolen is found, its RBN is returned to the caller. Otherwise, sets zero return code to indicate that the specified real block number could not be found.		
3 If the call is to convert, the input RBN is used to identify the physical storage unit to be converted from non-preferred to preferred storage. Every PFTE in the unit is updated by turning on the PFTPREF flag to indicate to RSM that the preferred area has been expanded to include these frames.		
4 If message IEA988I has not been issued, it is written to the operator, and a flag is set indicating that the message has been written. Control returns to the caller.		

Virtual storage is the name given to the entire span of addresses available on a System/370 system with the dynamic address translation feature enabled. The size of virtual storage is equal to the size of real storage when the system is operating with the dynamic address translation feature disabled. When the system is in extended control mode, with the dynamic address translation feature enabled, the size of virtual storage is limited only by the addressing capability of the system, not by the size of real storage.

Like other system resources, virtual storage can be shared by many system users. Consequently, the allocation of virtual storage must be supervised. Space must be allocated to a user when it is needed and freed when it is no longer needed. The supervisor routines that control the allocation and release of virtual storage are referred to as VSM (virtual storage management) routines.

The VSM routines service two macro instructions: GETMAIN (used to allocate storage) and FREEMAIN (used to release previously allocated storage). When executed, each macro instruction results in an SVC interruption and passage of control to the appropriate VSM routines.

Requests for allocation of virtual storage are serviced by the GETMAIN routines. These routines service all requests for virtual storage, including requests for a new region, space within an existing region, space within a system queue area, space within a local system queue area. The GETMAIN routines create, reference, and continually update queues of control blocks to determine whether a request for storage can be satisfied, and from where the storage is to be allocated. The GETMAIN routines pass the address of the allocated area to the requesting routine.

Requests to free virtual storage are serviced by FREEMAIN routines. These routines update control block queues to relfect the release of previously allocated space, thereby making the space available for reallocation. The FREEMAIN routines service all requests to free virtual storage, including requests to free an entire region, space within a region, space within a local system queue area, and space within the system queue area.

The VSM routines assign blocks of storage to the various tasks according to their needs. The VSM routines:

- Allocate virtual storage blocks on request.
- Release virtual storage blocks on request.
- Ensure that real (fixed by definition) page-frames exist for all SQA, LSQA, and nonpageable (V=R) region space allocated.
- Maintain storage usage information for use by System Management Facilities.
- Protect storage with fetch protection and storage protection keys.

The GETMAIN and FREEMAIN routines are supported by the GETPART and FREEPART routines, which allocate and free regions and their associated control block space.

The GETPART and FREEPART routines are called by GETMAIN and FREEMAIN to allocate and free space for an entire virtual region. These routines can process requests for both pageable (V=V) and nonpageable (V=R) regions. For V=R region requests, VSM passes control to Real Storage Management to allocate real storage frames to match the virtual pages allocated for the V=R region.

VSM also comprises a set of routines which handle intialization and termination of VSM resources within an address space.

The Create/Free Address Space routines are called by Address Space Create and Address Space Termination to allocate and initialize or delete address space control block space for a new address space. The Create Address Space routine calls Real Storage Management to initialize the RSM control blocks for the new address space. A subroutine within the Create/Free Address Space routines performs storage clean-up when a task terminates. It frees all local storage being used by the terminating task.

Another set of routines satisfy requests for quick cells, small fixed-length blocks of storage in the SQA or in the LSQAs that can be allocated quickly and that can be expected to be used repeatedly during short periods of time.

The Build Quick Cell Pool routine establishes a set of quick cells within an area of storage specified by the requester, a system routine. It formats the storage into a "best fit" number of quick cells or extends an established pool by formatting the new space and enqueuing it from the old pool space.

The GETCELL routine allocates a quick cell from an established cell pool. The FREECELL routine frees a quick cell for further use by returning it to the pool from which it was allocated.

The Delete Quick Cell Pool routine deletes all or part of a pool of quick cells, either freeing the storage or enqueuing the storage to be freed by the user.

Another routine allows the protection key for one or more areas of virtual storage to be manipulated. Both the storage protection key and the fetch protection key for a page that has been allocated by GETMAIN can be changed by using the change key routine (CHANGKEY).

Subpools

A subpool is a group of logically related storage blocks identified by a subpool number. The subpool number indicates to VSM the kind of storage that is requested. Figure 2-44 summarizes the subpool assignments.

Subpool Number	Indicates Request for	Attributes of Subpool	Notes
0-127	Space within a region	Job-oriented Pageable Job step's protection key Fetch-protected	These are the only valid subpool numbers for problem programs. A request for a higher number will cause the problem program to be abnormally terminated. When subpool 0 is requested by programs in supervisor state and key 0, subpool 252 is assigned.
128			Reserved for compatibility with VS1. Treated as an error.
129-226			Undefined.
227	Fixed global space (explicitly assigned and freed)	User protection key Fixed System-oriented Explicitly assigned and freed Fetch-protected	Multiple-key system queue area. Space is obtained from the Common Service Area (CSA).
228	Fixed global space (explicitly assigned and freed)	User protection key Fixed System-oriented Explicitly assigned and freed Not fetch-protected	Multiple-key system queue area. Space is obtained from the Common Service Area (CSA).
229	Private Area Storage	User protection key Pageable Fetch-protected	Automatically freed at task termination. Assigned from top of private area.
230	Private Area Storage	User protection key Pageable Not fetch-protected	Freed automatically at task termination. Assigned from top of private area.
231	Space within CSA (explicitly assigned and freed)	User protection key Pageable Fetch-protected System-oriented Explicitly assigned and freed	Assigned in Common Service Area.
232			Reserved. Treated as an error. Used in OS/VS2 Release 1 for TSO external page storage.
233	Space within LSQA (task-related)	Job-oriented Fixed Protection key = 0 Task-related Swappable Not fetch-protected	Allows a task running in key 0 to acquire accountable, fixed, protected storage that is job-oriented and freed at end of task. Space is assigned from subpool 253.
234	Space within LSQA (job-step-related)	Job-oriented Fixed Protection key = 0 Job-step-related Swappable Not fetch-protected	Allows a task running in key 0 to acquire accountable, fixed, protected storage that is job-oriented and freed at end of job step. Space is assigned from subpool 254.
235	Space within LSQA (explicitly assigned and freed)	Job-oriented Fixed Protection key = 0 Explicitly assigned and freed Not fetch-protected Swappable	Allows a task running in key 0 to acquire non-accountable, fixed, protected storage that is job-oriented. Space is assigned from subpool 255.

Figure 2-44. Subpool Assignments (Part 1 of 3)

Subpool Number	Indicates Request for	Attributes of Subpool	Notes
236	Space within SWA	For system use only Protection key = 1 Not fetch-protected	To assign or free pageable virtual storage for the scheduler work area.
237	Space within SWA	For system use only Protection key = 1 Not fetch-protected	To assign or free pageable virtual storage for the scheduler work area.
238			Reserved for compatibility with OS/VS1. Treated as an error.
239	Fixed, Global Space (explicitly assigned and freed)	Fetch-protected Protection key = 0 Explicitly assigned and freed	System queue area space obtained from the Common Service Area (CSA). Treated as subpool 227 key-zero space.
240	Space within a region (job-step-related)	Job-oriented Pageable Job step's protection key Fetch-protected Job-step-related	Treated as subpool 250 to maintain compatibility with MFT and OS/VSI. Automatically freed at end of step.
241	Space within CSA	System-oriented Pageable User protection key Explicitly assigned and freed Not fetch-protected	Assigned in the Common Service Area.
242	Nonpageable V = R region	For scheduler use only	A new nonpageable (V = R) region is assigned or an existing nonpageable region is freed.
243			Reserved. Treated as an error. Used in OS/VS2 Release 1 for SQA space.
244			Reserved. Treated as an error. Used in OS/VS2 Release 1 for SQA space.
245	Space within SQA (explicitly assigned and freed)	System-oriented Fixed Protection key = 0 Explicitly assigned and freed Not fetch-protected	Allows a task running in key 0 to acquire non-accountable, fixed, protected storage that is system-oriented.
246			Reserved. Treated as an error. Used in MVT to exchange regions.
247	Pageable (V = V) region	For scheduler use only	A new pageable (V = V) region is assigned or an existing pageable region is freed. External page storage allocation is assumed when using this subpool.
248			Reserved. Treated as an error. Used in MVT for rollout/rollin.
249			Reserved. Treated as an error. Used in OS/VS2 Release 1 for LSQA segments.
250	Space within a region	Job-oriented Pageable Job step's protection key Job-step-related Fetch-protected	Allows a task running in supervisor state and key 0 state to acquire unprotected storage in the user's region. All subpool 250 requests are assigned subpool 0 of the associated task.

Figure 2-44. Subpool Assignments (Part 2 of 3)

Subpool Number	Indicates Request for	Attributes of Subpool	Notes
0-127	Space within a region	Job-oriented Pageable Job step's protection key Fetch-protected	These are the only valid subpool numbers for problem programs. A request for a higher number will cause the problem program to be abnormally terminated. When subpool 0 is requested by programs in supervisor state and key 0, subpool 252 is assigned.
128			Reserved for compatibility with VSI. Treated as an error.
129-226			Undefined.
227	Fixed global space (explicitly assigned and freed)	User protection key Fixed System-oriented Explicitly assigned and freed Fetch-protected	Multiple-key system queue area. Space is obtained from the Common Service Area (CSA).
228	Fixed global space (explicitly assigned and freed)	User protection key Fixed System-oriented Explicitly assigned and freed Not fetch-protected	Multiple-key system queue area. Space is obtained from the Common Service Area (CSA).
229	Private Area Storage	User protection key Pageable Fetch-protected	Automatically freed at task termination. Assigned from top of private area.
230	Private Area Storage	User protection key Pageable Not fetch-protected	Freed automatically at task termination. Assigned from top of private area.
231	Space within CSA (explicitly assigned and freed)	User protection key Pageable Fetch-protected System-oriented Explicitly assigned and freed	Assigned in Common Service Area.
232			Reserved. Treated as an error. Used in OS/VS2 Release 1 for TSO external page storage.
233	Space within LSQA (task-related)	Job-oriented Fixed Protection key = 0 Task-related Swappable Not fetch-protected	Allows a task running in key 0 to acquire accountable, fixed, protected storage that is job-oriented and freed at end of task. Space is assigned from subpool 253.
234	Space within LSQA (job-step-related)	Job-oriented Fixed Protection key = 0 Job-step-related Swappable Not fetch-protected	Allows a task running in key 0 to acquire accountable, fixed, protected storage that is job-oriented and freed at end of job step. Space is assigned from subpool 254.
235	Space within LSQA (explicitly assigned and freed)	Job-oriented Fixed Protection key = 0 Explicitly assigned and freed Not fetch-protected Swappable	Allows a task running in key 0 to acquire non-accountable, fixed, protected storage that is job-oriented. Space is assigned from subpool 255.

Figure 2-44. Subpool Assignments (Part 1 of 3)

Subpool Number	Indicates Request for	Attributes of Subpool	Notes
236	Space within SWA	For system use only Protection key = 1 Not fetch-protected	To assign or free pageable virtual storage for the scheduler work area.
237	Space within SWA	For system use only Protection key = 1 Not fetch-protected	To assign or free pageable virtual storage for the scheduler work area.
238			Reserved for compatibility with OS/VS1. Treated as an error.
239	Fixed, Global Space (explicitly assigned and freed)	Fetch-protected Protection key = 0 Explicitly assigned and freed	System queue area space obtained from the Common Service Area (CSA). Treated as subpool 227 key-zero space.
240	Space within a region (job-step-related)	Job-oriented Pageable Job step's protection key Fetch-protected Job-step-related	Treated as subpool 250 to maintain compatibility with MFT and OS/VSI. Automatically freed at end of step.
241	Space within CSA	System-oriented Pageable User protection key Explicitly assigned and freed Not fetch-protected	Assigned in the Common Service Area.
242	Nonpageable V = R region	For scheduler use only	A new nonpageable (V = R) region is assigned or an existing nonpageable region is freed.
243			Reserved. Treated as an error. Used in OS/VS2 Release 1 for SQA space.
244			Reserved. Treated as an error. Used in OS/VS2 Release 1 for SQA space.
245	Space within SQA (explicitly assigned and freed)	System-oriented Fixed Protection key = 0 Explicitly assigned and freed Not fetch-protected	Allows a task running in key 0 to acquire non-accountable, fixed, protected storage that is system-oriented.
246			Reserved. Treated as an error. Used in MVT to exchange regions.
247	Pageable (V = V) region	For scheduler use only	A new pageable (V = V) region is assigned or an existing pageable region is freed. External page storage allocation is assumed when using this subpool.
248			Reserved. Treated as an error. Used in MVT for rollout/ rollin.
249			Reserved. Treated as an error. Used in OS/VS2 Release 1 for LSQA segments.
250	Space within a region	Job-oriented Pageable Job step's protection key Job-step-related Fetch-protected	Allows a task running in supervisor state and key 0 state to acquire unprotected storage in the user's region. All subpool 250 requests are assigned subpool 0 of the associated task.

Figure 2-44. Subpool Assignments (Part 2 of 3)

Subpool Number	Indicates Request for	Attributes of Subpool	Notes
251	Space within a region	Job-oriented Job step's protection key Job-step-related Fetch-protected	Allows an authorized task to acquire accountable, unprotected, pageable storage in the user's partition. Space is job-oriented and automatically freed at the termination of the job step. Used for modules not loaded into Subpool 252 from the low end of storage.
252	Space within a region	Job-oriented Protection key = 0 Job-step-related Not fetch-protected	Allows a task running in key 0 to acquire accountable, pageable, protected storage in the user's region that is job-oriented and automatically freed at the termination of the job-step task. Used for reenterable modules from authorized libraries.
253	Space within LSQA (task-related)	Job-oriented Fixed Protection key = 0 Task-related Not fetch-protected Swappable	Allows a task running in key 0 to acquire fixed, accountable, protected storage in the LSQA for the user's region that is job-oriented and freed when the task terminates.
254	Space within LSQA (job-step related)	Job-oriented Fixed Protection key = 0 Job-step-related Swappable Not fetch-protected	Allows a task running in key 0 to acquire fixed, accountable, protected storage in the LSQA for the user's region that is job-oriented and freed when the job step terminates.
255	Space within LSQA (explicitly assigned and freed)	Job-oriented Fixed Protection key = 0 Explicitly assigned and freed Swappable Not fetch-protected	Allows a task running in key 0 to acquire fixed, non-accountable, protected storage in the LSQA that is job-oriented and must be explicitly freed.

Figure 2-44. Subpool Assignments (Part 3 of 3)

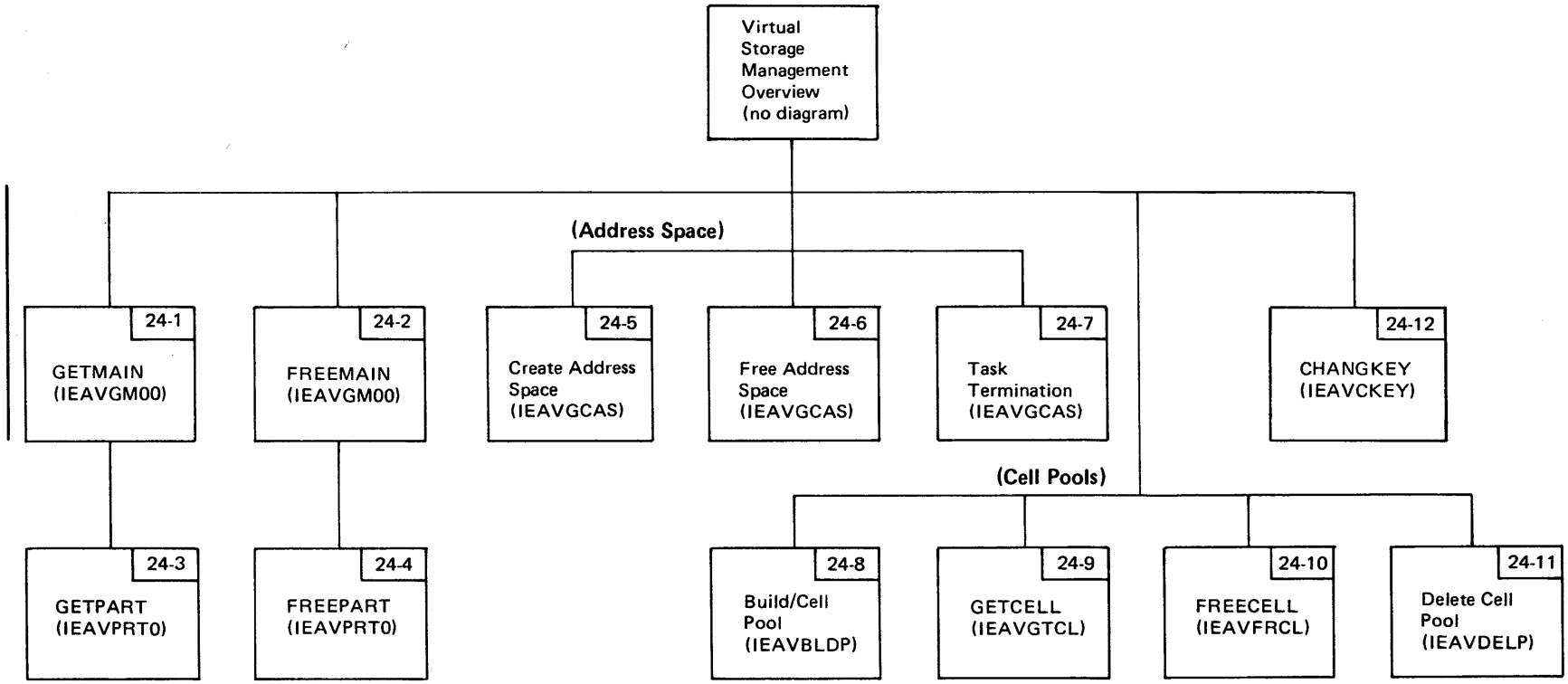


Figure 2-45. Virtual Storage Management Visual Contents

Diagram 24-1. GETMAIN Routine (IEAVGM00) (Part 1 of 2)

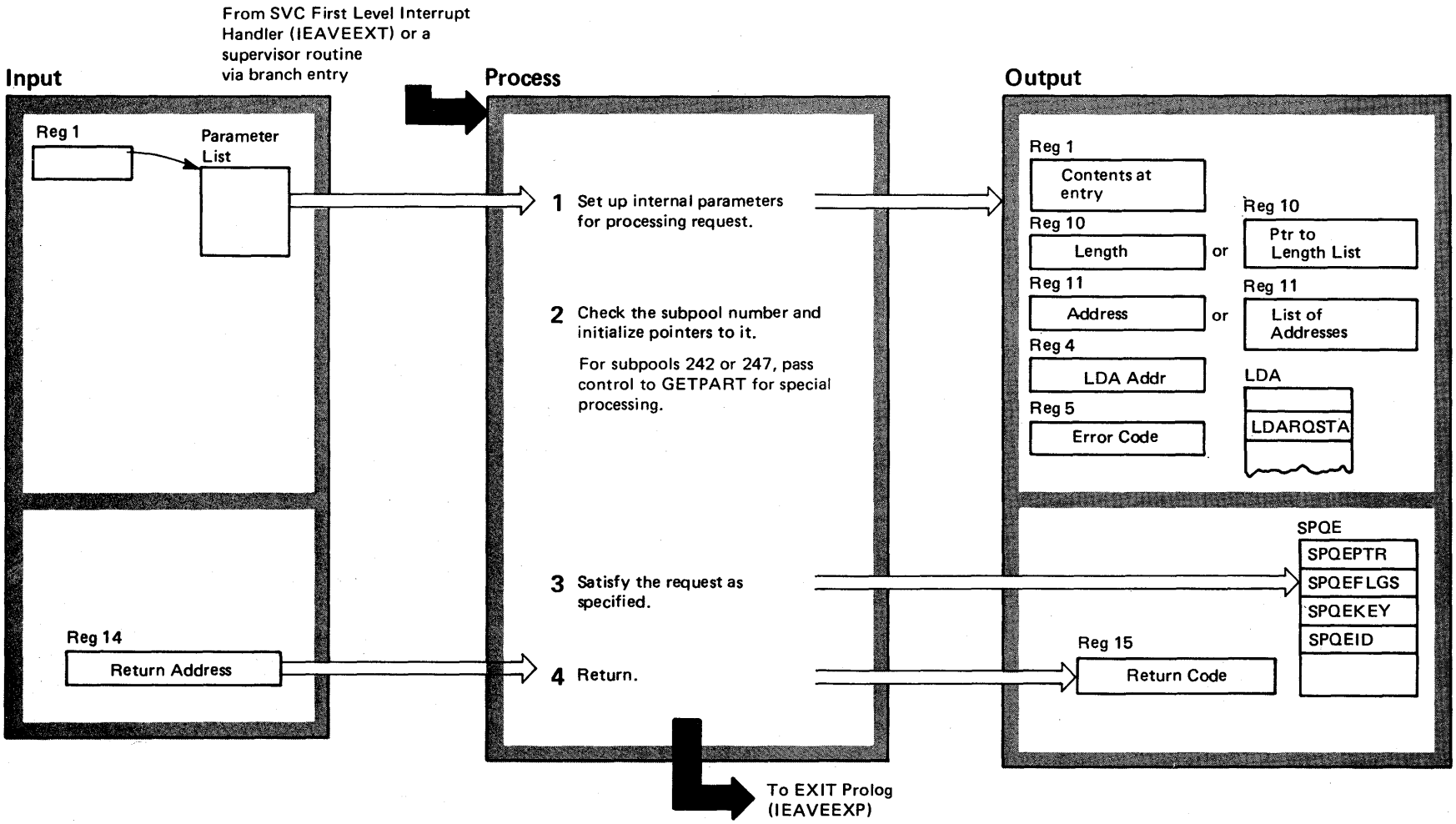


Diagram 24-1. GETMAIN Routine (IEAVGM00) (Part 2 of 2)

Extended Description	Module	Label
The GETMAIN routine (IEAVGM00) allocates virtual storage in the SQA and CSA and in the LSQA, SWA, and user region of each virtual memory. It also provides storage-used figures for System Management Facilities use.		
1 For entry points IGC004 and IGC005, GETMAIN checks the validity of all input parameters and lists.	IEAVGM00	GMBASE
	IEAVEVAL	IEA0VL00
For all other entry points, no validity checking occurs.	IEAVEVAL	IEA0VL01
GETMAIN then sets up internal parameters describing the operation to be performed and the information needed to perform it.		
2 GETMAIN checks the subpool number in the parameter list. If subpools 242 or 247 are requested, GETMAIN	IEAVGM00	CSPCHK
passes control to GETPART (IEAVPRT0). For other subpools, GETMAIN checks the validity of the subpool request and the authorization of the user. If the subpool request is invalid, GETMAIN abnormally terminates the user with a code of Bxy, where xy is the hexadecimal SVC under which GETMAIN was called. For authorized subpool requests, GETMAIN obtains pointers to the relevant control blocks, such as the TCB, GDA (Global Data Area), and the SPQE (Subpool Queue Element).	IEAVPRT0	IEAVPRT0
3 GETMAIN creates an SPQE if no SPQE exists. Then it searches for virtual storage to satisfy the request. If	IEAVGM00	GSPQESPC
	IEAVGM00	GETMAINB
the requested space is not available, GETMAIN sets register 15 to 4 or 8 for conditional requests. For unconditional requests, GETMAIN abnormally terminates the task. If	IEAVGM00	GRRECORE
the space is available, GETMAIN updates the FQE to show	IEAVGM00	GFQEUPDT
the allocated storage and notifies SMF and SRM how much	IEAVGM00	SRMSTART
has been allocated.	IEAVGM00	GMSMFCRE
4 GETMAIN returns control to the caller with a return code of 0 for a successful allocation or an error return	IEAVGM00	GERROR
code of 4 or 8 if the request is conditional; GETMAIN schedules an abnormal termination if the request is unconditional.		
Error Processing	IEAVGFRR	IEAVGFRR
When an error occurs in GETMAIN, Recovery Termination passes control to the FRR. The FRR records information on SYS1.LOGREC, calls for an SVC DUMP, and tries to repair the subpool queues. Then for unexpected errors (machine check, program check, etc), the FRR percolates the error for higher level recovery to RTM. For SALLOC lock release or page release failures, the FRR returns control for execution to continue. For other errors, the FRR issues a completion code of 7xy (where xy is the SVC number under which GETMAIN was called) and then percolates the error through RTM.		

Diagram 24-2. FREEMAIN Routine (IEAVGM00) (Part 1 of 2)

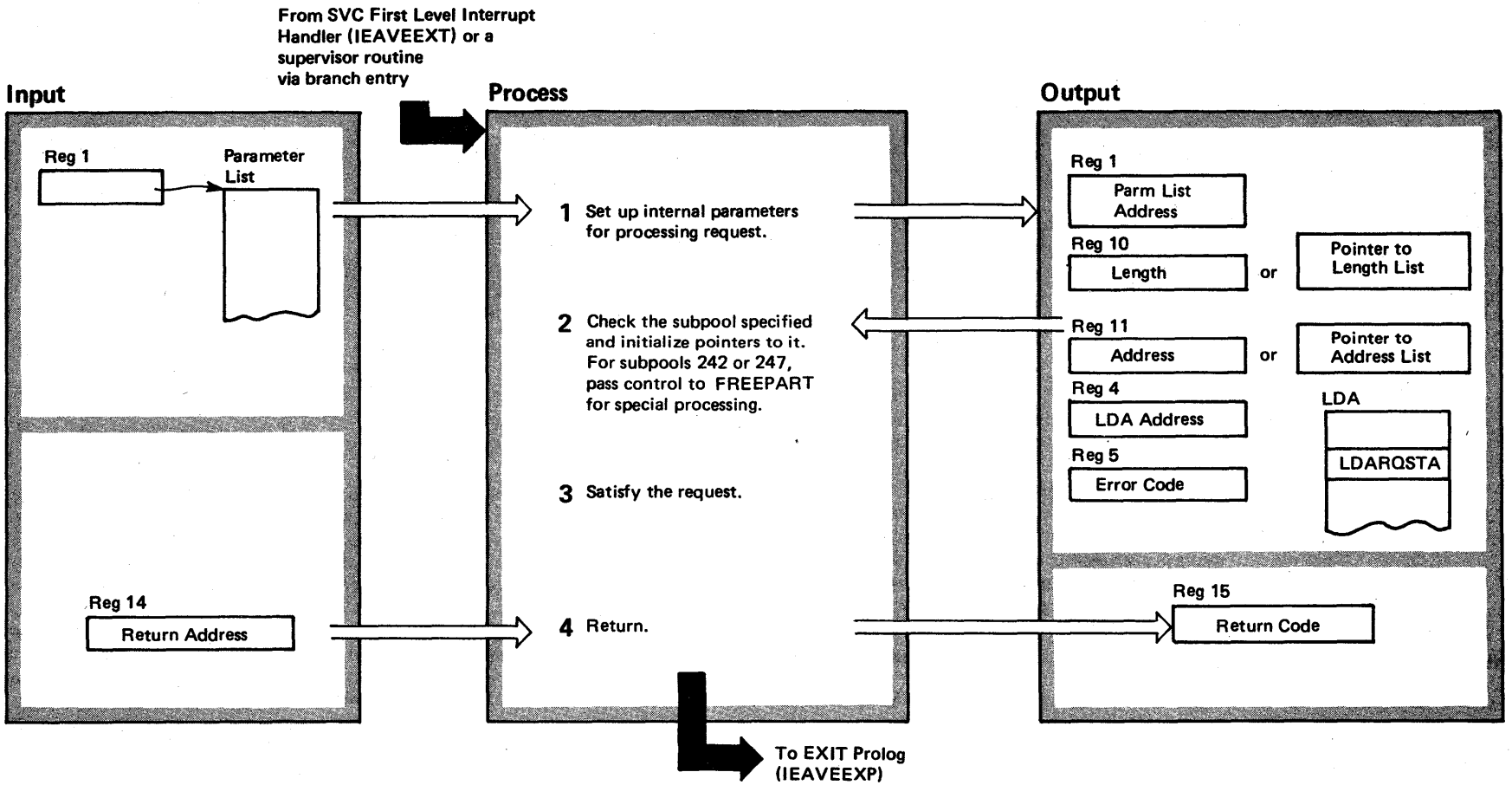


Diagram 24-2. FREEMAIN Routine (IEAVGM00) (Part 2 of 2)

Extended Description	Module	Label
The FREEMAIN routine (IEAVGM00) frees virtual storage in the SQA and CSA and in the LSQA, SWA, and user region of each virtual address space.		
1 For an SVC 5 request, FREEMAIN checks the input parameters and parameter lists. For all other entries, FREEMAIN only indicates the type of entry. The FREEMAIN sets up registers with internal parameters to allow common routines to process FREEMAIN requests.	IEAVGM00	GMBASE
2 FREEMAIN checks the subpool requested. For subpools 242 and 247, FREEMAIN passes control to FREEPART to free the storage. For subpools not in LSQA or SQA, FREEMAIN searches for an SPQE. If no SPQE is found, FREEMAIN sets an error return code of 4 if the request is conditional.	IEAVGM00 IEAVPRT0	FMCOMMON IEAVPRT0
3 FREEMAIN rounds the request up to an 8-byte multiple and searches for the requested storage. It removes the appropriate storage from the allocated space and updates the FQE to show freed space. The AQEs (Allocated Queue Element) for the freed space are removed. FREEMAIN determines whether one or more complete pages of virtual storage have been freed. If so, FREEMAIN calls the RSM PGRLSE routine to release the real pages. Then FREEMAIN releases the virtual pages and updates the FBQE (Free Block Queue Element) associated with the type of storage released. FREEMAIN also notifies the SRM how much space is available in CSA or SQA. In addition, for 4K block releases, FREEMAIN updates the storage-used fields in the TCT for SMF use.	IEAVGM00 IEAVRELS	FMCOM IEAVRELV

Extended Description	Module	Label
4 FREEMAIN returns to the caller with a code of 0 in register 15 for a successful operation; failures are indicated with codes of 4 or 8 in register 15 if the request is conditional. For an unconditional FREEMAIN or a parameter error on a conditional FREEMAIN, FREEMAIN calls for an abnormal termination of the user task. of the user task.	IEAVGM00	CKERRCDE
Error Processing When an error occurs in a FREEMAIN operation, Recovery Termination passes control to the FRR (functional recovery routine). The FRR records information on SYS1.LOGREC, calls for an SVC DUMP operation if necessary, and tries to repair the subpool queues. For unexpected errors, such as machine checks, the FRR returns control to RTM for higher-level error recovery. For SALLOC lock release or page release failures, the FRR returns control and allows execution to continue. For other errors, the FRR issues a completion code of 7xy (where xy is the SVC number through which FREEMAIN was entered) and then passes the error back to RTM for further recovery.	IEAVGFRR	IEAVGFRR

Diagram 24-3. GETPART Routine (IEAVPRT0) (Part 1 of 2)

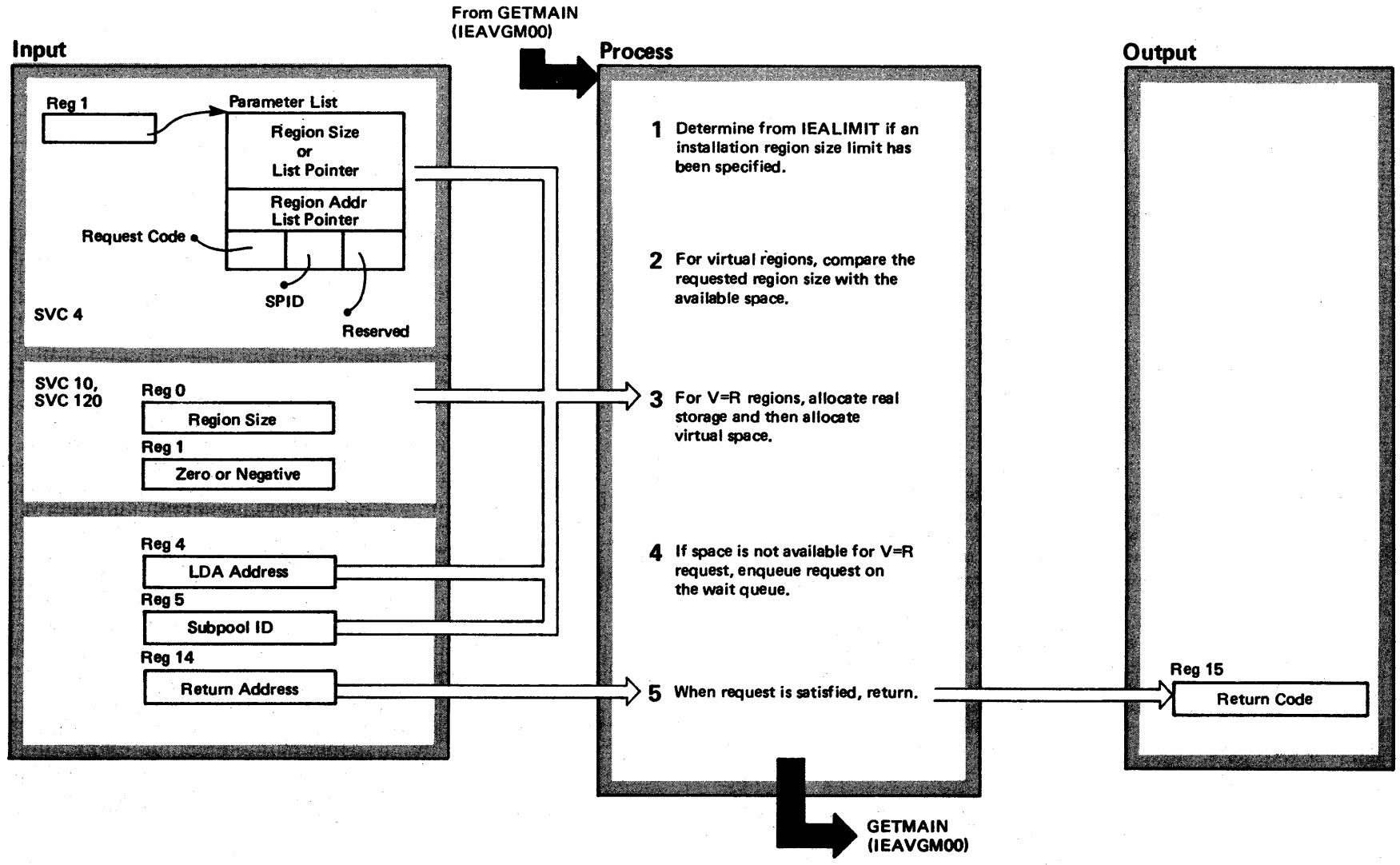


Diagram 24-3. GETPART Routine (IEAVPRT0) (Part 2 of 2)

Extended Description	Module	Label
The GETPART routine (IEAVPRT0) allocates region space at the request of the system. Both V=V and V=R requests are processed by GETPART.		
1 GETPART calls IEALIMIT, the user exit routine, to determine if an installation-supplied limit is to be applied to the region request.	IEAVPRT0	IEALIMIT
2 GETPART checks the V=V region size requested against the total space available for regions within the address space. If not enough space is available, GETPART puts a return code of 8 in register 15 and returns. If not enough contiguous space is available, GETPART puts a return code of 20 in register 15. A region size of zero is taken by GETPART as a request for the system default region size.	IEAVPRT0	IEAVPRT0
3 For a V=R request, GETPART finds an FBQE (Free Block Queue Element) to satisfy the request and then calls RSM to allocate the corresponding real pages. If RSM returns a code of 8, indicating it found assigned frames already allocated in the area requested by GETPART, GETPART recalls RSM with the next available free address. If GETPART can't find sufficient space, it puts a return code of 20 in register 15. If RSM passes a return code of 16, GETPART puts a return code of 16 in register 15.	IEAVPRT0	IEAVPRT0
	IEAVEQR	IEAVEQR
	IEAVPRT0	IEAVPRT0
4 GETPART enqueues a VRWPQEL on the global queue of waiting requests. This occurs when GETPART cannot initially find virtual space to satisfy the V=R region request. When the space becomes free, GETPART posts the ECB for the request. The initiator then reissues the GETPART request.	IEAVPRT0	IEAVPRT0
	IEA0PT01	IEA0PT01
5 When the required region has been allocated, GETPART returns a code of 0 in register 15.	IEAVPRT0	IEAVPRT0
Error Processing	IEAVGPRR	IEAVGPRR
When an error occurs in GETPART processing, Recovery Termination passes control to the GETPART Functional Recovery routine (FRR). For machine checks and program checks in GETPART, the FRR retries the GETPART routine (for V=V requests) or retries the specific section where failure occurred (V=R requests). Where no retry can be made, the FRR cleans up storage already allotted and queues processed and calls for termination to continue. In all cases, the FRR initializes the SDWA. Then the FRR returns to R/TM.		
If an error in the XMPOST routine occurs during the waiting period, the FRR abnormally terminates the waiting initiator with a code of X'304'.	IEAVGPRR	PRT0ERTN

Diagram 24-4. FREEPART Routine (IEAVPRT0) (Part 1 of 2)

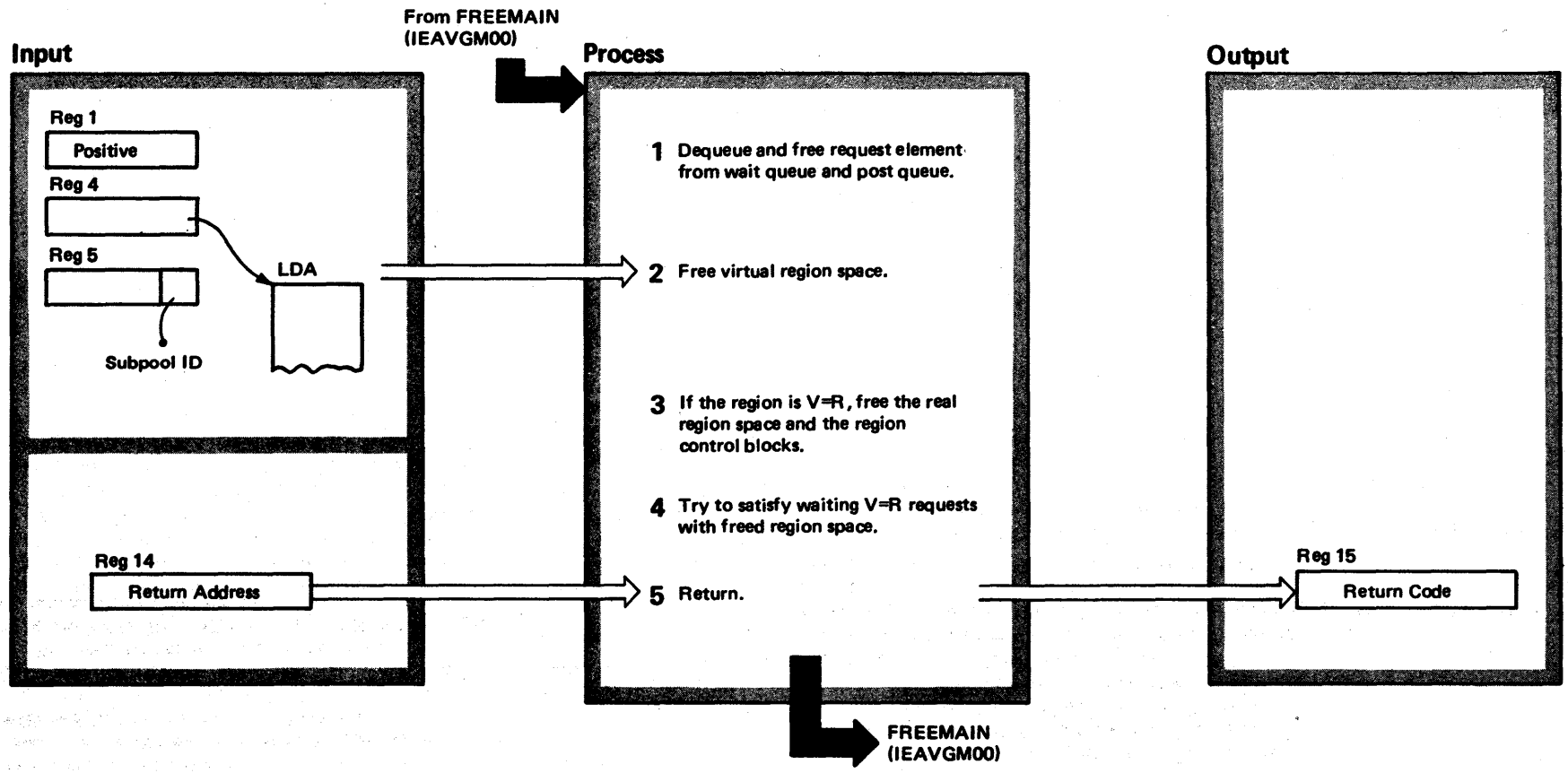


Diagram 24-4. FREEPART Routine (IEAVPRT0) (Part 2 of 2)

Extended Description	Module	Label
The FREEPART routine (IEAVPRT0) processes requests from initiators and Started Task Control to return virtual or real region space to available space. The routine also dequeues and frees the control blocks defining the region.		
1 FREEPART checks the WAIT queue and the POST queue for requests relating to the region being released. If found, the elements are dequeued and the space freed.	IEAVPRT0	IEAVPRT0
2 FREEPART releases any remaining allocated space within the region and the SPQEs identifying it. Then the space representing the region is returned to the system queues. For a V=R region, FREEPART also releases the DPQE and PQE for the region.	IEAVPRT0 IEAVGM00 IEAVGM00	IEAVPRT0 RMBRANCH MRELEASE
3 For a V=R region, FREEPART calls RSM to release the real pages and their identifying control blocks. If the return code from RSM is not zero, FREEPART puts a return code of 4 in register 15.	IEAVEQR	IEAVEQRF
4 FREEPART checks the VRWAITQ for requests that can be satisfied by the region space just freed if the FREEPART was for a V=R region. It posts requests that can use up to, but not more than, the available space.	IEAVPRT0	IEAVPRT0
5 When processing is completed, FREEPART returns to the caller. If the FREEPART is successful, a return code of 0 is placed in register 15.	IEAVPRT0	IEAVPRT0
Error Processing When Recovery Termination passes an error to IEAVGPRR, the routine looks for program checks and machine checks. For these errors, IEAVGPRR tests to determine the extent of processing and calls for retry at that point. For other errors, termination is indicated. IEAVGPRR sets up the SDWA and returns to Recovery Termination. For errors in posting routines from the WAIT queue, IEAVGPRR abnormally terminates the Initiator for the address space with a code of X'304'.	IEAVGPRR IEAVGPRR	IEAVGPRR PRT0ERTN

Diagram 24-5. Create Address Space (IEAVGCAS) (Part 1 of 2)

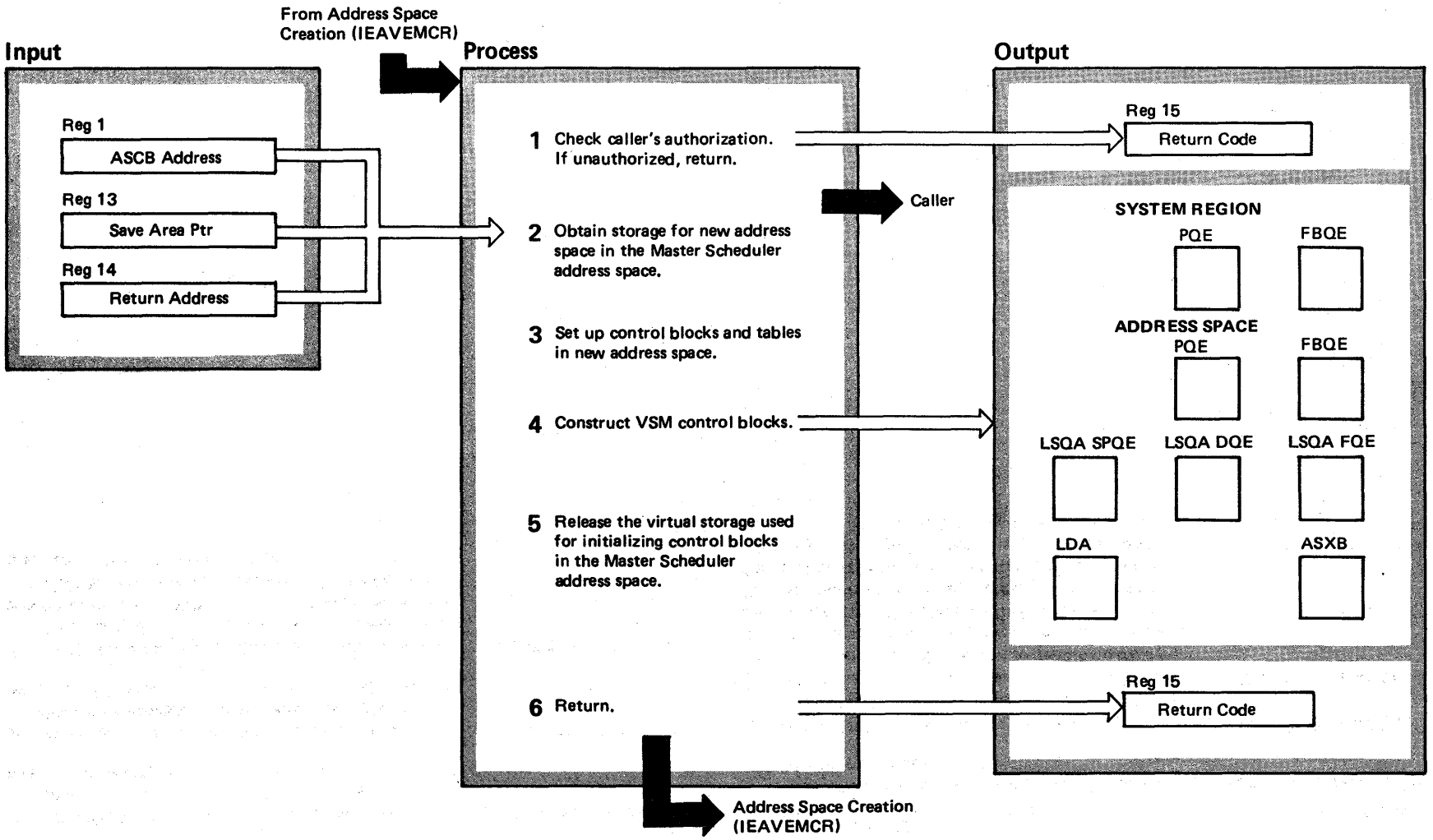


Diagram 24-5. Create Address Space (IEAVGCAS) (Part 2 of 2)

Extended Description	Module	Label
IEAVGCAS (VSM Address Space Creation) processes requests to set up a new address space. It initializes the address space control blocks and calls RSM to set up the RSM control blocks.		
1 IEAVGCAS checks the caller's authorization. If the caller is not authorized, IEAVGCAS puts a return code of 4 in register 15.	IEAVGCAS	IEAVGCAS
2 IEAVGCAS gets a page of storage in the Master Scheduler address space. If the storage can't be obtained, IEAVGCAS puts a return code of 4 in register 15.	IEAVGCAS	IEAVGCAS
3 IEAVGCAS calls RSM Address Space Initialization to set up global and local address control blocks in the new address space. If RSM returns a non-zero return code, IEAVGCAS frees the page in the Master Scheduler address space and puts a return code of 4 in register 15.	IEAVITAS	IEAVITAS
4 IEAVGCAS builds the LDA (Local Data Area) in the top of the page obtained from the Master Scheduler address space. Then it initializes the various address space and region control blocks used by VSM: PQE, FBQE, SPQE, DQE, and FQE.	IEAVGCAS	IEAVGCAS
5 IEAVGCAS releases the virtual page in the Master Scheduler address space.	IEAVGCAS	IEAVGCAS
6 IEAVGCAS returns to the caller through register 14.	IEAVGCAS	IEAVGCAS
Error Processing When errors occur, IEAVCARR frees the page in Master Scheduler address space. For program checks and machine checks, IEAVCARR retries the IEAVGCAS routine unless RSM had been entered; if so, IEAVCARR returns to Address Space Creation with a return code of 4. For any other errors, IEAVCARR records information in the SDWA and routes control to R/TM to continue termination processing.	IEAVCARR	IEAVCARR

Diagram 24-6. Free Address Space (IEAVGFAS) (Part 1 of 2)

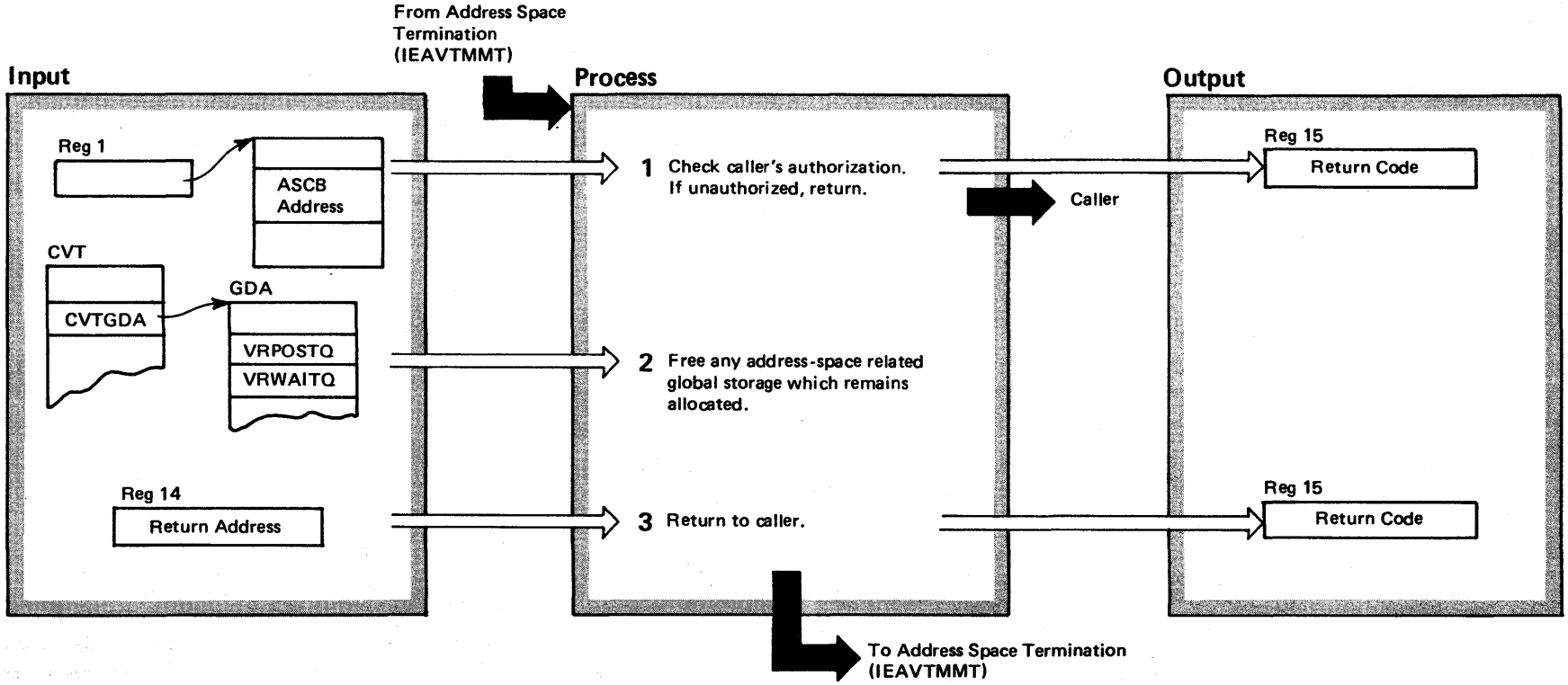


Diagram 24-6. Free Address Space (IEAVGFAS) (Part 2 of 2)

Extended Description	Module	Label
IEAVGFAS (VSM Address Space Freeing) processes the deletion of an address space. It dequeues and frees all queue elements relating to the address space and updates the system control blocks.		
1 IEAVGFAS checks the caller's address space against the Master Scheduler ASID. If not equal, the routine puts a return code of 4 in register 15.	IEAVGCAS	IEAVGFAS
2 IEAVGFAS checks the VRWAITQ and VRPOSTQ for an element identified for the specified address space, dequeues it, and frees the space.	IEAVGCAS	IEAVGFAS
	IEAVGCAS	FREEQEL
3 If no errors have occurred, IEAVGFAS puts a return code of 0 in register 15 and returns.	IEAVGCAS	IEAVGFAS
Error Processing	IEAVCARR	IEAVFARR
For retriable errors during dequeuing, IEAVFARR attempts to retry the dequeuing routine; for other retriable errors, IEAVFARR re-enters the IEAVGFAS routine. For other errors, IEAVFARR records information in the SDWA and returns to Recovery Termination.		

Diagram 24-7. Task Termination (IEAVGCAS) (Part 1 of 2)

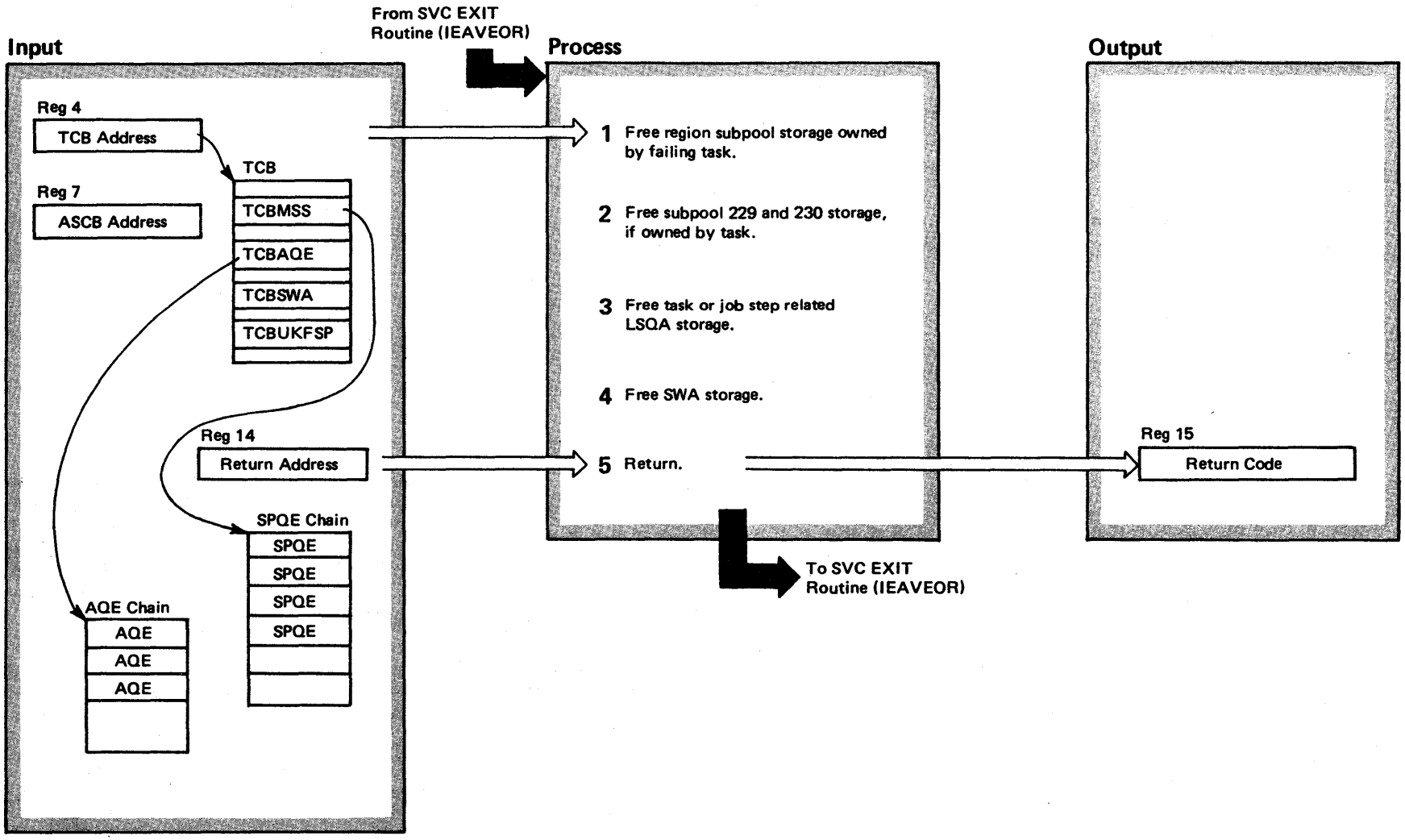


Diagram 24-7. Task Termination (IEAVGCAS) (Part 2 of 2)

Extended Description	Module	Label
IEAQSPET (VSM Task Termination) performs storage clean-up operations when a task is terminating. It frees all local storage owned by the task.		
1 IEAQSPET frees the subpool storage represented by the SPQEs chained from the TCBMSS field for the task unless the subpool is shared. Then it frees the SPQEs.	IEAVGCAS IEAVGCAS	IEAQSPET FREESPQE
2 IEAQSPET frees the subpool 229 and 230 storage and the SPQEs for the task.	IEAVGCAS IEAVGCAS	IEAQSPET FREESPQE
3 IEAQSPET frees the SWA space for the task unless the subpool is shared and then frees the SPQEs.	IEAVGCAS IEAVGCAS	IEAQSPET FREESPQE
4 When all control block queues have checked, IEAQSPET returns to EXIT with a return code of 0 in register 15. If any of the FREEMAIN operations failed, IEAQSPET places a return code of 4 in register 15.	IEAVGCAS	IEAQSPET
Error Processing When the error is a program check or a machine check, IEAVTTRR enters the IEAQSPET routine for retry. Otherwise, it returns to Recovery Termination after recording the SDWA information.	IEAVCARR	IEAVTTRR

Diagram 24-8. Build Quickcell Pool Routine (IEAVBLDP) (Part 1 of 2)

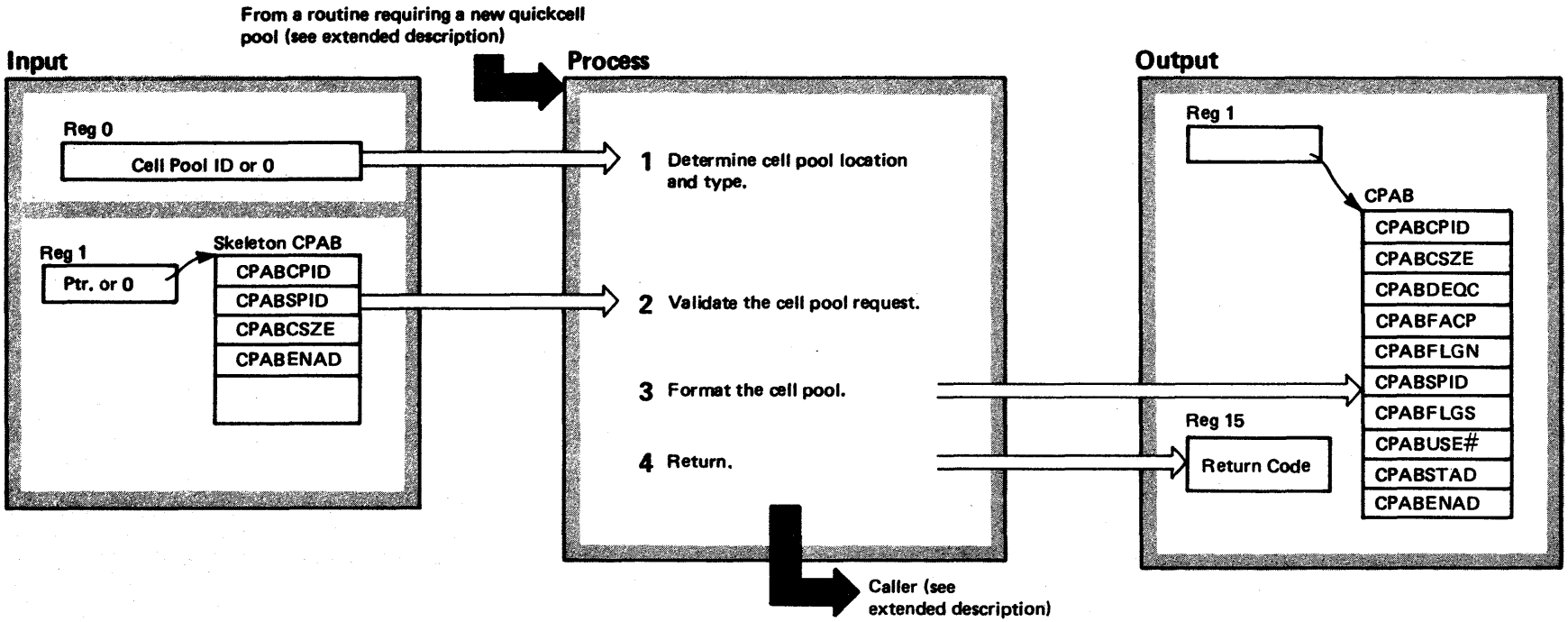


Diagram 24-8. Build Quickcell Pool Routine (IEAVBLDP) (Part 2 of 2)

Extended Description	Module	Label
The Build Quickcell Pool routine (IEAVBLDP) creates, extends, or reformats a pool of quickcells, as directed by the internal macro instruction (BLDCPOOL) that invokes it. Modules that can require a new quickcell pool are: IEAVEMIN, IEAVESVC, IEAVMDOM, IEAVMWTO, IEAVNIP0, IEAVNPA6, IEAVNP14, IEAVPCB, IEAVSWCH, IEAVVINT, IEAVVRP2, IEAVVWTO, IEEMB803, IEEMB804, and IRARMSRV.		
1 IEAVBLDP checks the CPAB (Cell Pool Anchor Block) and CPID (Cell Pool Identifier) passed to it. It determines whether a new cellpool must be created, whether a cell pool is to be extended, or whether a cell pool is to be reformatted.	IEAVBLDP	IEAVBLDP
2 IEAVBLDP verifies that all parameters passed are valid by checking them against the skeleton CPAB built by the macro processor.	IEAVBLDP	CPIDTEST
3 IEAVBLDP formats the new cell pool by dividing it into the number of cells that will fit into the specified area and storing pointer and size information in the CPAB. Then it formats each cell, linking it to its chain through linkage pointers.	IEAVBLDP	POOLFORM LOOPFORM
4 IEAVBLDP returns control to the caller with a return code indicating success (0) or an error:	IEAVBLDP	ERREXIT

Return Code	Error
8	Invalid CPID or unformatted pool.
12	Invalid Subpool
16	Invalid cell size
20	Incompatible concurrent request.

In each error return case, register 0 contains the extent subpool number and the extent length; register 1 contains the extent address.

Diagram 24-9. GETCELL Routine (IEAVGTCL) (Part 1 of 2)

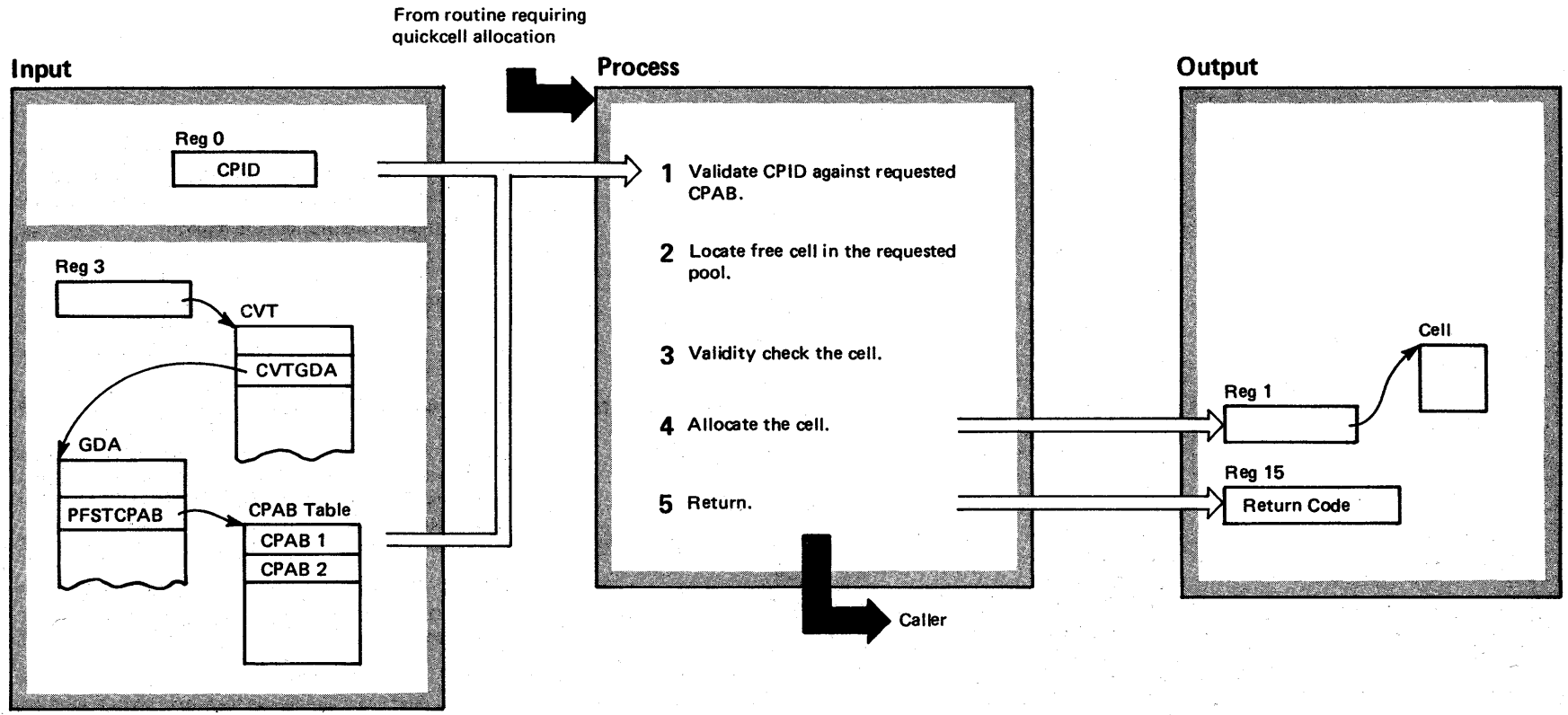


Diagram 24-9. GETCELL Routine (IEAVGTCL) (Part 2 of 2)

Extended Description	Module	Label
The GETCELL routine (IEAVGTCL) allocates a quick-cell from an established quickcell pool. The routine is invoked through the GETCELL internal macro instruction. Modules that can require quickcell allocation are: IEAVELK, IEAVEMIN, IEAVEPC, IEAVEQR, IEAVESVC, IEAVGPRR, IEAVMDOM, IEAVMWTO, IEAVPCB, IEAVPFTE, IEAVPIOP, IEAVRCF, IEAVRFR, IEAVSOUT, IEAVSWCH, IEAVTRTH, IEAVTRTM, IEAVTRTR, IEAVVRP2, IEAVVWTO, IEEMB804, and IRARMSRV.		
1 IEAVGTCL checks the CPID and CPAB passed for validity. It also checks for matching CPIDs and empty pools.	IEAVGTCL	IEAVGTCL
2 IEAVGTCL locates an empty cell in the requested pool by checking the CPABFACP field. It also verifies that no deletions are in process against the extent.	IEAVGTCL	PERMCPID
3 IEAVGTCL checks the cell for residence in the proper extent and for boundary alignment within the extent.	IEAVGTCL	DEQLOOP2
4 IEAVGTCL stores the CPID in the chosen cell and unlocks the pool extent for further operations.	IEAVGTCL	STORCPID
5 IEAVGTCL returns control to the caller with a return code of 0 for successful allocation or the following error return codes:		

Return Code	Error
4	Empty pool or extent being deleted.
8	Extent is unreliable
12	Pool is unformatted
16	Invalid CPID

Diagram 24-10. FREECELL Routine (IEAVFRCL) (Part 1 of 2)

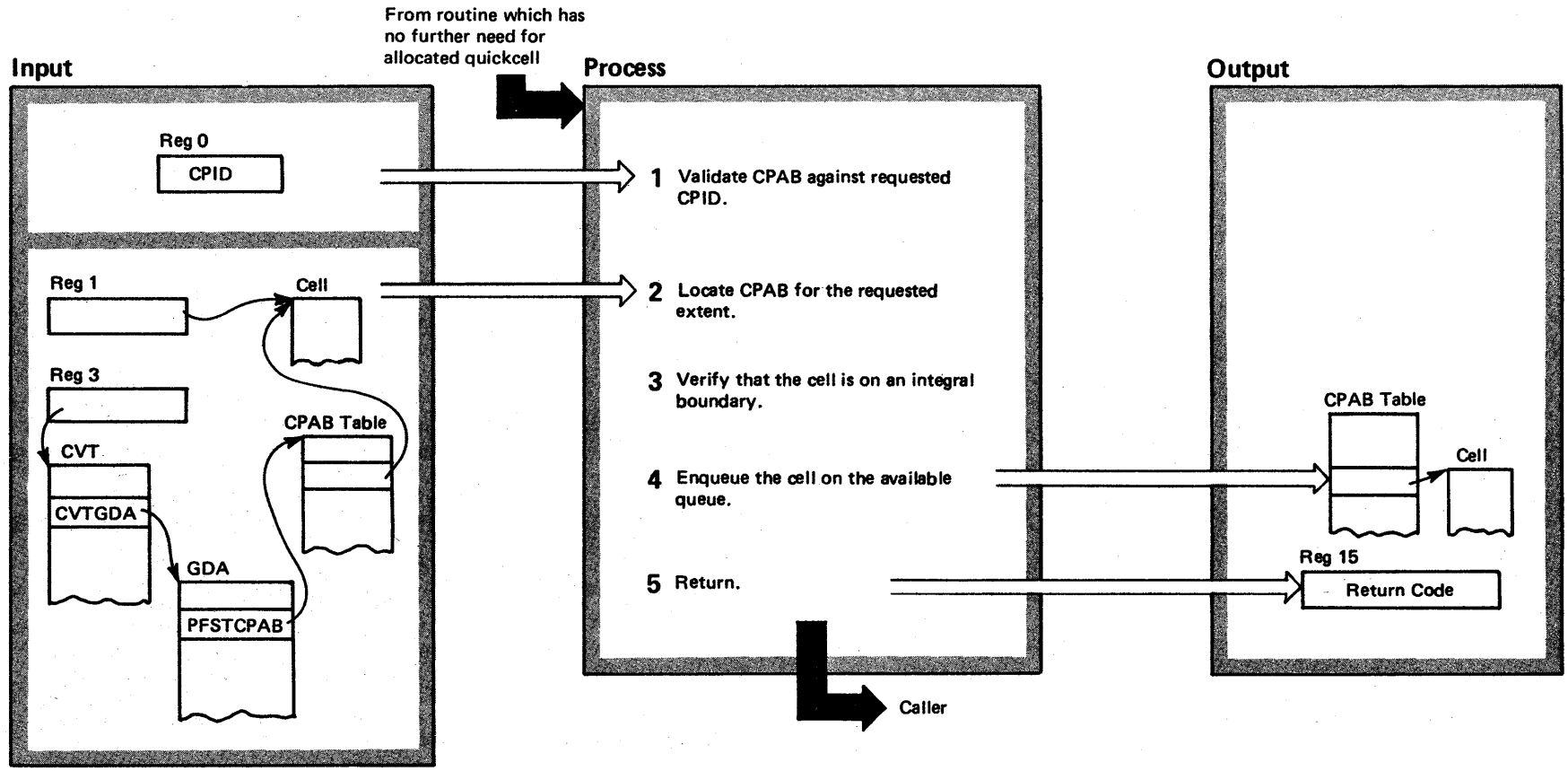


Diagram 24-10. FREECELL Routine (IEAVFRCL) (Part 2 of 2)

Extended Description	Module	Label
The FREECELL routine (IEAVFRCL) returns a quick-cell to a quickcell pool. It makes the cell available for use by adding it to a queue of available calls in the pool. Modules that may no longer require an allocated quickcell are: IEAVDLAS, IEAVEDSO, IEAVEEXP, IEAVEOR, IEAVEPCR, IEAVEQR, IEAVGFA, IEAVGPRR, IEAVIOCP, IEAVMDOM, IEAVMDSV, IEAVPIOI, IEAVRCF, IEAVRFR, IEAVSOUT, IEAVSWCH, IEAVSWIN, IEAVTRTR, IEAVTRT1, IEAVTRT2, IEAVVRP2, IEAVVWTO, IEEMB803, and IRARMSRV.		
1 IEAVFRCL checks the cell to determine that it was allocated from the cell pool specified.	IEAVFRCL	PERMCPID
2 IEAVFRCL locates the CPAB for the cell pool specified.	IEAVFRCL	GOTCPAB
3 IEAVFRCL verifies that the cell is on an integral boundary in the extent, and that no deletions are taking place concurrently.	IEAVFRCL	CPABLOOP
4 IEAVFRCL returns the cell to the pool of available cells and releases the extent for further operations.	IEAVFRCL	ENQLOOP
5 IEAVFRCL returns control to the user and passes a return code indicating success (0) or an error return code:	IEAVFRCL	FRCEXIT0

Return Code	Error
4	Cell not allocated from specified pool (CPID doesn't match)
8	The cell did not come from one of the tents in specified pool
12	Unformatted pool
16	Invalid CPID

Diagram 24-11. Delete Quickcell Pool (IEAVDELP) (Part 1 of 2)

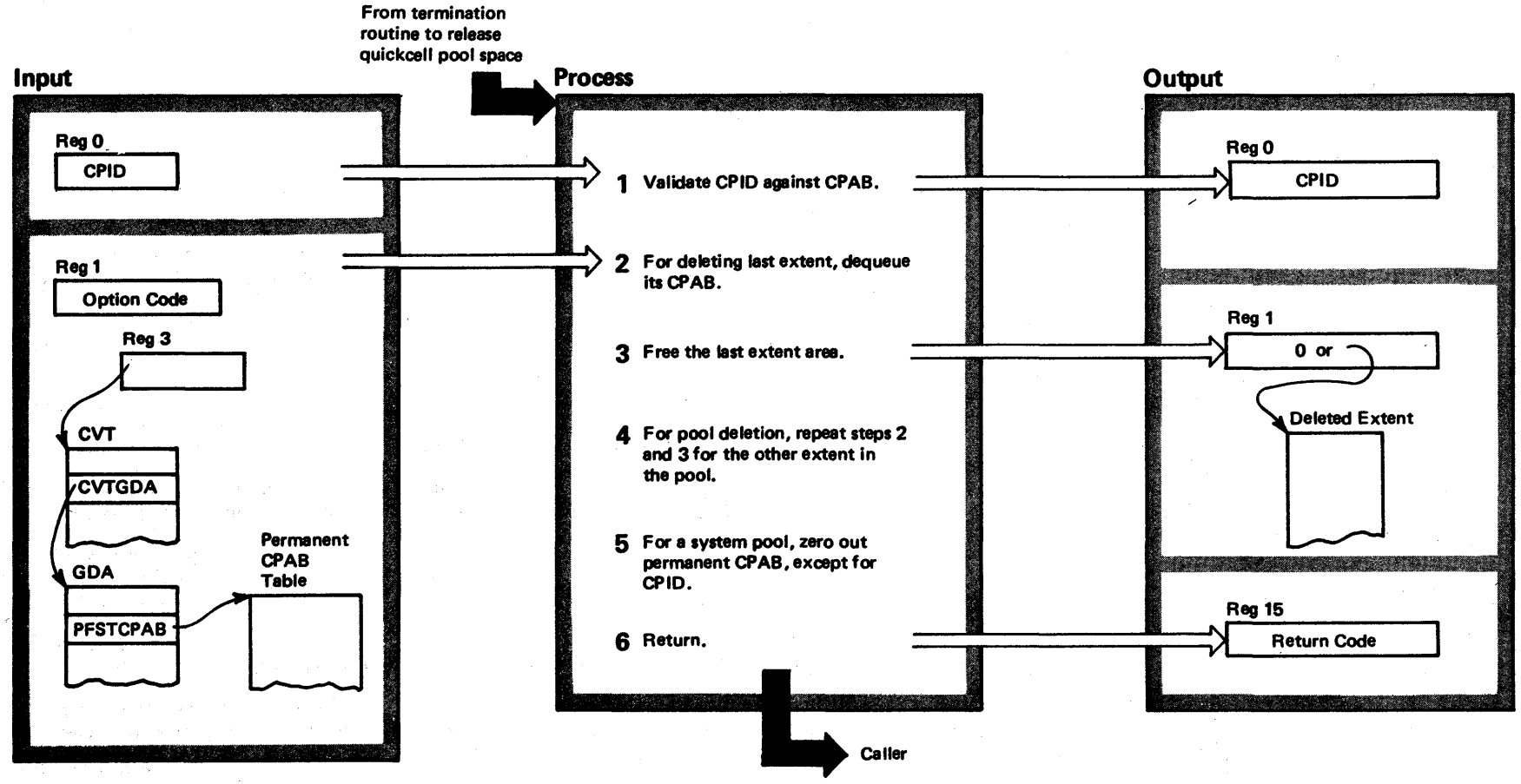


Diagram 24-11. Delete Quickcell Pool (IEAVDELP) (Part 2 of 2)

Extended Description	Module	Label
The Delete Quickcell Pool routine (IEAVDELP) removes all or part of a pool of quickcells, either freeing the storage or enqueuing the storage for user freeing. Either one extent, all extents, or the whole pool may be deleted, depending on which DELCPOOL macro instruction option is chosen.		
1 IEAVDELP checks the CPID against the CPAB for validity. It also checks to see if NIP created the cell pool.	IEAVDELP	GOTCPAB
2 IEAVDELP finds the last extent and checks for another operation in progress. If none, IEAVDELP dequeues the CPAB for the last extent.	IEAVDELP	EXTENT
3 If the suppress FREEMAIN option was chosen, IEAVDELP stores FREEMAIN information in the first two words of the extent. Otherwise, IEAVDELP frees the storage used by the extent and its CPAB.	IEAVDELP	REMOVEAB
4 IEAVDELP checks for pool deletion and, if requested, loops through the pool deleting all extents and CPABs.	IEAVDELP	REMOVED
5 IEAVDELP sets to zero all fields in the permanent CPAB, except CPABCPID, when the entire pool of quickcells has been deleted.	IEAVDELP	DELEXIT
6 IEAVDELP returns control to the caller with a return code of zero for success or one of the following error return codes:		

Return Code	Error
8	Attempt to delete a NIP-created pool.
12	Attempt to delete an unformatted pool.
16	Invalid or null CPID.
20	A conflicting function is pending for specified extent.

Diagram 24-12. Change Key Routine (IEAVCKEY) (Part 1 of 2)

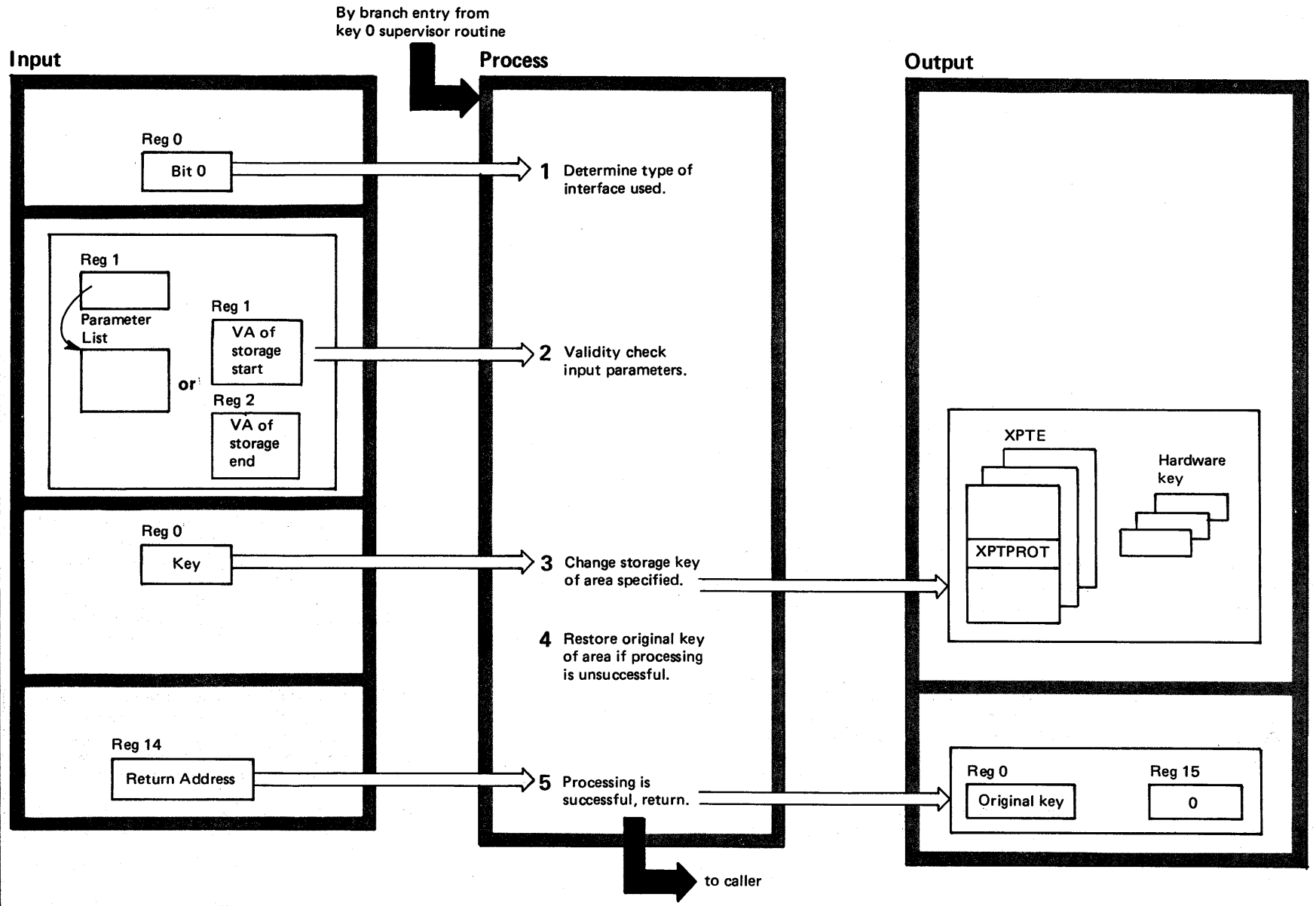


Diagram 24-12. Change Key Routine (IEAVCKEY) (Part 2 of 2)

Extended Description	Module	Label
The change key routine (IEAVCKEY) changes the key of areas of storage within the problem program subpools at the request of supervisor-state key 0 programs.		
1 Two types of interfaces are recognized, R-type and L-type. R-type interfaces (indicated by bit zero of input register 0 being zero) specify via general purpose registers 1 and 2 a single virtual address (VA). L-type interfaces (indicated by bit zero of input register 0 being one) specify one or more VA ranges via a parameter list.	IEAVCKEY	IEAVCKEY
2 For L-type interfaces, the parameter list supplied must be in fixed storage (L/SQA or PGFIX). For either interface, the VA range(s) must define storage from subpools 0-127, 251, and 252.	IEAVCKEY	REGPTOC LISTPROC ELTVCK PAGEVCK
3 For each VA range, the storage key and fetch protection flag at all pages in the range are changed to the new key and new fetch protect flag supplied. This is accomplished by: • Changing the XPTPROT field in the XPTE associated with each page. • Changing the hardware key of any pages that are assigned to a frame in real storage at the time of the request. The key of the first page that will be changed is saved for return to the caller upon successful completion of the change key function.	IEAVCKEY	ELTPROC KEYCHG
4 If a page within this area is found not to have been allocated (via GETMAIN) during the process of changing the key for the area of storage, an error condition is recognized and the original key of the area of storage is restored.	IEAVCKEY	ELTPROC RECOVER
5 At the successful completion of this routine, the caller receives control with a return code of 0 in register 15 and the key of the first page changed in register 0.	IEAVCKEY	IEAVCKEY
Error Processing: For any error that prevents successful completion of the change key function, the requesting program is abnormally terminated with an error code in register 15 reflecting the exact error that occurred. For unexpected errors in IEAVCKEY, recovery termination management (RTM) gives control to the change key FRR (IEAVCKRR). For system or machine errors, the FRR records information on SYS1.LOGREC and requests, when possible, a retry to recover the original storage key of all areas of storage that has been changed. For all other type error conditions, percolation is requested through RTM.	IEAVCKRR	IEAVCKRR

Auxiliary Storage Management (ASM)

Overview

ASM transfers virtual storage pages between real storage and auxiliary storage, either as a paging operation (a page at a time) or as a swapping operation (an address space at a time). Additionally, ASM manages auxiliary storage, and maintains the necessary copies of VIO data set pages.

ASM is called by RSM (Real Storage Management) and VBP (Virtual Block Processor). ASM interfaces more directly with RSM than before; RSM calls the appropriate modules in ASM for the specific function needed. Also, control blocks (XPTEs and AIAs) are shared with RSM. VBP calls one module (ILRGOS) to initiate VIO operations.

The ASM MO diagrams are presented in seven sections corresponding to the seven functions described here. There is an introduction to each section that contains a more complete description of each function, including control block usage.

ASM processing is divided into seven functions:

- *I/O Control* is the communication link through which Real Storage Management (RSM) makes paging and swapping requests. I/O Control determines the type of request, passes it to the Swap Driver part of I/O Control or to the I/O Subsystem, and is notified of its completion. I/O Control notifies RSM of the completion, and keeps track of the auxiliary storage locations of all virtual pages.
- *I/O* Subsystem receives control via an SRB from I/O Control, starts I/O Supervisor (IOS) processing by issuing the STARTIO macro, and returns control to I/O Control after the completion of the I/O. The message module, which produces the messages issued by ASM, is also a part of the I/O Subsystem.
- *VIO Control* coordinates and synchronizes all ASM processing required to support VIO data sets. This function interfaces with the Virtual Block Processor (VBP) for group-related requests. VIO Control and I/O Control process VIO page-related requests that RSM initiates.
- *VIO Group Operators* maintain the VIO data set information required by VBP. These operators are invoked only by VIO Control as the result of requests from VBP.
- *Recovery* provides the mechanism to handle two types of errors, those detected during normal ASM processing, and those detected by ASM recovery while it is in control. ASM recovery attempts to determine the severity of the error and then takes appropriate action.
- *Service Routines* include: an ASM control block formatting facility, which is invoked by the system dump-printing facility; an address space termination resource manager, whose main function is to reclaim auxiliary storage resources from an address space that is terminating; and a pool extender routine for adding storage to a virtual storage pool.
- *Page Expansion* gives the user the ability to add page and swap data sets to the system without having to do another IPL. This function is available to the installation through the PAGEADD operator command.

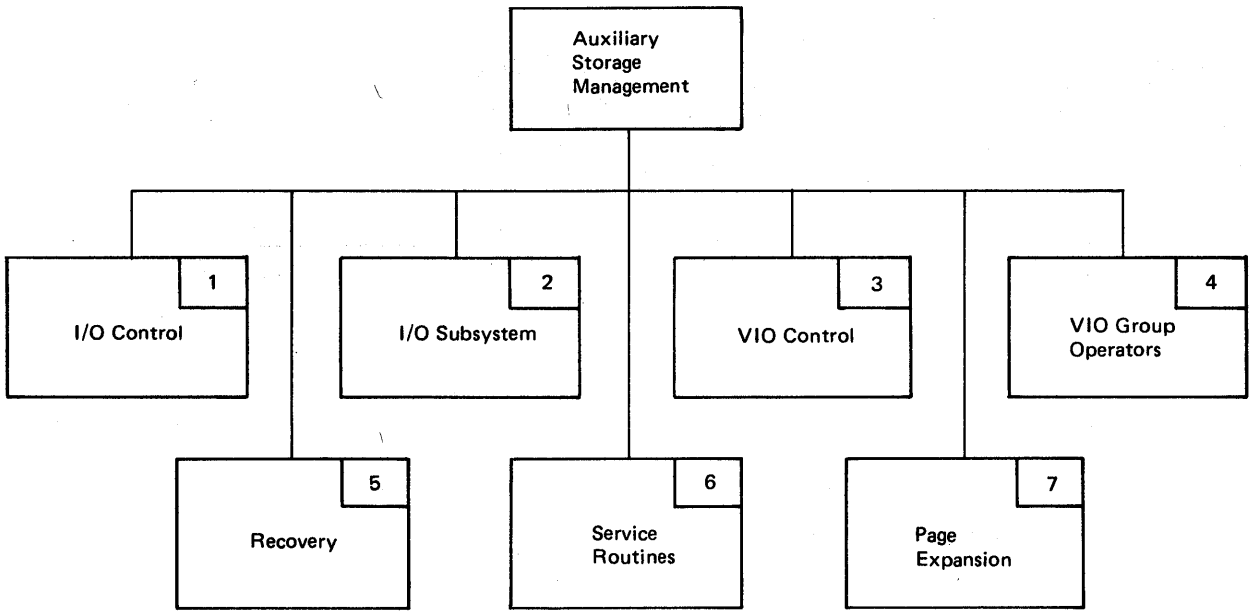


Figure 2-56. Auxiliary Storage Management Visual Table of Contents

I/O Control

In MVS, RSM initiates all paging and swapping I/O. The I/O Subsystem (part of ASM) and the I/O Supervisor (IOS) execute the paging/swapping I/O. I/O Control is the communication link between RSM and the I/O Subsystem-IOS.

I/O Control is divided into three functional units: Initial Page Processing, Initial Swap Processing, and Completion Processing.

Initial Page Processing

The ASM module ILRPAGIO performs Initial Page Processing. RSM or Initial Swap Processing sends a chain of ASM I/O Request Areas (AIAs) to ILRPAGIO. Each AIA represents a request for a paging operation (either in or out) against either VIO or non-VIO pages.

VIO Requests

ILRPAGIO sends requests for VIO paging to ILRPOS (the Page Operation Starter, part of VIO Control) for processing. See Chapter 3, "VIO Control " for a description. The return of an AIA address from ILRPOS to ILRPAGIO indicates an error AIA.

Non-VIO Requests

For non-VIO write requests, ILRPAGIO clears the XPTE (External Page Table Entry) of the page to be written and calls ILRFRSL1 (an entry point in the Free Slot module, also part of I/O Control) to free the slot of auxiliary storage that page currently occupies.

For non-VIO read requests, the XPTE is checked to see if the page to be read has a valid LSID (Logical Slot Identifier). If the LSID is valid, it is copied into the AIA. If it is invalid, or if there was a previous I/O error on this page (indicated by a flag in the XPTE), the AIA is in error.

ILRPAGIO puts valid AIAs on the staging queue (ASMSTAGQ). The ILRQIOE entry point of ILRPAGIO is then called to build an IOE (I/O Request Element) for each AIA on the queue. Initial Page Processing and the I/O Subsystem communicate via IOEs.

ILRQIOE queues write IOEs to the PART (Page Activity Reference Table); it queues read IOEs to the PART Entry of the page to be read. If there is no I/O currently outstanding, ILRQIOE then schedules an SRB for ILRPTM (the PART Monitor, part of the I/O Subsystem) to start the work represented by the IOEs. If there is I/O currently outstanding, Page Completion (ILRPAGCM) will schedule the SRB for ILRPTM when that I/O completes.

Each AIA received from RSM is processed until the entire chain of AIAs is exhausted or an error is found. If an error is found, the AIA chain is broken and the error AIA and any following AIAs are returned to RSM.

Initial Swap Processing

Two modules, ILRSWAP and ILRSWPDR, perform initial swap processing. RSM sends a chain of AIAs to ILRSWAP. ILRSWAP divides the chain into two groups: requests against non-LSQA pages and requests against LSQA pages. Non-LSQA requests are sent to ILRPAGIO to be processed to page data sets.

When ILRPAGIO returns, ILRSWAP determines if the ILRSLSQA entry point of ILRSWAP can be called to process the LSQA pages through special, high-speed, swap data sets. If all paging operations are complete, ILRSWAP calls ILRSLSQA. If all paging operations are not complete, the LSQA pages cannot be processed now and ILRWSAP returns to RSM. LSQA page processing is initiated later by ILRPAGCM, the page completion routine.

When ILRSLSQA gets control, if the AIA request is a swap-out and no swap data sets are available, or if it is a swap-in of LSQA pages previously written to page data sets, ILRSLSQA calls ILRPAGIO to process the AIA. Otherwise, ILRSLSQA builds a SCCW (Swap Channel Command Workarea) and a channel program for the request, and chains the SCCW from the SART Entry (Swap Activity Reference Table Entry) for the appropriate swap data set. An SRB for ILRSWPDR (the Swap Driver) is scheduled to start the work represented by the SCCW.

ILRSWPDR checks each SART Entry for work (represented by a SCCW chained from the SART Entry). When it finds work, it locks the SART entry and chains the SCCW to the IORB/IOSB (I/O Request Block and I/O Supervisor Block) that is also chained to the SARTE. ILRSWPDR then issues STARTIO to begin IOS processing against the swap data set.

Completion Processing

ILRPAGCM handles completion processing. The function of ILRPAGCM is to process completed page and swap requests and place the AIAs on queues to be retried or to be returned to RSM. ILRPAGCM divides the chain of AIAs that is passed to it into two groups; one group contains AIAs representing paging requests, the other contains

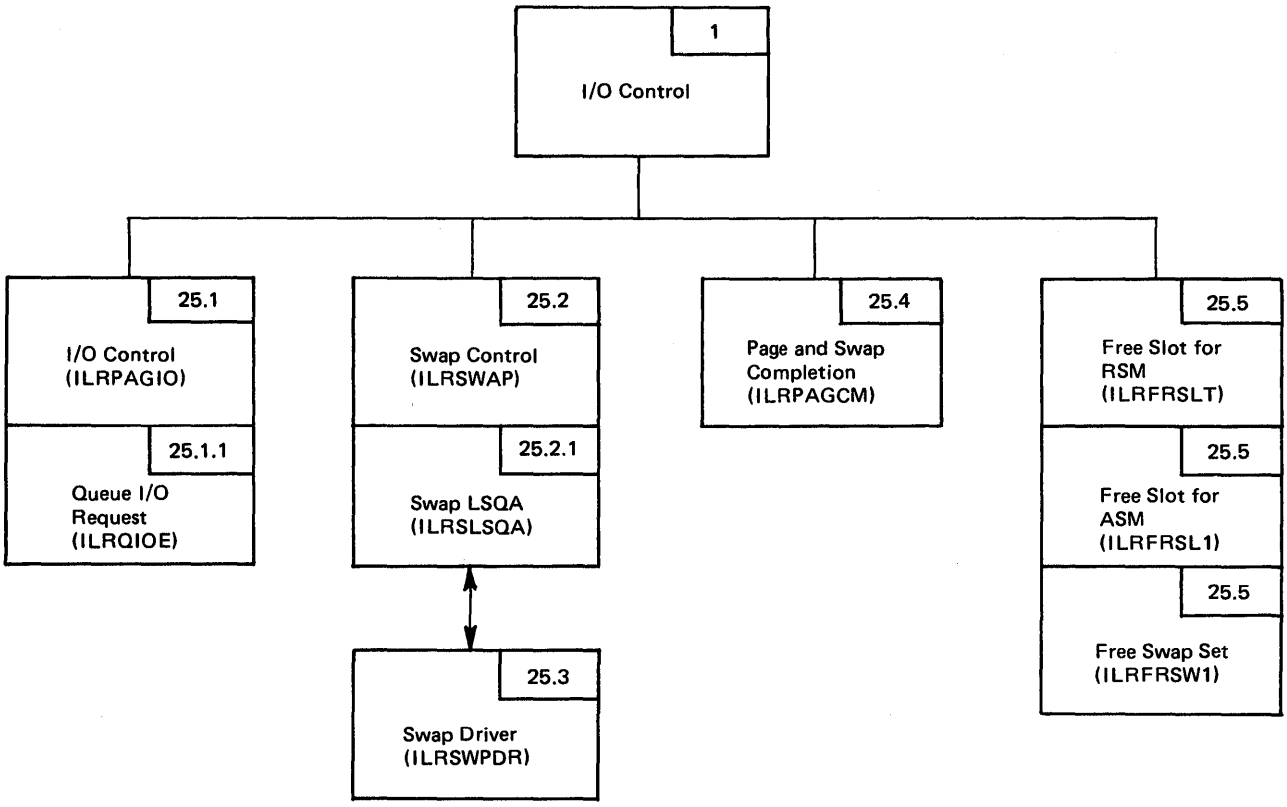
swapping requests. It processes each group separately.

Page Completion
This procedure handles all AIAs that are completed for page requests and VIO requests. When an AIA completes successfully, Page Completion puts it on a queue to be returned to RSM. If an error occurs on a read request and if there is a backup copy of the page, the request is retried. An error on a write request is always retried. If any additional I/O requests are queued to the PART, ILRPAGCM schedules an SRB for ILRPTM. If a swap-out is in process, Page Completion checks to see if all non-LSQA operations have completed. If they have, ILRSLSQA is called to start the LSQA swap.

Swap Completion
The swap completion routine handles all completions for LSQA pages regardless of whether they were processed through swap data sets or through page data sets. Swap Completion processes AIAs in the order in which they are received. AIAs that are grouped for swap data sets are rechained prior to being returned to RSM.

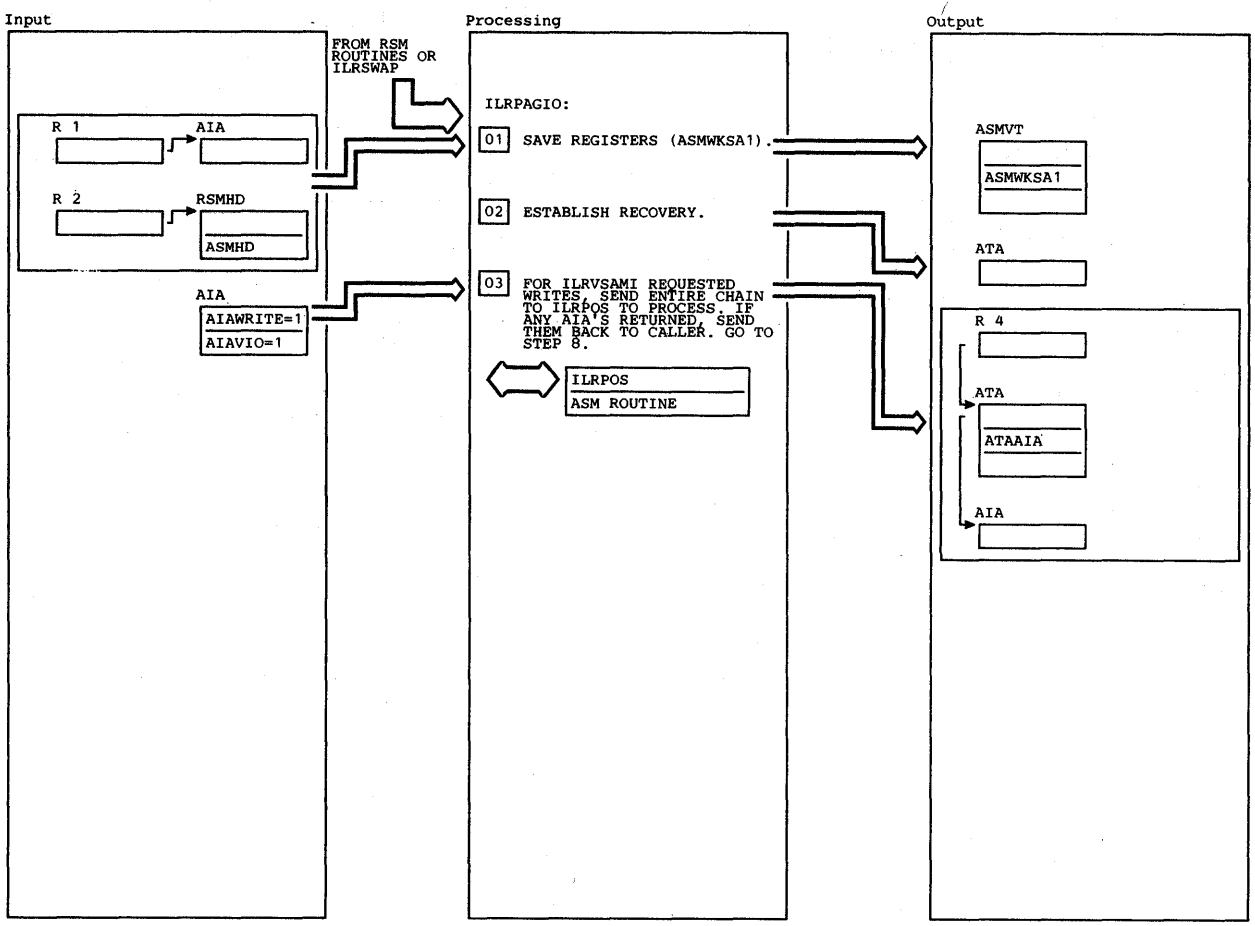
Completed swap-in AIAs are returned to RSM immediately unless the I/O retry flag (indicating IOS failure) is set in the AIA. In this case, the requests are retried by queueing the AIAs to the SARWAITQ or the ASMSTAGQ, depending on whether a swap data set or a page data set is being used.

Swap Completion queues normal swap-out completions to the Swap Capture Queue (ASHCAPQ). Swap-out completions that fail are retried by sending them to the SARWAITQ (for swap data sets) or the ASMSTAGQ (for page data sets). If there was an error and no more swap data sets are available, the AIA is sent to the capture queue and the captured error flag is set. When all AIAs for a particular address space have been placed on ASHCAPQ, Swap Completion determines if any AIAs have error flags set. If no errors occurred Swap Completion returns the entire group of AIAs to RSM. Otherwise, Swap Completion puts the entire group on the ASHSWPAQ to be retried by ILRSLSQA.



25.x. – Module
25.x.y. – Entry point in module 25.x.

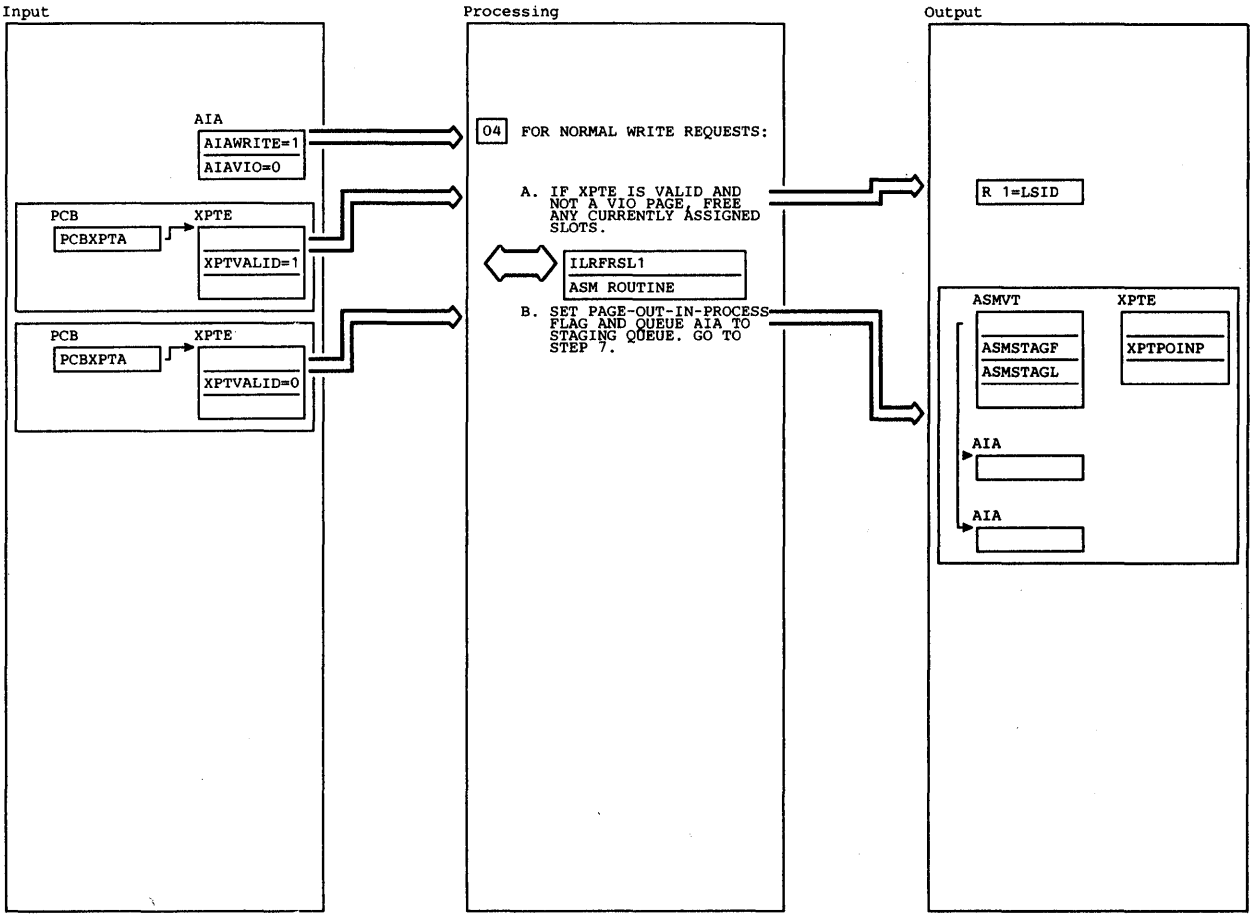
Figure 2-57. I/O Control Overview



Notes	Routine	Label	Ref
01 CALLED BY RSM FOR A PAGING OPERATION, OR FROM ILRSWAP FOR PAGING THE NON-LSQA PAGES ON A SWAP OPERATION. REGISTERS ARE SAVED IN THE ASMVT SAVE AREA DEFINED FOR THIS MODULES USE.			
02 SETFRR IS ISSUED FOR RECOVERY PURPOSES. ILRIOFRR RECOVERY ROUTINE HANDLES ERRORS OCCURRING IN ILRPAGIO.			
03 FOR VIO REQUESTED WRITES (AIAVIO=1 AND AIAWRITE=1) THE ENTIRE CHAIN OF AIA'S IS SENT TO THE VIO PAGE OPERATIONS STARTER SINCE THESE TYPES OF REQUESTS CANNOT BE MIXED WITH ANY OTHER TYPE. ILRPOS DETERMINES IF THE PAGES MAY BE STARTED IMMEDIATELY AND HOLDS THEM OR QUEUES THEM TO THE STAGING QUEUE AS APPROPRIATE. IF ANY ERRORS ARE DETECTED, THE AIA AND ANY SUCCEEDING AIA'S ARE RETURNED TO THIS MODULE, AND THEY WILL BE RETURNED TO THE CALLER AT EXIT. GO TO STEP8.	ILRPOS	ILRPOS	

Notes	Routine	Label	Ref

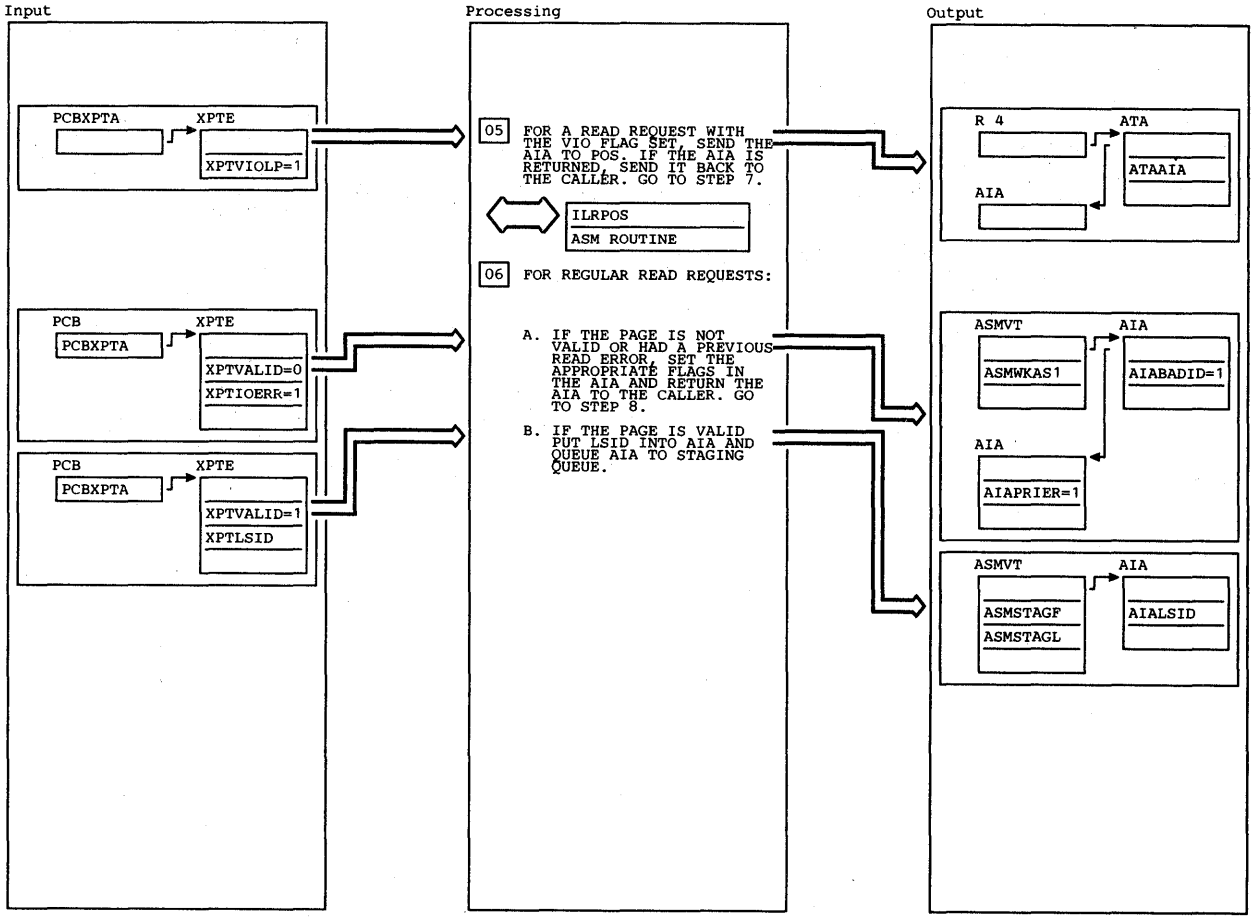
Diagram 25.1 ILRPAGIO (Part 1 of 4)



Notes	Routine	Label	Ref
04 FOR NORMAL WRITE REQUESTS (AIAWRITE=1 AND AIAVIO=0):			
A. IF THE XPTE IS VALID (A PREVIOUS WRITE HAS BEEN DONE FOR THIS PAGE) AND IS NOT A VIO DEFINED PAGE (XPTVALID='1') THE SLOT THAT IS CURRENTLY ASSIGNED TO THE PAGE IS FREED BY ILRFRSL1, BECAUSE SOME OTHER SLOT WILL BE USED FOR THE WRITE.	ILRFRSLT	ILRFRSL1	
B. MARK XPTE BEING A 'PAGE OUT IN PROGRESS' PAGE (XPTPOINP='1') AND PUT AIA ON THE STAGING QUEUE (ASMSTAGQ). GO TO STEP 7 TO CHECK FOR MORE AIA'S TO PROCESS.			

Notes	Routine	Label	Ref

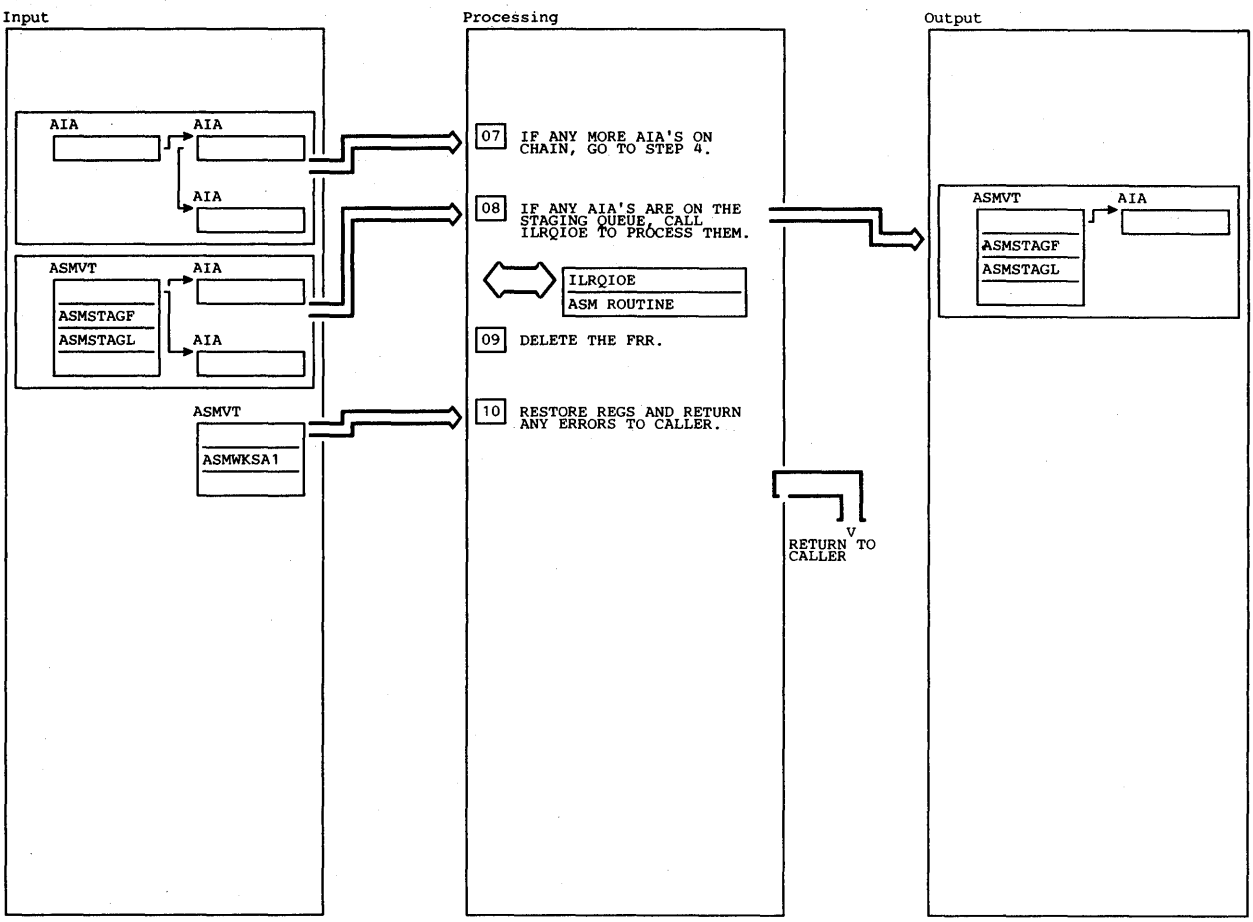
Diagram 25.1 ILRPAGIO (Part 2 of 4)



Notes	Routine	Label	Ref
05 FOR READ REQUESTS THAT HAVE VIO PAGES IN THE SLOT, THE AIA MUST BE SENT TO ILRPOS TO BE PROCESSED AS IN THE ABOVE STEPS. HOWEVER, IN THIS CASE EACH AIA FOR A VIO PAGE (XPTVIOLP='1') MUST BE SENT INDIVIDUALLY SINCE THEY MAY BE MIXED WITH OTHER TYPES OF REQUESTS. IF THE AIA IS RETURNED BY ILRPOS SOME TYPE OF ERROR WAS DETECTED AND THE AIA AND ANY SUCCEEDING AIA'S WILL BE RETURNED TO THE CALLER AT EXIT. GO TO STEP 7 TO CHECK FOR MORE AIA'S TO PROCESS.	ILRPOS	ILRPOS	
06 FOR REGULAR READ REQUESTS (XPTWRITE=0 AND XPTVIOLP=0).			
A. CHECK TO SEE IF THE PAGE SUFFERED A PREVIOUS READ ERROR AT SWAP IN TIME (XPTIOERR=1) OR IF IT IS NOT VALID (XPTVALID=0). IF EITHER CONDITION IS DETECTED SET THE CORRESPONDING ERROR FLAG IN THE AIA (AIAPRIER OR AIABADID) AND RETURN THE AIA AND ANY SUCCEEDING AIA'S TO THE CALLER AT EXIT. GO TO STEP 8 TO CHECK FOR WORK ON THE STAGING QUEUE.			
B. IF THE XPTE IS VALID (XPTVALID='1') THE LSID (LOGICAL SLOT IDENTIFIER) MUST BE COPIED FROM THE XPTE TO THE AIA. THE AIA IS THEN PLACED ON THE STAGING QUEUE.			

Notes	Routine	Label	Ref

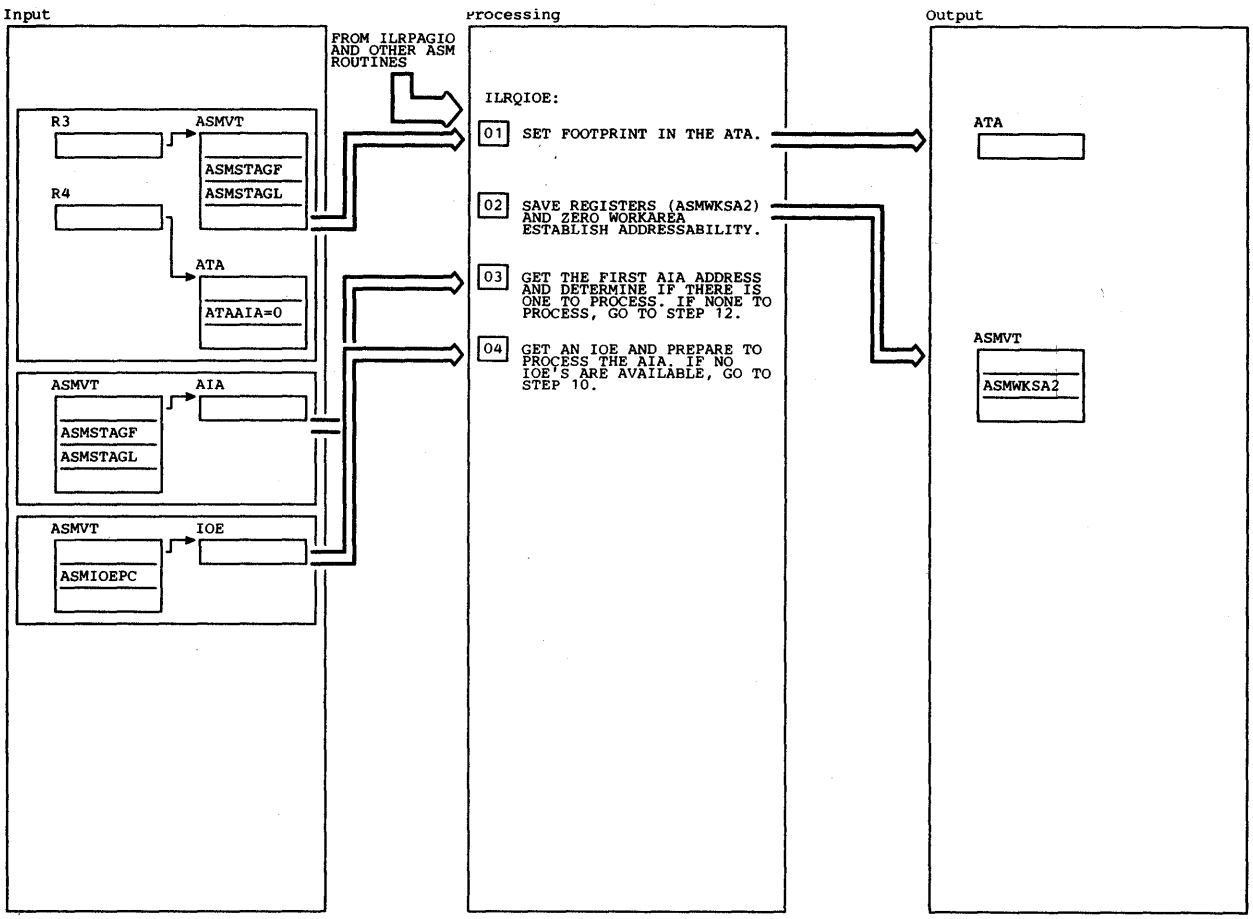
Diagram 25.1 ILRPAGIO (Part 3 of 4)



Notes	Routine	Label	Ref
07 IF ANY MORE AIA'S REMAIN TO BE PROCESSED (AIANXAIA NOT= 0), GO TO STEP 4.			
08 IF ANY AIA'S WERE PUT ON THE STAGING QUEUE, EITHER BY ILRPAGIO ITSELF OR BY ILRPOS, CALL ILRQIOE TO START THE PROCESSING.	ILRPAGIO	ILRQIOE	
09 REMOVE RECOVERY.			
10 RETURN TO CALLER.			

Notes	Routine	Label	Ref

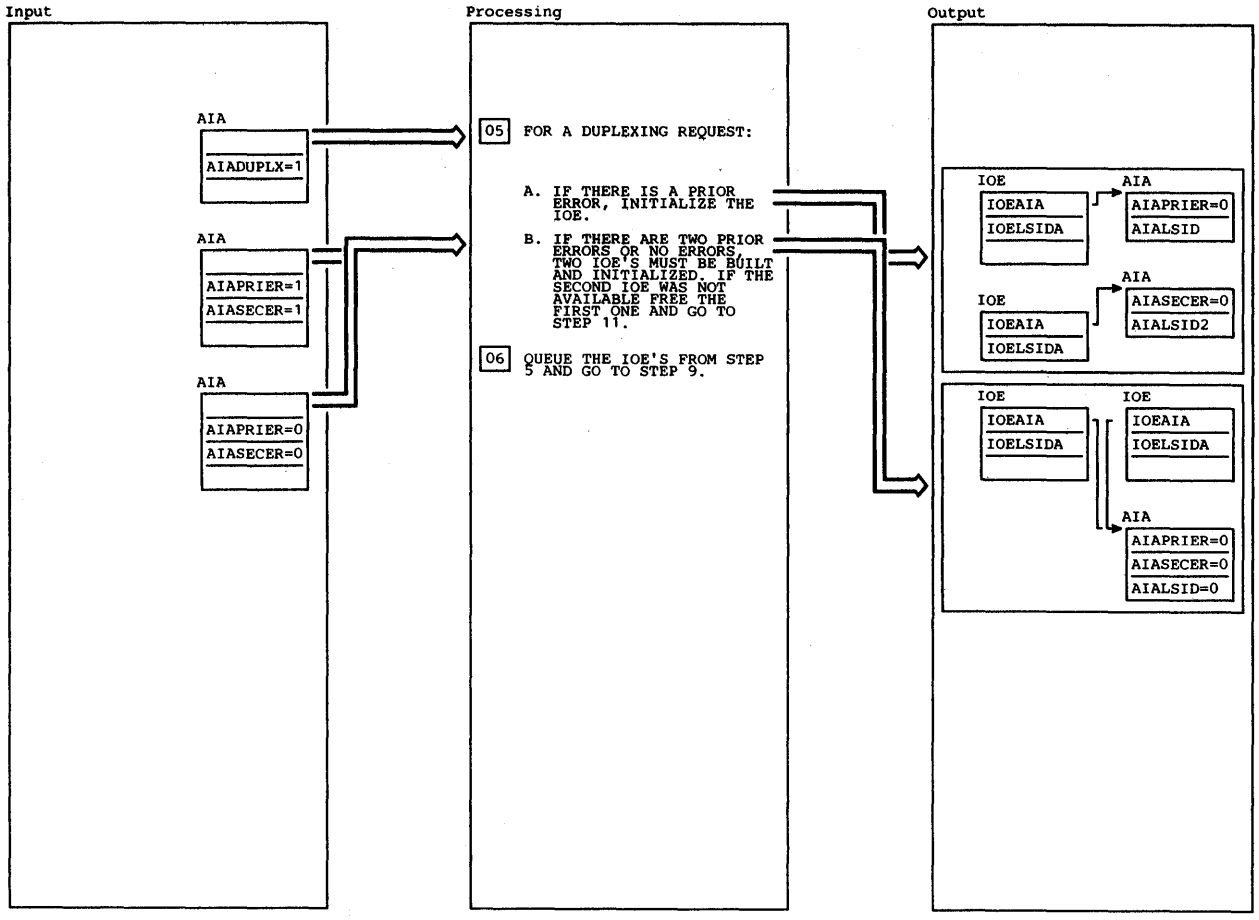
Diagram 25.1 ILRPAGIO (Part 4 of 4)



Notes	Routine	Label	Ref
01 ILRQIOE IS ENTERED FROM ILRESTRT, ILRPAGIO, ILRSWAP, AND ILRPAGCM WITH REGISTER 3 POINTING TO THE ASMVT, REGISTER 4 POINTING TO THE ATA AND AIA'S TO BE PROCESSED CHAINED FROM THE STAGING QUEUE (ASMSTAGQ). IOE'S WILL BE BUILT FOR THE AIA'S ACCORDING TO THE INDICATORS IN THEM. AN INDICATOR IS SET IN ATA FOR RECOVERY PURPOSES. ILRCQIOE (AN ENTRY IN ILRIOFRR RECOVERY ROUTINE) HANDLES ERRORS OCCURRING IN ILRQIOE.			
02 REGISTERS ARE SAVED AT ASMWKSA2 AND THE WORK AREA IS ZEROED.			
03 THE CHAIN OF AIA'S TO PROCESS IS QUEUED TO THE DOUBLE HEADED ASMSTAGQ. THE FIRST AIA (ASMSTAGF) IS PICKED UP AND IF NON-ZERO IT IS PROCESSED. IF THERE ARE NO AIA'S GO TO STEP 12.			
04 ISSUE THE ILRGMA MACRO TO GET AN IOE. THE IOE'S ARE TAKEN FROM A NON-EXPANDABLE POOL CONTROLLED IN THE ASMVT. THE IOE WILL BE RETURNED VIA REGISTER 1, BUT IF IT IS ZERO THERE ARE NO IOE'S AVAILABLE AND PROCESSING CONTINUES AT STEP 10.	ILRGMA		

Notes	Routine	Label	Ref

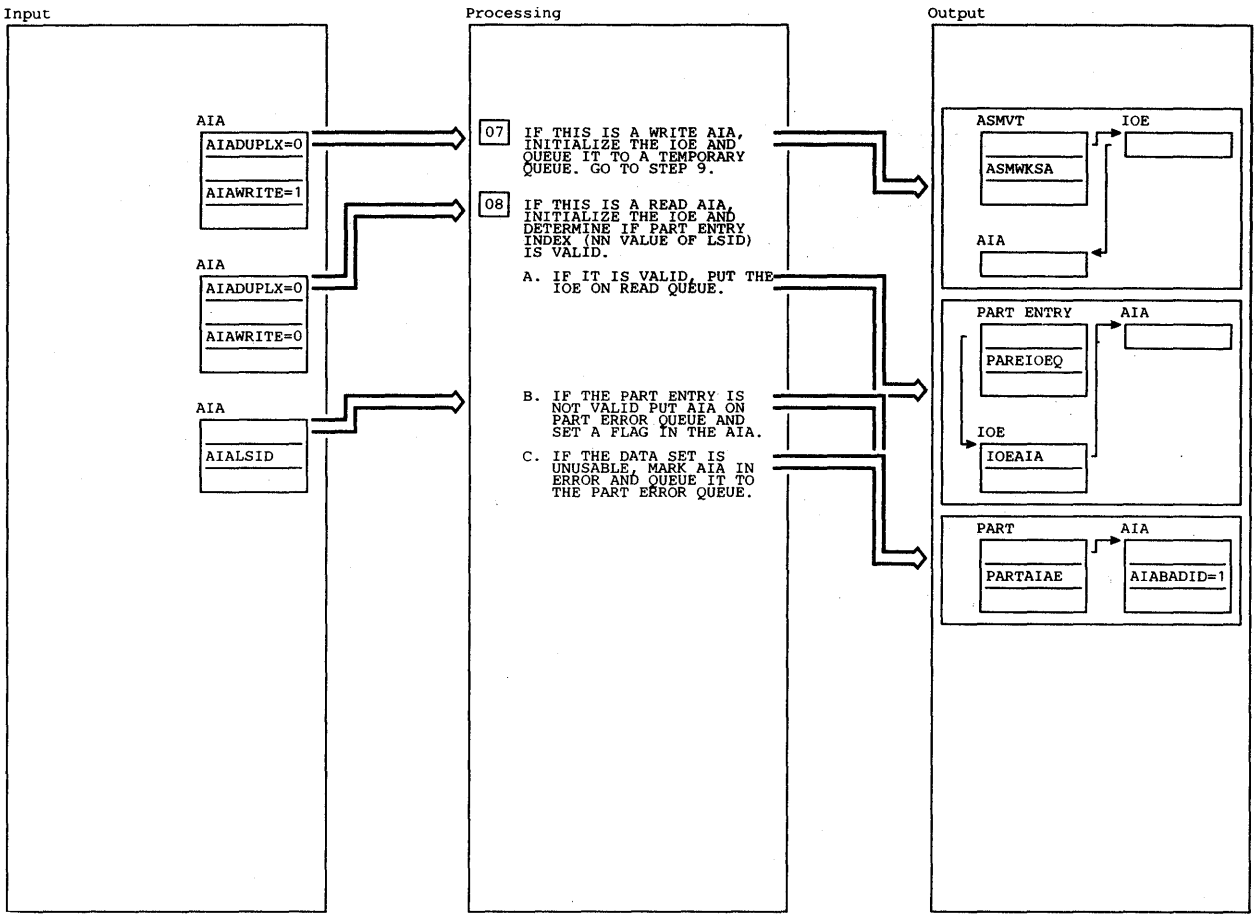
Diagram 25.1.1 ILRQIOE (Part 1 of 4)



Notes	Routine	Label	Ref
05 IF THIS IS A DUPLEXING REQUEST A CHECK IS MADE TO SEE IF THERE WERE ANY ERRORS. THE ERROR BITS INDICATE THAT AN ATTEMPT WAS MADE PREVIOUSLY BUT AN I/O ERROR OCCURRED AND IT MUST BE RETRIED. A. IF ONLY ONE OF THE ERROR BITS (AIAPRIER OR AIASECER) ARE ON THEN ONLY THAT ONE HAS TO BE RETRIED. A PRIMARY ERROR (AIAPRIER) INDICATES AN ERROR GOING TO THE PLPA OR COMMON DATA SET WHILE A SECONDARY ERROR INDICATES AN ERROR GOING TO THE DUPLEX DATA SET. ONLY ONE IOE WILL BE BUILT FOR THE ONE RETRY. B. IF BOTH ERROR BITS ARE ON OR IF NEITHER BIT IS ON THE REQUESTS MUST BE WRITTEN TO BOTH DATA SETS. ANOTHER IOE IS OBTAINED AND THEN BOTH ARE BUILT. IF A SECOND IOE IS NOT AVAILABLE THE FIRST IS GIVEN BACK AND PROCESSING CONTINUES AT STEP 11.			
06 THE IOE'S FROM STEP 5 ARE PUT ON LOCAL QUEUES AND A LOCAL FLAG (WRITEFLG) IS TURNED ON TO INDICATE THAT WRITE IOE'S MUST BE PUT ON THE PART HEADER QUEUES. GO TO STEP 9.			

Notes	Routine	Label	Ref

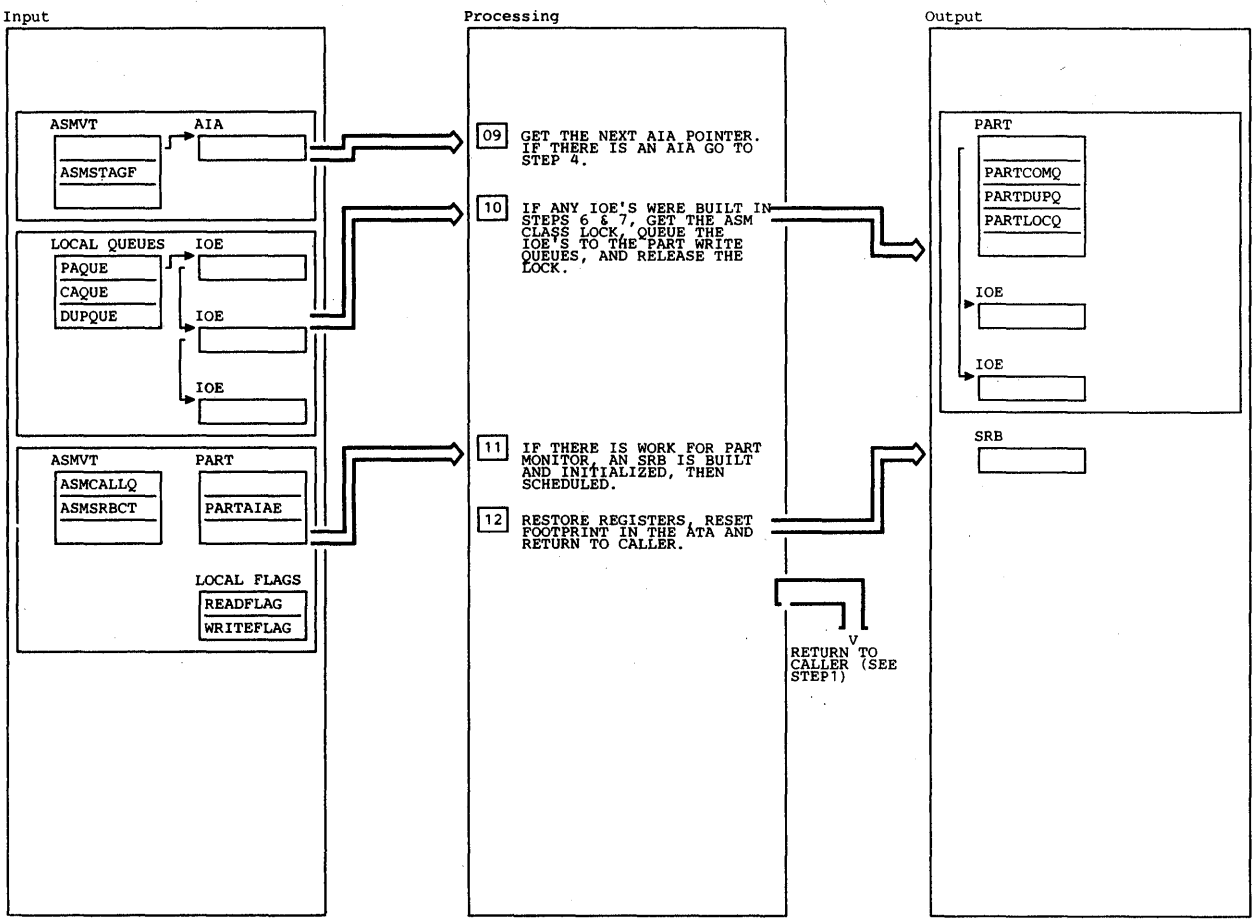
Diagram 25.1.1 ILRQIOE (Part 2 of 4)



Notes	Routine	Label	Ref
07 IF THIS IOE IS FOR A WRITE REQUESTS (AIAWRITE=1) INITIALIZE THE IOE AND QUEUE IT FOR THE LOCAL DATA SET. TURN ON THE LOCAL FLAG (WRITEFLG) AND GO TO STEP 9.			
08 IF THIS IS A READ REQUEST (AIAWRITE=0) CHECK THAT THE PAGE DATA SET (PART ENTRY) INDICATED EXISTS. THE SECOND BYTE (NN PORTION) OF LSID IS AN INDEX TO THE PART ENTRY.			
A. IF THE INDEX ENTRY IS WITHIN THE RANGE OF PART ENTRIES IN USE (PARTEUSE), IT IS VALID. THE IOE IS INITIALIZED AND PUT ON THE PART ENTRY READ QUEUE (PAREIOEQ). THE IORBS CHAINED FROM THE PART ENTRY ARE CHECKED TO DETERMINE THAT ONE IS AVAILABLE (IORFUSE=0). IF ONE IS, A LOCAL FLAG (READFLAG) IS SET TO INDICATE TO SCHEDULE PART MONITOR.			
B. IF THE INDEX ENTRY IS GREATER THAN PARTEUSE, THEN THE PART ENTRY IS INVALID. THE IOE IS RETURNED, THE AIA IS PUT ON THE PART ERROR QUEUE, AND AIABADID IS SET TO 1.			
C. IF THE DATA SET THESE REQUESTS ARE BEING MADE AGAINST IS NOT USABLE (PARENUSE=1), AIABADID IS SET TO 1 AND THE AIA IS QUEUED TO THE PART ERROR QUEUE (PARTAIAE).			

Notes	Routine	Label	Ref

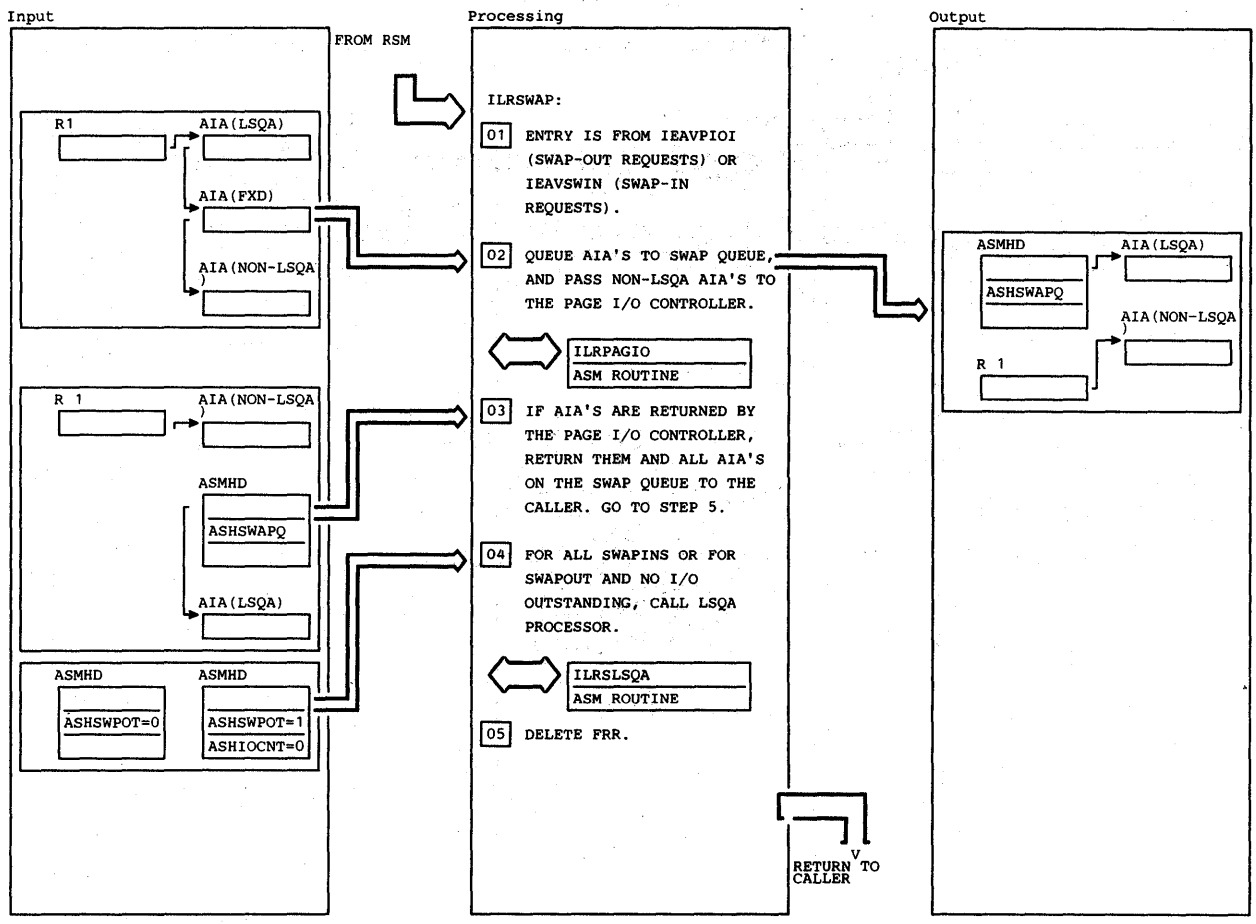
Diagram 25.1.1 ILRQIOE (Part 3 of 4)



Notes	Routine	Label	Ref
09 GET THE NEXT AIA FROM THE QUEUE (ASMSTAGF). IF THERE IS AN AIA TO PROCESS GO TO STEP4. IF THERE IS NO AIA SET THE LAST POINTER (ASMSTAGL) TO ZERO.			
10 AT THIS POINT ALL AIA'S HAVE BEEN PROCESSED OR THERE ARE NO IOE'S. THE INTERNAL FLAG (WRITEFLG) IS CHECKED TO SEE IF ANY WRITE IOE'S WERE PUT ON THE LOCAL QUEUES. IF WRITEFLG=0 THERE ARE NONE SO GO TO STEP 12. THE IOE'S ARE PUT ON THE WRITE QUEUES IN THE PART HEADER (PARTLOCQ, PARTCOMQ PARTDUPQ).			
11 THE PART MONITOR (ILRPTM) IS SCHEDULED IF THERE IS NO I/O OUTSTANDING (ASMIOCNT=0) IF ANY AIAS WERE PUT ON THE PART ERROR QUEUE (PARTAIAE), OR IF THE LOCAL FLAGS (READFLAG OR WRITEFLAG) ARE ON. HOWEVER, IF THE SRB FOR PART MONITOR IS ALREADY SCHEDULED(ASMSRBCT-=0) IT WILL NOT BE SCHEDULED AGAIN. ALSO IF COMPLETION (ILRPAGCM) IS THE CALLER (ASMCALLQ=1) THE SRB WILL NOT BE SCHEDULED, BECAUSE COMPLETION SCHEDULES IT.			
12 RESTORE REGISTERS FROM ASMWKSA2, RESET THE FOOTPRINT IN THE ATA AND RETURN TO THE CALLER.			

Notes	Routine	Label	Ref

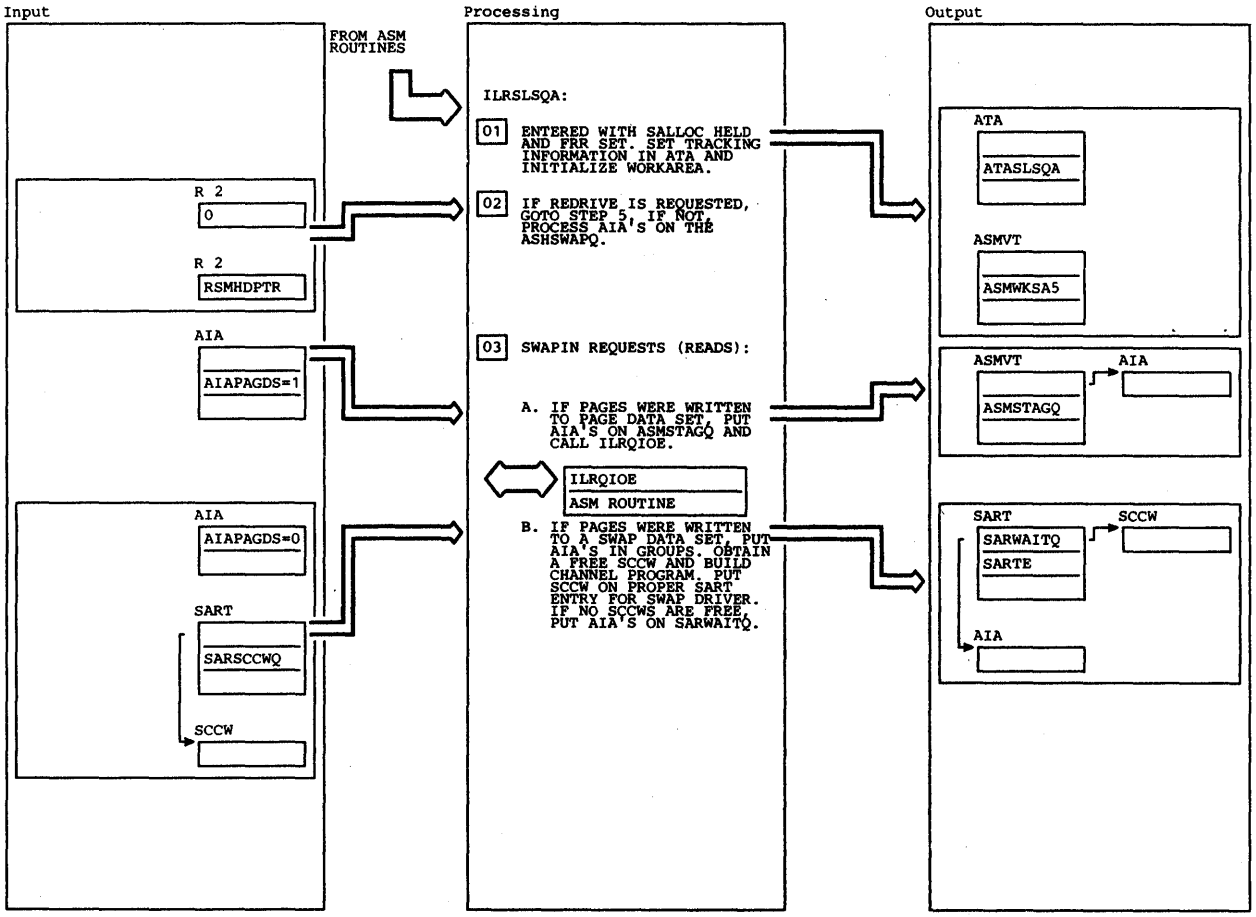
Diagram 25.1.1 ILRQIOE (Part 4 of 4)



Notes	Routine	Label	Ref
01 ILRSWAP INITIALIZES THE SWAP REQUESTS RECEIVED FROM RSM. ESTABLISHES ADDRESSABILITY, SAVES CALLER'S REGISTERS IN THE ASMVT, AND ISSUES SETFRR TO ESTABLISH A RECOVERY ENVIRONMENT. ILRCSWAP (ENTRY IN ILRSWP01 RECOVERY ROUTINE) HANDLES ERRORS OCCURRING IN ILRSWAP.			
02 THE INPUT AIA'S ARE QUEUED TO THE ASM HEADER SWAP QUEUE (ASHSWAPQ). THOSE AIA'S THAT REPRESENT NON-LSQA PAGES ARE DEQUEUED AND SENT TO ILRPAGIO, THE PAGE I/O CONTROLLER, TO BE PROCESSED. NON-LSQA PAGES ARE WRITTEN TO PAGE DATA SETS, NOT SWAP DATA SETS.	ILRPAGIO	ILRPAGIO	
03 ANY AIA'S RETURNED BY ILRPAGIO MUST BE SENT BACK TO THE CALLER ALONG WITH ANY OTHER AIA'S ON THE SWAP QUEUE. THE AIA'S RETURNED REPRESENT AN ERROR AIA AND THE SUBSEQUENTLY CHAINED AIA'S. SET A RETURN CODE OF 4 AND GO TO STEP 5.			

Notes	Routine	Label	Ref
04 IF THERE WERE NO NON-LSQA AIA'S TO BE PROCESSED FOR A SWAP OUT REQUEST AND NO I/O WAS OUTSTANDING AT THE TIME THE REQUEST WAS MADE, OR IF THE REQUEST WAS FOR A SWAPIN, CALL ILRSLSQA TO PROCESS THE LSQA PAGES BEING SWAPPED. IF THERE IS OUTSTANDING I/O, ILRPAGCM WILL CALL ILRSLSQA WHEN I/O COMPLETES.	ILRSWAP	ILRSLSQA	
05 DELETE THE FRR AND RESTORE CALLER'S REGISTERS.			

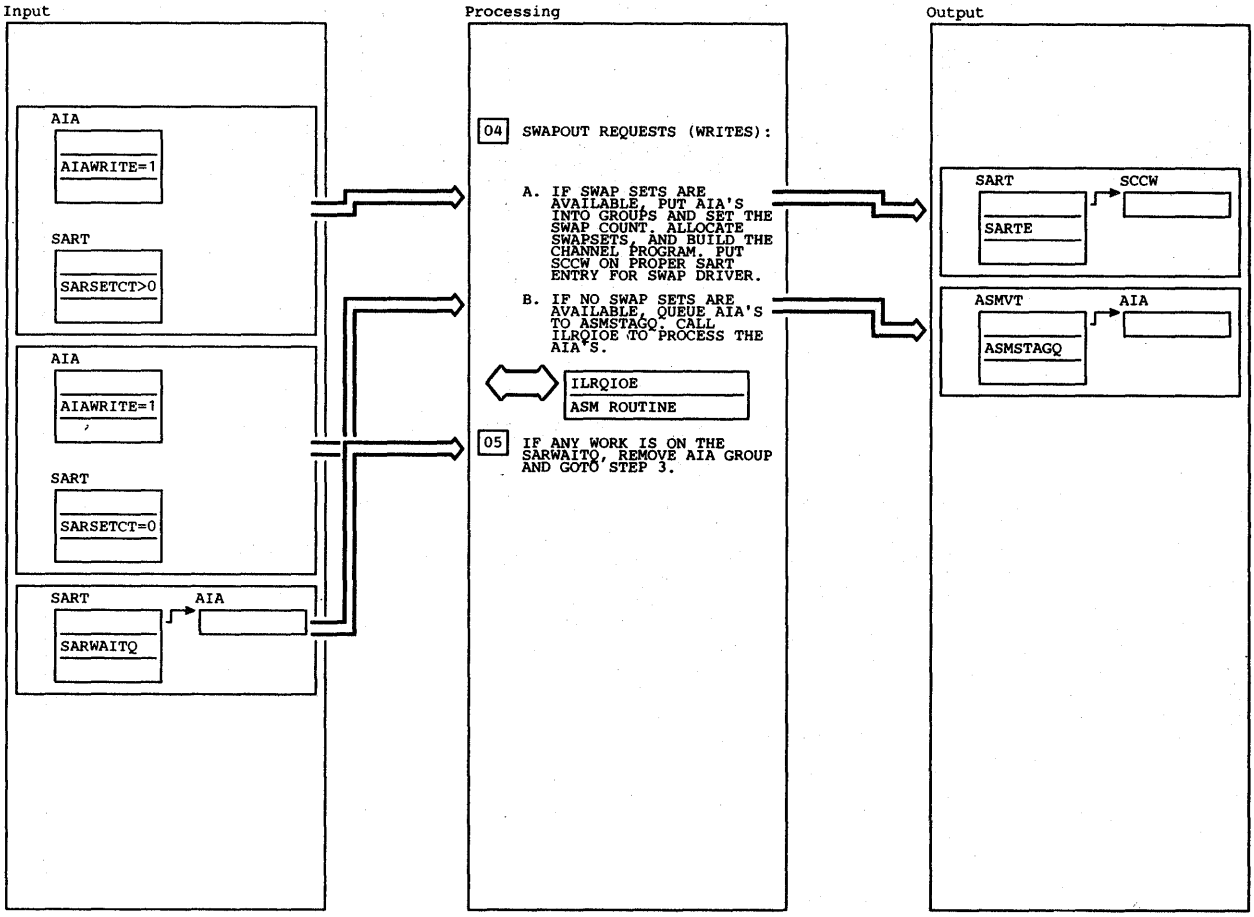
Diagram 25.2 ILRSWAP (Part 1 of 1)



Notes	Routine	Label	Ref
01 ILRSLSQA PROCESSES THE LSQA PAGES OF A SWAP PROCESS. ENTERED WITH SALLOC HELD AND FRR PREVIOUSLY SET. SET TRACKING INFORMATION IN THE ATA FOR RECOVERY AND CLEAR THE WORKAREA IN THE ASMVT. ILRCSLSQ (ENTRY IN ILRSWP01 RECOVERY ROUTINE) HANDLES ERRORS OCCURRING IN ILRSLSQA.			
02 IF R2=0, ENTRY WAS MADE FOR PROCESSING WORK THAT WAS PREVIOUSLY LEFT ON THE SARWAITQ BECAUSE OF A LACK OF RESOURCES (SCCW'S). GOTO STEP 5. IF R2¬=0, IT CONTAINS AN RSMHD POINTER. ENTRY WAS MADE TO PROCESS AIA'S FOR THAT ADDRESS SPACE, AND THE AIA'S ARE ON THE ASHSWAPQ.			
03 IF PAGES WERE WRITTEN TO A PAGE DATA SET AT SWAPOUT TIME (AIAPAGDS=1), THE AIA'S ARE PLACED ON THE ASMSTAGQ AND ILRQIOE IS CALLED TO PROCESS THEM. IF THE PAGES WERE WRITTEN TO A SWAP DATA SET A SCCW IS OBTAINED AND CHANNEL PROGRAMS BUILT. THE SCCW IS THEN PLACED ON THE PROPER SART ENTRY FOR THE SWAP DRIVER. IF NO SCCW'S ARE FREE, THE AIA'S ARE PUT ON THE SARWAITQ.	ILRPAGIO	ILRQIOE	

Notes	Routine	Label	Ref

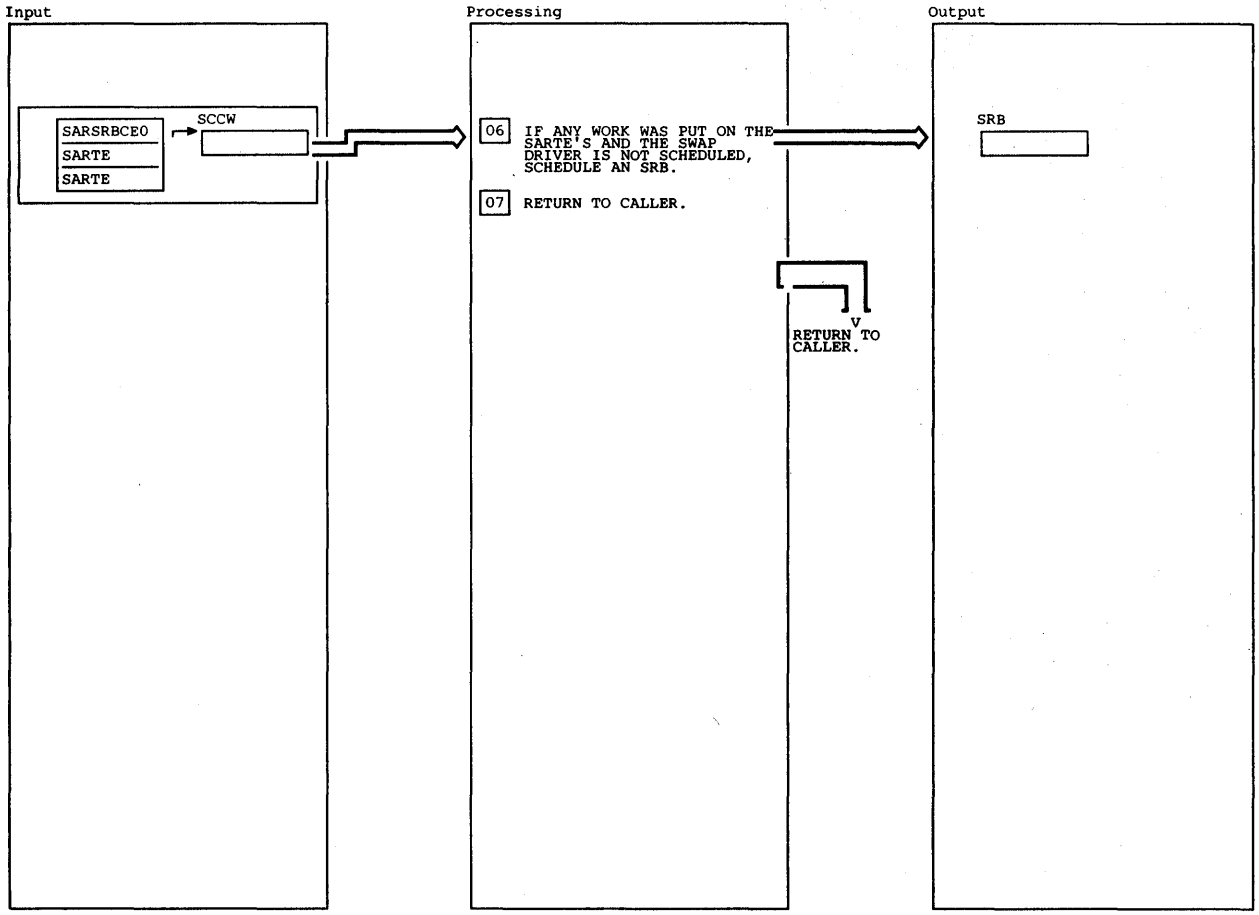
Diagram 25.2.1 ILRSLSQA (Part 1 of 3)



Notes	Routine	Label	Ref
04 FOR SWAP OUT REQUESTS, IF SWAP SETS ARE AVAILABLE THE AIA'S ARE DIVIDED INTO GROUPS AND SWAP SETS ARE ALLOCATED. THE CHANNEL PROGRAM IS BUILT AND THE SCCW PUT ON THE PROPER SART ENTRY FOR THE SWAP DRIVER. IF NO SWAP SETS ARE AVAILABLE, THE AIA'S ARE PLACED ON THE ASMSTAGQ AND ILRQIOE IS CALLED TO PROCESS THEM.	ILRPAGIO	ILRQIOE	
05 IF ANY WORK WAS LEFT BEHIND PREVIOUSLY (CAUSED BY AN SCCW SHORTAGE), REMOVE A GROUP FROM THE SARWAITQ AND GOTO STEP 3.			

Notes	Routine	Label	Ref

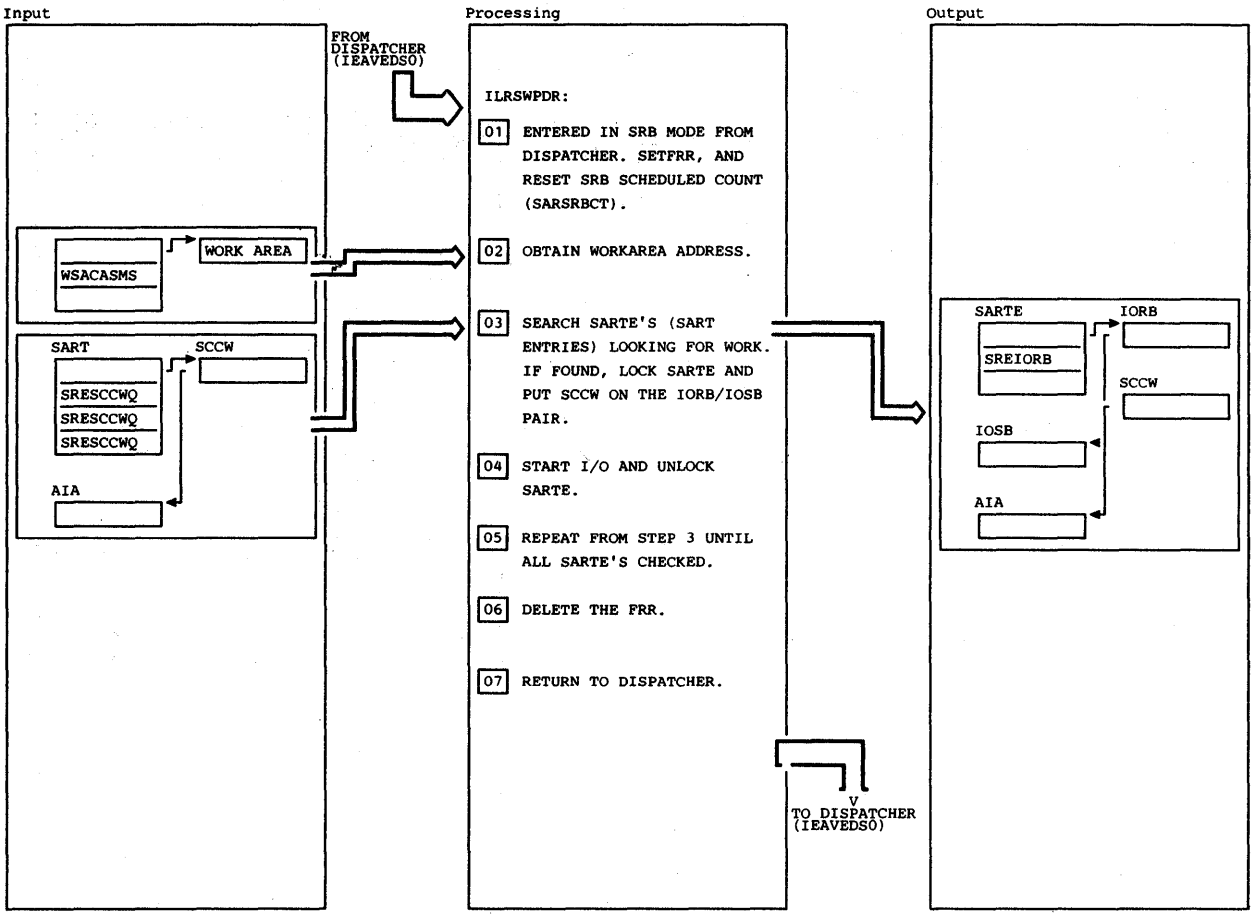
Diagram 25.2.1 ILRSLSQA (Part 2 of 3)



Notes	Routine	Label	Ref
06 IF ANY WORK WAS PUT ON A SART ENTRY, AND THE SWAP DRIVER IS NOT CURRENTLY SCHEDULED (SARSRBCT=0), SCHEDULE AN SRB AND SET THE SCHEDULED COUNT (SARSRBCT=1) TO PREVENT ANOTHER CPU FROM SCHEDULING.			
07 RETURN TO CALLER.			

Notes	Routine	Label	Ref

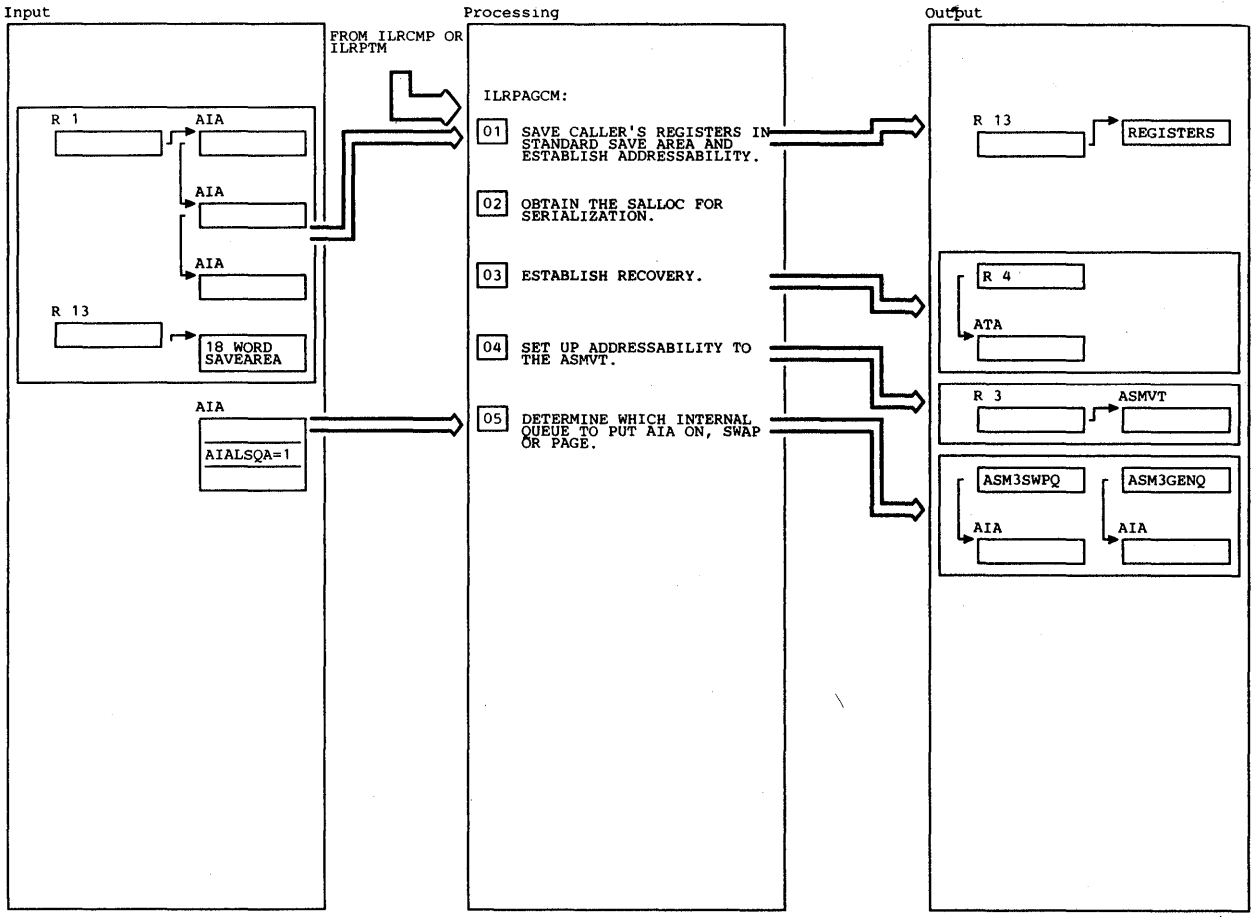
Diagram 25.2.1 ILRSLSQA (Part 3 of 3)



Notes	Routine	Label	Ref
01 ENTERED IN SRB MODE FROM DISPATCHER, ILRSWPDR FINDS AND STARTS I/O REQUESTS TO THE SWAP DATA SETS (ILRPTM AND ILRSRT TOGETHER DO THIS PROCESSING FOR THE PAGE DATA SETS). SET FRR FOR RECOVERY AND RESET THE SCHEDULED COUNT (SARSRBCT=0) SO ANY SUBSEQUENT WORK PUT ON QUEUES WILL CAUSE A RE-SCHEDULE OF SWAP DRIVER. ILRSWP01 RECOVERY ROUTINE HANDLES ERRORS OCCURRING IN ILRSWPDR.			
02 OBTAIN ADDRESS OF WORKAREA USED TO STORE REGISTERS ACROSS START I/O.			
03 SEARCH SART ENTRIES (SARTE) LOOKING FOR WORK TO PROCESS. IF A SCCW IS FOUND TO PROCESS (SRESCCWQ) AND AN IORB/IOSB PAIR IS AVAILABLE (ILRFUSE=0) TRY TO LOCK THE SARTE TO PREVENT INTERFERENCE FROM ANOTHER COPY OF THE SWAP DRIVER. IF PREVIOUSLY LOCKED GO TO NEXT SARTE TO PROCESS. IF LOCK IS SUCCESSFULLY OBTAINED, PUT			

Notes	Routine	Label	Ref
SCCW'S ON IORB-IOSB PAIRS AND POINT TO FIRST CCW.			
04 ISSUE SIO (START I/O) MACRO TO IOS TO START THE OPERATION.	SIO		
05 UNLOCK THE SARTE AND REPEAT FROM STEP 3 UNTIL ALL SART ENTRIES HAVE BEEN CHECKED.			
06 ISSUE SETFRR DELETE TO RESET THE RECOVERY ENVIRONMENT.			
07 RETURN TO THE DISPATCHER.			

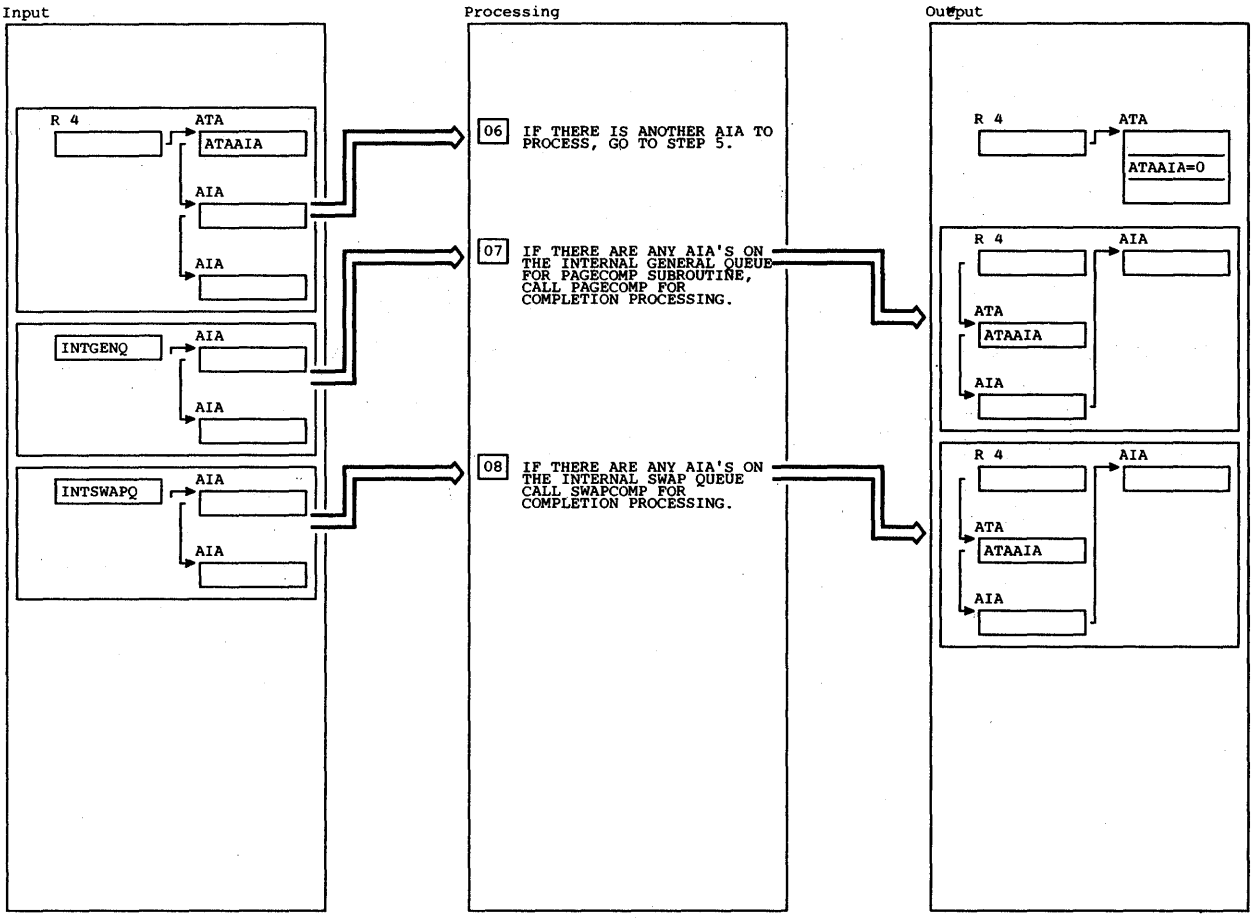
Diagram 25.3 ILRSWPDR (Part 1 of 1)



Notes	Routine	Label	Ref
01 CALLED BY I/O COMPLETION, OR BY PART MONITOR, WITH REGISTER 1 POINTING TO A CHAIN OF AIA'S AND REGISTER 13 POINTING TO AN 18 WORD SAVEAREA. THE REGISTERS ARE SAVED IN THE CALLER'S SAVE AREA.			
02 OBTAIN THE SALLOC FOR CONTROL BLOCK SERIALIZATION.			
03 ISSUE SETFRR TO ESTABLISH ERROR RECOVERY. ILRIOFRR HANDLES ERRORS OCCURRING IN ILRPAGCM.			
04 GET THE ADDRESS OF THE ASMVT AND PUT IT IN REGISTER 3, THE STANDARD ASM CONVENTION.			
05 THE AIA'S ARE PUT ON ONE OF TWO PUSH DOWN STACKS. IF THE AIA IS FOR A SWAP LSQA PAGE (AIALSQA=1) IT GOES ON THE SWAP QUEUE (ASM3SWPQ), OTHERWISE IT GOES ON THE GENERAL QUEUE (ASM3GENQ). THE TWO QUEUES ARE MAINTAINED IN THE WORK AREA (ASMWKSA3).			

Notes	Routine	Label	Ref

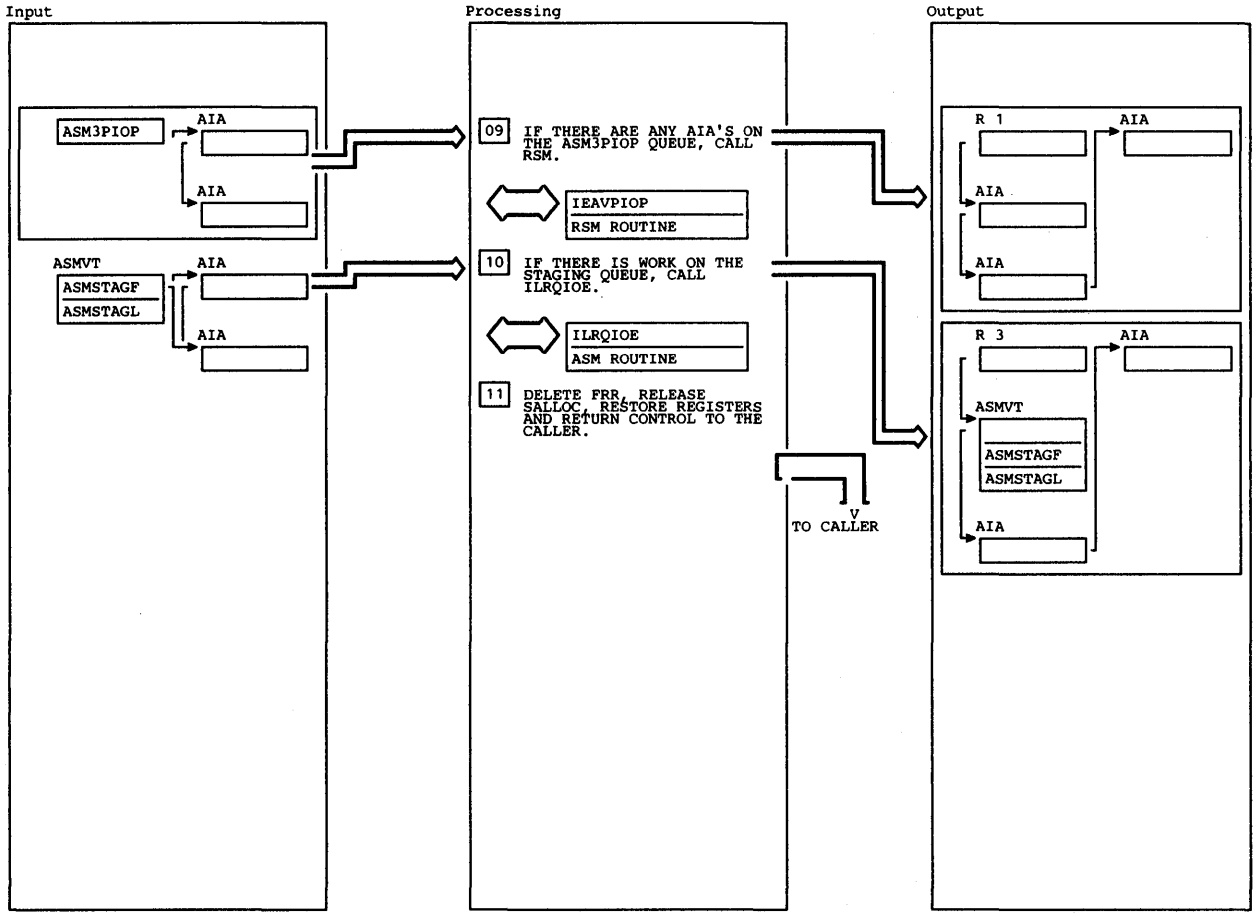
Diagram 25.4 ILRPAGCM (Part 1 of 3)



Notes	Routine	Label	Ref
06 WHILE PROCESSING IN THIS LOOP THE ATAAIA ALWAYS POINTS TO THE NEXT AIA TO PROCESS. PICK UP ATAAIA AND IF THERE IS ANOTHER AIA TO PROCESS GO TO STEP 5.			
07 IF ANY AIA'S WERE PUT ON THE INTERNAL GENERAL QUEUE, PASS THEM TO PAGECOMP FOR COMPLETION PROCESSING. THE AIA'S ARE PASSED VIA THE ATAAIA FIELD.		PAGECOMP	25.4.1
08 IF ANY AIA'S WERE PUT ON THE INTERNAL SWAP QUEUE, PASS THEM TO SWAPCOMP FOR COMPLETION PROCESSING. THE AIA'S ARE PASSED VIA THE ATAAIA FIELD.		SWAPCOMP	25.4.2

Notes	Routine	Label	Ref

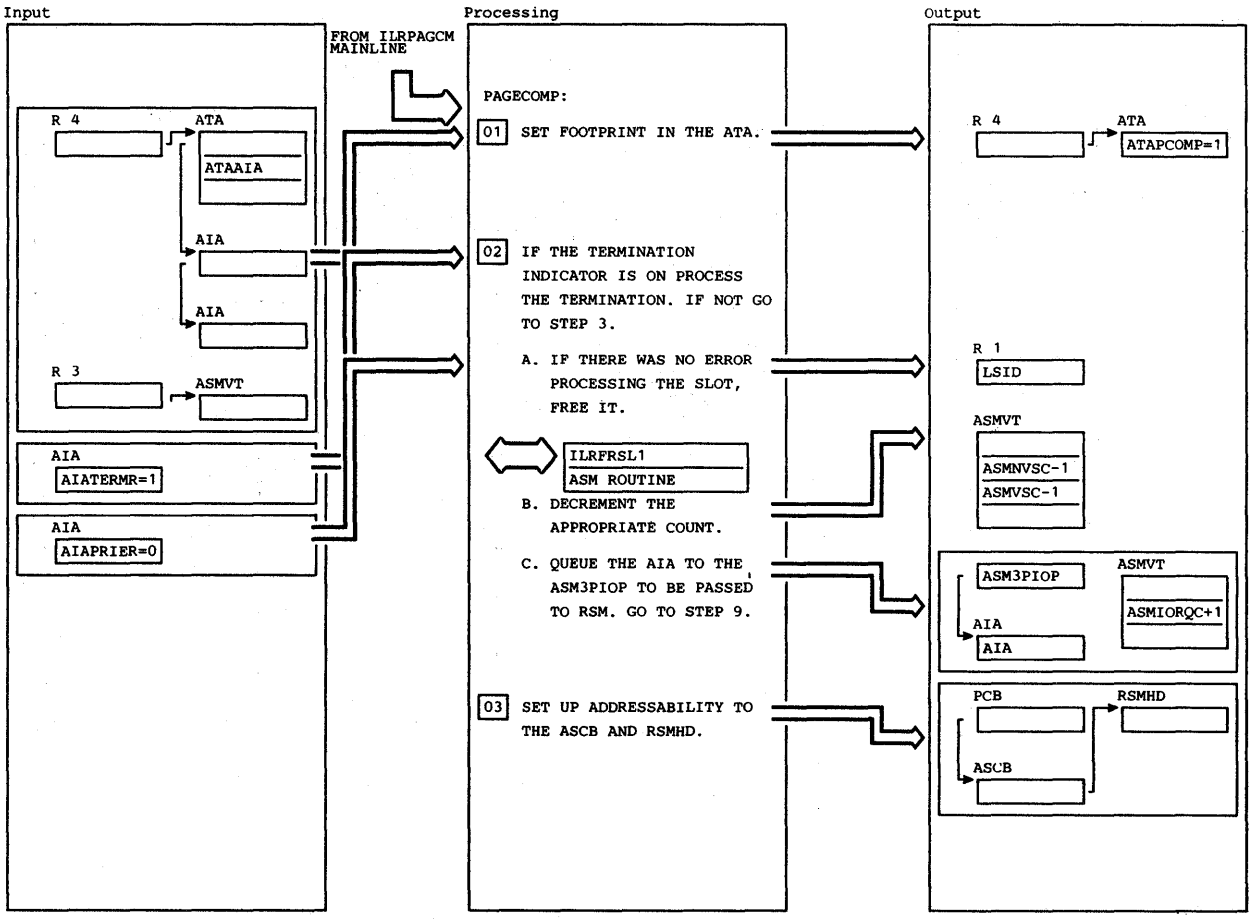
Diagram 25.4 ILRPAGCM (Part 2 of 3)



Notes	Routine	Label	Ref
09 WHILE PROCESSING IN PAGECOMP AND SWAPCOMP SOME AIA'S MAY HAVE BEEN PUT ON THE QUEUE (ASM3PIOP) TO PASS BACK TO RSM (IEAVPIOP). IF ANY ARE ON THE QUEUE THE ADDRESS IN ASM3PIOP IS PUT INTO REGISTER 1 AND CONTROL IS PASSED TO IEAVPIOP.	IEAVPIOP	IEAVPIOP	
10 THE STAGING QUEUE (ASMSTAGQ) IS CHECKED TO SEE IF ANY AIA'S ARE WAITING TO BE PROCESSED. IF THERE IS AN ADDRESS IN THE FIRST POINTER (ASMSTAGF = NON-ZERO) CONTROL IS PASSED TO ILRQIOE TO BUILD IOE'S.	ILRPAGIO	ILRQIOE	
11 THE FRR IS DELETED, THE SALLOC IS RELEASED, THE REGISTERS RESTORED AND CONTROL IS RETURNED TO THE CALLER.			

Notes	Routine	Label	Ref

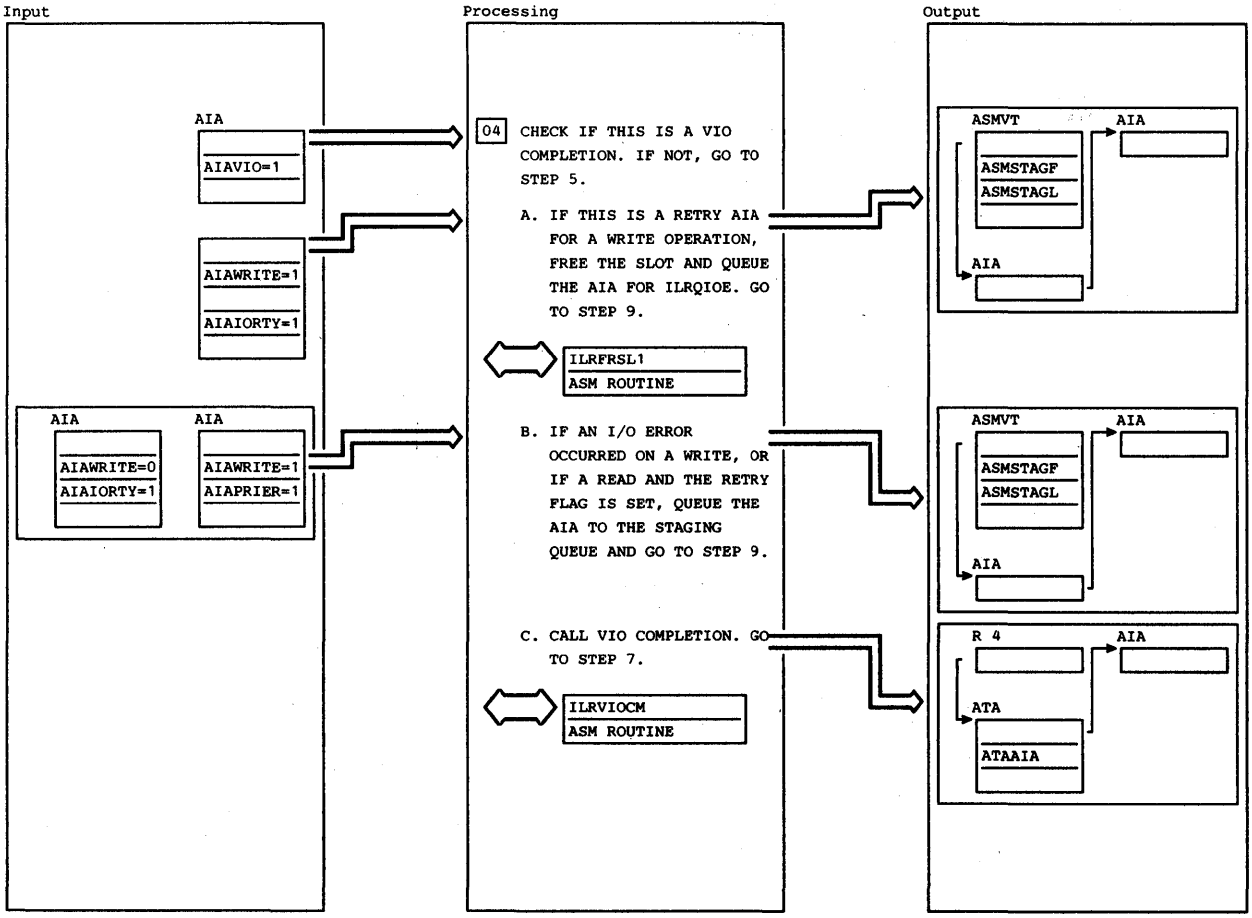
Diagram 25.4 ILRPAGCM (Part 3 of 3)



Notes	Routine	Label	Ref
01 CONTROL IS RECEIVED FROM MAINLINE ILRPAGCM TO PROCESS ALL PAGE COMPLETIONS. REGISTER 3 POINTS TO THE ASMVT AND REGISTER 4 POINTS TO THE ATA WHICH HAS THE ADDRESS OF THE AIA'S TO BE PROCESSED. FOR RECOVERY PURPOSES, PAGECOMP INDICATES IT HAS CONTROL BY SETTING A BIT IN THE ATA.			
02 THE TERMINATION INDICATOR (AIATERMR=1) MEANS THAT THE ADDRESS SPACE WAS TERMINATED AND THAT CONTROL BLOCKS ARE NO LONGER AVAILABLE. THE AIA MUST BE GIVEN BACK TO IEAVPIOP AND:			
A. IF THERE WAS NO ERROR FREE THE SLOT FOR FURTHER USE.	ILRFRSLT	ILRFRSL1	
B. DECREMENT THE VIO OR NON-VIO (ASMVSC OR ASMNVSC) SLOT COUNT DEPENDING ON THE TYPE OF PAGE.			
C. QUEUE THE AIA TO THE ASM3PIOP QUEUE (FOR RETURN TO IEAVPIOP) AND INCREMENT THE			

Notes	Routine	Label	Ref
COMPLETED COUNT (ASMIORQC). GO TO STEP 9.			
03 GET THE ADDRESSES FOR THE ASCB AND RSMHD FOR THE ADDRESS SPACE.			

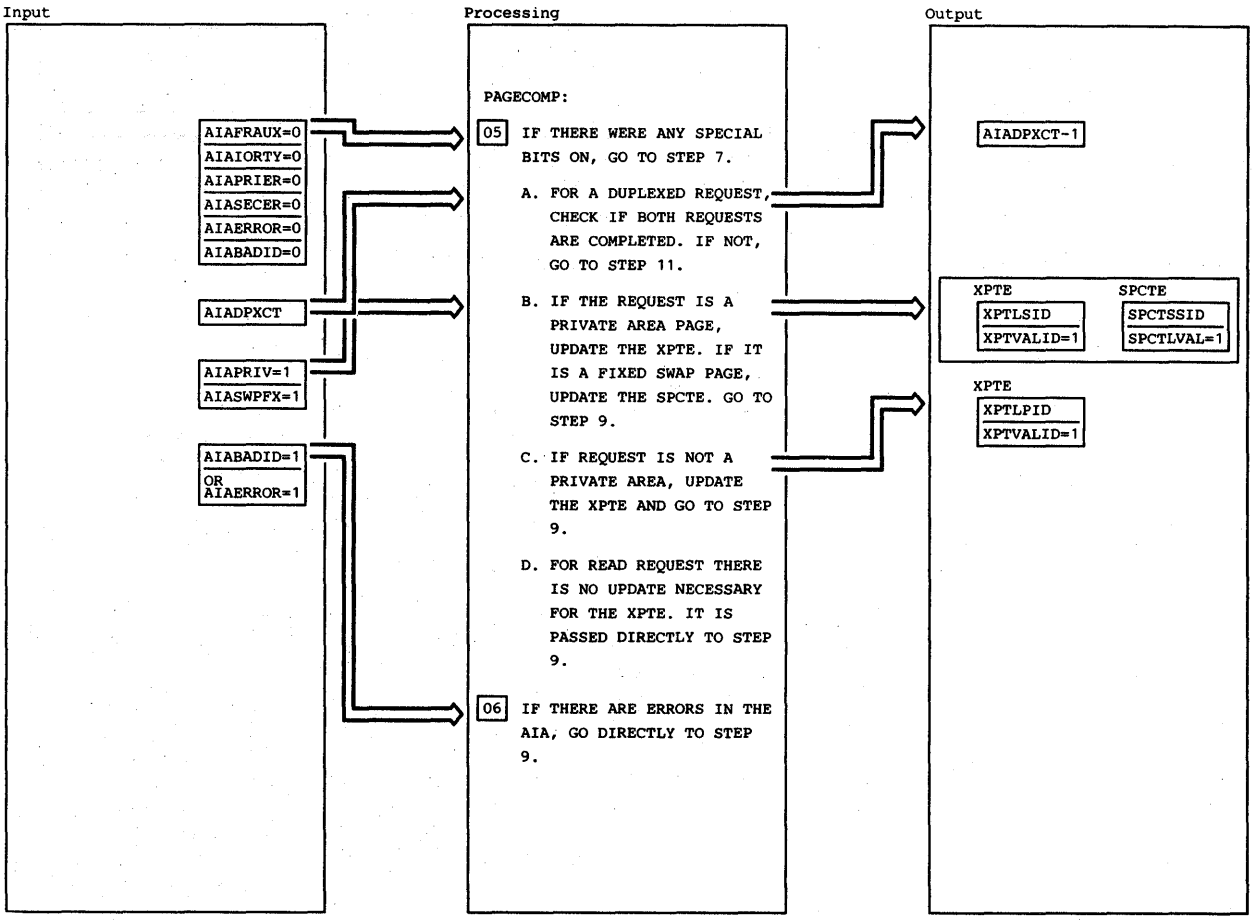
Diagram 25.4.1 PAGECOMP (Part 1 of 6)



Notes	Routine	Label	Ref
04 IF THIS IS NOT A VIO COMPLETION (AIAVIO=0) GO TO STEP5, ELSE:			
A. IF A RETRY WAS REQUESTED (AIAIORTY=1) THE AIA MUST BE SENT BACK TO ILRQIOE VIA THE STAGING QUEUE. IN ADDITION, IF IT WAS A WRITE (AIAWRITE=1) THE SLOT THAT WAS USED MUST BE FREED. GO TO STEP 9.	ILRFRSLT	ILRFRSL1	
B. IF THE REQUEST WAS FOR A WRITE AND AN ERROR OCCURRED (AIAWRITE=1 AND AIAPRIER=1) OR IF REQUEST WAS FOR READ AND RETRY IS SET (AIAWRITE=0 AND AIAIORTRY=1) QUEUE THE AIA TO THE STAGING QUEUE. GO TO STEP 9.			
C. IN ANY OTHER CASE THE AIA WILL BE SENT TO ILRVIOCM FOR PROCESSING GO TO STEP 7.	ILRVIOCM	ILRVIOCM	

Notes	Routine	Label	Ref

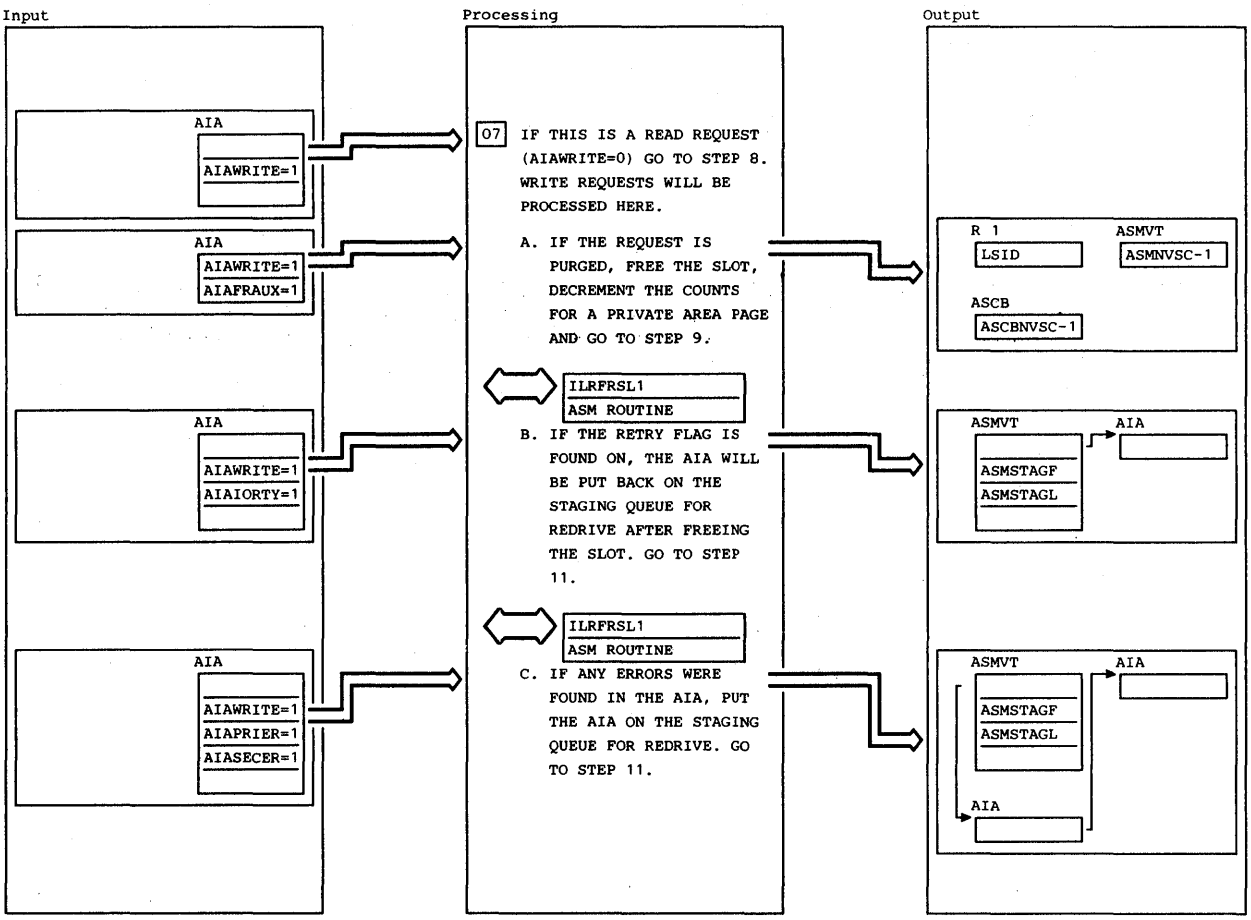
Diagram 25.4.1 PAGECOMP (Part 2 of 6)



Notes	Routine	Label	Ref
05 IF THERE WERE NO SPECIAL BITS ON IN AIAFLG2, THE REQUEST IS A NORMAL COMPLETION AND IS HANDLED FIRST. A. FOR A DUPLEX REQUEST (AIADUPLX=1), DECREMENT AIADPXCT. SET TO TWO ORIGINALLY, AIADPXCT MUST BE ZERO, BOTH REQUESTS COMPLETED, BEFORE RETURNING AIA TO RSM. IF AIADPXCT DOES NOT GO TO ZERO WHEN DECREMENTED, GO TO STEP 11 TO GET NEXT AIA. B. FOR A PRIVATE AREA COMPLETION (AIAPRIV=1), A TRANSFER ADDRESS SPACE IS DONE TO ACCESS THE XPTE. THE LSID IS MOVED IN AND IT IS VALIDATED (XPTVALID=1). FOR FIXED SWAP PAGES (AIASWPFX=1) THE SPCTE IS ALSO UPDATED (SPCTSSID AND SPCTLVAL). C. FOR A NONERIVATE AREA COMPLETION A TRANSFER ADDRESS SPACE IS NOT REQUIRED. THE XPTE IS UPDATED AND			

Notes	Routine	Label	Ref
VALIDATED. D. FOR READ REQUESTS NO PROCESSING IS REQUIRED. 06 IF ANY SEVERE ERRORS (AIABADID OR AIAERROR) OCCURRED THE AIA'S ARE QUEUED FOR DIRECT RETURN TO IEAVPIOP (DONE IN STEP 9).			

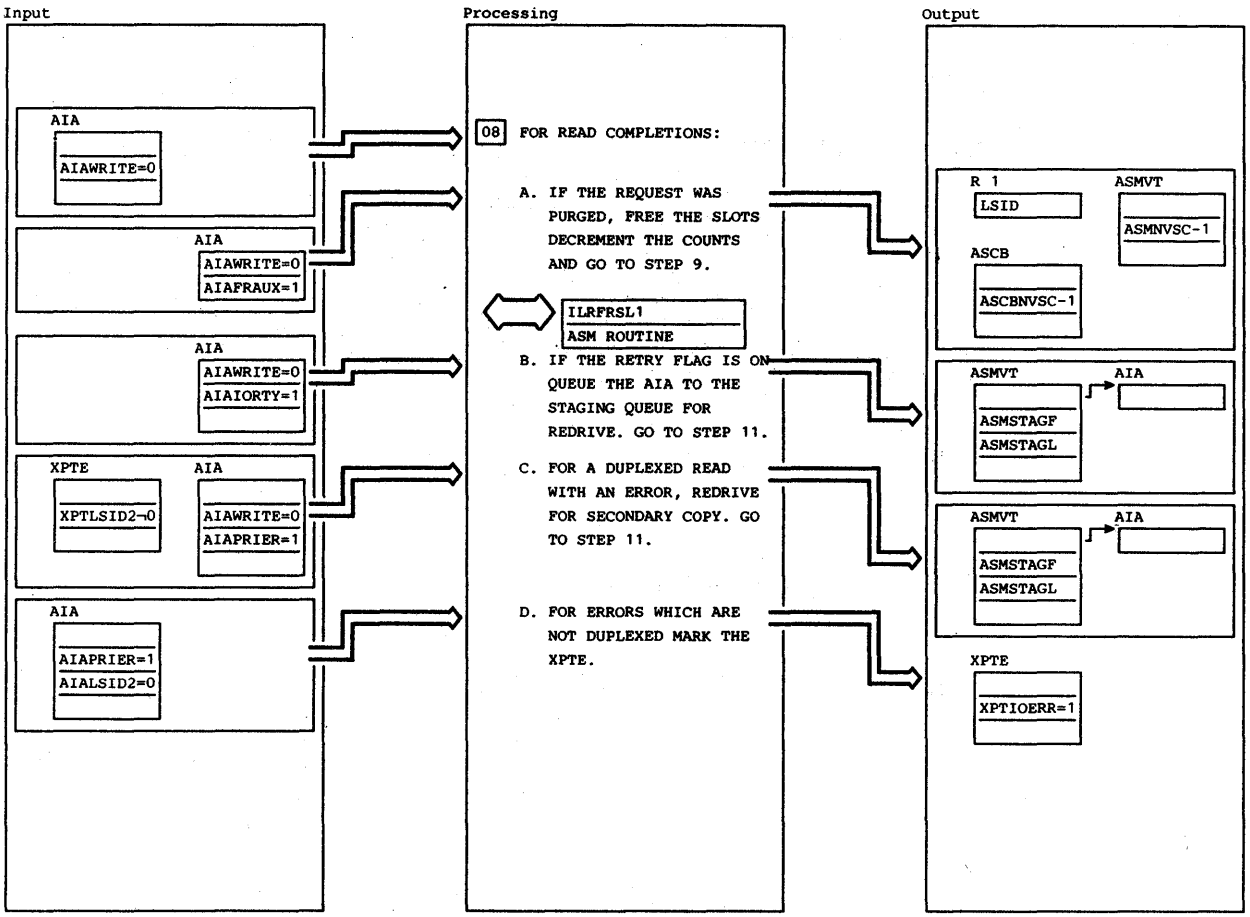
Diagram 25.4.1 PAGECOMP (Part 3 of 6)



Notes	Routine	Label	Ref
07 FOR WRITE REQUESTS:			
A. IF THE REQUEST WAS PURGED (AIAFRAUX=1) THE SLOT IS NOT NEEDED SO IT IS FREED UNLESS THERE WERE I/O ERRORS. THE USE COUNTS (ASCBNVSC, ASMNVSC) ARE DECREMENTED FOR PRIVATE AREA PAGES. GO TO STEP 9.	ILRFRSLT	ILRFRSL1	
B. IF THE AIA SPECIFIES THAT A RETRY SHOULD BE ATTEMPTED (AIAIORTY=1) THE SLOTS WHICH WERE ALLOCATED ARE FREED AND THE AIA IS QUEUED TO THE ASMSTAGQ. GO TO STEP 11.	ILRFRSLT	ILRFRSL1	
C. IF ANY I/O ERRORS OCCURRED (AIAPRIER=1 OR AIASECER=1) QUEUE THE AIA TO THE ASMSTAGQ FOR REDRIVE. GO TO STEP 11.			

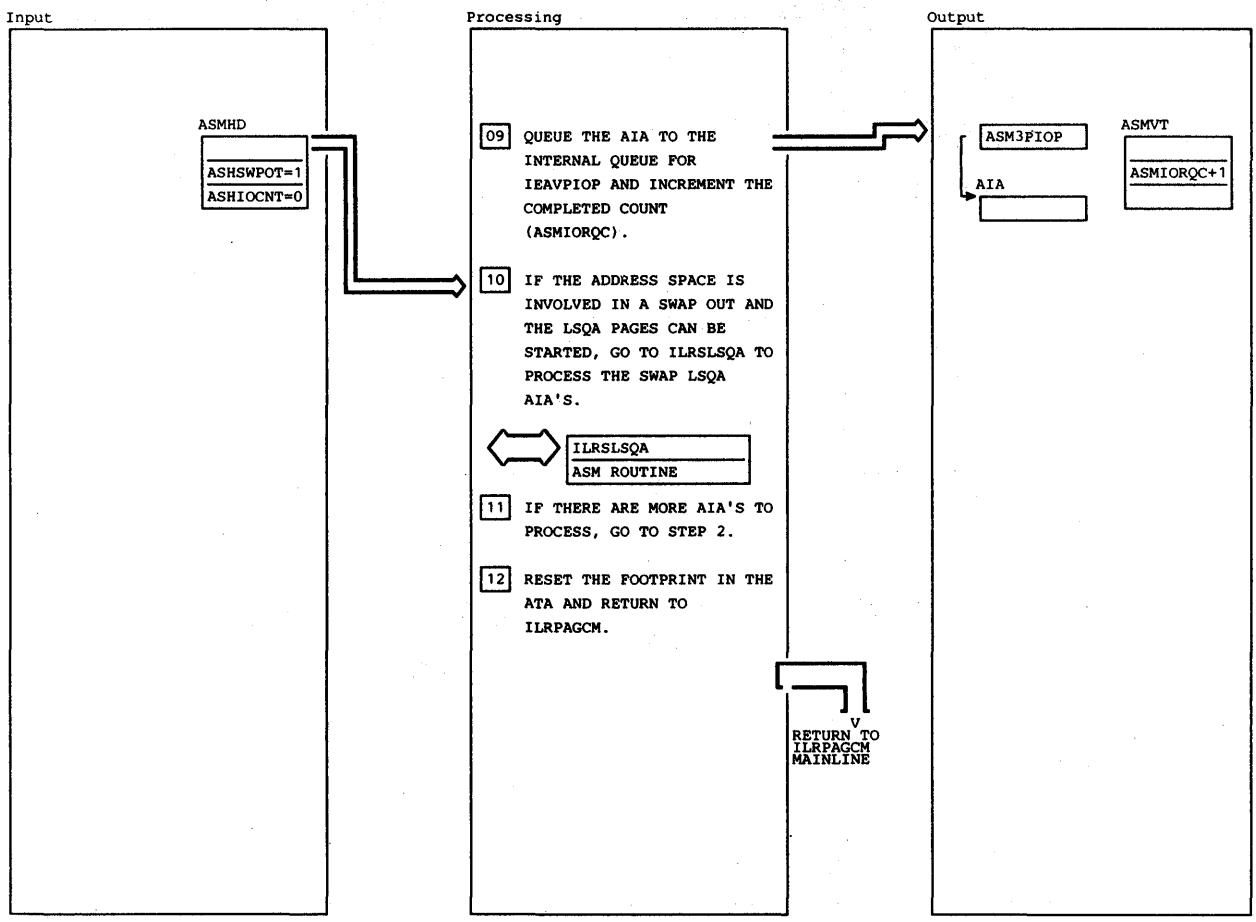
Notes	Routine	Label	Ref

Diagram 25.4.1 PAGECOMP (Part 4 of 6)



Notes	Routine	Label	Ref
08 FOR READ REQUESTS:			
A. IF THE SLOT WAS PURGED (AIAFRAUX=1) WHILE THE PAGE OPERATION WAS IN PROGRESS THE SLOT CAN BE FREED AND USE COUNTS DECREMENTED (ASCBNVSC, ASMNVSC) FOR PRIVATE AREA PAGES. IF NO I/O ERRORS OCCURRED, THE SLOT IS FREED. GO TO STEP 9.	ILRFRSLT	ILRFRSL1	
B. IF THE RETRY FLAG IS ON THE AIA MUST BE QUEUED FOR REDRIVE TO THE ASMSTAGQ. GO TO STEP 11.			
C. FOR A READ REQUEST THAT WAS DUPLEXED, A PRIMARY ERROR CAN BE REDRIVEN TO TRY TO READ THE SECONDARY COPY. THE SECONDARY LSID (XPTLSID2) IS MOVED INTO THE PRIMARY FIELD (XPTLSID) AND INTO THE AIA (AIALSID) AND THEN QUEUED TO ASMSTAGQ FOR REDRIVE. GO TO STEP 11.			
D. FOR A NON-DUPLEXED READ ERROR THE ERROR IS INDICATED IN THE XPT (XPTIOERR=1).			

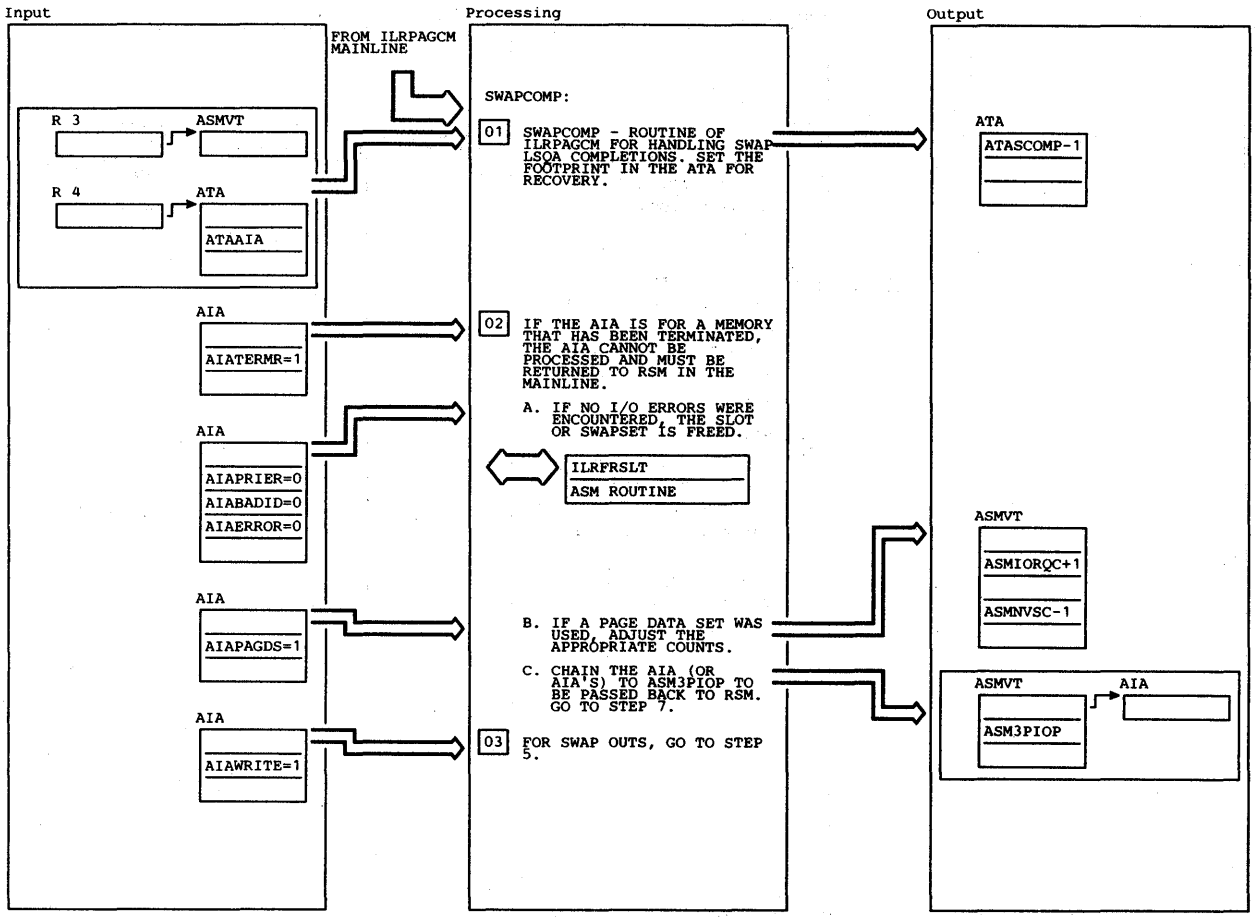
Diagram 25.4.1 PAGECOMP (Part 5 of 6)



Notes	Routine	Label	Ref
09 THE AIA IS NEXT QUEUED TO THE INTERNAL QUEUE (ASM3PIOP) FOR IEAVPIOP AND THE AIA COMPLETED COUNT (ASMIORQC) IS INCREMENTED.			
10 DETERMINE IF A SWAPOUT IS IN PROGRESS (ASHSWPOT=1) AND IF SO, IF ALL I/O FOR THE PRIVATE AREA PAGES HAS COMPLETED (ASHIOCNT=0), THE LSQA PAGES CAN BE STARTED. IF THE LSQA CAN BE STARTED, ILRSLSQA IS CALLED TO START THEM.	ILRSWAP	ILRSLSQA	
11 THE NEXT AIA TO PROCESS IS PICKED UP FROM ATAAIA AND IF THERE IS ONE TO PROCESS GO TO STEP 2.			
12 ALL THE AIA'S ARE NOW PROCESSED - RESET THE FOOTPRINT IN THE ATA AND RETURN TO ILRPAGCM.			

Notes	Routine	Label	Ref

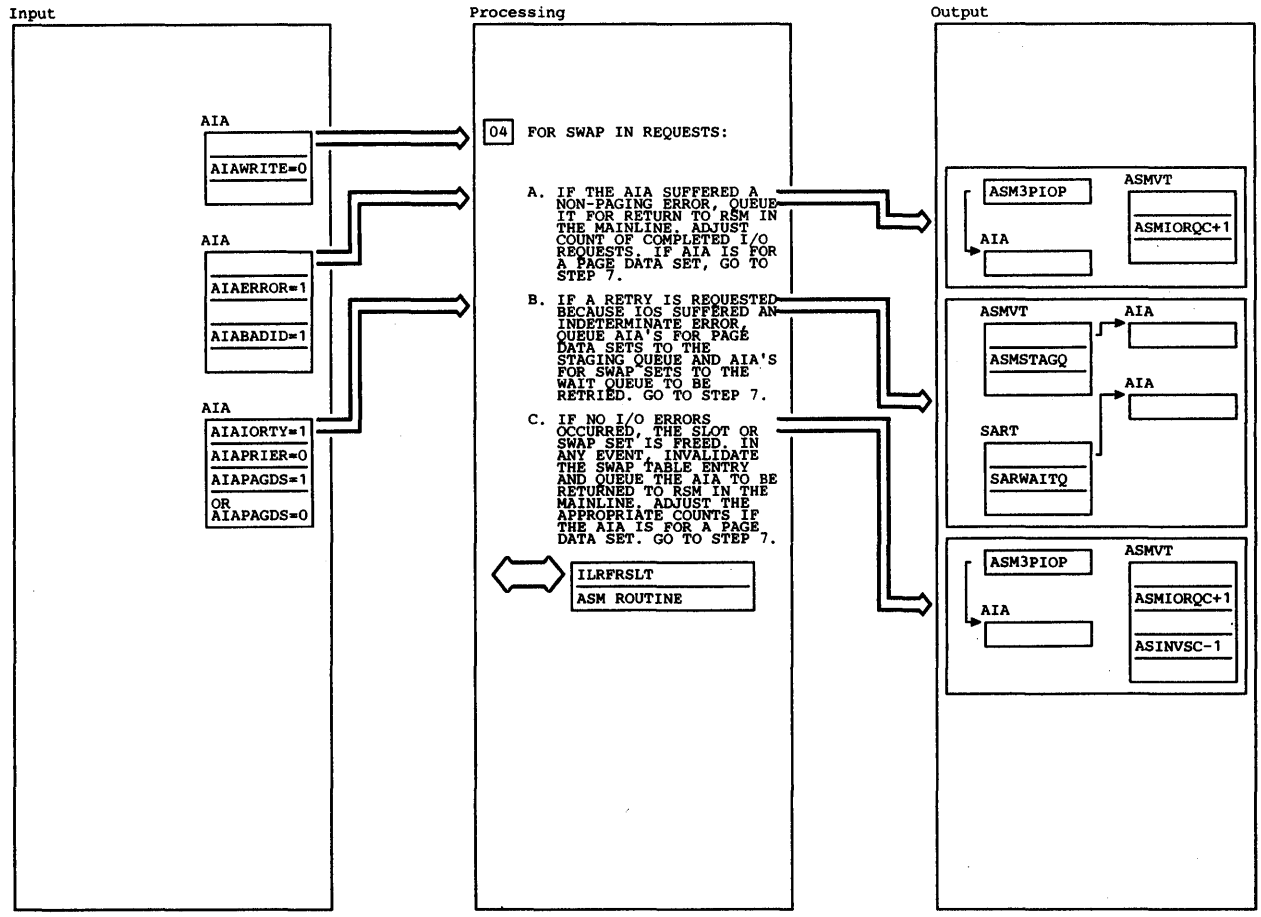
Diagram 25.4.1 PAGECOMP (Part 6 of 6)



Notes	Routine	Label	Ref
01 SET FOOTPRINT IN THE ATA TO INDICATE TO RECOVERY THAT PROCESSING IS NOW IN THE SWAP COMPLETION PART OF ILRPAGCM.			
02 IF THE TERMINATION BIT IS SET IN THE AIA (AIATERMR=1), THE MEMORY THAT THIS AIA WAS CREATED FOR HAS BEEN TERMINATED AND THE ASMHD, SPCTE AND OTHER MEMORY RELATED CONTROL BLOCKS HAVE BEEN DESTROYED. FOR THIS REASON ONLY THE AIA CAN BE REFERENCED.			
A. IF THE AIA DID NOT HAVE AN I/O ERROR, FREE THE PAGE DATA SET SLOT (ILRFRSL1) OR THE SWAP DATA SET (ILRFRSW1).	ILRFRSLT ILRFRSLT	ILRFRSL1 ILRFRSW1	
B. INCREMENT THE COUNT OF COMPLETED I/O REQUESTS (ASMIORQC), AND DECREMENT THE COUNT OF SLOTS FOR NON-VIO PAGES (ASMNVSC).			
C. QUEUE THE AIA (OR GROUP OF AIA'S) TO ASM3PIOP FOR THE MAINLINE TO RETURN TO RSM.			
03 FOR SWAP OUTS (AIAWRITE=1) GO TO STEP 5.			

Notes	Routine	Label	Ref

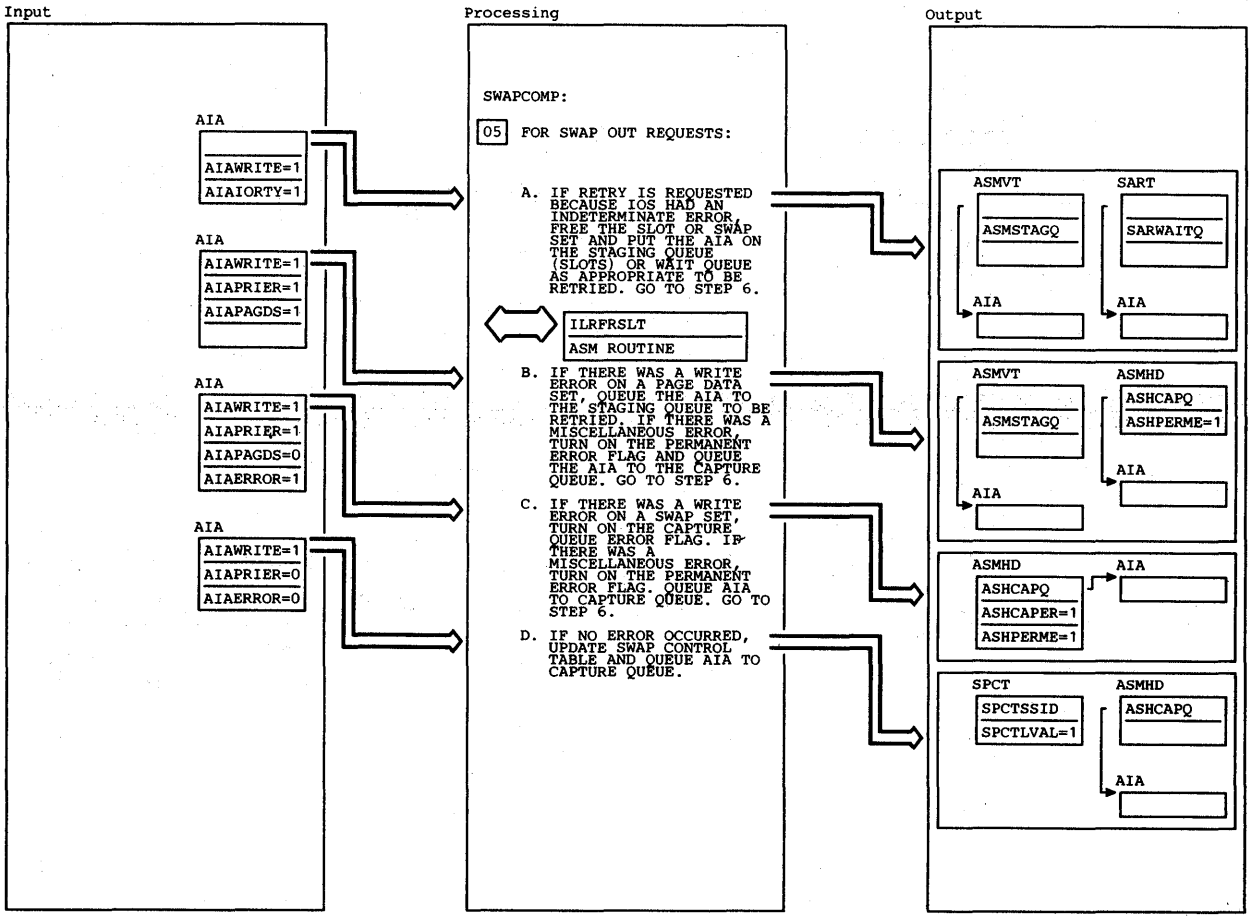
Diagram 25.4.2 SWAPCOMP (Part 1 of 5)



Notes	Routine	Label	Ref
04 FOR SWAP IN REQUESTS (AIAWRITE=0):			
A. IF THE AIA HAD A NON-PAGING ERROR SUCH AS A BAD LSID PASSED BY RSM (AIABADID=1) OR AN INDETERMINATE ERROR (AIAERROR=1) THE OPERATION CANNOT BE RETRIED AND THE AIA IS QUEUED TO THE INTERNAL QUEUE TO BE RETURNED TO RSM BY THE MAINLINE. INCREMENT COUNT OF COMPLETED I/O REQUESTS (ASMIORQC) IF AIA WAS FOR A PAGE DATA SET.			
B. IF IOS HAD AN INDETERMINATE ERROR BUT NOTHING IS WRONG WITH THE ASM REQUEST (AIA) THE OPERATION SHOULD BE RETRIED (AIAIORTY=1). IF THE READ REQUEST WAS FOR A PAGE ON A PAGE DATA SET (AIAPAGDS=1) THE AIA IS PLACED ON THE ASMSTAGQ AND THE MAINLINE CALLS ILRQIOE TO RETRY THE OPERATION. IF THE REQUEST WAS FOR A GROUP OF PAGES ON A SWAP SET (AIAPAGDS=0) THE AIA GROUP IS PLACED ON THE SARWAITQ SO THAT ILRSLSQA IS CALLED TO RETRY THE OPERATION (STEP 8).			
C. IF NO ERRORS OCCURRED, THE SLOT OR SWAP SET IS FREED. THE RELATED ENTRY(S) IN THE SWAP CONTROL TABLE ARE INVALIDATED (SPCTLVAL=0) BECAUSE THE AUXILIARY SLOTS NO LONGER EXIST. THE AIA(S) ARE PLACED ON THE QUEUE TO BE RETURNED TO RSM BY THE MAINLINE. IF THE AIA WAS FOR A PAGE DATA SET, INCREMENT THE COUNT OF COMPLETED I/O REQUESTS (ASMIORQC) AND DECREMENT THE COUNT OF SLOTS FOR NON-VIO PAGES (ASMNVSC).	ILRFRSLT ILRFRSLT	ILRFRSL1 ILRFRSW1	

Notes	Routine	Label	Ref

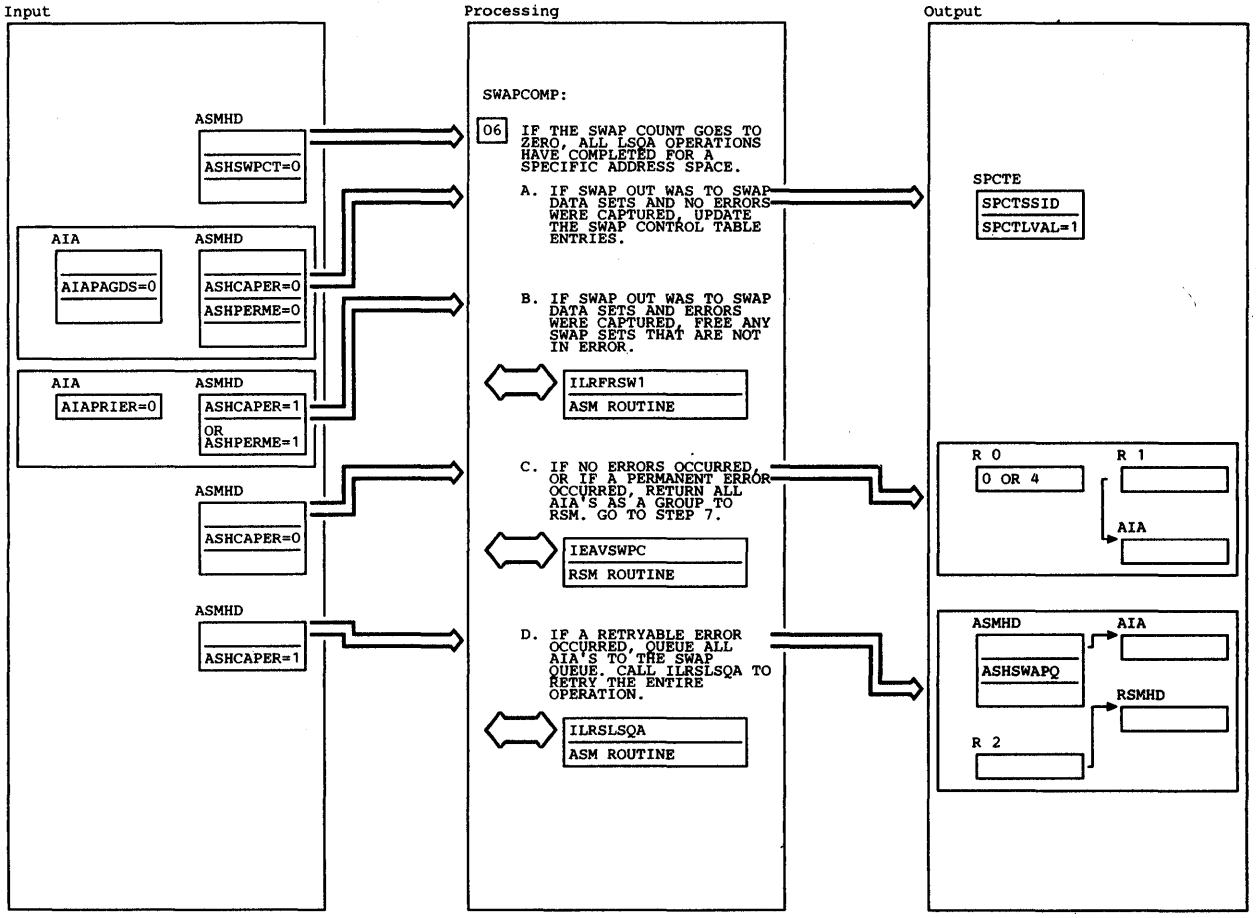
Diagram 25.4.2 SWAPCOMP (Part 2 of 5)



Notes	Routine	Label	Ref
05 SWAP OUT (AIAWRITE=1) REQUESTS:			
A. IF IOS SUFFERED AN INDETERMINATE ERROR, THE AIA(S) MAY BE RETRIED. THE SLOT OR SWAPSETS ARE FREED AND THE AIA(S) QUEUED TO EITHER THE ASMSTAGQ (FOR PAGE DATA SETS) OR THE SARWAITQ (FOR SWAP SET). ILRQIOE OR ILRSLSQA ARE CALLED TO RETRY FOR AIA'S ON THE QUEUES LATER IN PROCESSING.	ILRFRSLT ILRFRSLT	ILRFRSL1 ILRFRSW1	
B. IF THERE WAS A WRITE ERROR ON THE OPERATION THIS AIA REPRESENTS AND THE ERROR OCCURRED ON A PAGE DATA SET, THE AIA IS QUEUED TO THE ASMSTAGQ SO THAT IT MAY BE WRITTEN TO A DIFFERENT SLOT. IF THERE WAS A MISCELLANEOUS ERROR (AIAERROR=1), TURN ON THE PERMANENT ERROR FLAG (ASHPERME=1) AND QUEUE THE AIA TO THE CAPTURE QUEUE.			
C. IF THERE WAS A WRITE ERROR ON A SWAP SET, TURN ON THE CAPTURE QUEUE ERROR FLAG (ASHCAPER=1) TO INDICATE THAT AN ERROR AIA CAPABLE FOR REDRIVE IS ON THE CAPTURE QUEUE. IF THERE WAS A MISCELLANEOUS ERROR (AIAERROR=1) TURN ON THE PERMANENT ERROR FLAG (ASHPERME=1) WHICH MEANS THE AIA'S CANNOT BE REDRIVEN. QUEUE THE AIA TO THE CAPTURE QUEUE.			
D. IF NO ERROR OCCURRED, UPDATE SWAP CONTROL TABLE ENTRY WITH LSID FROM AIA AND VALIDATE THE ENTRY.			

Notes	Routine	Label	Ref

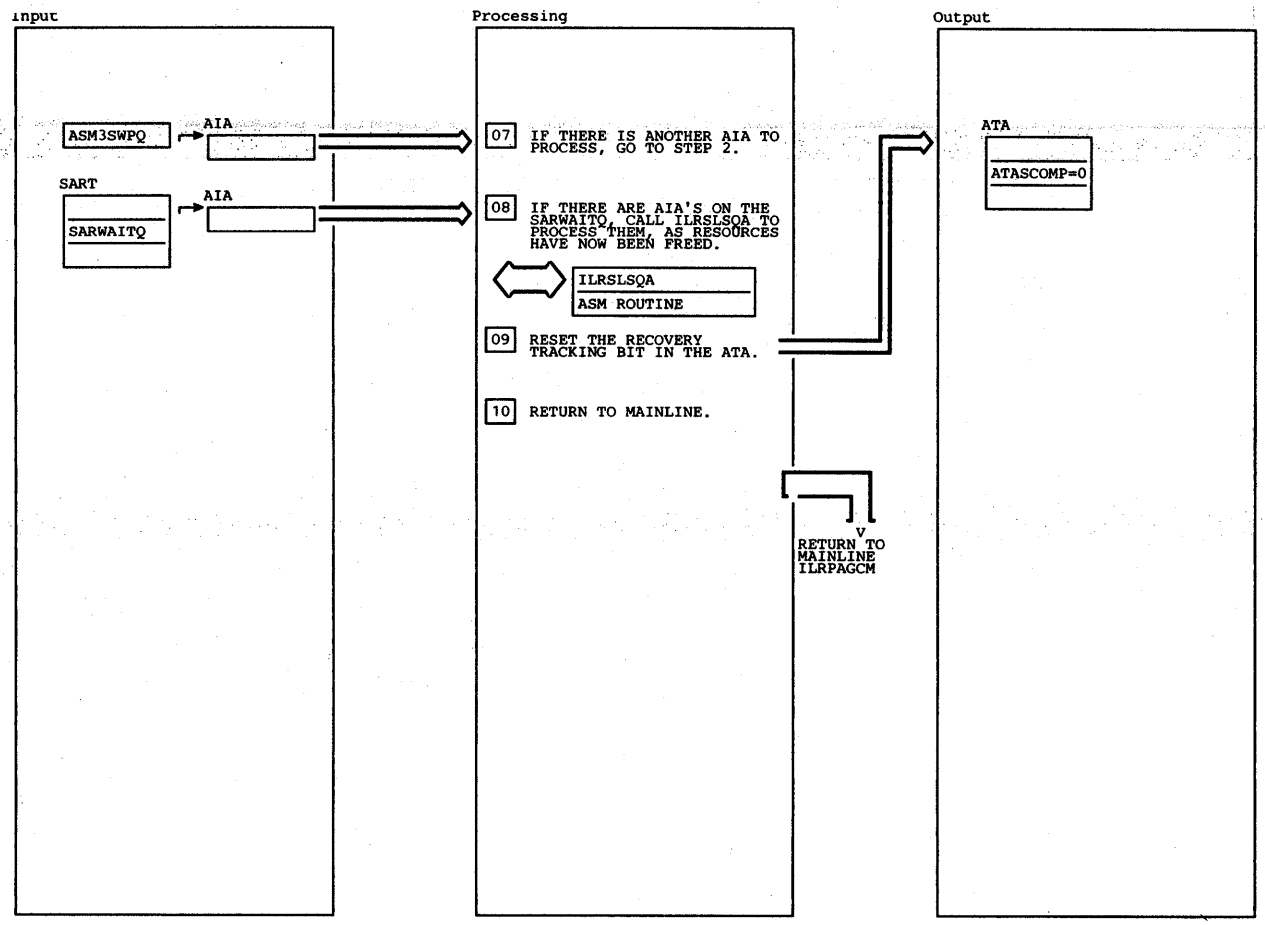
Diagram 25.4.2 SWAPCOMP (Part 3 of 5)



Notes	Routine	Label	Ref
06 WHEN THE SWAP COUNT IS ZERO (ASMSWPCT=0), ALL LSQA AIA'S FOR THAT SPECIFIC ADDRESS SPACE HAVE COMPLETED AND BEEN CAPTURED.			
A. IF SWAP OUT WAS TO SWAP DATA SETS (AIAPAGDS=0), AND IF NO ERRORS HAVE BEEN CAPTURED (ASHCAPER=0 AND ASHPERME=0), THE SPCTE'S (SWAP CONTROL ENTRIES) ARE VALIDATED (SPCTLVAL=1) AND THE LSID'S FOR EACH AIA ARE PLACED IN THE SWAP CONTROL TABLE FOR THE SUBSEQUENT SWAP IN.			
B. IF SWAP OUT WAS TO SWAP DATA SETS (AIAPAGDS=0), AND ERRORS OCCURRED (ASHCAPER=1 OR ASHPERME=1), ALL SWAP SETS THAT DO NOT CONTAIN ERRORS (AIAERROR=0 AND AIAPRIER=0) ARE FREED.	ILRFRSLT	ILRFRSW1	
C. IF NO ERROR OCCURRED, REGISTER 0 IS SET TO 0, IF A PERMANENT ERROR OCCURRED, REGISTER 0 IS SET TO 4 (TO INDICATE THE ABSENCE OR PRESENCE OF THE ERROR TO RSM). IN EITHER CASE, ALL THE AIA'S ARE RETURNED TO RSM'S SWAP COMPLETION ROUTINE.	IEAVSWPC	IEAVSWPC	
D. IF AN ERROR OCCURRED THAT CAN BE RETRIED (ASHCAPER=1), ALL THE AIA'S ARE QUEUED TO THE SWAP QUEUE (ASHSWAPQ). ILRSLSQA IS CALLED TO REDRIVE THE ENTIRE LSQA SWAP OPERATION.	ILRSWAP	ILRSLSQA	

Notes	Routine	Label	Ref

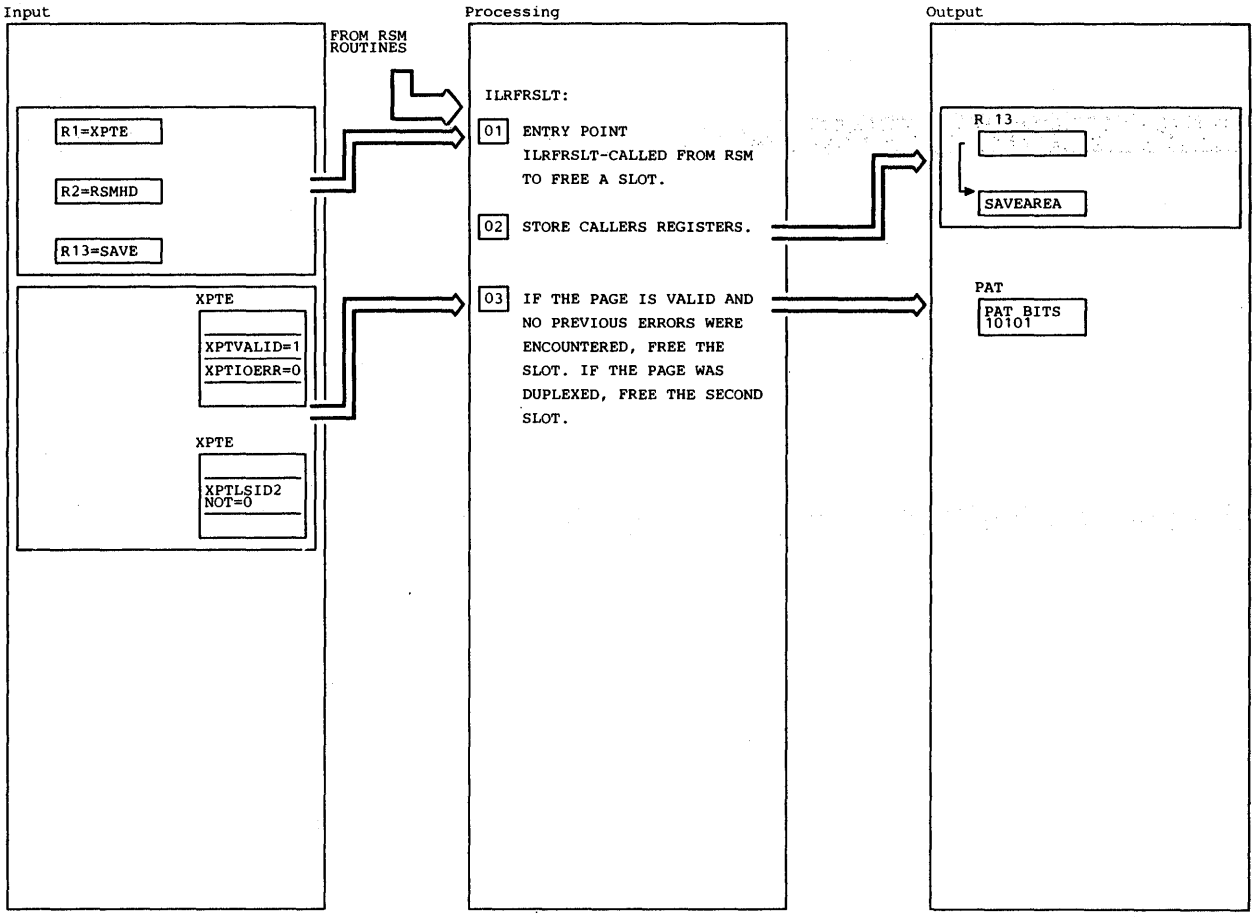
Diagram 25.4.2 SWAPCOMP (Part 4 of 5)



Notes	Routine	Label	Ref
07 IF THERE ARE MORE AIA'S ON THE INTERNAL SWAP QUEUE TO BE PROCESSED, GO TO STEP 2.			
08 IF THERE ARE AIA'S ON THE WAIT QUEUE ILRSLSQA IS CALLED TO PROCESS THEM, BECAUSE RESOURCES (SCCW'S) ARE NOW FREE TO PROCESS THEM.	ILRSWAP	ILRSLSQA	
09 RESET THE TRACKING BIT IN THE ATA FOR RECOVERY.			
10 RETURN TO THE MAINLINE.			

Notes	Routine	Label	Ref

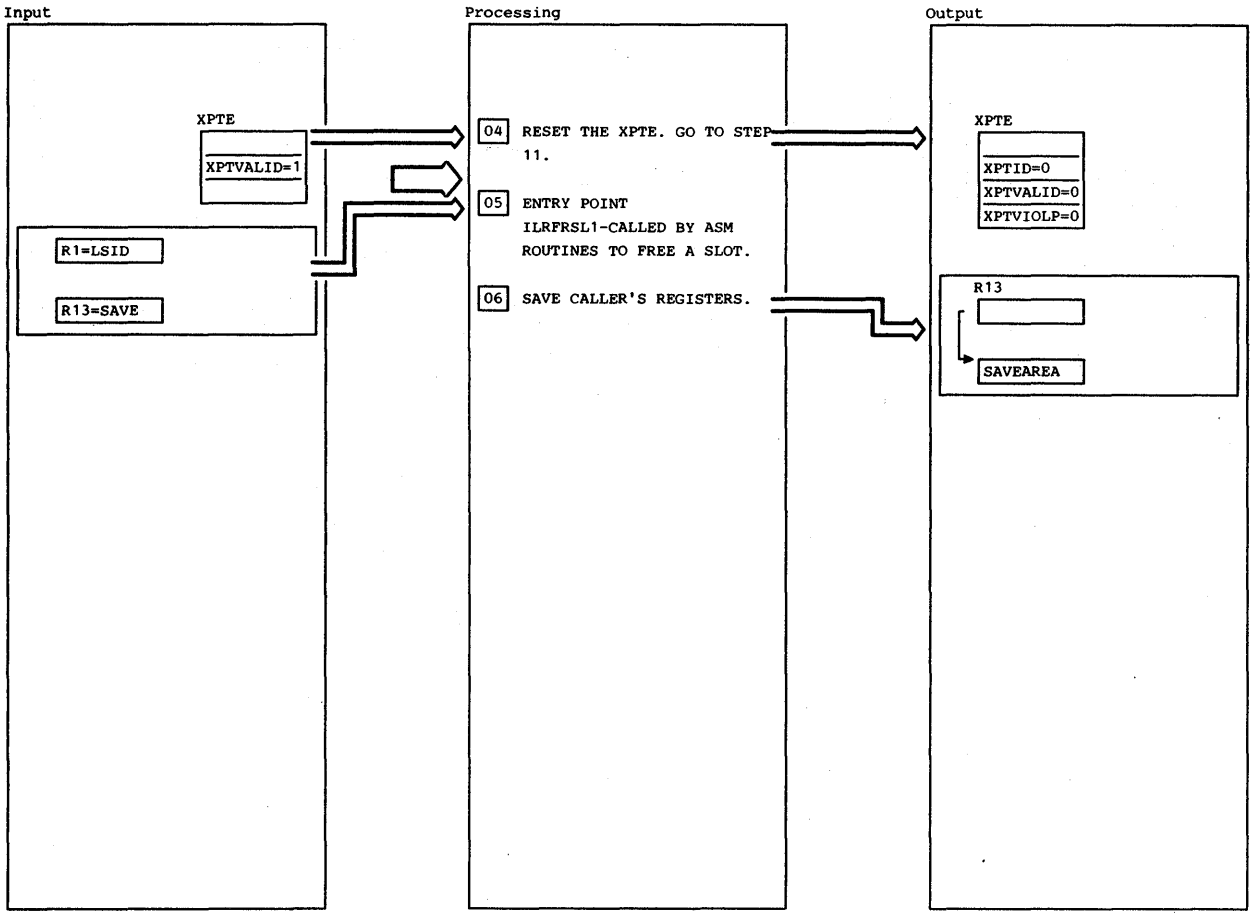
Diagram 25.4.2 SWAPCOMP (Part 5 of 5)



Notes	Routine	Label	Ref
01 ILRFRSLT IS THE MAIN ENTRY POINT. SECONDARY ENTRY POINTS ARE ILRFRSL1 AND ILRFRSW1. THE MAIN ENTRY POINT IS USED BY RSM ROUTINES TO FREE A SLOT SUCH AS WHEN A PAGE IS FREED. THIS MODULE SETS NO FRR OR TRACKING BIT, IT MERELY RUNS AS A SUBROUTINE OF THE CALLER.			
02 STORE THE CALLERS REGISTERS IN THE SAVEAREA PASSED IN REGISTER 13.			
03 IF THE XPTE IS VALID (XPTVALID=1) AND NO PREVIOUS ERRORS WERE DETECTED (XPTPRIER=0), FREE THE SLOT BY SETTING THE APPROPRIATE PAT BIT TO 0. IF THE PAGE WAS DUPLEXED (XPTLSID2¬=0), FREE THE SECOND SLOT IN THE SAME MANNER.			

Notes	Routine	Label	Ref

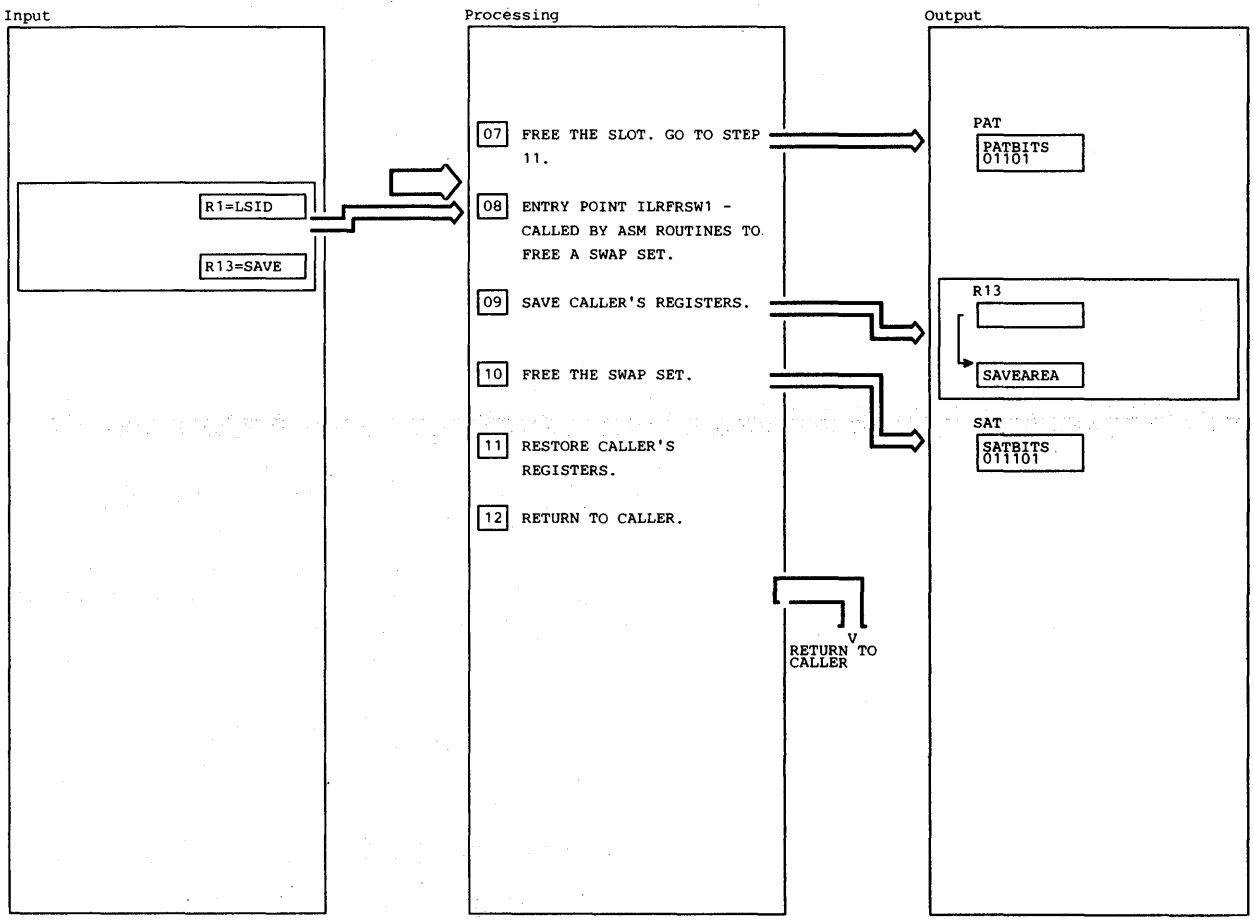
Diagram 25.5 ILRFRSLT (Part 1 of 3)



Notes	Routine	Label	Ref
04 RESET THE XPTE BY SETTING XPTVALID=0, XPTVIOLP=0 AND XPTID (THE TWO LSID'S) TO 0. GO TO STEP 11.			
05 ENTRY POINT ILRFRSL1 IS CALLED BY ASM ROUTINES TO FREE A SLOT.			
06 SAVE THE CALLERS REGISTERS IN THE CALLER PROVIDED SAVE AREA.			

Notes	Routine	Label	Ref

Diagram 25.5 ILRFRSLT (Part 2 of 3)



Notes	Routine	Label	Ref
07 FREE THE SLOT BY SETTING THE APPROPRIATE PAT BIT TO 0. GO TO STEP 11.			
08 ENTRY POINT ILRFRSW1 IS CALLED BY ASM ROUTINES TO FREE A SWAP SET.			
09 SAVE THE CALLERS REGISTERS IN THE SAVE AREA PASSED BY THE CALLER.			
10 FREE THE SWAP SET BY SETTING THE APPROPRIATE SAT BIT TO 0.			
11 RESTORE THE CALLER'S REGISTERS.			
12 RETURN TO THE CALLER.			

Notes	Routine	Label	Ref

Diagram 25.5 ILRFRSLT (Part 3 of 3)

I/O Subsystem

The I/O Subsystem communicates with IOS to effect the physical transfer of data between real and auxiliary storage. When paging I/O is required, I/O Control schedules an SRB to start the I/O Subsystem processing. The I/O Subsystem selects the page data sets for which paging is pending, allocates slots if necessary, builds channel programs, calls IOS through the STARTIO macro to initiate the actual I/O, and, after return from IOS, returns the completed requests to I/O Control.

The I/O Subsystem is "completion" driven, that is, the page completion portion of I/O Control drives the I/O Subsystem when previously scheduled I/O completes. Only when no I/O is currently outstanding is the I/O Subsystem driven by the initial page processing portion of I/O Control.

I/O Subsystem processing is done by three modules: the Part Monitor (ILRPTM), Slot Sort (ILRSRT), and Completion (ILRCMP). The I/O Subsystem contains one other module, the Message module (ILRMSG00), which produces the messages issued by ASM.

I/O Subsystem can be divided into two basic parts: Initial Processing (prior to the call to IOS), and Completion Processing (upon return from IOS).

Initial Processing

Initial Processing starts when ILRPTM receives control (SRB mode) from I/O Control. Paging requests are represented by IOEs (I/O Request Elements) pointing to AIAs (ASM I/O Request Areas). Before passing control to ILRPTM, I/O Control queues each IOE to the PARTE (Page Activity Reference Table Entry) for the ASM data set against which the paging request is being made.

There are three queues of PARTEs. The first is a straight queue of PLPA, Common, and Duplex data set PARTEs. The other two are circular queues (the last PARTE on the queue points to the first) for local page data sets, one for fixed- and one for movable-head devices. ILRPTM examines all PARTEs on these two queues each time it is called. ILRPTM processes the PARTEs in the following order: PLPA, Common, Duplex, fixed-head queue, movable-head queue.

ILRPTM scans each PARTE and calls Slot Sort (ILSRT) if all the following conditions are met:

- There is a read or write request on the PARTE.
- If the request is a write, there is at least one slot available in the data set represented by that PARTE.
- The PARTE is not locked (in use by another CPU). If it is not, ILRPTM turns on an in-process flag to lock out this PARTE from other CPUs.
- A PCCW (Program Channel Command Workarea, used by ASM to identify a page I/O request), is available.
- An IORB (I/O Request Block) is available. The IORB is the main interface with IOS.

There are two read queues on a PARTE, one sorted and one unsorted. Before passing control to ILRSRT, ILRPTM sorts the unsorted reads and inserts them onto the sorted queue.

During processing, if the PLPA data set fills before all PLPA pages are written, the remainder are written to the Common paging data set. Conversely, Common writes can spill over into PLPA. If PLPA and Common are both full or one is unusable, ILRPTM calls the message routine (ILRMSG00). If duplexing is active, ILRMSG00 notifies the operator that the system is relying on the secondary copy. If duplexing is not active, ILRMSG00 terminates the system. If the Duplex data set is unusable, but both PLPA and Common are not, the operator is notified and the system continues, relying on the primary copy. If Duplex is unusable and if PLPA or Common is unusable or both are full, ILRMSG00 terminates the system.

ILRSRT sorts the I/O requests against a page data set in such a way that they can be processed with a minimum number of device revolutions. ILRSRT chooses a cylinder between the one it last used and the end of the data set. If none can be selected, ILRSRT starts again from the beginning of the data set. (If there is only one read and no writes to be done, ILRSRT takes a quick path to process the read, bypassing the cylinder selection process.) If no cylinder is found on the data set (no reads and no more available slots), the data-set-full return code is set and I/O processing for this data set on this ILRSRT invocation ceases.

After ILRSRT selects a cylinder, it selects a slot, dequeues the IOE to be processed to that slot, builds a PCCW for the operation, and chains the PCCW from the IORB. When all requests possible for a specific cylinder are processed, ILRSRT finds the next cylinder for I/O.

Processing of cylinders and slots stops when resources (PCCWs, IOEs, available slots, etc.) are exhausted or when the current service burst (the

maximum amount of time the channel/device can be tied up for a given set of operations) is met. Finally, ILRSRT completes initialization of an IORB-IOSB-SRB chain (IOSB is the I/O Supervisor Block) and branch enters IOS via the STARTIO macro.

Completion Processing

When the physical I/O operation completes, IOS returns control to the Completion module (ILRCMP) of the I/O Subsystem. The function of ILRCMP is to return AIAs to page completion (ILRPAGCM, part of I/O Control). If an error has occurred and retry is possible, ILRCMP causes an AIA to be reprocessed before returning it to ILRPAGCM. ILRCMP has four major routines:

- Disabled Interrupt Exit (entry point ILRCMPDI).
- Normal End Appendage (entry point ILRCMPNE).
- Abnormal End Appendage (entry point ILRCMPAE).
- Termination (ILRCMP).

Any I/O completion involves at least two calls to ILRCMP entry points. For successful I/O, both calls are to ILCMPDI. If there were errors on the I/O, the first call is to ILCMPDI, and the subsequent calls are to other entries of ILRCMP depending on the types of errors.

Disabled Interrupt Exit

IOS first returns control to ILRCMPDI, passing it the address of an IOSB. ILRCMPDI follows the IOSB-IORB-S/PCCW chain, and processes the individual requests represented by the S/PCCWs. After a successful I/O, on the first branch entry to ILRCMPDI it frees the S/PCCWs, returns the associated AIAs to ILRPAGCM, and returns to IOS. IOS branch enters ILRCMPDI a second time so that ILRCMPDI can release the IORB of the successfully completed request, and, if work remains on the associated PARTE/SARTE, schedule ILRPTM or ILRSWPDR as appropriate. On the first branch entry after an unsuccessful I/O, ILRCMPDI returns to IOS with a code indicating that the Post Status routine (IECVPST) should get control. IOS must schedule an SRB for POST STATUS, who calls the appropriate entry in ILRCMP.

Normal End Appendage

IECVPST calls ILRCMPNE if the error is a wrong-length record or a unit exception. ILRCMPNE immediately returns to IECVPST with a code indicating that control should be passed to DASD ERP (Error Recovery Procedure) for retry.

IECVPST also calls ILRCMPNE if DASD ERP retried successfully. In this case, ILRCMPNE removes the S/PCCWs from the IOSB-IORB, returns the S/PCCWs to the appropriate available queue, and returns the processed AIAs to ILRPAGCM.

Abnormal End Appendage

IECVPST calls ILRCMPAE on all errors other than the two mentioned in the previous section. If ILRCMPAE determines that the error is temporary, it returns immediately to IECVPST with a code indicating that control should be passed to DASD ERP for retry.

If the error is permanent, it indicates that either an entire page/swap data set or a slot is unusable. If it is a data set error, control is passed to ILRMSG00 to determine whether the system can continue and to take appropriate action - either sending a message or taking the system down. Additionally, ILRCMPAE calls a subroutine to mark all the S/PCCWs as errors. If it is a slot error, ILRCMPAE records the LSID in a bad slot list in SQA and queues the error S/PCCW AIA to be returned to ILRPAGCM. ILRCMPAE rechains the S/PCCWs following the one in error and returns to IECVPST with a code indicating that a new STARTIO should be issued to retry them.

Termination

IECVPST calls ILRCMP if an ABEND occurs within IOS or within ILRCMPNE or ILRCMPAE. This means the status of the I/O is indeterminable, so ILRCMP marks all the AIAs for retry, returns them to ILRPAGCM, frees the IORB associated with the AIAs and schedules ILRPTM or ILRSWPDR as appropriate.

IECVPST also calls ILRCMP after ILRCMPAE or ILRCMPNE has freed all the S/PCCWs. In this case, ILRCMP frees the IORB and schedules the SRB if necessary.

Message Module

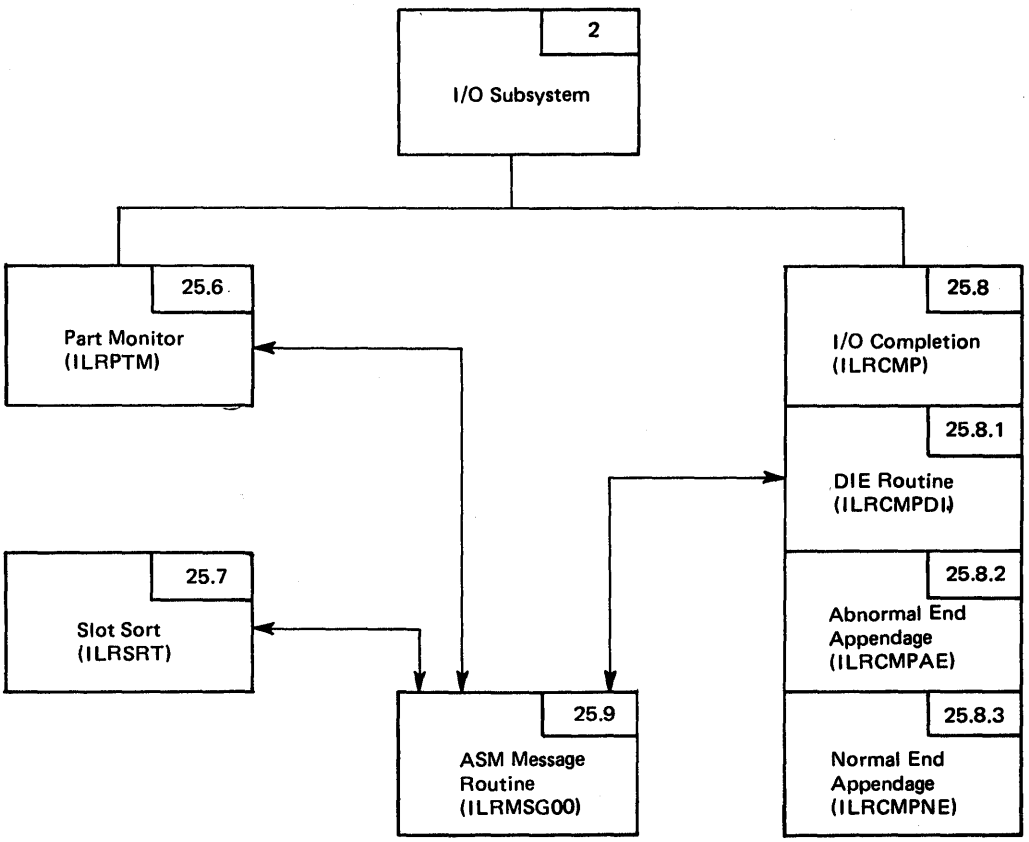
The Message Module (ILRMSG00) is used by the Part Monitor (ILRPTM), I/O Completion (ILRCMP), and the recovery modules ILRCMP01, ILRSRT01 and ILRSWP01.

ILRMSG00 has two primary functions: to write messages to the operator concerning the status of all page and swap data sets; to terminate the system when ASM is unable to continue. Reasons for termination are: PLPA or Common has become unusable and there is no Duplex data set available; Duplex has become unusable and PLPA/Common is

unusable or both are full; the last Local page data set has become unusable.

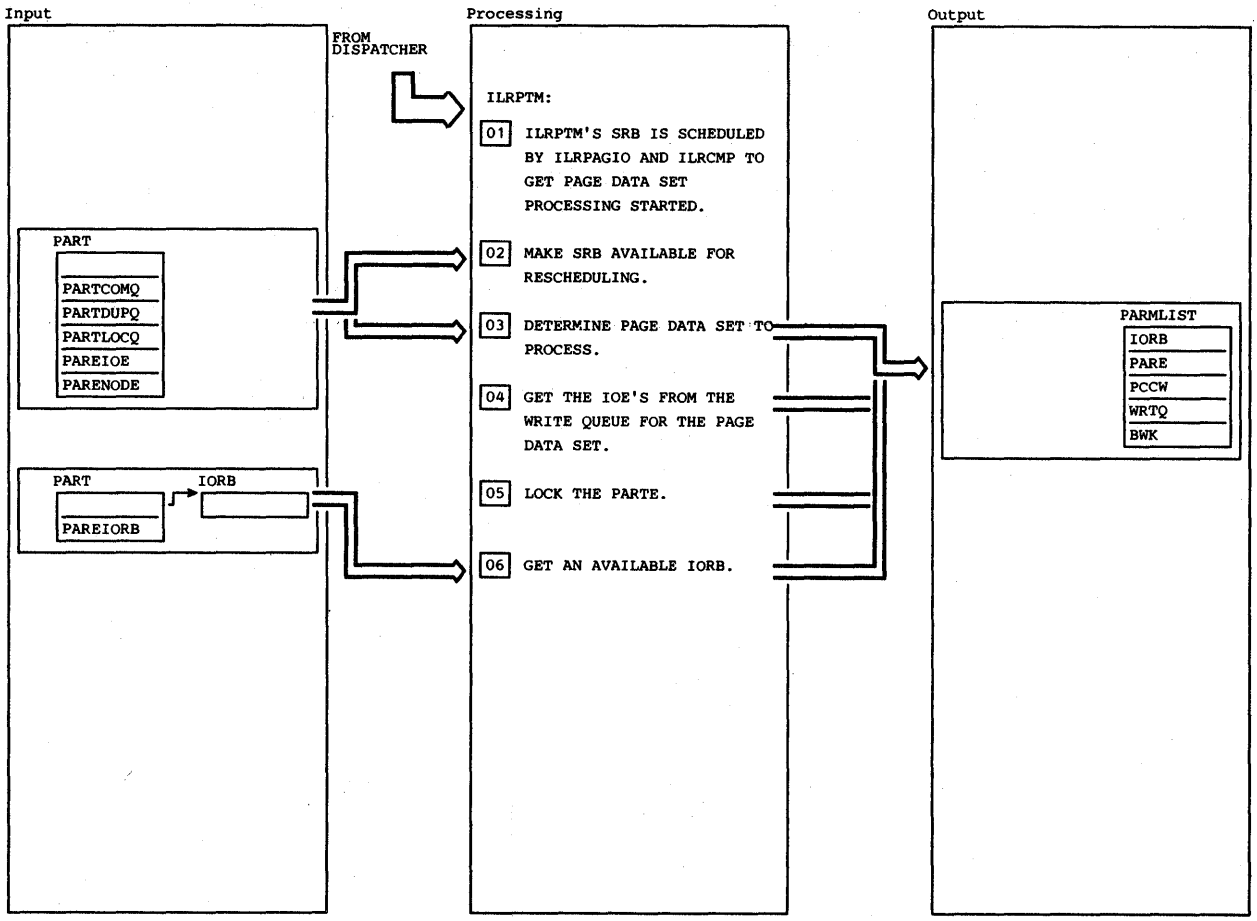
When ILRMSG00 is provided with a message number, it issues that message to the operator and returns to the caller. If the message number provided is eight, ILRMSG00 passes control to IGFPTERM to terminate the system.

If a message number is not provided to ILRMSG00, it determines what message to issue and updates the appropriate flag and count fields in the ASMVT, PART, and SART. If necessary, ILRMSG00 passes control to IGFPTERM to terminate the system.



25.x. — Module
25.x.y. — Entry point in module 25.x.

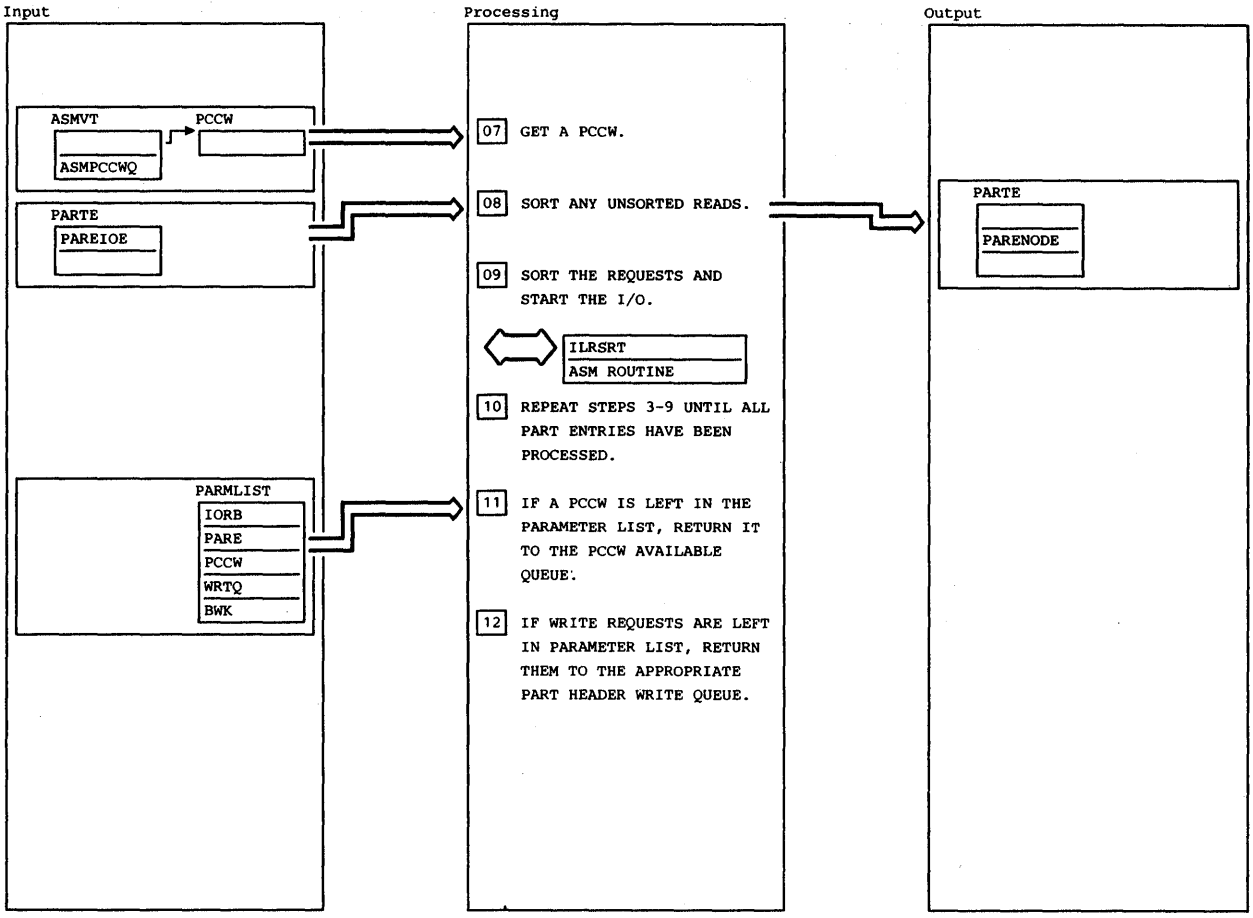
Figure 2-58. I/O Subsystem Overview



Notes	Routine	Label	Ref
01 ILRPTM (PART MONITOR) RECEIVES CONTROL IN SRB MODE TO INITIATE WORK ON PAGE DATA SETS. THE INFORMATION ABOUT A PAGE DATA SET IS CONTAINED IN A PARTE IN THE PART. THERE IS ONE PARTE FOR EACH PAGE DATA SET. FOR RECOVERY PURPOSES, ILRSRT01 RECOVERY ROUTINE HANDLES ERRORS OCCURRING IN ILRPTM.			
02 ONLY 1 SRB FOR ILRPTM CAN BE SCHEDULED AT A TIME. IF ANY WORK IS ADDED AFTER THIS ENTRY, ILRPTM WILL BE SCHEDULED AGAIN.			
03 A PAGE DATA SET WILL BE PROCESSED IF THERE ARE WRITES ON ITS CORRESPONDING WRITE QUEUE IN THE PART HEADER, OR READS IN EITHER OF ITS READ QUEUES (SORTED AND UNSORTED) IN THE PARTE, AND THAT PARTE IS NOT CURRENTLY BEING PROCESSED. THE POSSIBLE WRITE QUEUES ARE PARTCOMQ, PARTDUPQ, AND PARTLOCQ. MORE THAN ONE PARTE CAN POINT TO THE SAME WRITE QUEUE. THE SORTED AND UNSORTED			

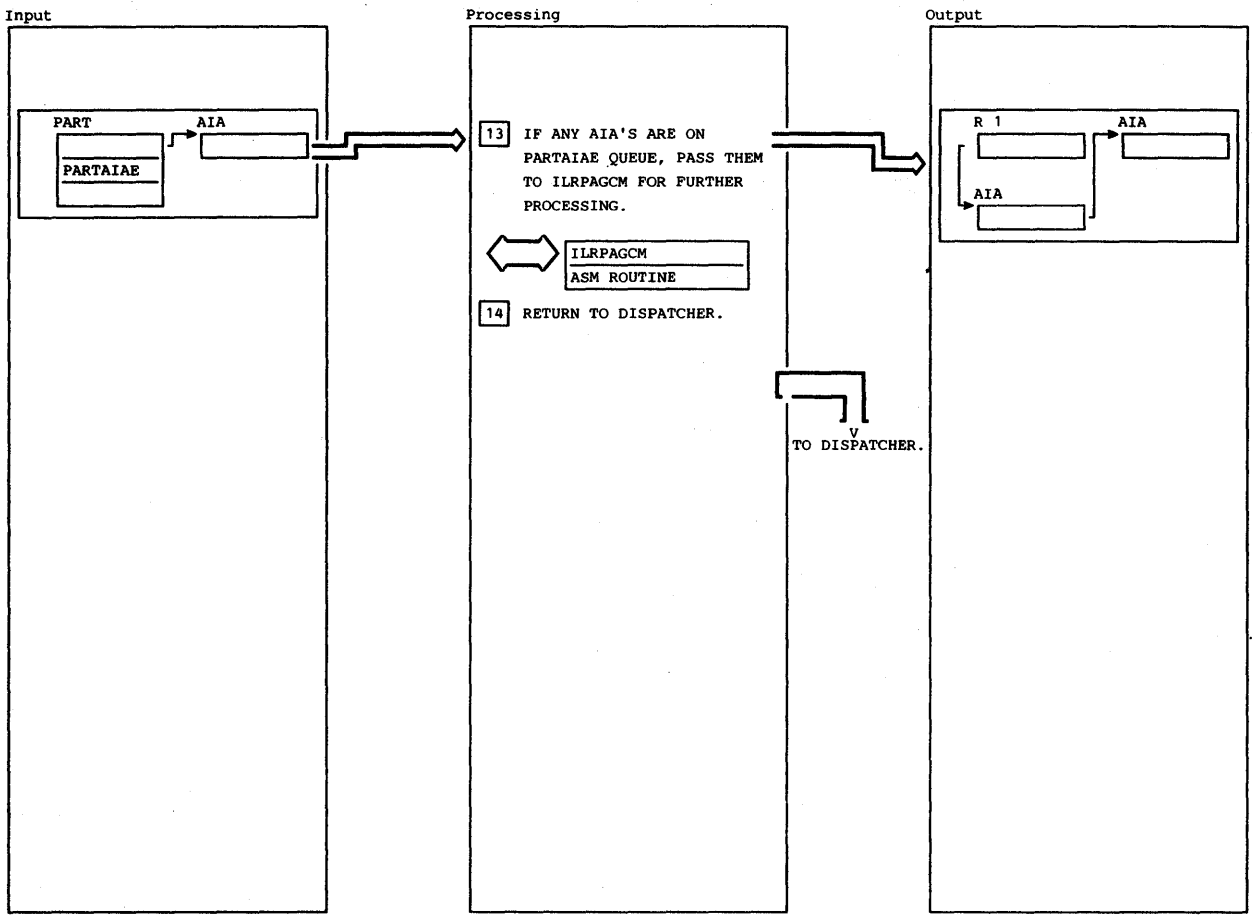
Notes	Routine	Label	Ref
READ QUEUES ARE PARENODE AND PAREIOE, RESPECTIVELY. THE ADDRESS OF THE PARTE SELECTED IS PUT IN THE PARAMETER LIST.			
04 ALL IOES CHAINED ON THE WRITE QUEUE ARE REMOVED AND PUT ON THE WRITE QUEUE IN THE PARAMETER LIST. THE ASM CLASS LOCK WILL SERIALIZE THE WRITE QUEUES.		GETWRTQ	25.6.1
05 THE PART ENTRY IS LOCKED TO SERIALIZE PROCESSING OF THE PAGE DATA SET.			
06 PUT THE IORB ADDRESS IN THE PARAMETER LIST. IF THERE ARE NO IORB'S FOR THIS PARTE AT ALL, PART MONITOR ABENDS 084 SO THAT RECOVERY CAN BUILD AN IORB FOR THIS PARTE. IF NO IORB IS AVAILABLE, THE PART ENTRY IS UNLOCKED AND PROCESSING CONTINUES AT THE NEXT ENTRY.			

Diagram 25.6 ILRPTM (Part 1 of 3)



Notes	Routine	Label	Ref
07 PUT THE PCCW ADDRESS IN THE PARAMETER LIST. IF NO PCCW IS AVAILABLE THE IORB IS MADE AVAILABLE, THE PARTE UNLOCKED, AND PART MONITOR EXITS.			
08 AN ADDITIONAL CHECK IS MADE TO DETERMINE IF THERE IS STILL WORK TO DO, AND ANY UNSORTED READS ARE SORTED ACCORDING TO CYLINDER LOCATION AND PUT ONTO THE SORTED READ QUEUE (PARENODE).		SORTREAD	25.6.2
09 ILRSRT IS CALLED TO SORT REQUESTS, BUILD THE CHANNEL PROGRAMS, AND START THE I/O.	ILRSRT	ILRSRT	
10 THE PART ENTRIES ARE PROCESSED IN THE FOLLOWING ORDER - PLPA, COMMON, DUPLEX, QUEUE OF ALL FIXED HEAD LOCALS, QUEUEU OF ALL MOVABLE HEAD LOCALS. CALL DSFULL TO DETERMINE WHICH DATA SETS ARE FULL AND WHICH CAN ACCEPT A WRITE. FOR DRUMS, IF MORE WORK REMAINS, STEPS 6-9 ARE REPEATED.		DSFULL	25.6.3
11 THE PCCW ADDRESS IS KEPT IN THE PARAMETER LIST. IF ILRSRT RETURNS A PCCW, STEP 7 WILL NOT HAVE TO BE DONE FOR THE NEXT PARTE. FINAL CLEANUP REQUIRES THAT ANY RESOURCES REMAINING BE RETURNED.			
12 IF WRITES ARE RETURNED FROM ILRSRT, STEP 4 WILL HAVE TO BE EXPANDED FOR THE NEXT PARTE AS FOLLOWS. IF THE PREVIOUS WRITE QUEUE IS THE SAME AS THE CURRENT WRITE QUEUE, THE NEW WRITES ARE JUST ADDED TO THE QUEUE IN THE PARAMETER LIST. IF THEY ARE NOT THE SAME, THE OLD WRITES WILL BE PUT BACK ON THEIR WRITE QUEUE BEFORE THE NEW ONES ARE OBTAINED. FINAL CLEANUP REQUIRES THAT ANY WRITES REMAINING BE PUT BACK ON THE APPROPRIATE QUEUE.			

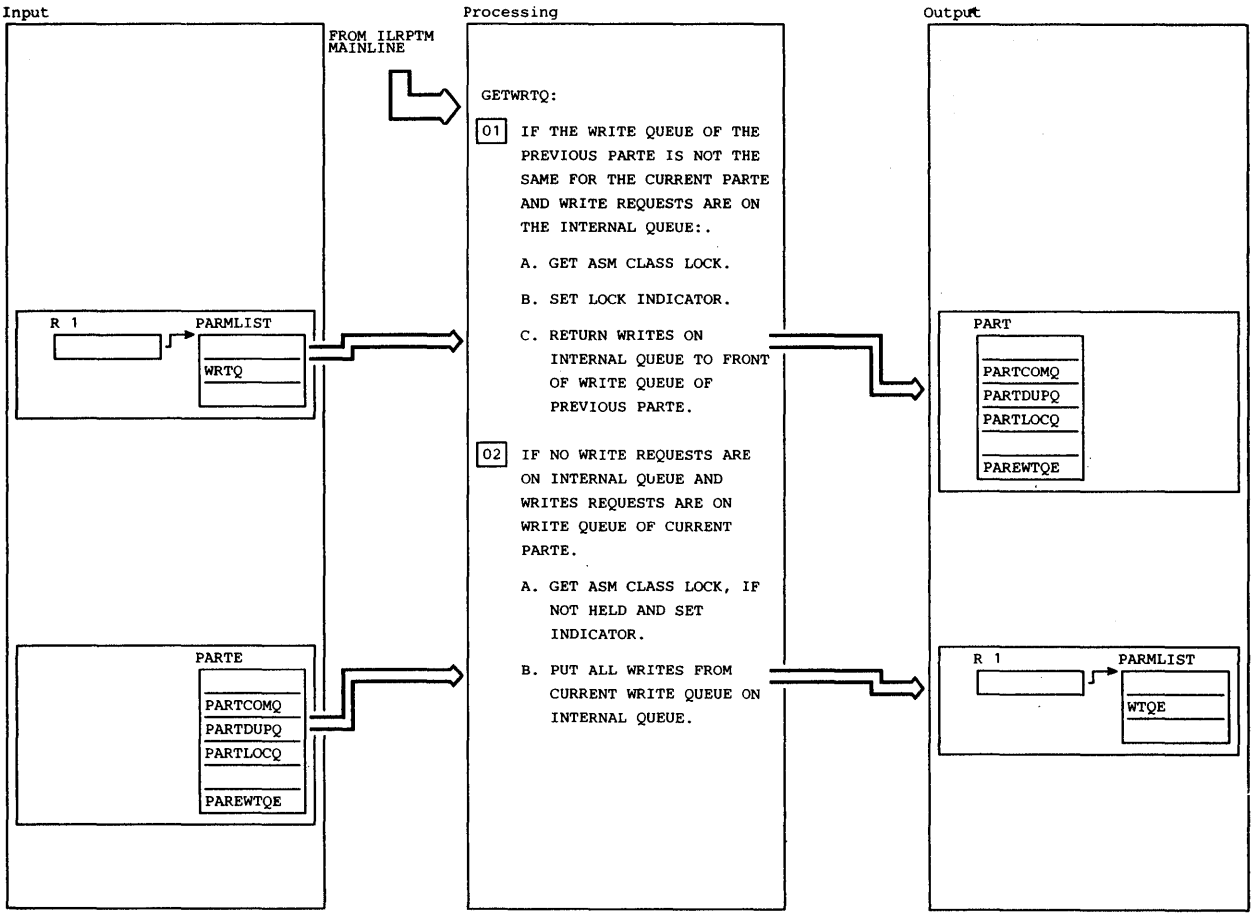
Diagram 25.6 ILRPTM (Part 2 of 3)



Notes	Routine	Label	Ref
13 ANY ERROR AIA'S RETURNED BY ILRSRT WILL BE REMOVED FROM THE PARTAIAE QUEUE TO BE PASSED TO ILRPAGCM.	ILRPAGCM	ILRPAGCM	
14 ILRPTM RUNS IN SRB MODE SO CONTROL IS RETURNED TO THE DISPATCHER.			

Notes	Routine	Label	Ref

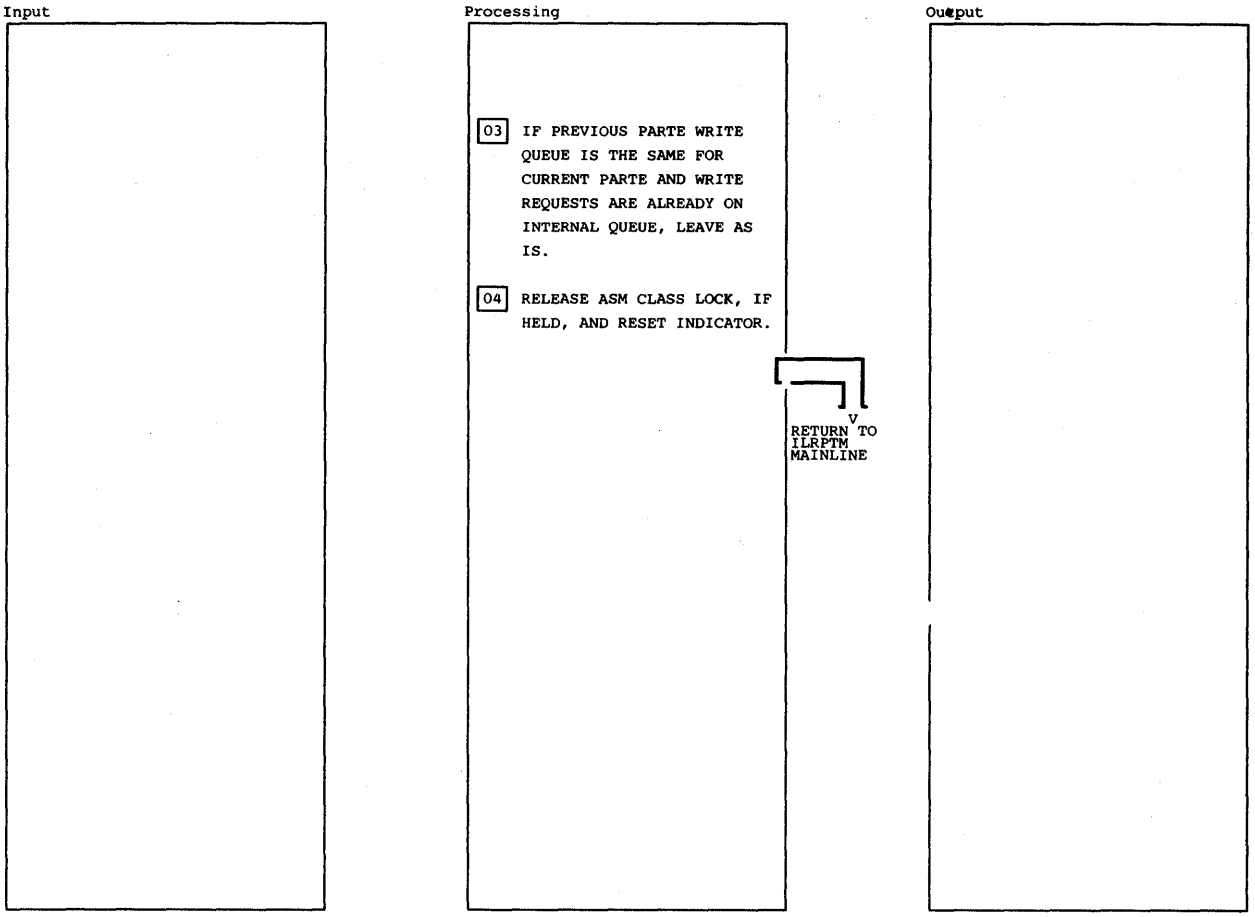
Diagram 25.6 ILRPTM (Part 3 of 3)



Notes	Routine	Label	Ref
01 WRITE REQUESTS FOR THE CURRENT PARTE (PAGE DATA SET) ARE TO BE OBTAINED. ALL WRITE REQUESTS ARE IN QUEUES IN THE PART HEADER OR ON THE INTERNAL QUEUE BEING PASSED TO ILRSRT. ANY WRITE REQUESTS NOT PROCESSED BY ILRSRT FOR THE PREVIOUS PARTE ARE STILL ON THIS INTERNAL QUEUE. IF A PARTE HEADER WRITE QUEUE DIFFERENT FROM THE PREVIOUS ONE IS TO BE USED FOR THE PARTE AND REQUESTS ARE LEFT ON THE INTERNAL QUEUE, THE INTERNAL QUEUE IS CLEARED (WRITE REQUESTS RETURNED TO ORIGINAL QUEUE). THE INTERNAL QUEUE WILL BE FILLED IN STEP 2.			
02 IF THERE ARE NO WRITE REQUESTS ON THE INTERNAL QUEUE, IT IS FILLED WITH NEW REQUESTS FROM THE APPROPRIATE PART HEADER QUEUE FOR THE CURRENT PARTE.			

Notes	Routine	Label	Ref

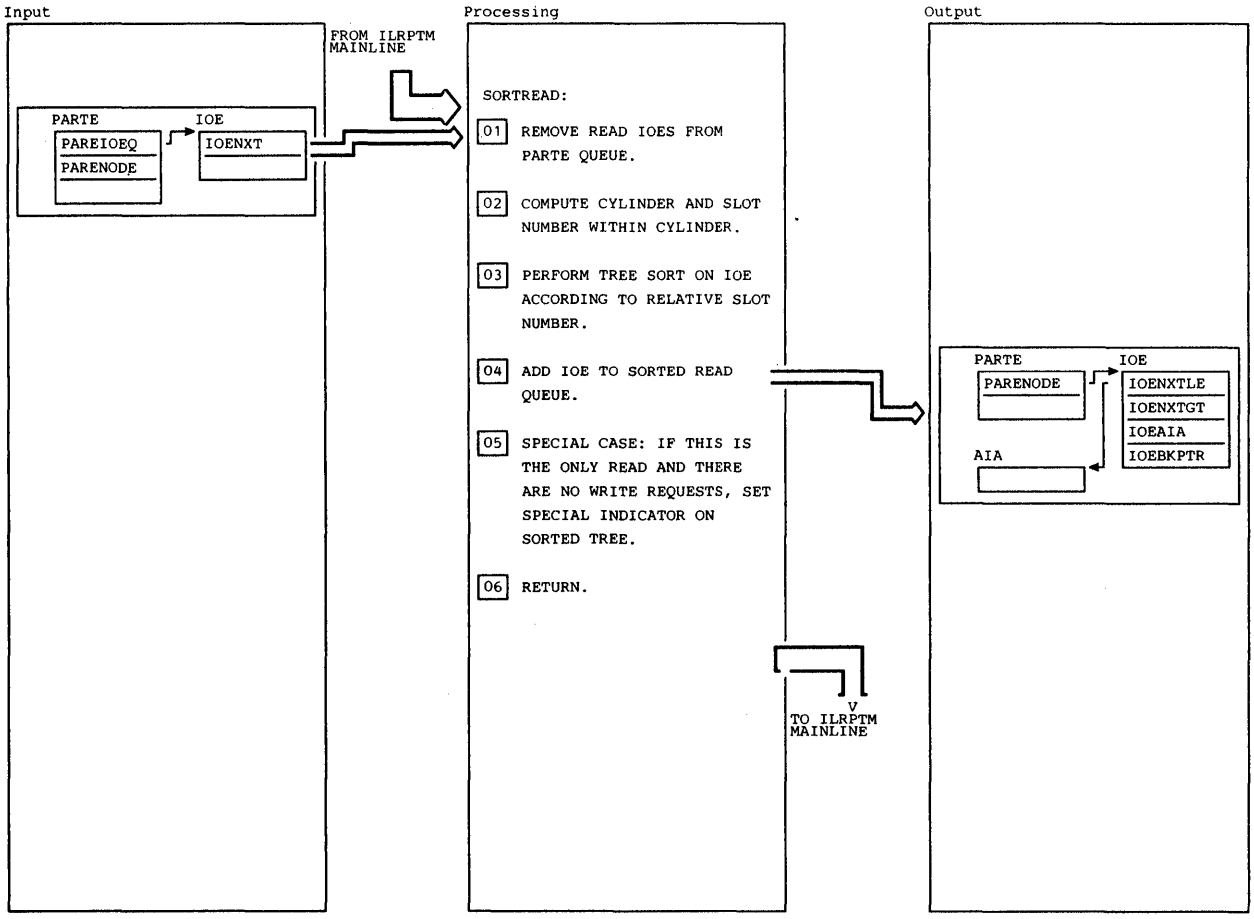
Diagram 25.6.1 GETWRTQ (Part 1 of 2)



Notes	Routine	Label	Ref
03 SINCE BOTH PARTES ARE TO USE THE SAME PART HEADER QUEUE OF WRITE REQUESTS, USE THE REQUESTS LEFT OVER FROM LAST PARTE PROCESSING.			
04 LOCK INDICATORS SET SO THAT LOCK WILL BE OBTAINED ONLY ONCE AND FREED ONLY ONCE.			

Notes	Routine	Label	Ref

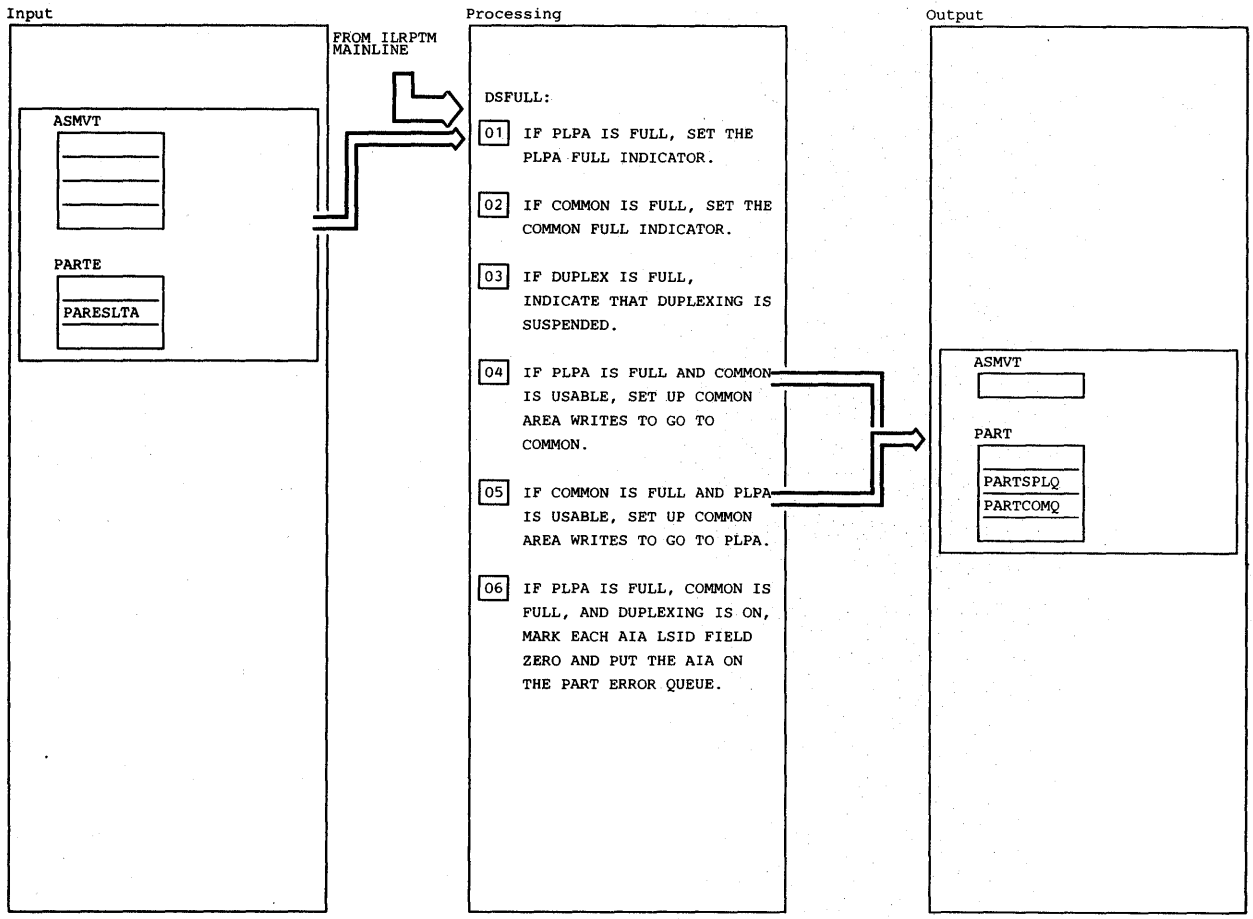
Diagram 25.6.1 GETWRTQ (Part 2 of 2)



Notes	Routine	Label	Ref
01 THE ENTIRE CHAIN OF READ REQUESTS (IOES) IS REMOVED FROM PAREIOEQ AND PAREIOEQ IS SET TO ZERO. COMPARE AND SWAP (CS) IS USED FOR SERIALIZATION.			
02 THE READ IOES ARE SORTED ACCORDING TO CYLINDER AND RELATIVE SLOT NUMBER.			
03 EACH IOE IS PLACED ON A TWO-DIRECTION TREE ACCORDING TO ITS RELATIVE SLOT NUMBER. ONE DIRECTION REPRESENTS LESS THAN OR EQUAL, THE OTHER DIRECTION REPRESENTS GREATER THAN. IN ORDER TO GROUP IOES REPRESENTING READ REQUESTS FROM THE SAME CYLINDER TOGETHER AN INSERTION IS SOMETIMES NECESSARY IN THE MIDDLE OF A 'LEG' OF THE TREE. THE TOP OF THE TREE IS POINTED TO BY PARENODE.			
04 IF THERE ARE NO READS ALREADY ON THE TREE, JUST SET TOP NODE (PARENODE) TO POINT TO THIS IOE. OTHERWISE, CALL ADRTTREE TO ADD		ADRTTREE	25.6.4

Notes	Routine	Label	Ref
THE IOE TO THE SORTED READ QUEUE.			
05 IF THIS IS THE ONLY READ AND THERE ARE NO WRITES, SET PARENODE TO ITS COMPLEMENT FOR A SPECIAL PATH THROUGH ILRSRT.			

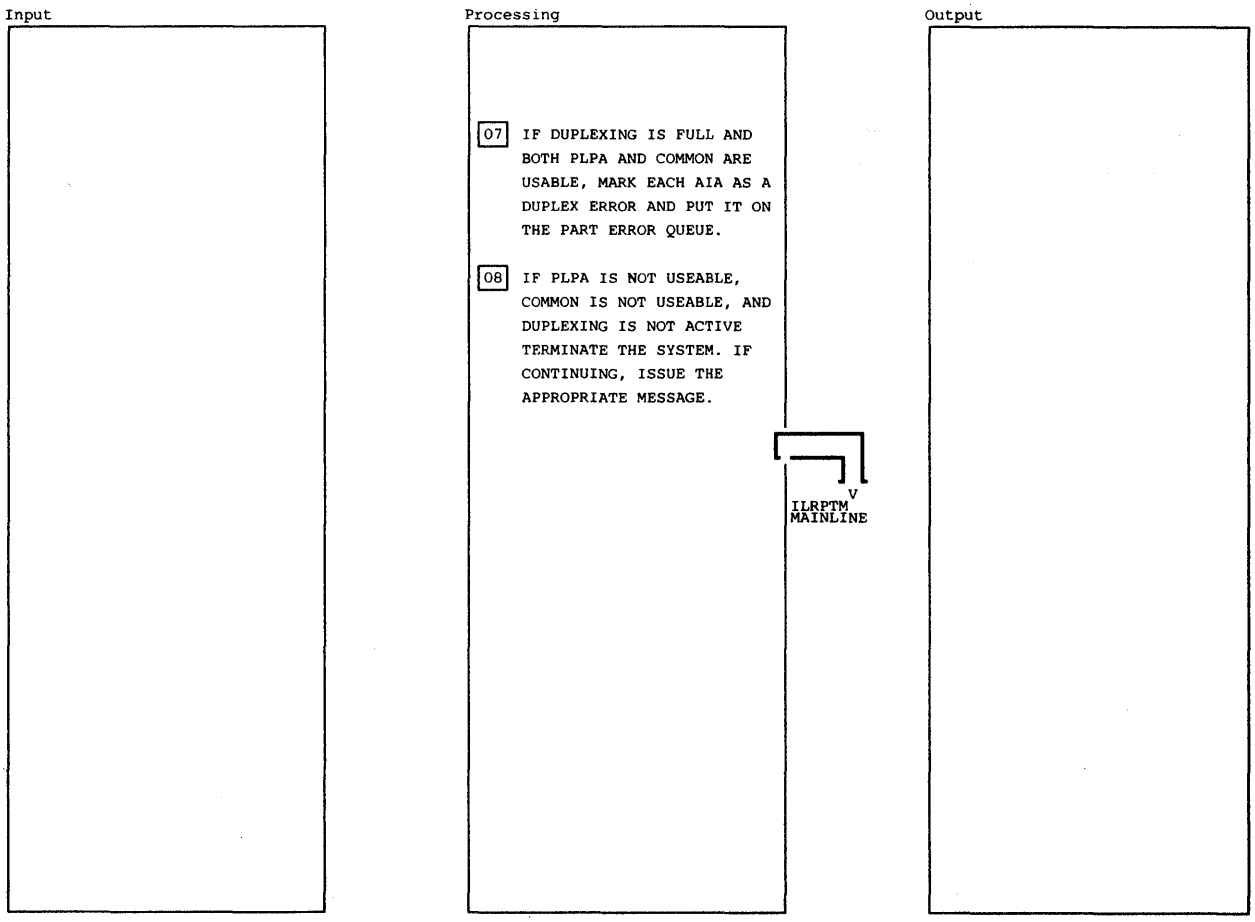
Diagram 25.6.2 SORTREAD (Part 1 of 1)



Notes	Routine	Label	Ref
01 IF PLPA DATA SET IS FULL (NO MORE SLOTS AVAILABLE), SET THE FLAG IN ASMVT (ASMPLPAF).			
02 IF COMMON DATA SET IS FULL, SET THE FLAG IN ASMVT (ASMCOMMF).			
03 IF DUPLEX DATA SET IS FULL, SET THE FLAGS INDICATING DUPLEXING IS SUSPENDED (ASMDUPLX OFF, ASMNODPX ON).			
04 IF PLPA IS FULL, COMMON IS NOT MARKED BAD, AND COMMON IS NOT FULL, SET UP FOR WRITES TO GO TO THE COMMON DATA SET BY SETTING PAREWTQE TO 0 FOR PLPA AND PAREWTQE TO THE ADDRESS OF THE PARTCOMQ FOR COMMON.			
05 IF COMMON IS FULL AND PLPA IS NOT MARKED AS BAD, AND PLPA IS NOT FULL, SET UP FOR THESE WRITES TO GO TO PLPA BY MOVING WRITES TO THE SPECIAL SPILL WRITE QUEUE.			

Notes	Routine	Label	Ref
06 IF PLPA AND COMMON ARE FULL AND DUPLEXING IS STILL ACTIVE, EACH AIA ON THE WRITE QUEUE IS MARKED AS AIALSID EQUAL ZERO AND PUT ON THE PART ERROR QUEUE. THESE AIAS WILL LATER BE SENT TO ILRPAGCM TO HANDLE.			

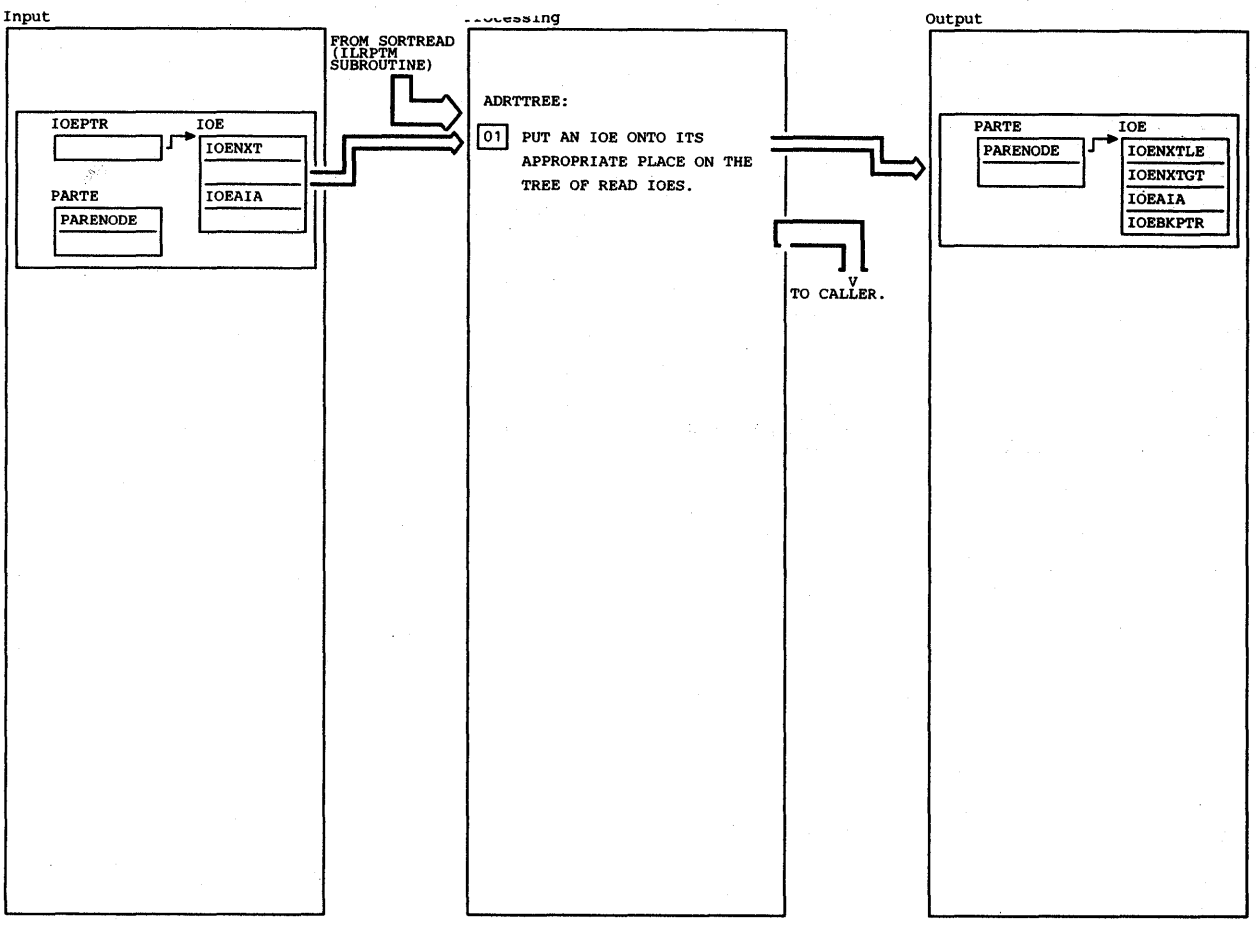
Diagram 25.6.3 DSFULL (Part 1 of 2)



Notes	Routine	Label	Ref
07 IF DUPLEX DATA SET IS FULL AND BOTH PLPA AND COMMON ARE USEABLE, ALL AIAS ON THE DUPLEX WRITE QUEUE WILL BE MARKED AS SECONDARY ERROR AND PUT ON THE PARTAIAE QUEUE. THESE AIAS WILL LATER BE SENT TO ILRPAGCM TO HANDLE.			
08 IF ACCESS TO SOME PLPA OR COMMON PAGES HAS BEEN LOST, ILRMSG00 TERMINATES THE SYSTEM. IF PROCESSING CAN CONTINUE, ILRMSG00 INFORMS THE OPERATOR OF WHAT HAS JUST HAPPENED, IF HE HAS NOT ALREADY BEEN INFORMED.	ILRMSG00	ILRMSG00	

Notes	Routine	Label	Ref

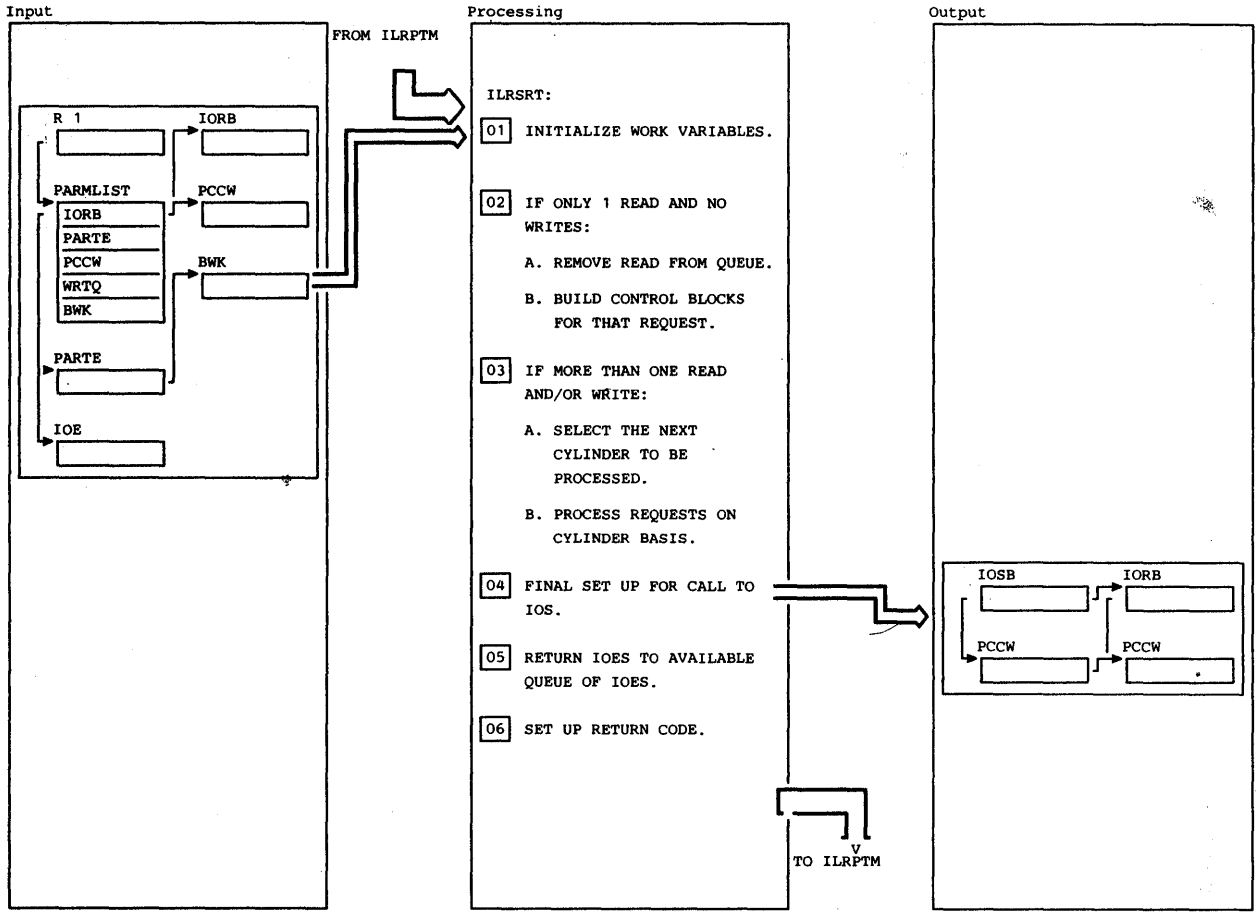
Diagram 25.6.3 DSFULL (Part 2 of 2)



Notes	Routine	Label	Ref
01 THE TREE OF READ IOES IS POINTED TO BY PARENODE. A BACKWARD POINTER IS USED TO ALLOW UPWARD AS WELL AS DOWNWARD MOVEMENT WHEN SCANNING THE TREE. AN INSERTION IS NECESSARY WHEN NORMAL SORTING WOULD SEPARATE TWO NODES ASSOCIATED WITH REQUESTS FOR THE SAME CYLINDER.			

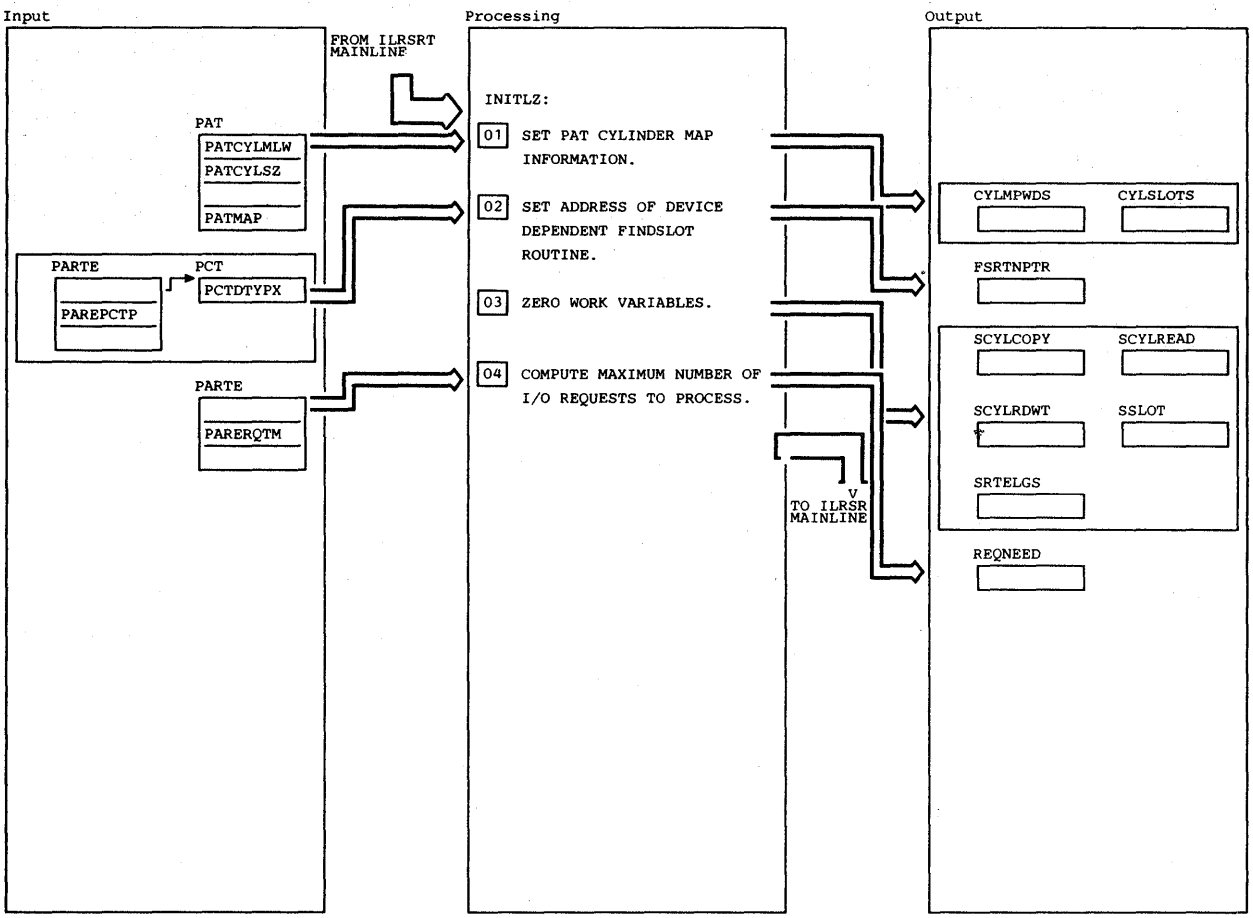
Notes	Routine	Label	Ref

Diagram 25.6.4 ADRTTREE (Part 1 of 1)



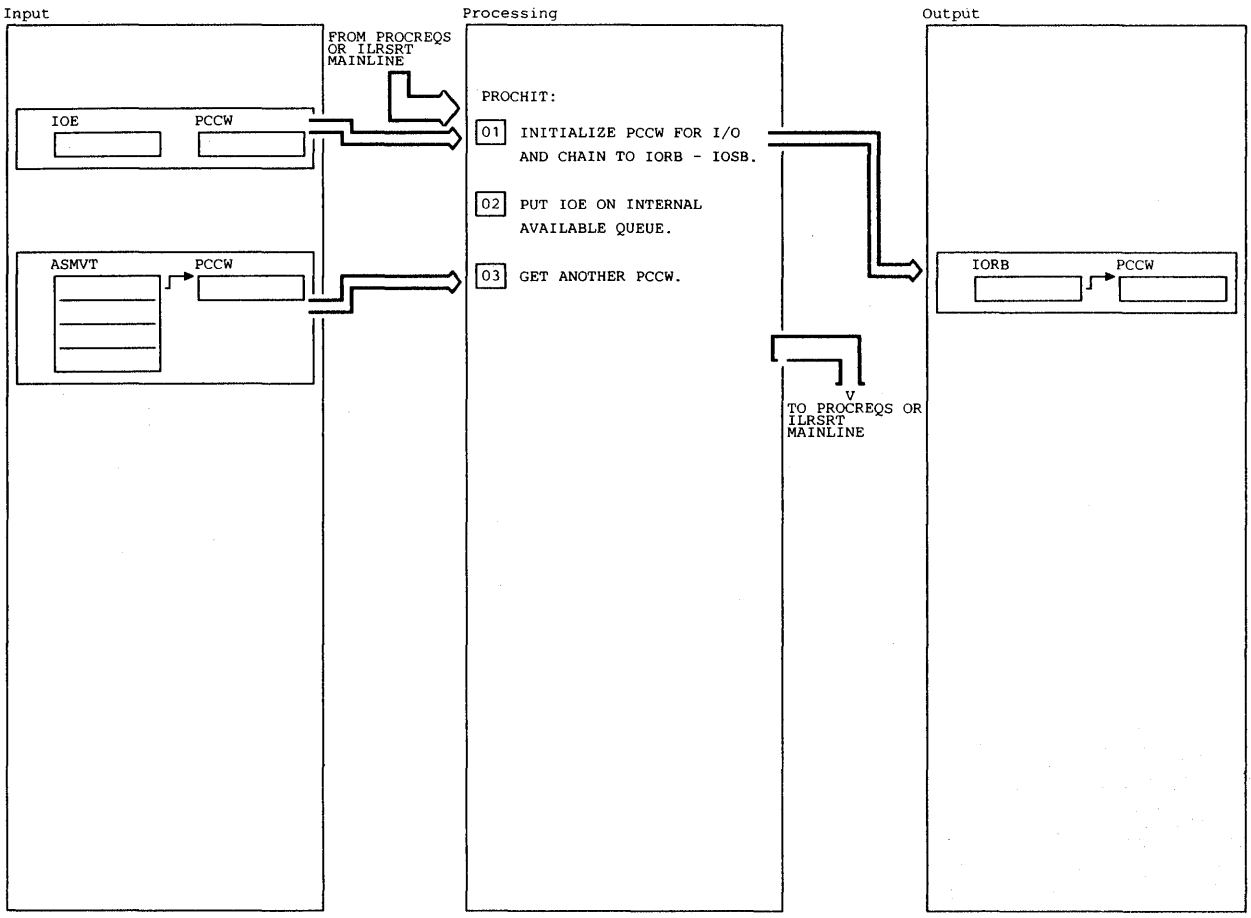
Notes	Routine	Label	Ref
01 ILRSRT IS CALLED BY ILRPTM TO PROCESS ONE PAGE DATA SET. ILRSRT PREPARES THE I/O REQUESTS FOR A SERVICE BURST OF WORK AND STARTS THE I/O. INITIALIZE WORKING VARIABLES, ESTABLISH CONTROL BLOCK ADDRESSABILITY. FOR RECOVERY PURPOSES, ILRSRT01 RECOVERY ROUTINE HANDLES ERRORS OCCURRING IN ILRSRT.		INITLZ	25.7.1
02 A SPECIAL PATH FOR ONE READ AND NO WRITES - PARENODE (THE SORTED READ QUEUE) WILL BE NEGATIVE.			
A. REMOVE READ FROM QUEUE, ZERO QUEUE AND COMPLEMENT ADDRESS TO GET VALID ADDRESS.			
B. CALL PROCHIT TO BUILD CCWS FOR THIS REQUEST.		PROCHIT	25.7.2
03 NORMAL PATH THROUGH SLOT SORT:			
A. DETERMINE CYLINDER TO PROCESS BASED ON THE CURRENT POSITION OF THE CYLINDER.			
B. CALL PROCREQS TO DETERMINE (BASED ON FEWEST REQUIRED ROTATIONS) WHICH REQUESTS WILL BE PROCESSED. PROCESSING CONTINUES UNTIL ENOUGH REQUESTS TO FILL THE SERVICE BURST ARE BUILT, NO MORE PCCWS ARE AVAILABLE, OR THERE ARE NO MORE REQUESTS.		PROCREQS	25.7.4
04 CALL IO TO COMPLETE SET UP AND TO ISSUE THE SIO (START I/O) MACRO.		IO	25.7.5
05 USE COMPARE AND SWAP (CS) TO RETURN STRING OF ALL THE IOES.			
06 RETURN CODES: 0 - SUCCESSFUL, NO WORK REMAINING. 4 - SUCCESSFUL, READS AND OR WRITES LEFT. 8 - DATA SET FULL, NO READS LEFT. 12 - DATA SET FULL, READS LEFT.			

Diagram 25.7 ILRSRT (Part 1 of 1)



Notes	Routine	Label	Ref
01 OBTAIN INFORMATION NEEDED TO ACCESS APPROPRIATE SECTION OF PATMAP. THIS INFORMATION IS THE NUMBER OF WORDS TO MAP A CYLINDER AND THE NUMBER OF SLOTS IN A CYLINDER, AND IT IS DEVICE-TYPE DEPENDENT.			
02 DEVICE TYPE IS DETERMINED FROM THE PCTCTYPX FIELD IN THE PCT. A SEPARATE FINDSLOT ROUTINE EXISTS FOR EACH DEVICE TYPE.			
03 ZERO LAST CHOSEN SLOT NUMBER IN FINDSLOT PARM LIST, INITIALIZE ALL SLOT FLAGS OFF IN FINDSLOT PARM LIST, ZERO READ CYLINDER VALUE (INDICATING READ CYLINDER TO BE FOUND), INITIALIZE ALL INTERNAL FLAGS OFF, ZERO RETURN CODE.			
04 THE COMPUTATION CONSISTS OF THE LENGTH OF A 'SERVICE BURST' (ASMBURST) DIVIDED BY THE TIME TO PROCESS A SINGLE REQUEST (PARERQTM) PLUS TWO. A MINIMUM OF TWO REQUESTS WILL ALWAYS BE SET IN REQNEED.			

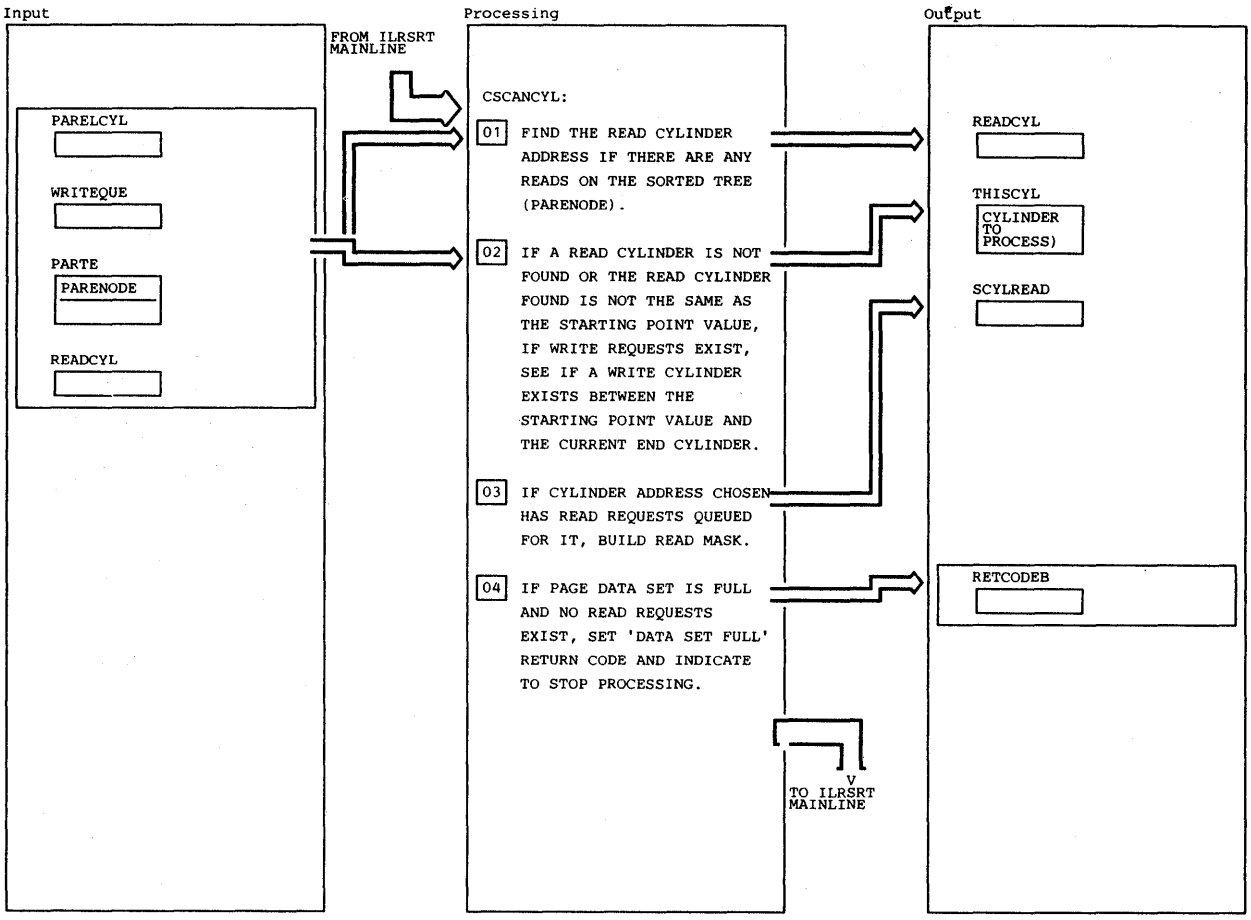
Diagram 25.7.1 INITLZ (Part 1 of 1)



Notes	Routine	Label	Ref
01 CALL IOCHAIN TO INITIALIZE PCCW AND CHAIN IT TO IOSB-IORB.		IOCHAIN	25.7.6
02 PUT IOE ON INTERNAL QUEUE. WHEN PROCESSING COMPLETE, ALL IOES WILL BE FREED USING ONE COMPARE AND SWAP (CS).			
03 IF MORE REQUESTS TO PROCESS AND SERVICE BURST NOT MET YET, GET A ANOTHER PCCW FROM AVAILABLE QUEUE.			

Notes	Routine	Label	Ref

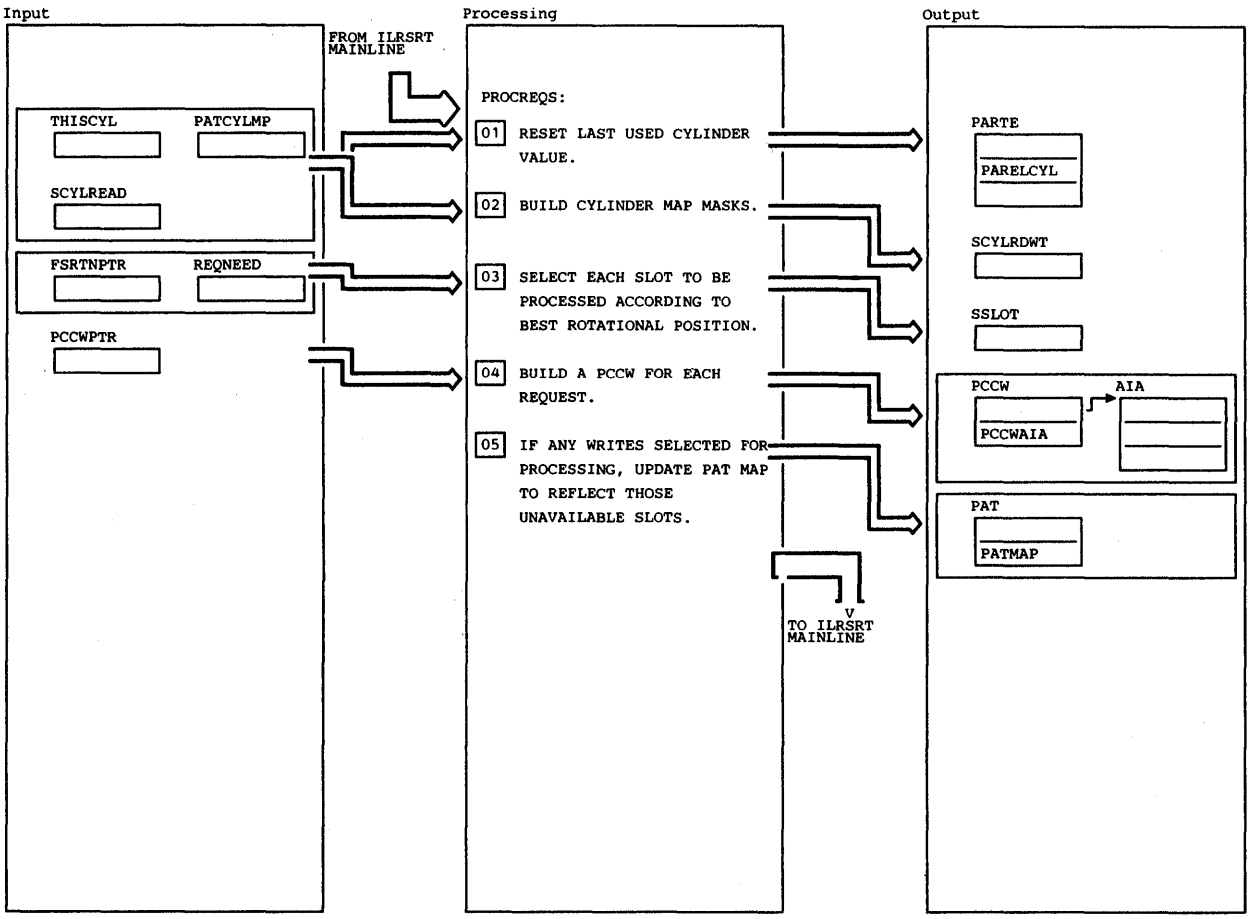
Diagram 25.7.2 PROCHIT (Part 1 of 1)



Notes	Routine	Label	Ref
01 STARTING POINT VALIE IS THE CYLINDER OF THIS DATA SET LAST PROCESSED. END CYLINDER IS INITIALLY SET TO THE END OF THE DATA SET. IF A NEW READ CYLINDER MUST BE FOUND, CALL GETRDCYL. SET THISCYL AND END CYLINDER EQUAL TO READCYL IF A READ CYLINDER IS FOUND.		GETRDCYL	25.7.7
02 THE CURRENT END CYLINDER IS EITHER THE NEXT READ CYLINDER OR THE END OF THE DATA SET. WHEN THE END OF THE DATA SET IS REACHED, THE STARTING CYLINDER IS RESET TO THE BEGINNING OF THE DATA SET. CALL GETWCYL TO FIND WRITE CYLINDER.		GETWCYL	25.7.8
03 CALL BRDMASK TO BUILD MASK OF READ REQUESTS FOR THIS CYLINDER.		BRDMASK	25.7.9
04 INDICATE RETURN CODE 64 TO INDICATE 'DATA SET FULL'.			

Notes	Routine	Label	Ref

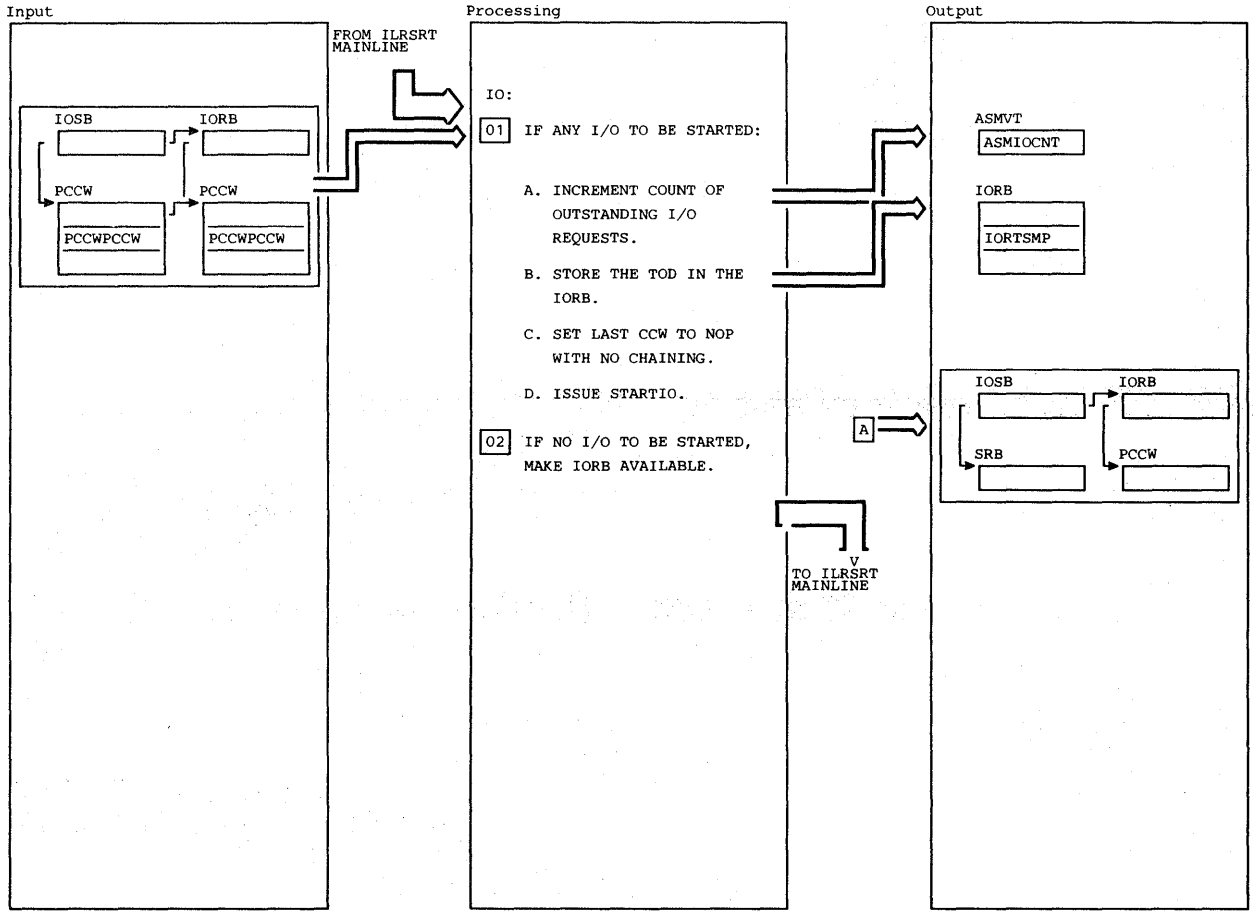
Diagram 25.7.3 CSCANCYL (Part 1 of 1)



Notes	Routine	Label	Ref
01 SAVE CYLINDER NUMBER FOR NEXT CYLINDER SCAN SO THAT READ REQUESTS ARE PROCESSED WITH MINIMUM ROTATION.			
02 FINISH BUILDING THE CYLINDER MAP MASKS TO BE USED IN PROCESSING REQUESTS ON THIS CYLINDER.		BILDMSKS	25.7.10
03 FIND NEXT BEST (REQUIRING FEWEST ROTATIONS) SLOT TO BE USED.		FINDSLOT	25.7.11
04 ASSIGN I/O TO SLOT FOUND VIA FINDSLOT ROUTINE. REPEAT STEPS 3 AND 4 UNTIL NO MORE REQUESTS FOR THIS CYLINDER, A RESOURCE HAS RUN OUT, OR THE REQUEST QUOTA FOR THE SERVICE BURST HAS BEEN MET.		PROCHIT	25.7.2
05 UPDATE PAT CYLINDER MAP AND PART ENTRY SLOTS AVAILABLE COUNT.		WRTUPDTE	25.7.12

Notes	Routine	Label	Ref

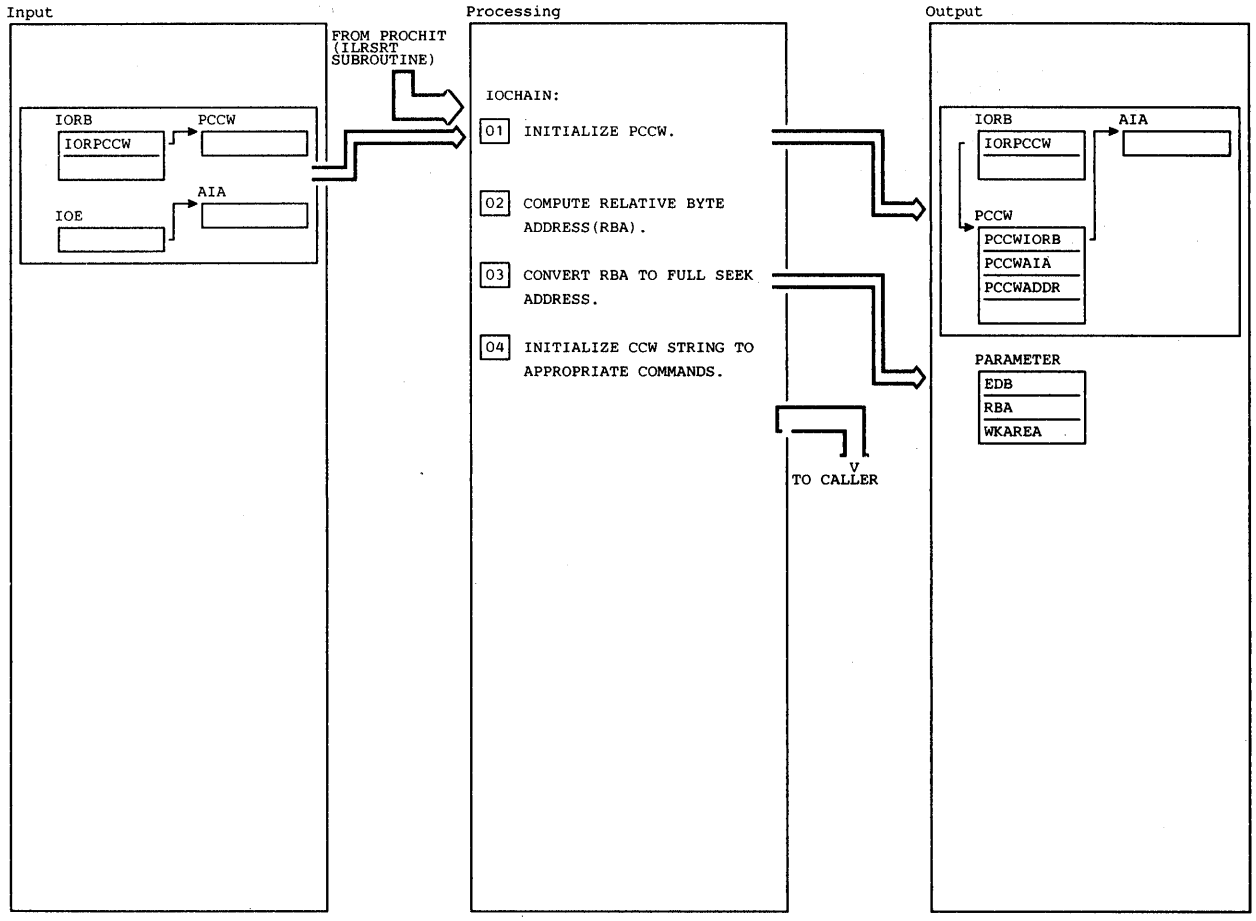
Diagram 25.7.4 PROCREQS (Part 1 of 1)



Notes	Routine	Label	Ref
01 IF ANY REQUESTS QUEUED: A. INCREMENT COUNT OF OUTSTANDING IOSBS FOR ASM. B. IF THE SERVICE BURST HAS BEEN FILLED, STORE THE CLOCK. OTHERWISE, ZERO THE TOD(TIME OF DAY). C. SET LAST CCW TO NOP (NO OPERATION) AND STOP CHAINING. THIS WILL END THE CHANNEL PROGRAM FOR THE CHANNEL. D. GO TO IOS VIA THE STARTIO MACRO.	STARTIO		
02 IF NO I/O IS TO BE STARTED, TURN OFF THE 'IN USE' FLAG IN THE IORB.			

Notes	Routine	Label	Ref

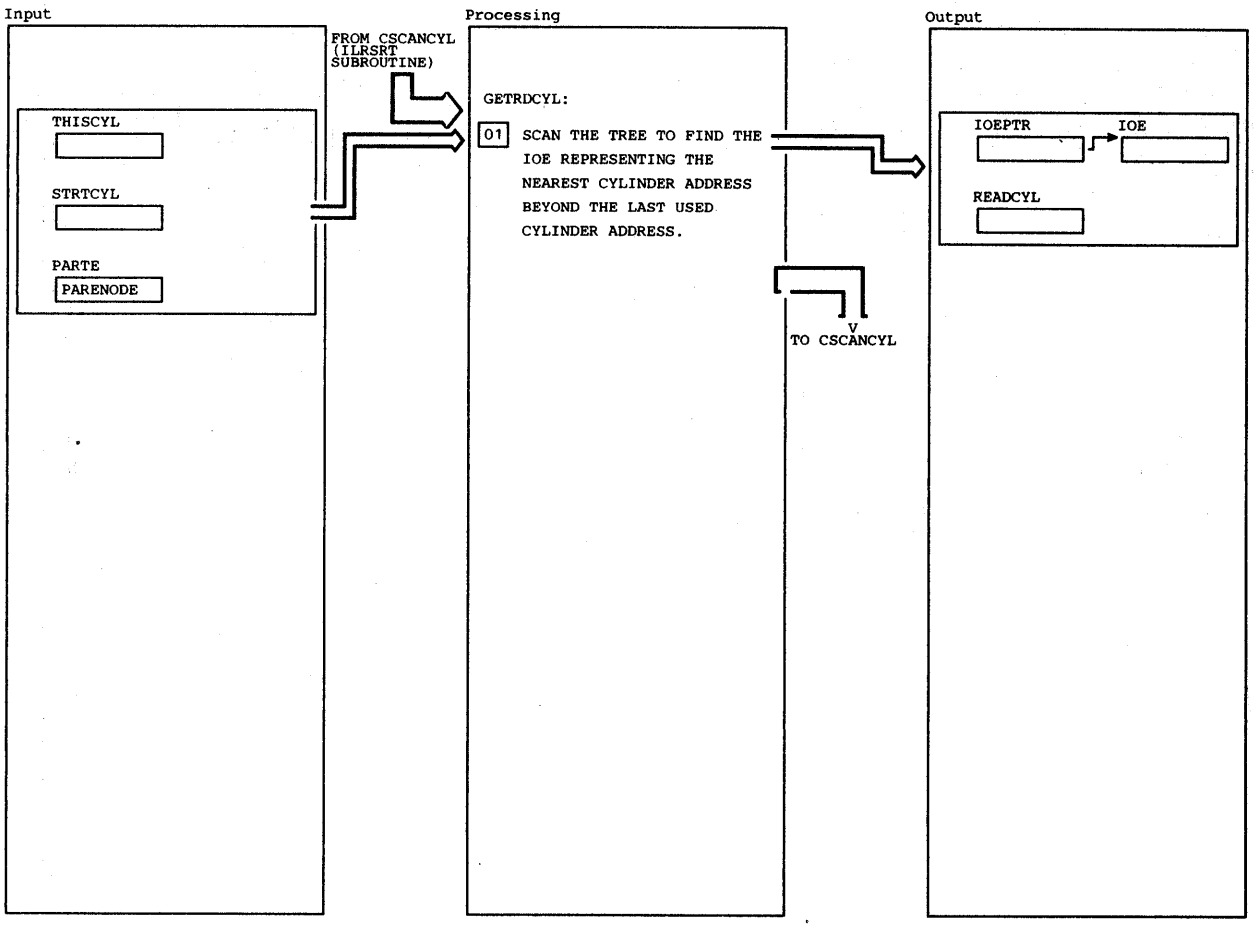
Diagram 25.7.5 IO (Part 1 of 1)



Notes	Routine	Label	Ref
01 PCCWAIA IS SET TO POINT TO THE AIA CURRENTLY POINTED TO BY THE IOE FOR THIS REQUEST. PCCWIORB IS A BACKWARD POINTER TO THE IORB. THE REAL ADDRESS OF AREA TO WRITE OUT OR READ INTO IS PUT IN PCCWADDR.			
02 THE SLOT NUMBER IS CONVERTED TO AN RBA BY MULTIPLYING THE SLOT NUMBER BY 4096.			
03 CONVERT RBA TO A FULL SEEK ADDRESS (MBBCCHHR). IF AN ERROR IS ENCOUNTERED DURING THE CONVERT A X'083' ABEND IS ISSUED SINCE EITHER THE EDB (EXTENSION DATA BLOCK) OR THE PAT HAS BEEN OVERLAID.			
04 THE APPROPRIATE STRING OF CCWS IS SET UP. FOR THE FIRST PCCW, THE CHANNEL PROGRAM STARTS WITH THE SET SECTOR FOR RPS (ROTATIONAL POSITION SENSING) AND THE SEARCH FOR NON RPS. THE READ/WRITE CCW IS CONVERTED TO THE APPROPRIATE CODE FOR READ OR			

Notes	Routine	Label	Ref
WRITE. FOR ALL OTHER PCCWS, THE CHANNEL PROGRAM MAY START WITH A SEEK CYLINDER, SEEK HEAD, SET SECTOR, OR SEARCH DEPENDING ON THE PREVIOUS CCW STRING. A SET SECTOR IS ONLY USED WHEN THERE IS ENOUGH ROOM TO DO A SET SECTOR AND NOT LOSE A REVOLUTION. THE PREVIOUS LAST CCW IS SET TO A TIC TO THE FIRST CCW IN THIS PCCW. THE READ/WRITE CCW IS SET TO THE APPROPRIATE CODE.			

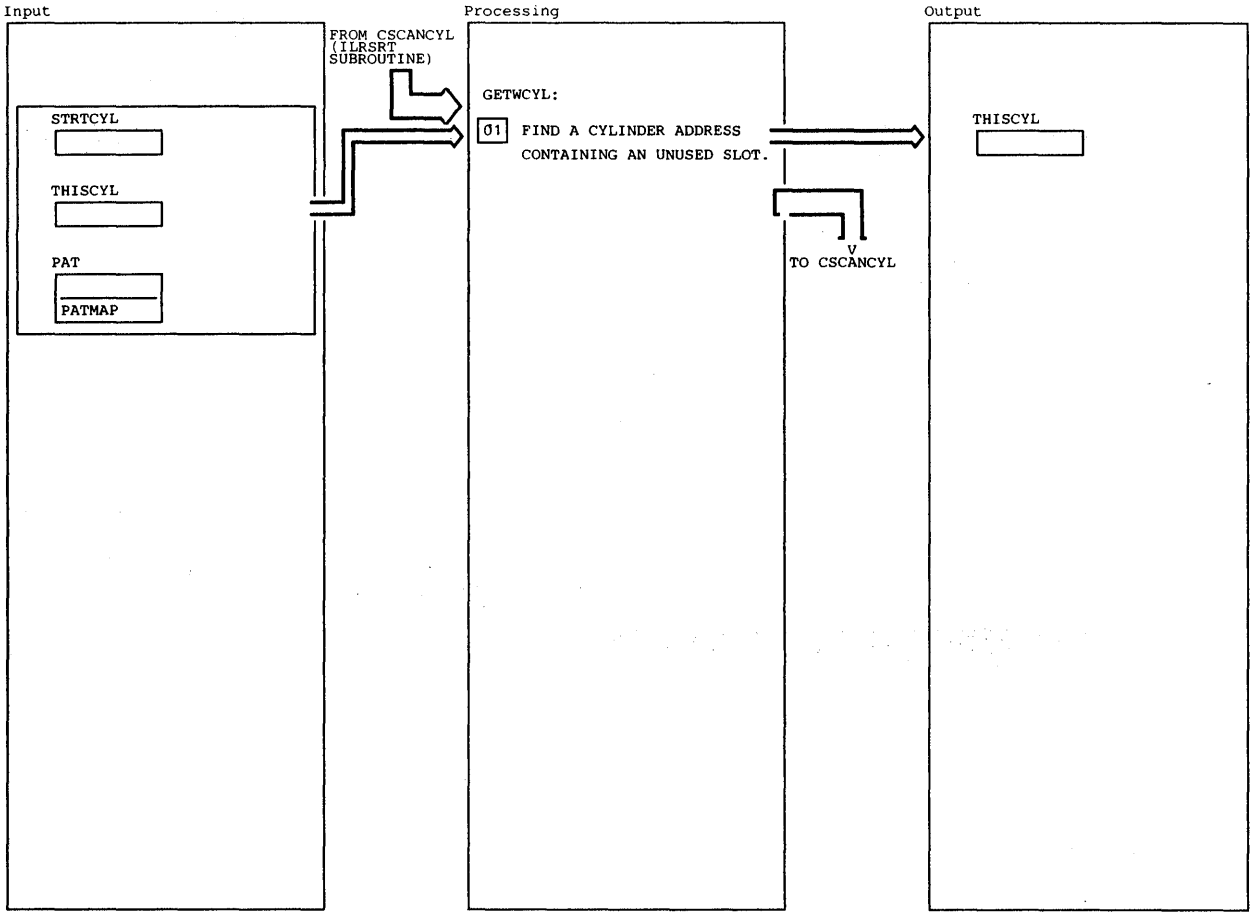
Diagram 25.7.6 IOCHAIN (Part 1 of 1)



Notes	Routine	Label	Ref
01 THE TREE, POINTED TO BY PARENODE, IS EXAMINED, MOVING DOWN ONE NODE AT A TIME ALONG THE APPROPRIATE LEG, KEYING ON THE CYLINDER ADDRESS VALUE ASSOCIATED WITH EACH IOE, IN SEARCH OF ONE OR MORE IOES REPRESENTING REQUESTS FOR THE NEAREST CYLINDER ADDRESS TO THE LAST USED CYLINDER ADDRESS. IF NO READ CYLINDER ADDRESS IS FOUND BETWEEN THE CURRENT C-SCAN (CYLINDER SCAN) END POINTS, NO WORK VARIABLES ARE ALTERED. OTHERWISE READCYL IS SET TO NEW READY CYLINDER ADDRESS.			

Notes	Routine	Label	Ref

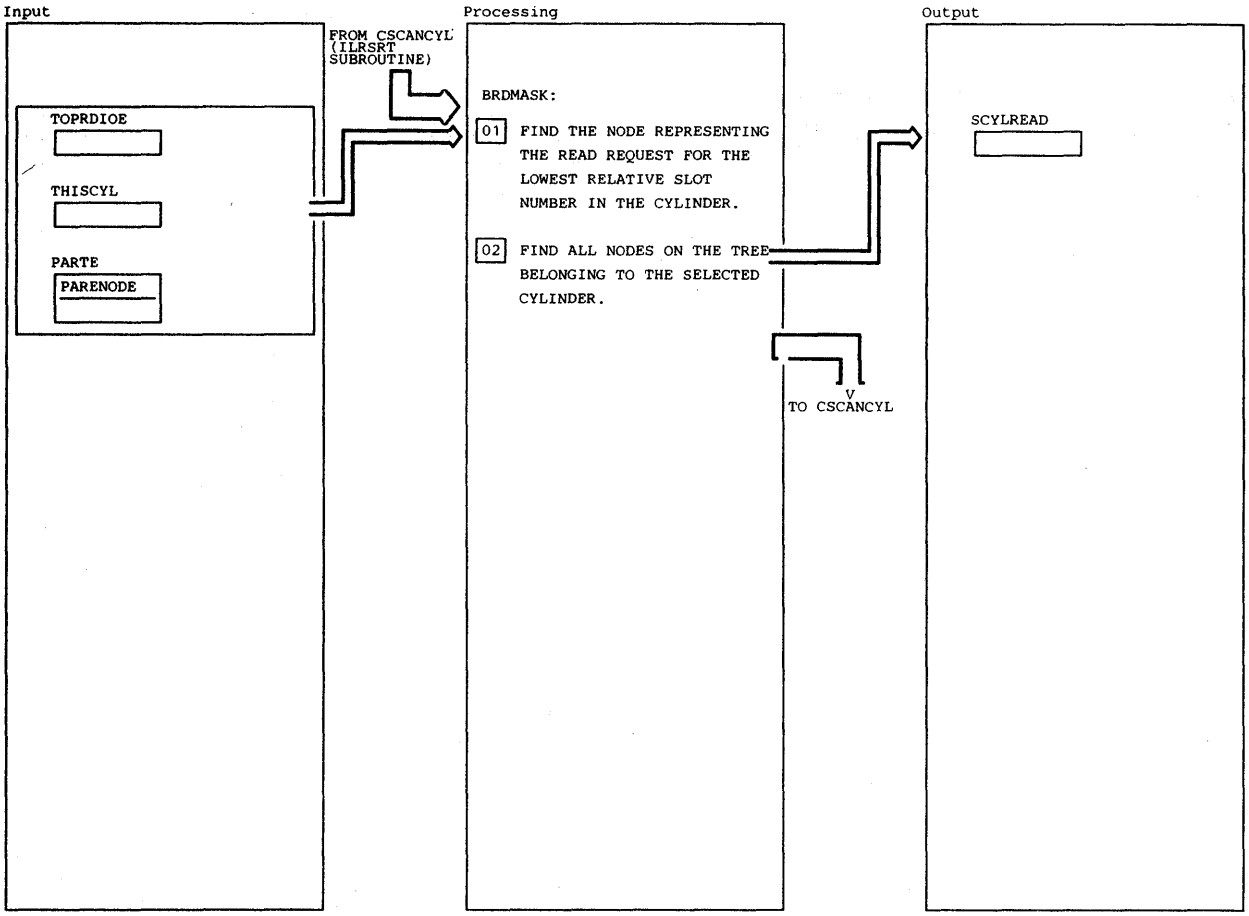
Diagram 25.7.7 GETRDCYL (Part 1 of 1)



Notes	Routine	Label	Ref
01 A FREE SLOT IS REPRESENTED BY A BIT OFF IN THE PAT MAP. THE SCAN IS PERFORMED BETWEEN THE BEGINNING AND ENDING CYLINDER VALUES PASSED AS INPUT PARAMETERS. IF NO CYLINDER ADDRESS IS FOUND THISCYL WILL NOT BE RESET. OTHERWISE, THISCYL WILL BE SET TO WRITE CYLINDER ADDRESS.			

Notes	Routine	Label	Ref

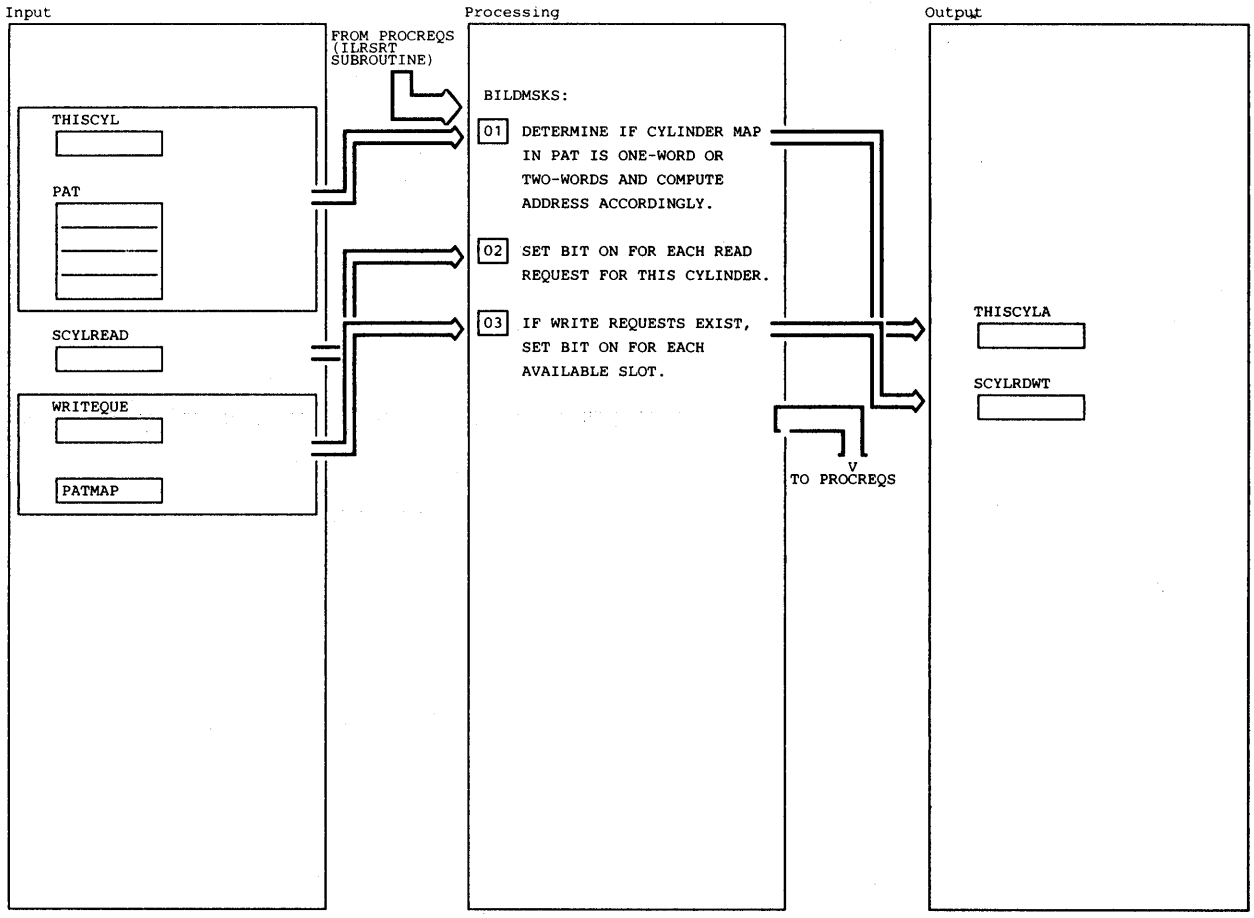
Diagram 25.7.8 GETWCYL (Part 1 of 1)



Notes	Routine	Label	Ref
01 CALL GETLOLEC TO GET LESS-THAN-EQUAL-TO NODE FOR THIS CYLINDER.		GETLOLEC	25.7.13
02 THE APPROPRIATE BIT IN THE CYLINDER READ MASK (SCYLREAD) IS SET FOR EACH RELATIVE SLOT NUMBER FOUND REPRESENTING A READ REQUEST FOR THIS CYLINDER.			

Notes	Routine	Label	Ref

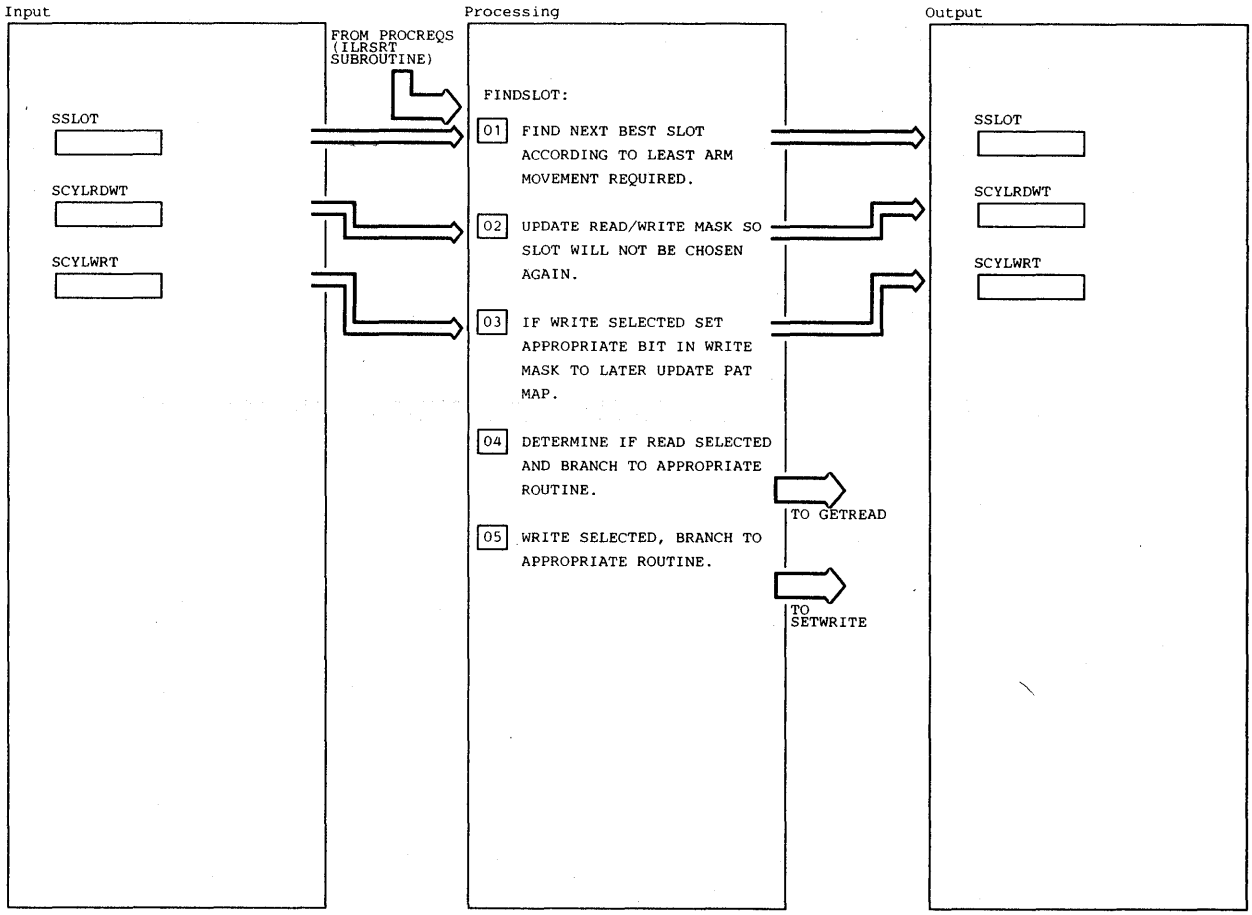
Diagram 25.7.9 BRDMASK (Part 1 of 1)



Notes	Routine	Label	Ref
01 THE 3330 IS THE ONLY DEVICE TYPE WHICH REQUIRES A 2-WORD MAP TO DESCRIBE ONE CYLINDER, AS THERE ARE 58 SLOTS(RECORDS) PER CYLINDER.			
02 A BIT IS TURNED ON IN SCYLRDWT FOR EACH READ REQUEST FOR THIS CYLINDER.			
03 IF WRITE REQUESTS EXIST, TURN ON A BIT IN SCYLRDWT FOR EACH AVAILABLE SLOT.			

Notes	Routine	Label	Ref

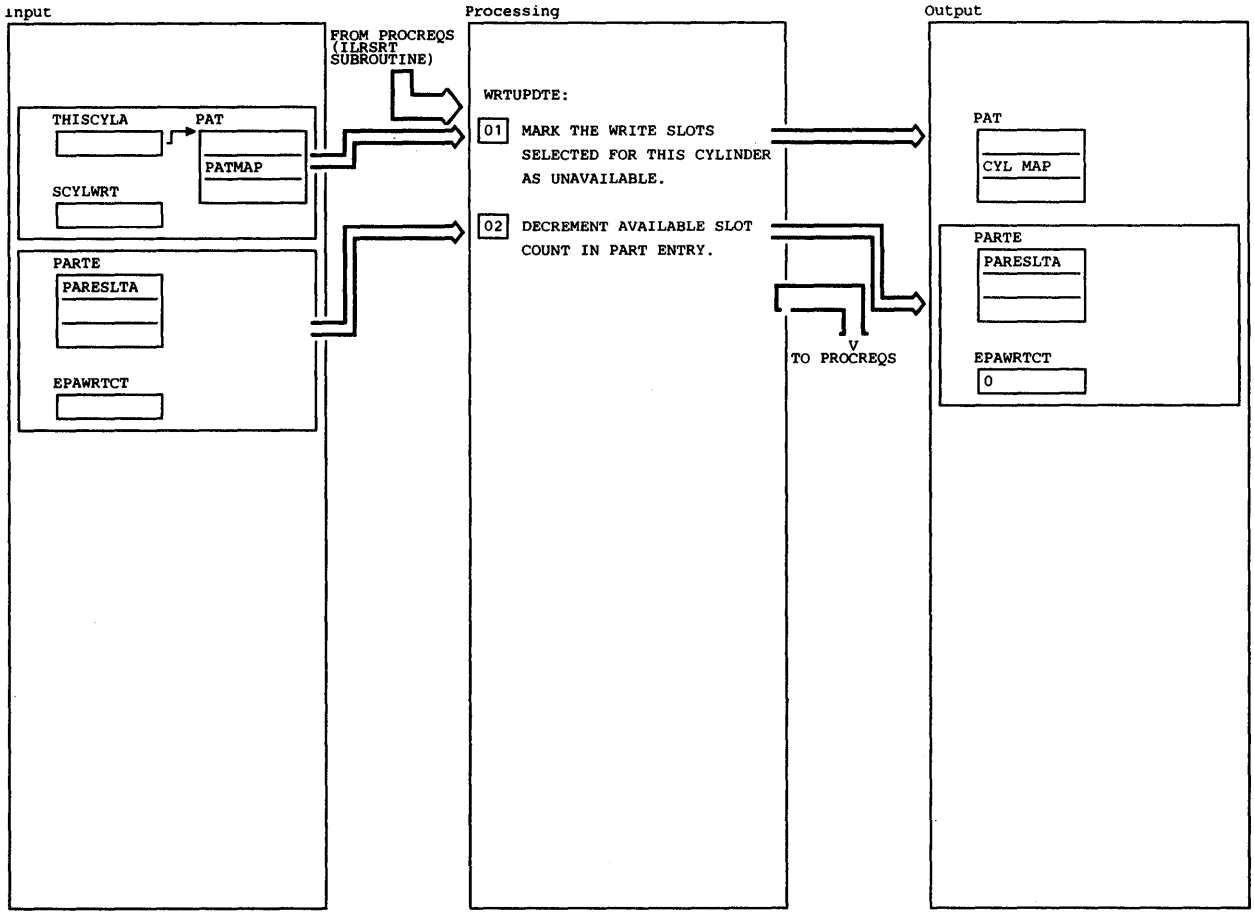
Diagram 25.7.10 BILDMSKS (Part 1 of 1)



Notes	Routine	Label	Ref
01 EACH DEVICE TYPE HAS A DIFFERENT TRACK LAYOUT DUE TO TRACK SIZE AND NUMBER OF TRACKS PER CYLINDER. A SEPARATE FINDSLOT ROUTINE IS USED TO DETERMINE THE NEXT BEST SLOT ON EACH DEVICE TYPE.			
02 ROUTINE IS CALLED FOR EACH I/O REQUEST FOR THIS CYLINDER WITH SCYLRDWT AS INPUT.			
03 FOR WRITE SLOTS SELECTED, PATMAP MUST BE UPDATED TO INDICATE SLOT IS ALLOCATED.			
04 SLOT SELECTED IS FOR READ REQUEST.		GETREAD	25.7.14
05 SLOT SELECTED IS FOR WRITE REQUEST.		SETWRITE	25.7.15

Notes	Routine	Label	Ref

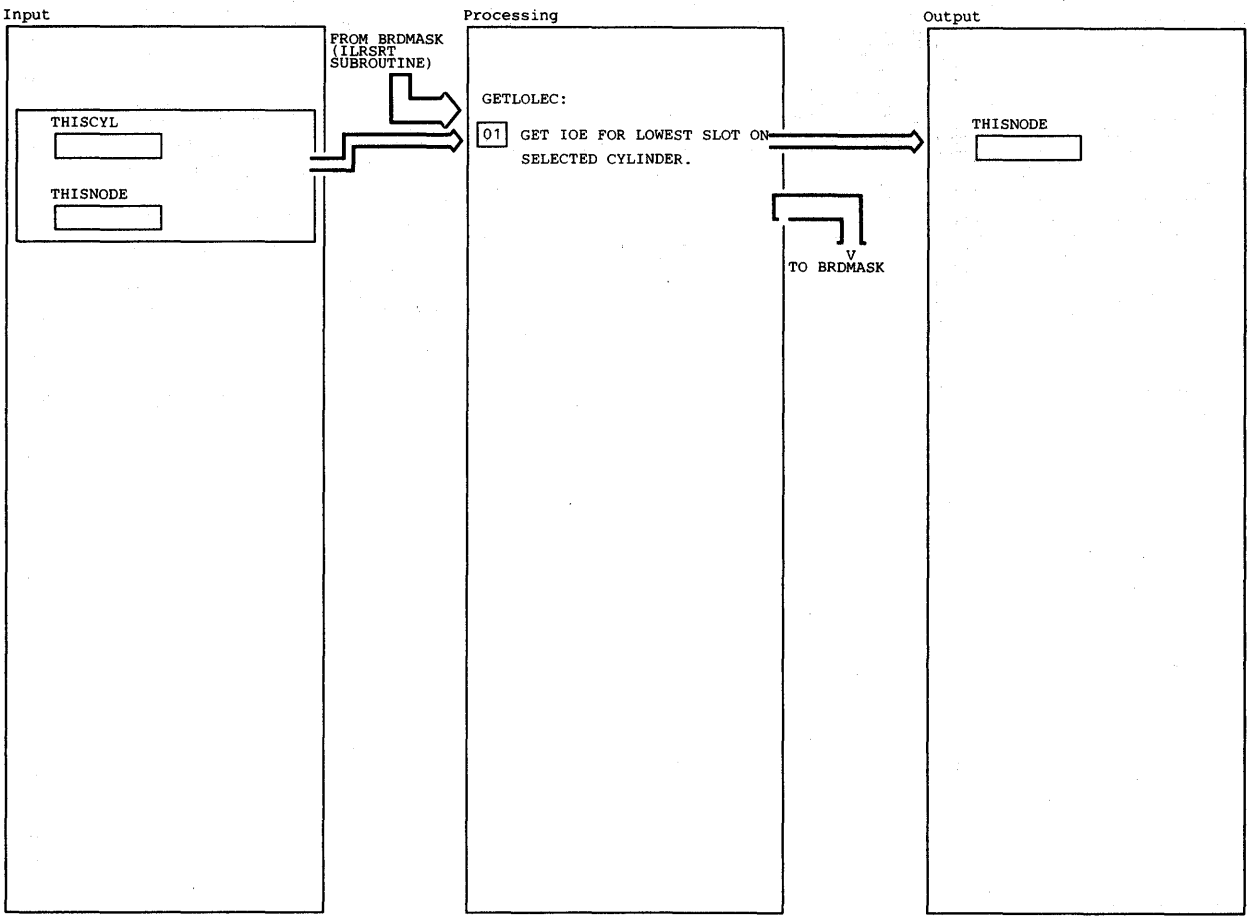
Diagram 25.7.11 FINDSLOT (Part 1 of 1)



Notes	Routine	Label	Ref
01 THISCYLA CONTAINS THE ADDRESS OF THE CYLINDER MAP WITHIN THE PAT FOR THE CYLINDER SELECTED.			
02 EPAWRTCT IS AN INITIAL COUNT OF SLOTS THAT HAVE BEEN ASSIGNED FOR WRITES. PARESLTA, AVAILABLE SLOTS, IS DECREMENTED BY EPAWRTCT. EPAWRTCT IS SET TO ZERO.			

Notes	Routine	Label	Ref

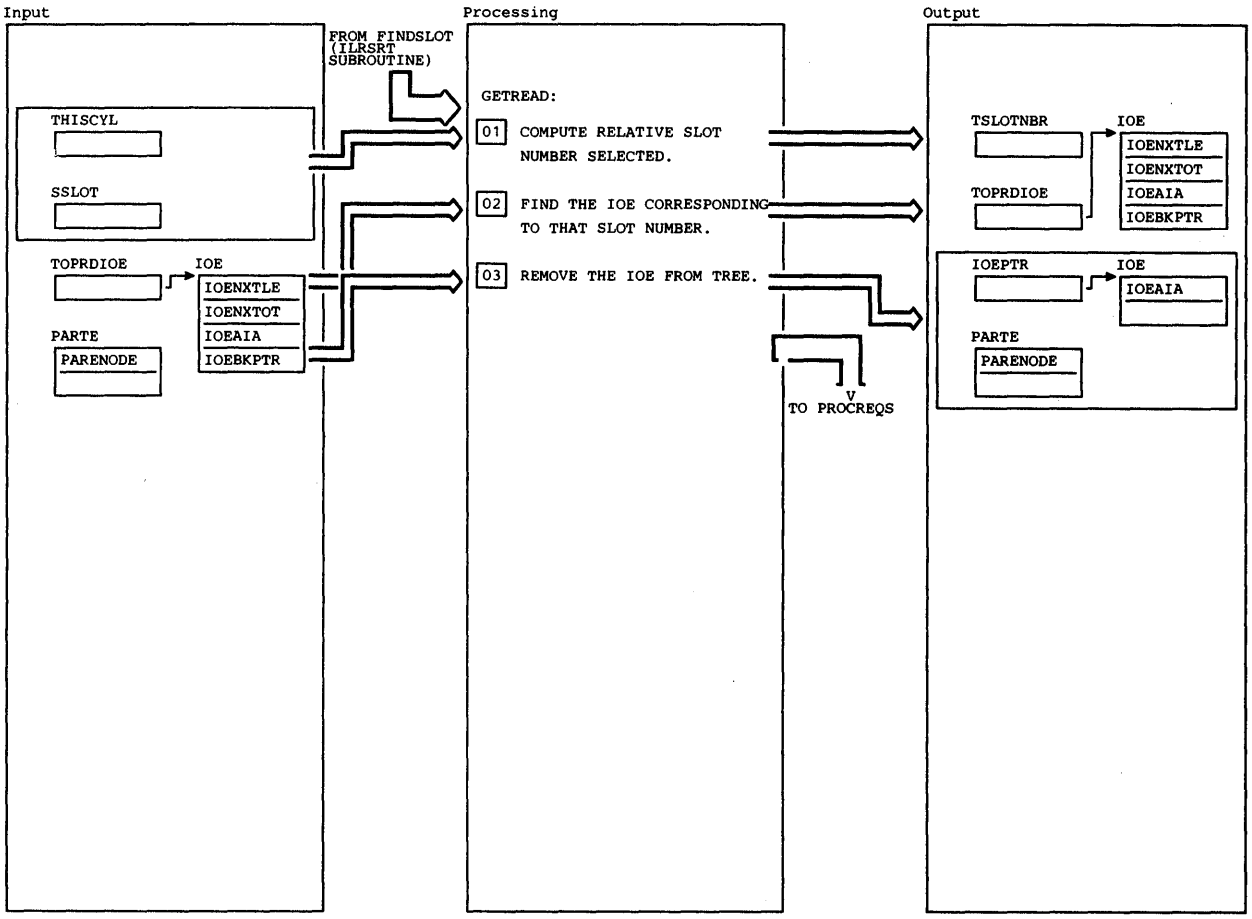
Diagram 25.7.12 WRTUPDTE (Part 1 of 1)



Notes	Routine	Label	Ref
01 MOVE DOWN THE LEG OF THE TREE, STARTING FROM THE NODE PASSED AS INPUT, UNTIL THE END OF THE LEG IS REACHED OR THE CYLINDER NUMBER CHANGES. RETURN A POINTER TO THE LOWEST NODE STILL ON THE SELECTED CYLINDER.			

Notes	Routine	Label	Ref

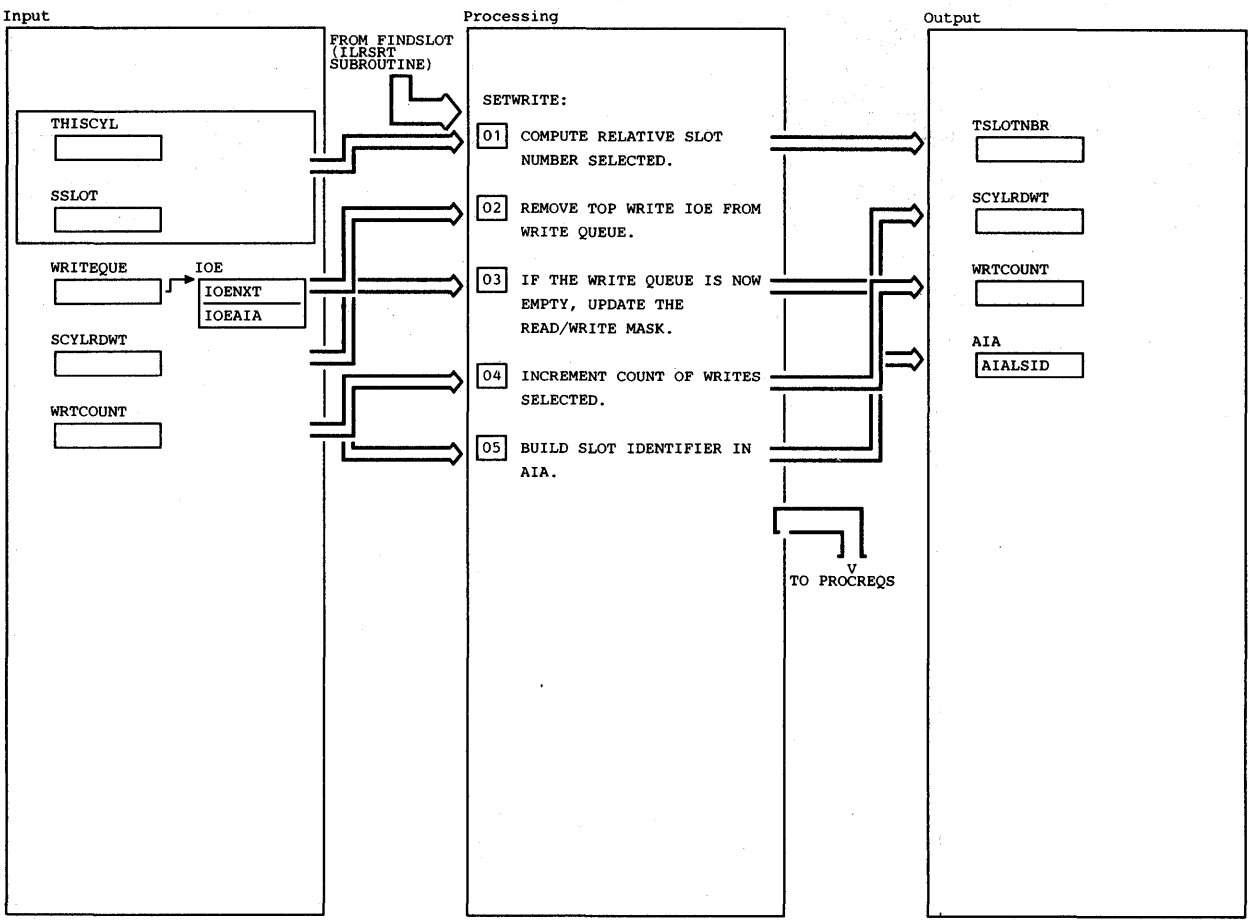
Diagram 25.7.13 GETLOLEC (Part 1 of 1)



Notes	Routine	Label	Ref
01 THE FINDSLOT ROUTINE PASSES A SLOT VALUE RELATIVE TO THE START OF A CYLINDER. THIS VALUE MUST BE CONVERTED TO THE RELATIVE SLOT NUMBER FROM THE BEGINNING OF THE PAGE DATA SET.			
02 FIND READ REQUEST ON TREE CORRESPONDING TO THE SLOT FOUND BY THE FINDSLOT ROUTINE.			
03 CALL REMVNODE TO REMOVE IOE FROM THE TREE (SORTED READ QUEUE - PARENODE).		REMVNODE	25.7.16

Notes	Routine	Label	Ref

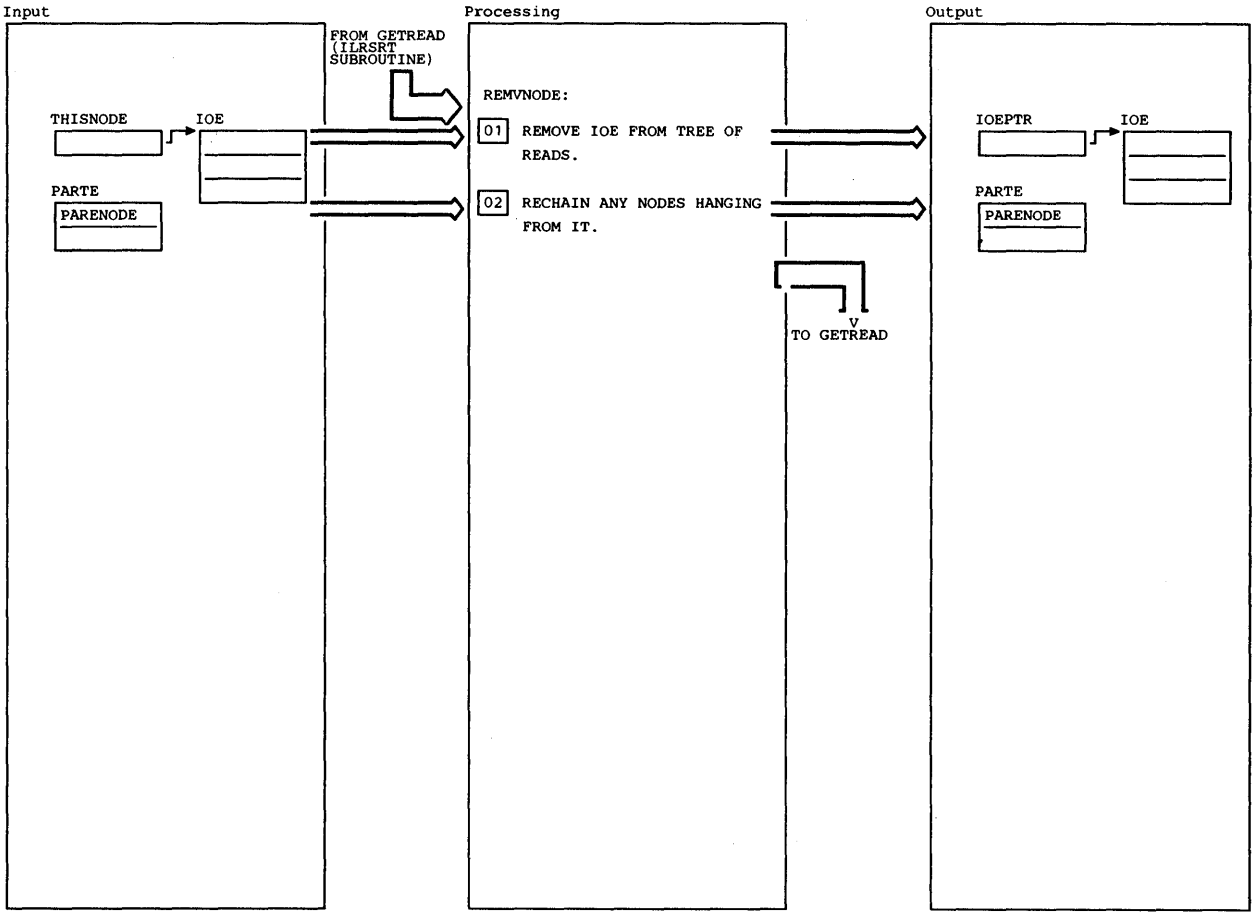
Diagram 25.7.14 GETREAD (Part 1 of 1)



Notes	Routine	Label	Ref
01 THE FINDSLOT ROUTINE PASSES A SLOT VALUE RELATIVE TO THE START OF THE SELECTED CYLINDER. THIS MUST BE CONVERTED TO THE RELATIVE SLOT NUMBER FROM THE BEGINNING OF THE PAGE SPACE.			
02 THE FIRST WRITE IOE IS SELECTED TO USE THE SLOT CHOSEN BY FINDSLOT.			
03 THE READ/WRITE MASK IS UPDATED SO THAT NO MORE WRITES WILL BE SELECTED.			
04 THIS COUNT ÍS LATER USED TO UPDATE THE COUNT OF ALLOCATED SLOTS IN THE PART ENTRY (PARESLTA).			
05 BUILD LOGICAL SLOT ID(LSID) OF THE SLOT BEING WRITTEN TO IN THE AIA.			

Notes	Routine	Label	Ref

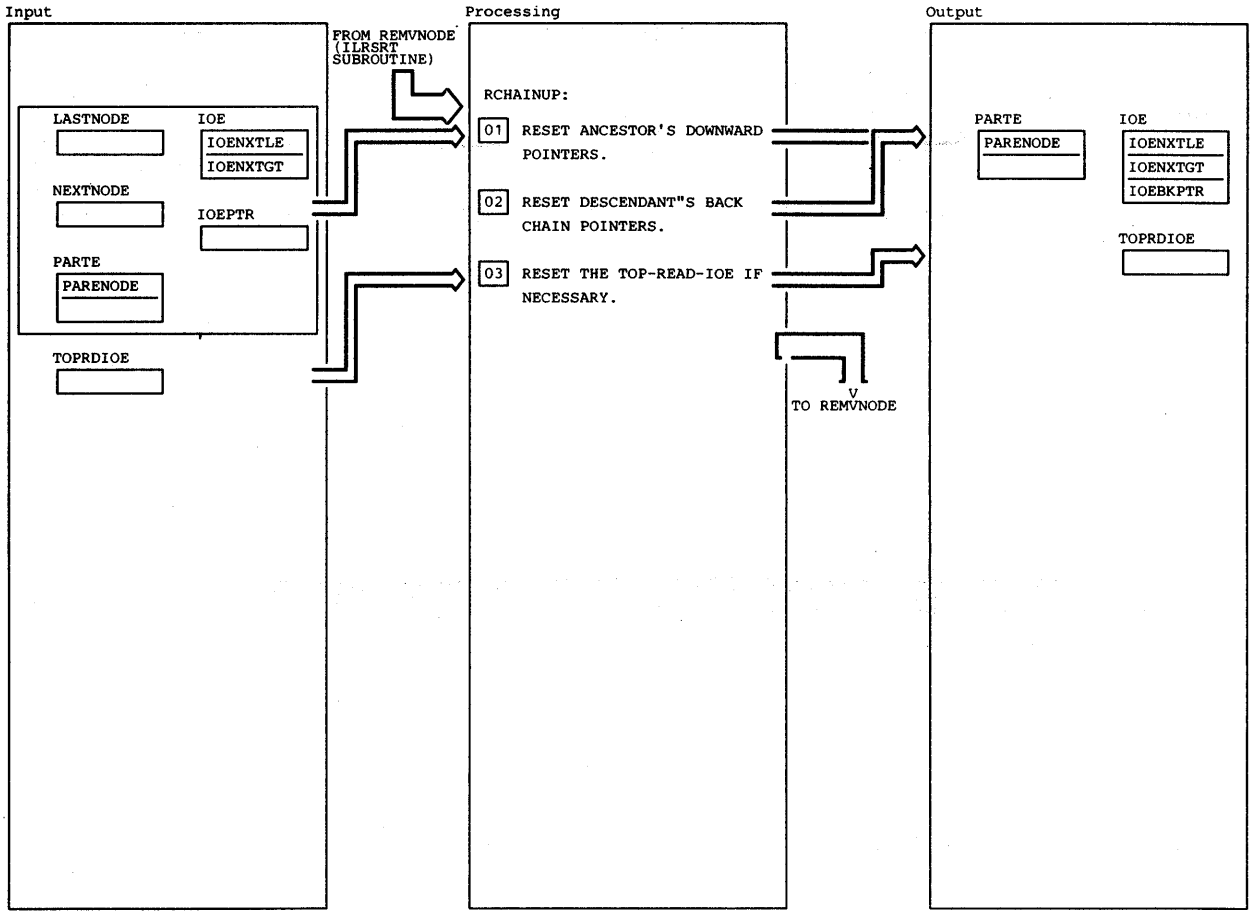
Diagram 25.7.15 SETWRITE (Part 1 of 1)



Notes	Routine	Label	Ref
01 THE TREE OF READS IS POINTED TO BY THE PART ENTRY(PARENODE). REMOVAL OF A NODE(IOE) FROM THE TREE REQUIRES UPDATING OF AN ANCESTOR NODE POINTING TO THE NODE BEING REMOVED.			
02 CALL RCHAINUP TO RECHAIN NODE, WHOSE ANCESTOR IS BEING REMOVED FROM THE TREE, TO ITS ANCESTOR'S ANCESTOR.		RCHAINUP	25.7.1 7

Notes	Routine	Label	Ref

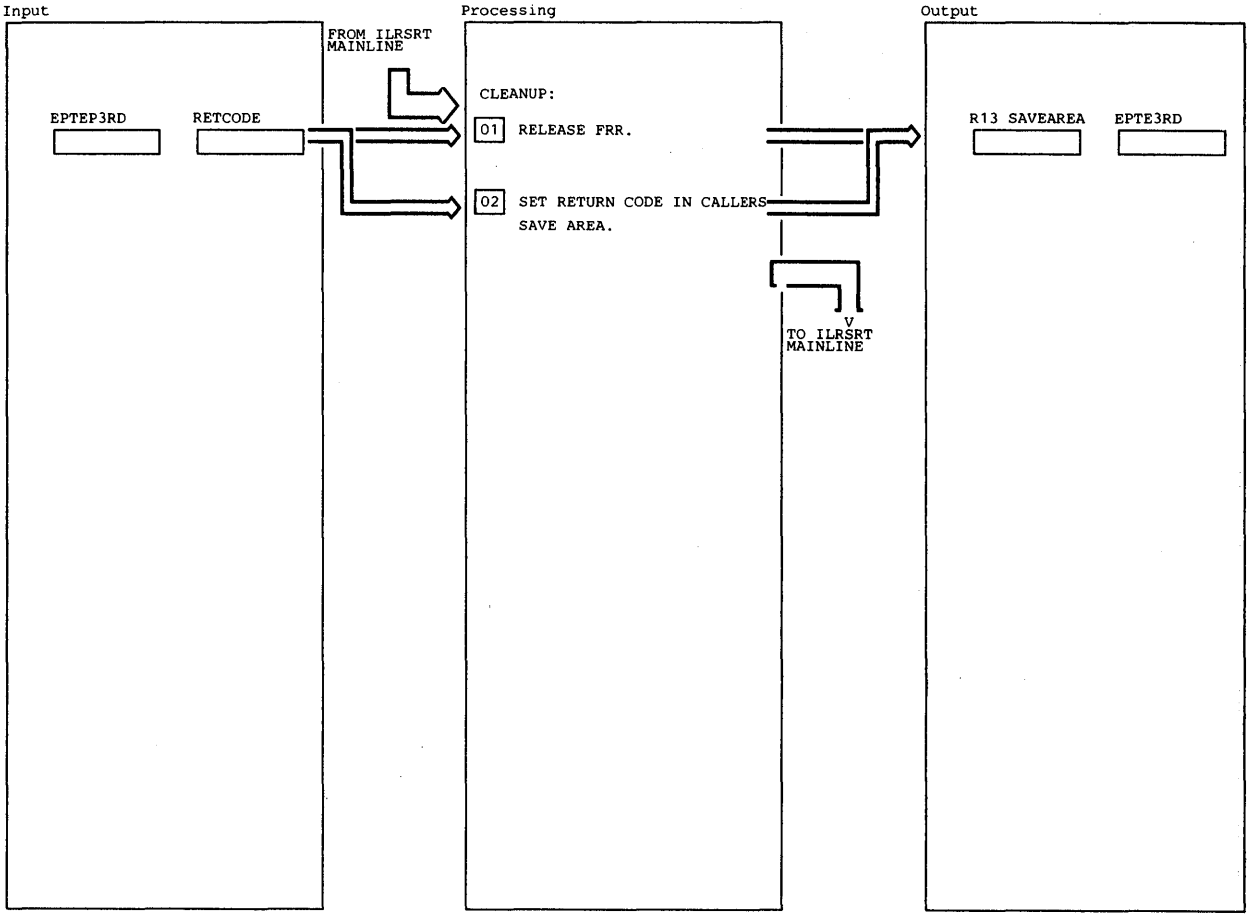
Diagram 25.7.16 REMVNODE (Part 1 of 1)



Notes	Routine	Label	Ref
01 IF THE ANCESTOR NODE POINTS TO THE NODE(IOE) TO BE REMOVED THROUGH THE GREATER-THAN(GT) LEG, UPDATE THE ANCESTOR'S GT POINTER(IOENXTGT). OTHERWISE, UPDATE THE LESS-THAN-OR-EQUAL-TO(LE) POINTER(IOENXTLE). IF THE NODE BEING REMOVED IS THE TOP NODE OF THE TREE, THEN PARENODE MUST BE RESET.			
02 IF A NEXT NODE EXISTS, RESET THE BACKWARD POINTER OF THE NEXT NODE.			
03 IF THE IOE JUST REMOVED FROM THE TREE IS THE FIRST IOE OF READS FROM A SPECIFIC CYLINDER, THEN RESET THE TOP-READ-IOE PTR FOR THE GROUP.			

Notes	Routine	Label	Ref

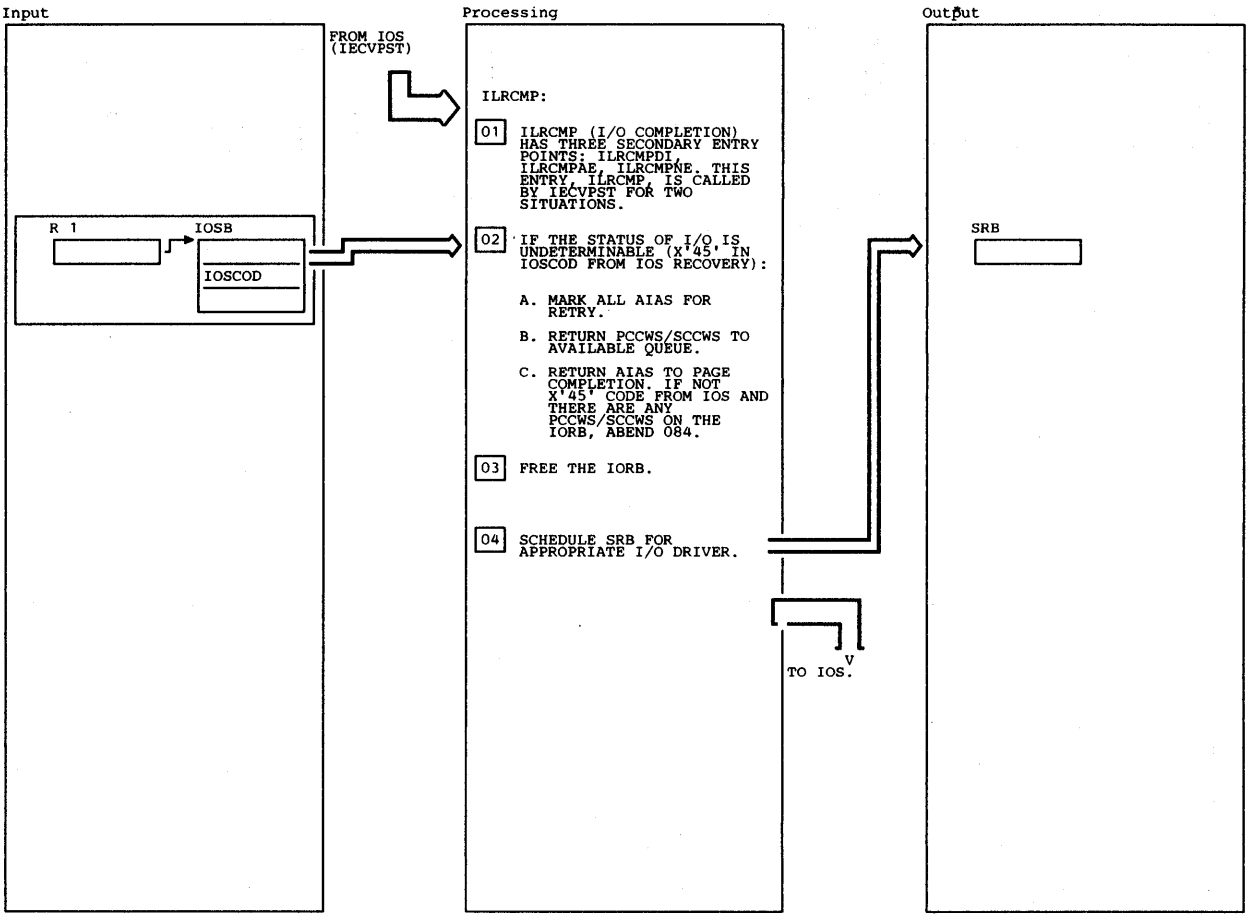
Diagram 25.7.17 RCHAINUP (Part 1 of 1)



Notes	Routine	Label	Ref
01 ZERO THE ILRSRT ENTRY IN THE EPATH (RECOVERY CONTROL BLOCK).			
02 RETURN CODE HAS BEEN PREVIOUSLY SET BY SUBROUTINES OF ILRSRT, AND IS FURTHER CHANGED BY THIS ROUTINE TO INDICATE TO PART MONITOR THAT MORE WORK EXISTS, IF NECESSARY.			

Notes	Routine	Label	Ref

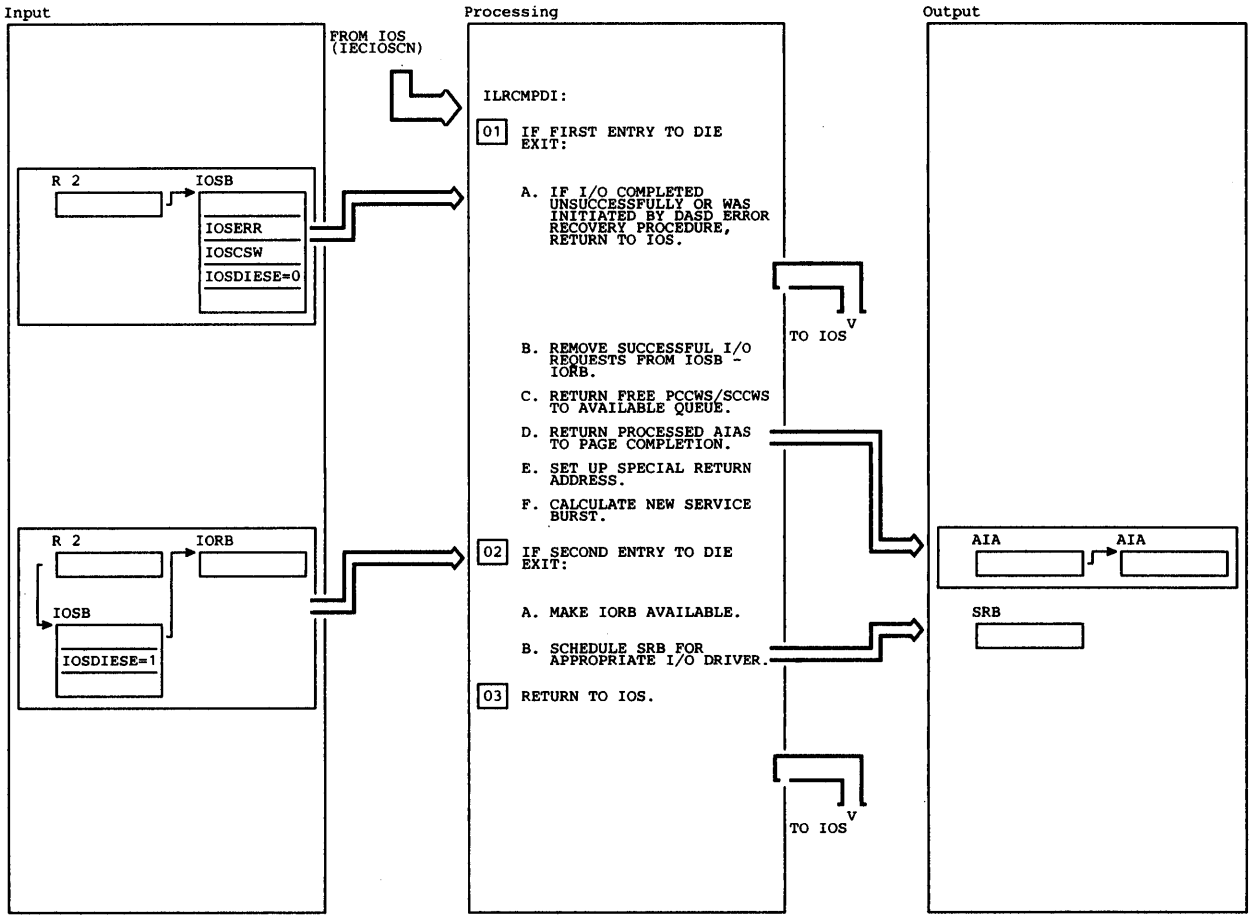
Diagram 25.7.18. CLEANUP (Part 1 of 1)



Notes	Routine	Label	Ref
01 FOR RECOVERY PURPOSES, ILRCMP01 RECOVERY ROUTINE HANDLES ERRORS OCCURRING IN ILRCMP (ALL FOUR ENTRY POINTS). ILRCMP ENTRY (TERMINATION ROUTINE FOR ASM) IS CALLED BY IECVPST (POST STATUS) ON TWO PATHS. ONE IS IF IOS RECOVERY IS ENTERED WHILE IOS WAS PROCESSING THIS IOSB. CODE X'45' IS PUT IN THE IOSCOD, IECVPST SCHEDULED, AND IECVPST CALLS ILRCMP. THE OTHER PATH IS WHEN IECVPST CALLS ILRCMP AFTER ALL THE PCCWS/SCCWS ARE FREED BY ILRCMPAE AND ILRCMPNE.			
02 IF X'45' CODE, ALL I/O SHOULD BE RETRIED SINCE STATUS OF I/O IS UNDETERMINABLE. A. ALL AIAS ARE MARKED FOR RETRY SO THAT PAGE COMPLETION CAN REDRIVE REQUESTS. B. PCCWS/SCCWS ARE RETURNED TO THE APPROPRIATE QUEUE. C. ALL AIAS ARE RETURNED TO PAGE COMPLETION. HAVING ADDITIONAL PCCWS/SCCWS ON THE IORB WITHOUT THE CODE X'45' CONDITION INDICATES ENTRY IN THE NORMAL END APPENDAGE WITHOUT IOSEX ON. THIS WILL ONLY OCCUR WITH CERTAIN HARDWARE MALFUNCTIONS. THE 084 ABEND IS ISSUED TO CAUSE RE-ENTRY HERE WITH CODE X'45'		ABNTERM	25.8.9
03 THE IORB IS MADE AVAILABLE.			
04 THE APPROPRIATE I/O DRIVER, ILRPTM OR ILRSWPDR, IS SCHEDULED WITH NO CHECKS FOR WORK.			

Notes	Routine	Label	Ref

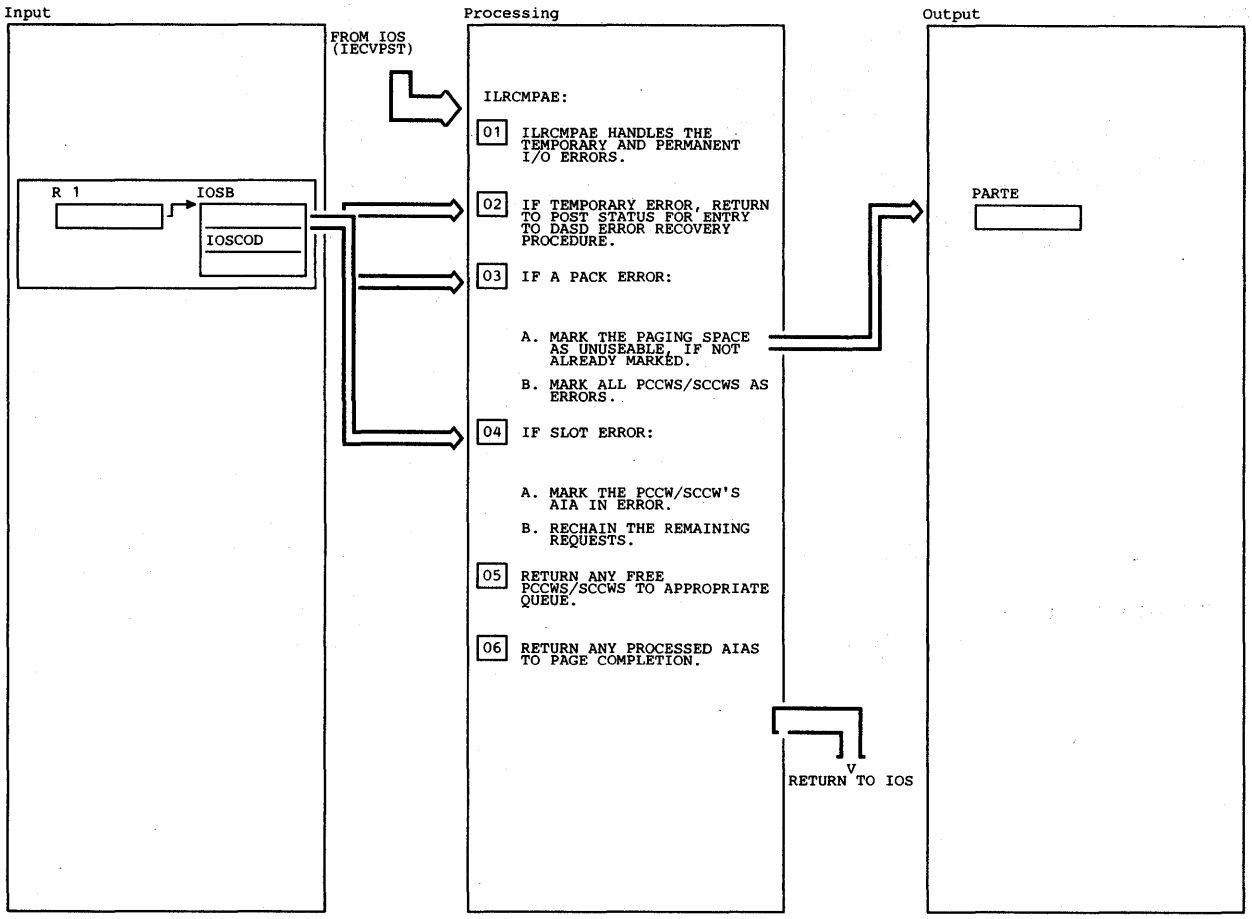
Diagram 25.8 ILRCMP (Part 1 of 1)



Notes	Routine	Label	Ref
01 ILRCMPDI, AN ENTRY IN ILRCMP, RECEIVES CONTROL WHEN I/O COMPLETES WHETHER IT IS STARTED BY ILRSWPDR OR ILRSRT OR RESTARTED BY DASD ERP FOR RETRIES. IF THE I/O IS UNSUCCESSFUL OR RESTARTED BY DASD ERP, ILRCMPDI RETURNS TO IOS IMMEDIATELY. IF FIRST ENTRY TO ILRCMPDI (DIE EXIT), IOSDIESE FLAG IS OFF.			
A. THE STATUS BITS IN THE CSW STORED IN THE IOSB ARE TESTED. IF ANY BITS ARE ON OTHER THAN CHANNEL END, DEVICE END, OR CONTROL UNIT END, THE I/O WAS NOT SUCCESSFUL. IN THIS CASE OR IF IOSERR (RESTARTED BY DASD ERP) IS ON, THE DIE RETURNS TO MAINLINE IOS TO HAVE THE POST STATUS DRIVER APPENDAGES PROCESS THIS I/O COMPLETION.			
B. ALL PCCW/SCCWS WILL BE REMOVED.		PROCCCWS	25.8.4
C. THE FREE PCCWS/SCCWS WILL BE RETURNED TO THE APPROPRIATE QUEUE.		POSTCMP	25.8.6
D. ALL AIAS REMOVED WILL BE RETURNED TO ILRPAGCM.		POSTCMP	25.8.6
E. THE RETURN ADDRESS IS SET TO R14+8 TO CAUSE SECOND ENTRY TO ILRCMPDI.			
F. THE TOD WILL HAVE BEEN STORED UPON ENTRY TO ILRCMPDI. IF ILRSRT'S TOD IS NON-ZERO, A NEW BURST TIME WILL BE CALCULATED.			
02 IF SECOND ENTRY TO DIE EXIT, IOSDIESE FLAG WILL BE ON, ENTRY IS FOR CLEAN UP OF RESOURCES.			
A. THE IORB IS MADE AVAILABLE.			

Notes	Routine	Label	Ref
B. IF WORK REMAINS FOR THE CORRESPONDING PARTE OR SARTE, THE APPROPRIATE I/O DRIVER IS SCHEDULED - ILRPTM OR ILRSWPDR.			

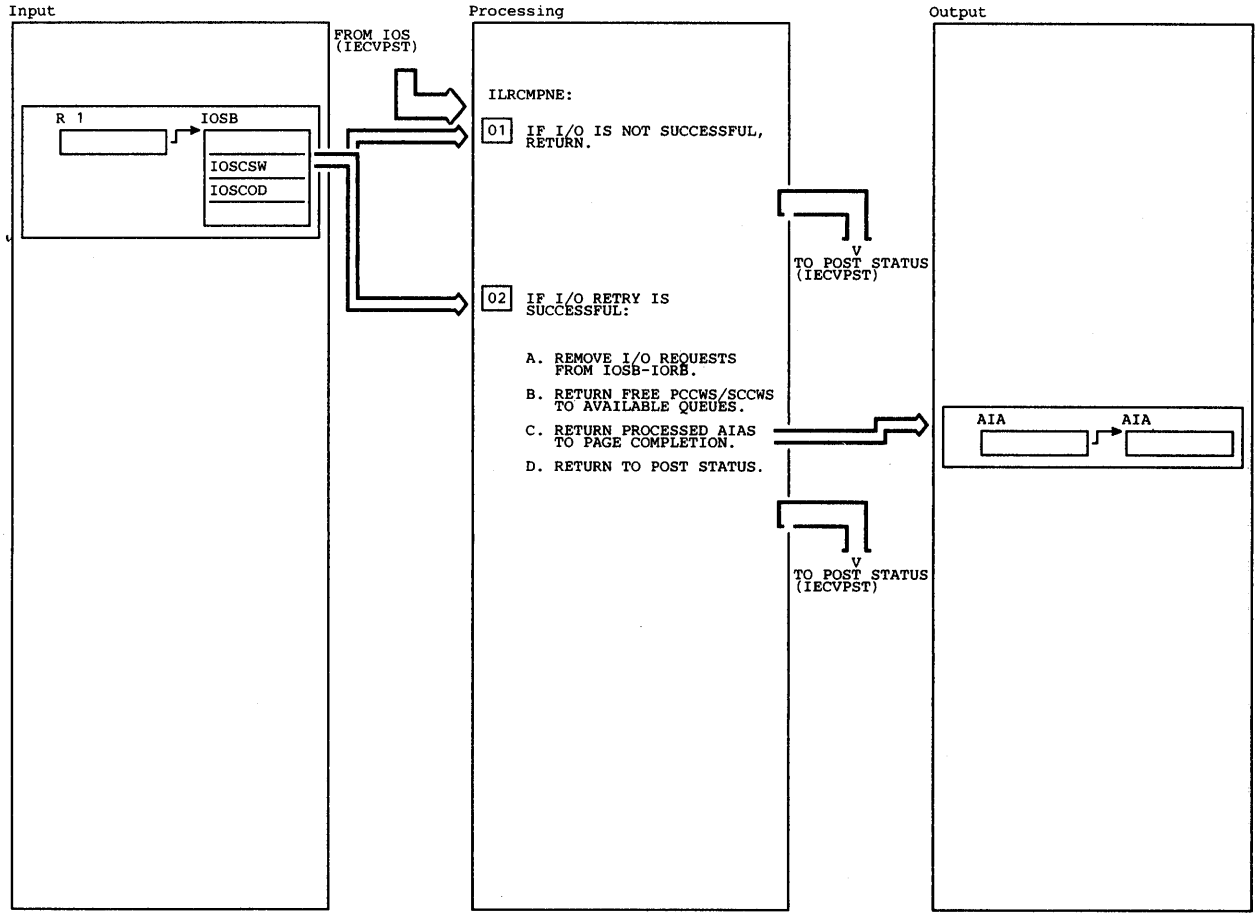
Diagram 25.8.1 ILRCMPDI (Part 1 of 1)



Notes	Routine	Label	Ref
01 ILRCMPAE IS ABNORMAL END APPENDAGE FOR ASM. IECVPST (POST STATUS) GIVEN CONTROL VIA ILRCMPDI (UPON UNSUCCESSFUL I/O COMPLETION) CALLS ILRCMPAE TO HANDLE THE CODES THAT ARE NOT X'45' AND NOT X'7F'. THE CODES ARE IN IOSCOD.			
02 A TEMPORARY ERROR HAS A CODE OF X'7X' AND WILL BE RETURNED TO POST STATUS FOR A CALL TO DASD ERP TO INTERPRET AND RETRY ERROR.			
03 A PACK ERROR IS A CODE OF X'51', X'6D' OR A X'41' CODE WITH SPECIFIC CHANNEL ERRORS OR A TOTAL OF 176 I/O ERRORS ON THE PACK.			
A. THE PARTE/SARTE IS MARKED AS UNUSEABLE AND THE APPROPRIATE ACTION IS TAKEN BY ILRMSG00 VIA BADPACK IF THE SYSTEM MUST BE TERMINATED.		BADPACK	25.8.10
B. ALL PCCWS/SCCWS WILL BE MARKED AS ERRORS.		BADSLOT	25.8.7
04 A SLOT ERROR, ANY OTHER IOSCOD CODE GIVEN TO ILRCMPAE:			
A. THE PCCW/SCCW AIA IN ERROR IS SO MARKED.		BADSLOT	25.8.7
B. THE REMAINING PCCWS/SCCWS WILL BE RECHAINED AND THE RETURN ADDRESS SET SO THAT POST STATUS WILL ISSUE A START IO.		RECHAIN	25.8.5
05 ALL FREED PCCWS/SCCWS WILL BE RETURNED TO THE APPROPRIATE QUEUE.		POSTCMP	25.8.6

Notes	Routine	Label	Ref
06 ANY PROCESSED AIAS WILL BE RETURNED TO PAGE COMPLETION TO COMPLETE PROCESSING FOR THE REQUESTS.		POSTCMP	25.8.6

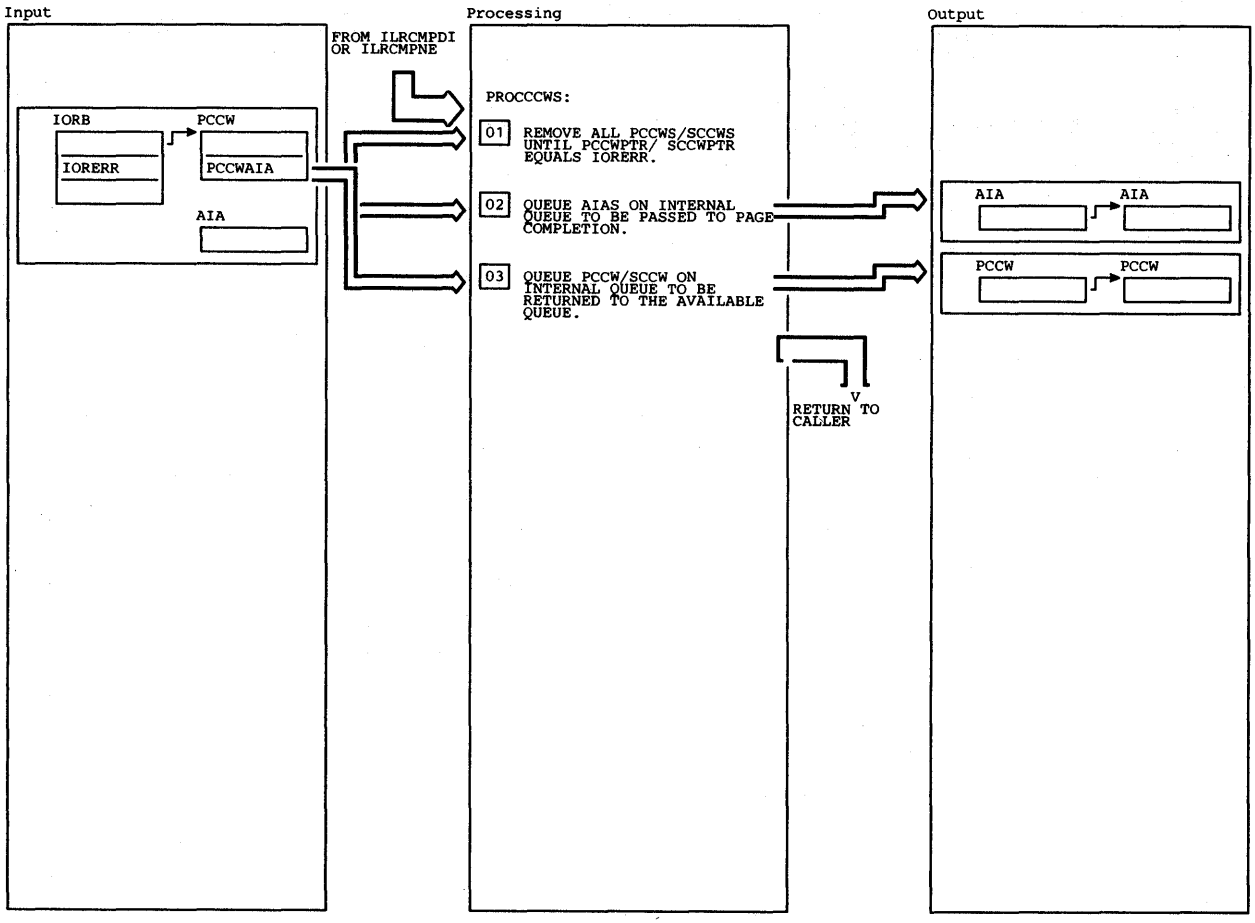
Diagram 25.8.2 ILRCMPAE (Part 1 of 1)



Notes	Routine	Label	Ref
01 THIS IS NORMAL END APPENDAGE FOR ASM. THE IOSCOD WILL CONTAIN X'7F'. SINCE THE FIRST ENTRY TO DIE HAS ALREADY HANDLED ALL THE INITIALLY SUCCESSFUL I/O, ONLY THE UNEXPECTED 'NORMAL' SITUATIONS REMAIN. UNIT EXCEPTION OR WRONG LENGTH RECORD WITH IOSEX BIT ON - EITHER SITUATION IS CONSIDERED AN ERROR AND SENT TO DASD ERP VIA IECVPST.			
02 NORMAL END APPENDAGE ALSO HANDLES SUCCESSFUL I/O FROM DASD ERP RETRIES.			
A. ALL PCCWS/SCCWS ARE REMOVED.		PROCCWS	25.8.4
B. THE FREE PCCWS/SCCWS ARE RETURNED TO THE APPROPRIATE QUEUE.		POSTCMP	25.8.6
C. ALL AIAS REMOVED ARE RETURNED TO ILRPAGCM.		POSTCMP	25.8.6
D. RETURN TO POST STATUS.			

Notes	Routine	Label	Ref

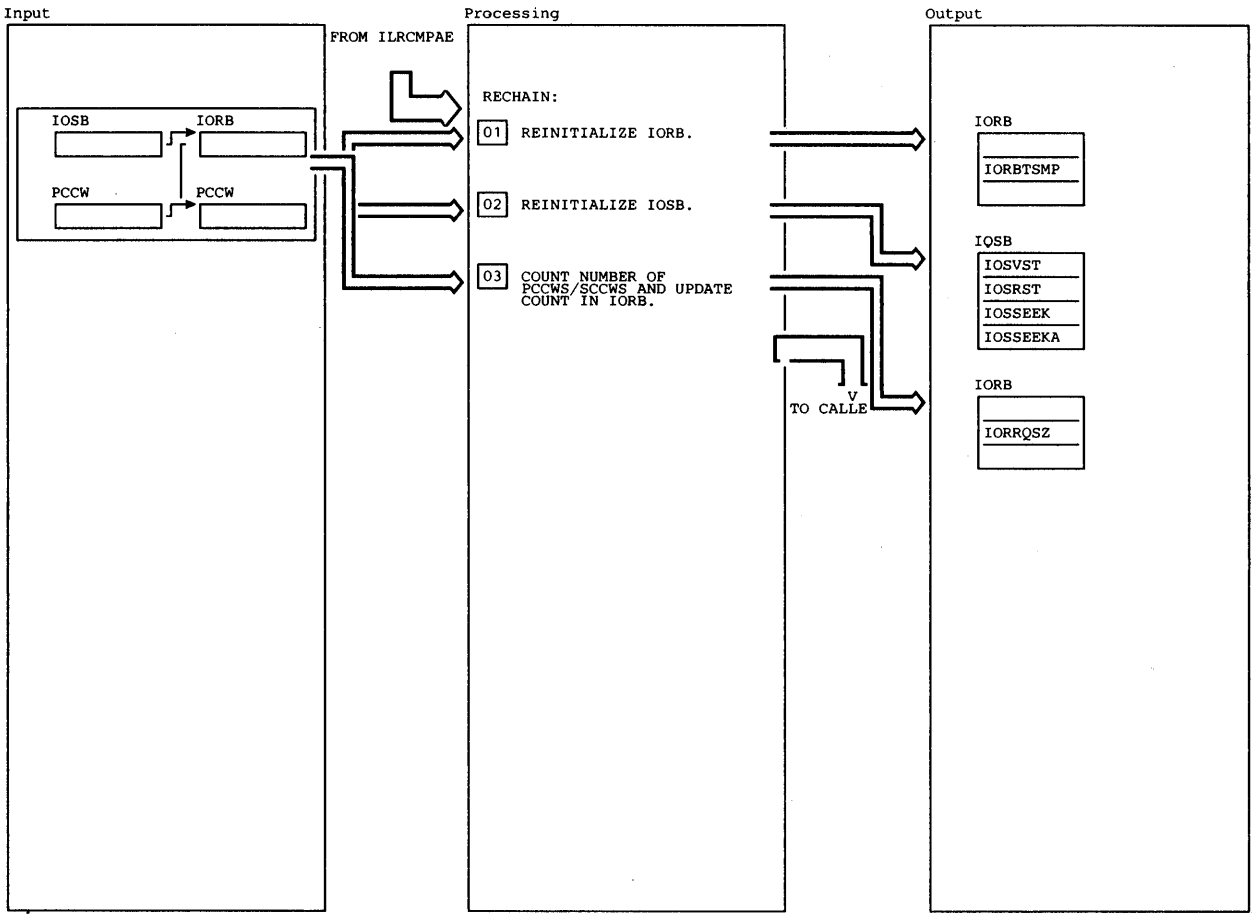
Diagram 25.8.3 ILRCMPNE (Part 1 of 1)



Notes	Routine	Label	Ref
01 IORERR IS THE ADDRESS OF THE PCCW/SCCW IN ERROR. FOLLOW THE CHAIN OF PCCWS/SCCWS. AS LONG AS THE PCCWPTR/SCCWPTR DOES NOT EQUAL IORERR THE PCCW/SCCW WILL BE DECHAINED FROM THE IORB.			
02 ALL OF THE AIAS ARE CHAINED TOGETHER AND PASSED TO PAGE COMPLETION ON ONE CALL.			
03 ALL OF THE PCCWS/SCCWS WILL BE CHAINED TOGETHER ON AN INTERNAL QUEUE AS THEY ARE FREED. THEY WILL THEN BE PUT BACK ON THE AVAILABLE QUEUE WITH ONE COMPARE AND SWAP (CS).			

Notes	Routine	Label	Ref

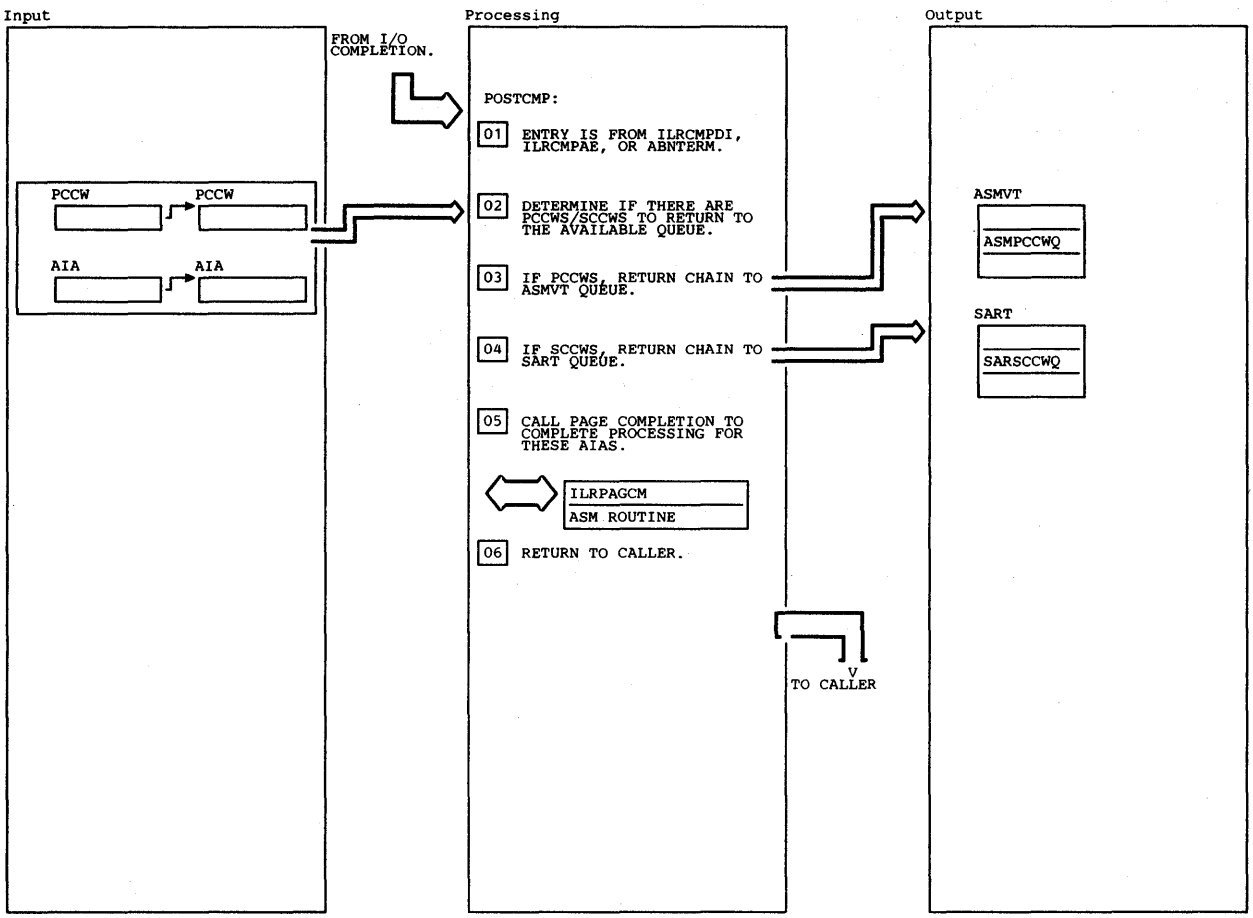
Diagram 25.8.4 PROCCCWS (Part 1 of 1)



Notes	Routine	Label	Ref
01 POINT THE IORB TO THE FIRST PCCW/SCCW. ZERO ILRSRT'S TOD (TIME OF DAY) IN THE IORB SINCE SERVICE BURST RECALCULATION WILL NOT BE DONE BASED ON A PARTIAL SERVICE BURST OF I/O REQUESTS.			
02 RESET THE FOLLOWING IOSB FIELDS RELATIVE TO THE FIRST PCCW/SCCW: IOSVST, IOSRST, IOSSEEK, IOSSEEKA.			
03 COUNT NUMBER OF PCCW/SCCWS AND PUT COUNT IN IORRQSZ.			

Notes	Routine	Label	Ref

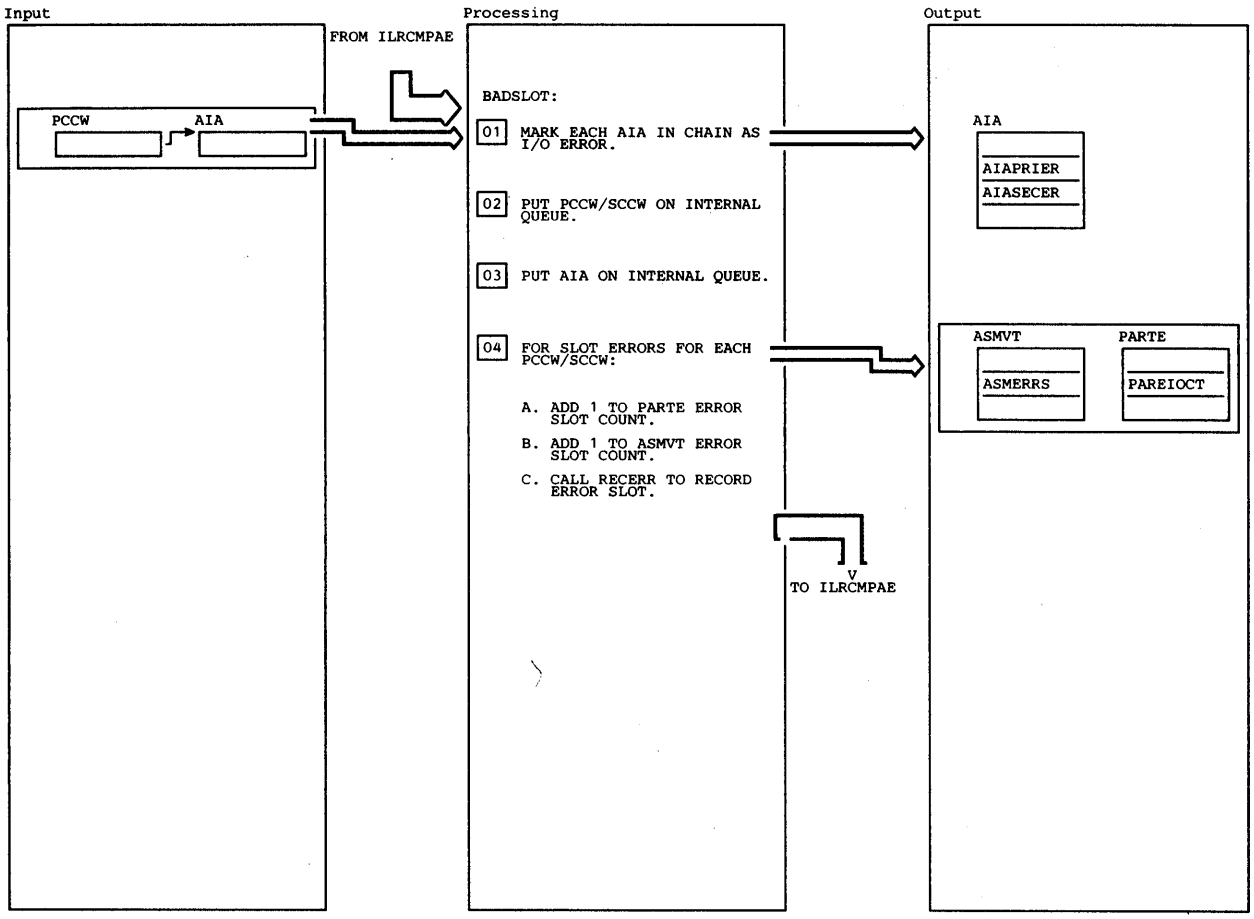
Diagram 25.8.5 RECHAIN (Part 1 of 1)



Notes	Routine	Label	Ref
01 POSTCMP MAKES PCCW/SCCW AVAILABLE AND CALLS ILRPAGCM TO HANDLE AIAS.			
02 CHECK INTERNAL QUEUE TO SEE IF ANY PCCWS/ SCCWS HAVE BEEN FREED.			
03 ALL PCCWS WILL BE CHAINED FROM THE ASMVT.			
04 ALL SCCWS WILL BE CHAINED FROM THE SART. ONLY ONE KIND OF CCW BLOCK IS PROCESSED AT ANY ONE INVOCATION.			
05 CALL ILRPAGCM TO RETURN PROCESSED AIAS.	ILRPAGCM	ILRPAGCM	

Notes	Routine	Label	Ref

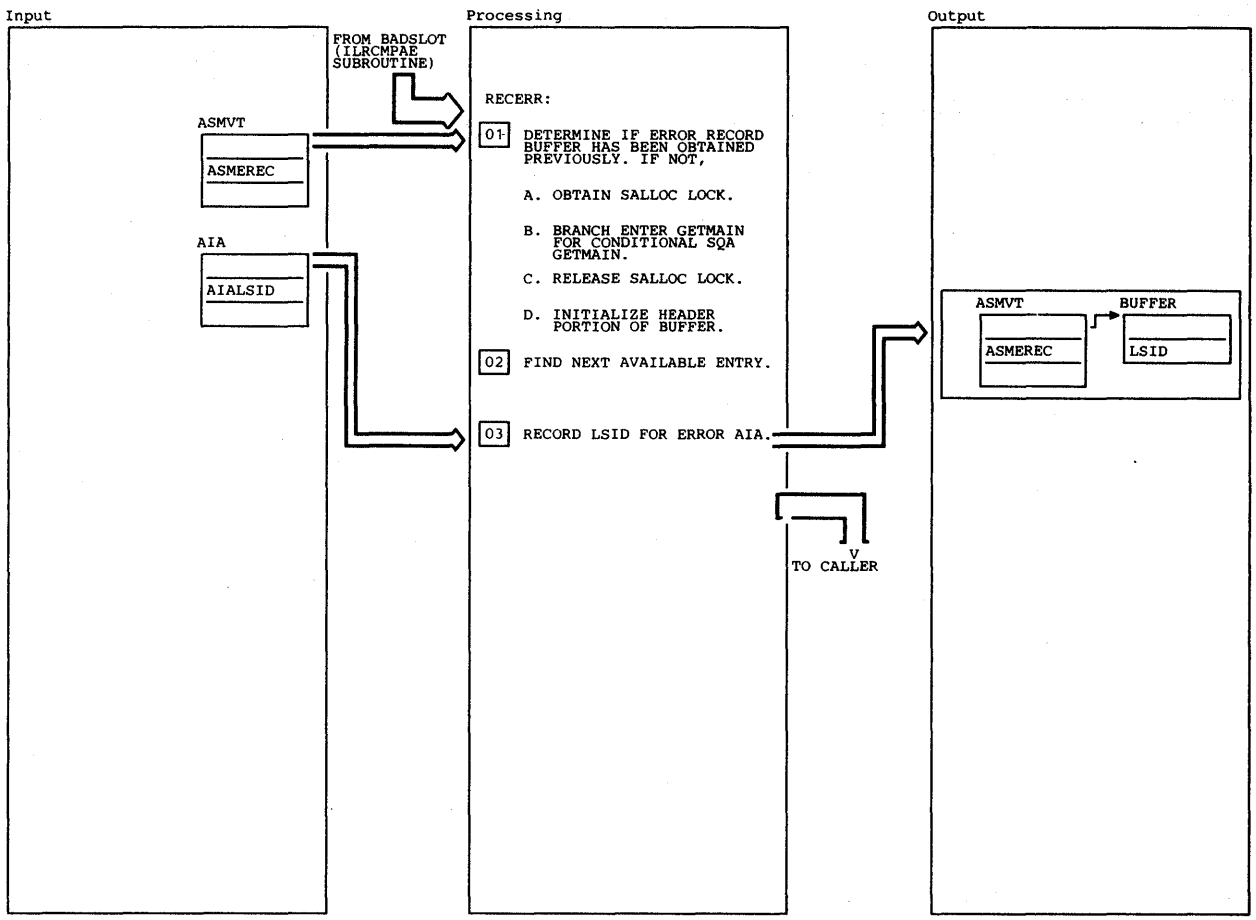
Diagram 25.8.6 POSTCMP (Part 1 of 1)



Notes	Routine	Label	Ref
01 IF THE AIA IS FOR THE DUPLEX DATA SET, MARK AIA AS A SECONDARY ERROR. OTHERWISE MARK IT AS A PRIMARY ERROR.			
02 FREE PCCW/SCCW WILL BE PUT ON INTERNAL QUEUE.			
03 AIA'S WILL BE PUT ON INTERNAL QUEUE TO BE RETURNED TO PAGE COMPLETION.			
04 IF THE ERROR IS AN I/O ERROR, THE ERROR SLOT COUNTS MUST BE UPDATED.			
A. THE PARTE ERROR COUNT KEEPS TRACK OF THE NUMBER OF I/O ERRORS RECEIVED FOR THIS DATA SET. WHEN 176 ERRORS ARE RECEIVED, THE PACK IS NO LONGER CONSIDERED USEABLE.			
B. THE ASMVT ERROR COUNT IS A TOTAL COUNT FOR ALL LOCAL PAGE DATA SETS.			
C. THE LSID WILL BE RECORDED IN A SQA BUFFER.		RECERR	25.8.8

Notes	Routine	Label	Ref

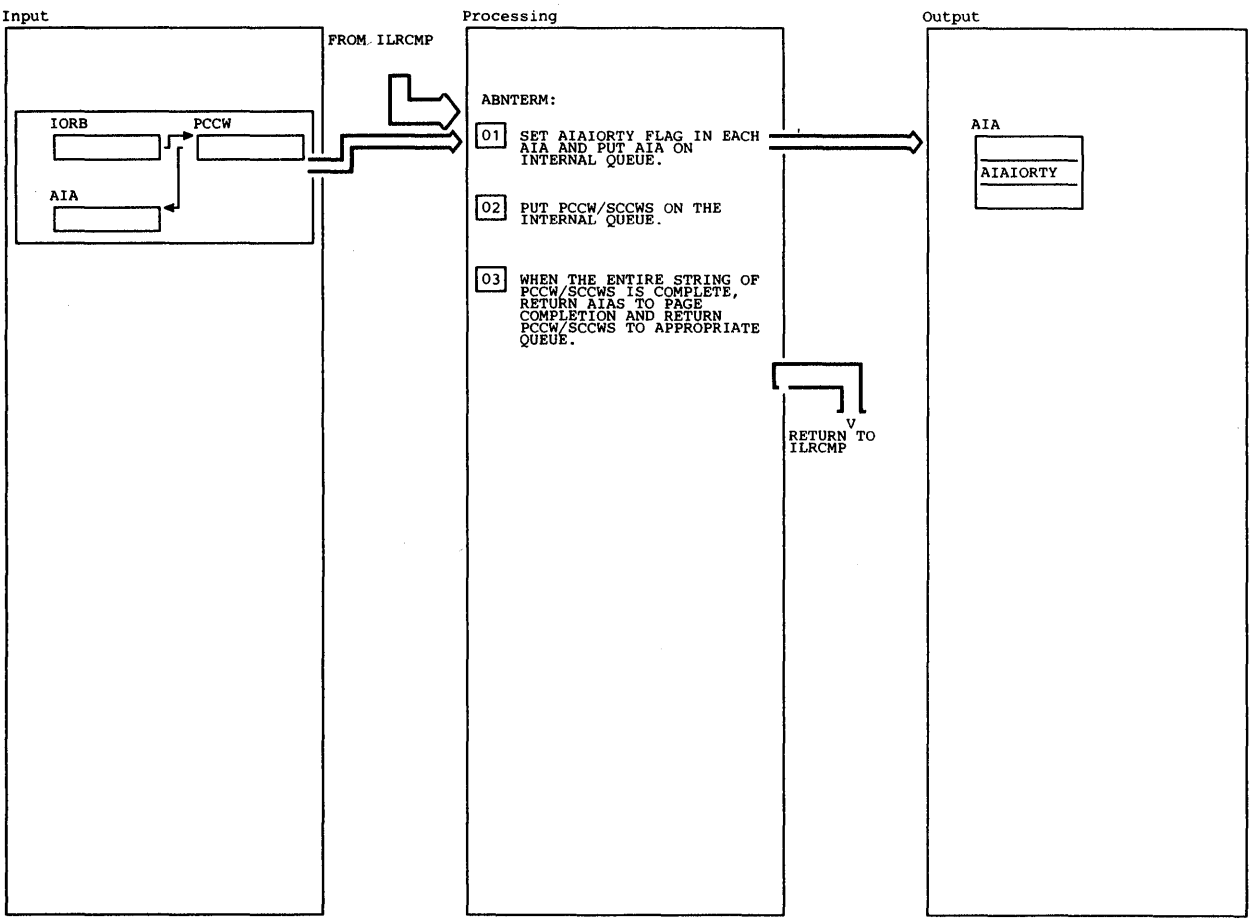
Diagram 25.8.7 BADSLOT (Part 1 of 1)



Notes	Routine	Label	Ref
01 A BUFFER FROM SQA IS USED TO RECORD THE LSID FOR ERROR SLOTS. THIS BUFFER IS NOT OBTAINED UNTIL THE FIRST ERROR IS ENCOUNTERED. THE POINTER IN THE ASMVT IS INITIALIZED TO ZERO AND SET TO THE ADDRESS OF THE BUFFER ONCE IT IS OBTAINED.			
02 A FIELD IN THE HEADER POINTS TO THE CURRENT ENTRY FOR RECORDING.			
03 RECORD THE THREE BYTE LSID IN THE NEXT AVAILABLE BUFFER ENTRY. IF RECORDING AN ERROR ON A SWAP DATA SET, TURN ON THE HIGH ORDER BIT OF THE HIGH ORDER BYTE.			

Notes	Routine	Label	Ref

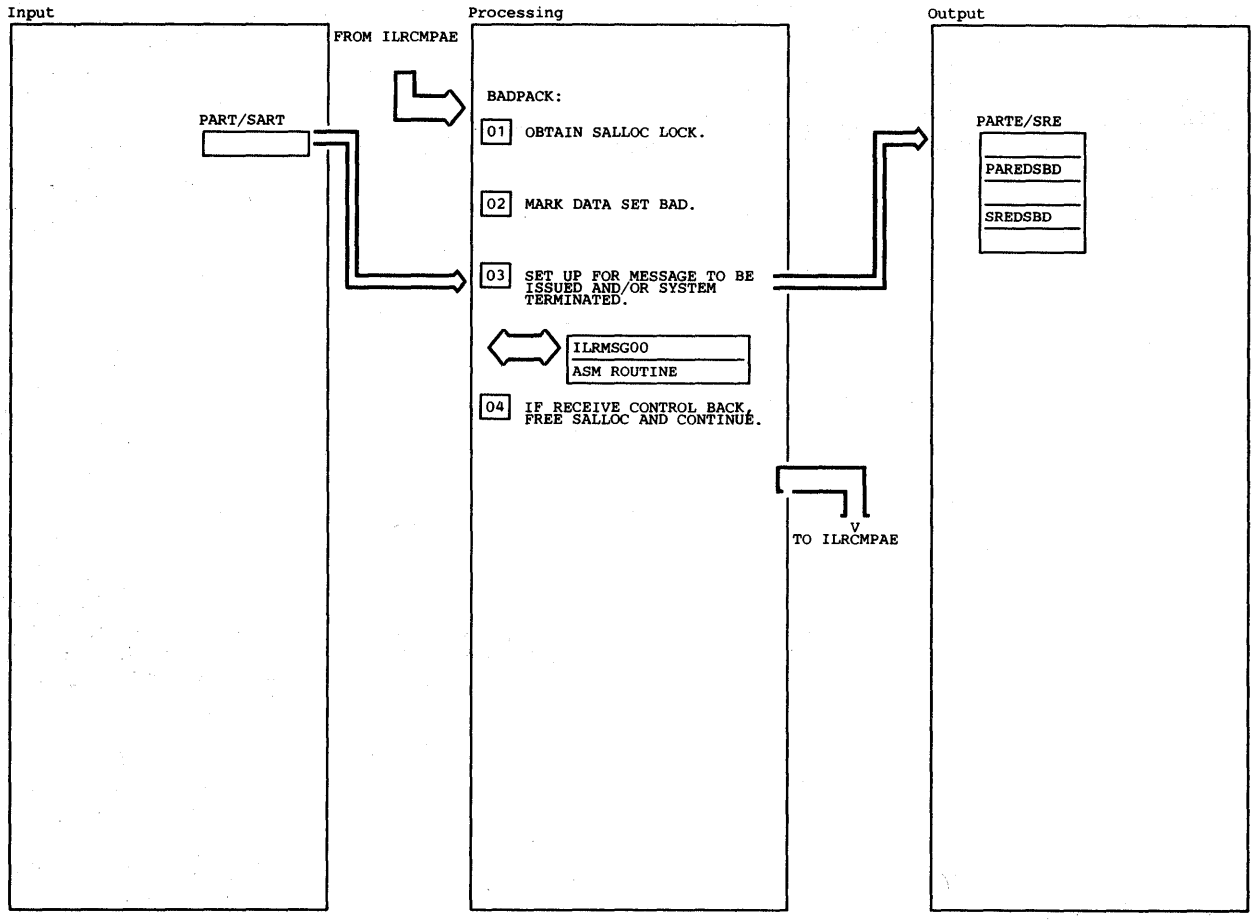
Diagram 25.8.8 RECERR (Part 1 of 1)



Notes	Routine	Label	Ref
01 THE AIAIORTY FLAG INDICATES TO PAGE COMPLETION TO REDRIVE THE REQUEST.			
02 AN INTERNAL QUEUE OF PCCW/SCCWS IS USED TO FREE THE PCCW/SCCWS.			
03 CALL POSTCMP TO RETURN PCCW/SCCWS TO APPROPRIATE QUEUE AND TO CALL ILRPAGCM TO RETURN STRING OF AIAS.		POSTCMP	25.8.6

Notes	Routine	Label	Ref

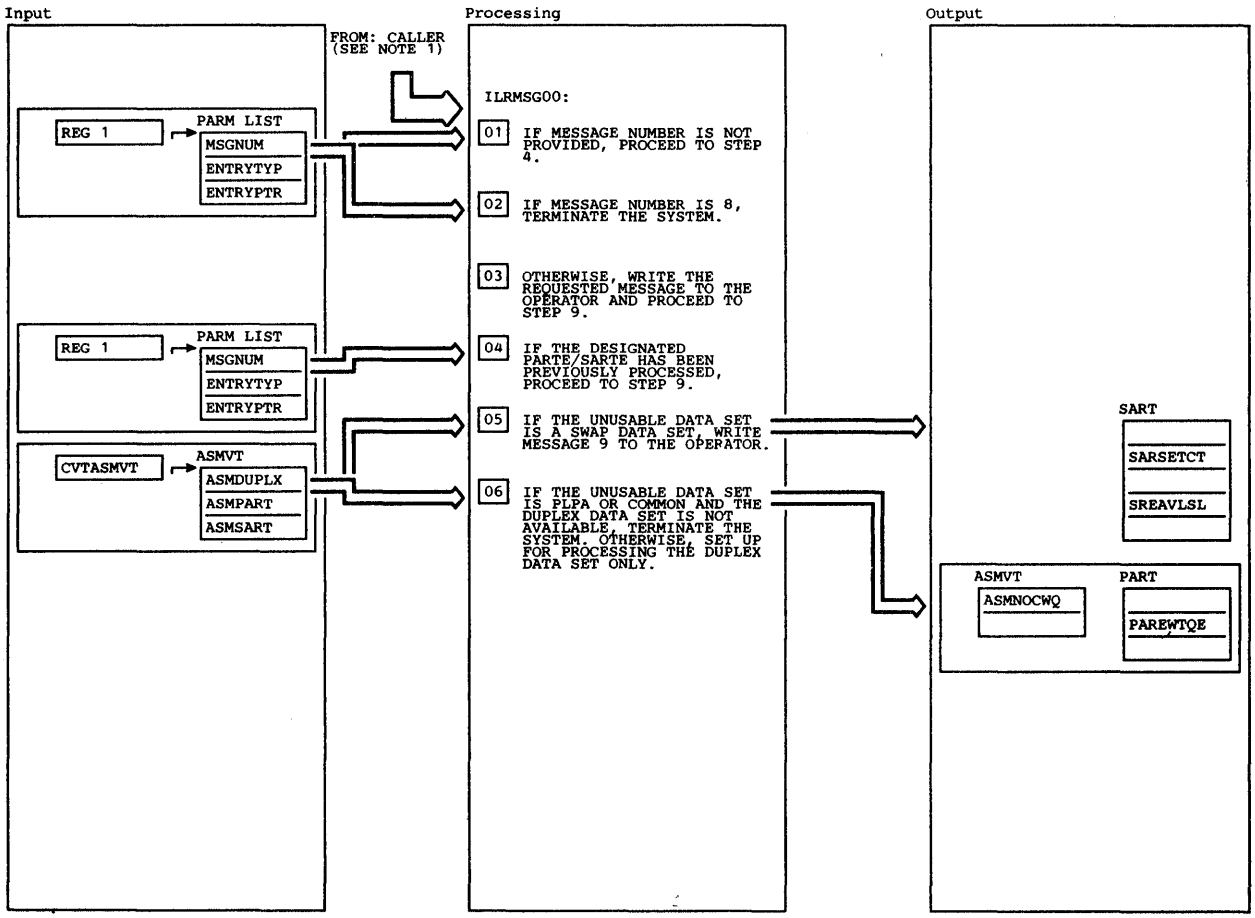
Diagram 25.8.9 ABNTERM (Part 1 of 1)



Notes	Routine	Label	Ref
01 SALLOC LOCK IS OBTAINED TO SERIALIZE COUNTS THAT HAVE TO BE MANIPULATED AND TERMINATION IF IT BECOMES NECESSARY.			
02 INDICATE BAD PAGE/SWAP SPACE IN PARTE/ SARTE.			
03 SET UP MESSAGE PARAMETER LIST WITH PAGE/SWAP INDICATOR AND PARTE/SARTE ADDRESS. ZERO MESSAGE NUMBER FIELD INDICATING ILRMSG00 WILL DETERMINE APPROPRIATE SYSTEM ACTION. CALL ILRMSG00 TO ISSUE MESSAGE AND/OR TERMINATE SYSTEM.	ILRMSG00	ILRMSG00	
04 IF SYSTEM NOT TERMINATED, FREE SALLOC AND CONTINUE TO ILRCMPAE.			

Notes	Routine	Label	Ref

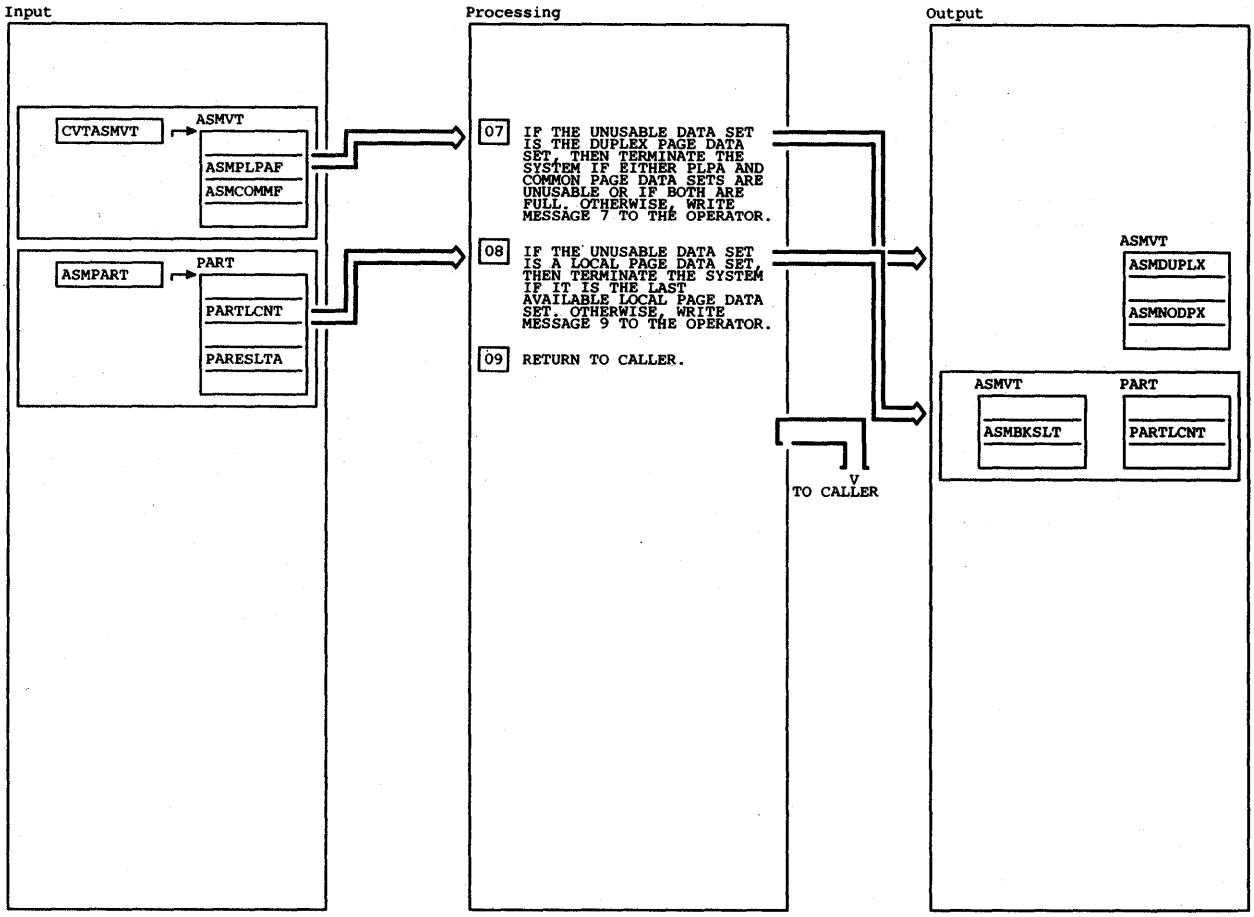
Diagram 25.8.10 BADPACK (Part 1 of 1)



Notes	Routine	Label	Ref
01 THE CALLER OF ASM MESSAGE ROUTINE (ILRMSG00) IS ONE OF THE FOLLOWING ROUTINES: ILRCMP, ILRCMP01, ILRSRT01, ILRSWP01, ILRPTM. THE SALLOC IS HELD UPON ENTRY. IF THE MESSAGE NUMBER HAS NOT BEEN SUPPLIED BY THE CALLER, PROCEED TO STEP 4 IN ORDER TO DETERMINE WHICH MESSAGE SHOULD BE WRITTEN. THE MESSAGE NUMBER IS NOT SUPPLIED BY ILRCMP, ILRSRT01, ILRCMP01, AND ILRSWP01 WHEN A PAGE OR SWAP DATA SET HAS BECOME UNUSABLE.			
02 IF THE MESSAGE NUMBER IS 8, CALL THE SUBROUTINE THAT TERMINATES THE SYSTEM.		TERMSYS	25.9.2
03 CALL THE SUBROUTINE THAT BUILDS THE REQUESTED MESSAGE AND WRITES IT TO THE OPERATOR. THEN PROCEED TO STEP 9.		WRITEMSG	25.9.3
04 IF THE DATA SET IS ALREADY MARKED AS NOT USABLE (PARENUSE/SRENUSE IS ON AND PAREDSBD/SREDSBD IS ON), A MESSAGE HAS ALREADY BEEN WRITTEN ABOUT THIS DATA SET. ALSO, IF NEITHER FLAG IS ON, NO FURTHER PROCESSING IS DONE. HENCE, PROCEED TO STEP 9.			
05 IF THE DATA SET THAT HAS BECOME UNUSABLE IS A SWAP SET, SET THE MESSAGE NUMBER TO 9. ALSO DECREMENT THE TOTAL NUMBER OF AVAILABLE SWAP SETS (SARSETCT) BY THE AVAILABLE SWAP SET COUNT FOR THIS ENTRY (SREAVLSL). ZERO THE AVAILABLE SWAP SET COUNT FOR THE UNUSABLE SWAP DATA SET (SREAVLSL). CALL THE SUBROUTINE TO BUILD MESSAGE 9 AND WRITE IT TO THE OPERATOR.		WRITEMSG	25.9.3
06 THE UNUSABLE DATA SET IS PLPA OR COMMON.			

Notes	Routine	Label	Ref
A. IF THE DUPLEX DATA SET IS NOT AVAILABLE (ASMDUPLX = '0' B), CALL THE SUBROUTINE TO TERMINATE THE SYSTEM.		TERMRTN	25.9.2
B. IF THE DUPLEX DATA SET IS AVAILABLE, CLEARWTQ IS CALLED TO EMPTY THE APPROPRIATE PART WRITE QUEUE, FREE THE IOES ON THE QUEUE, AND SCHEDULE SRB FOR ILRPTM TO BEGIN PROCESSING AGAINST THE DUPLEX DATA SET. THE ASMVT FLAG ASMNOCWQ (NO COMMON WRITE QUEUE) IS TURNED ON. WRITEMSG SUBROUTINE BUILDS AND WRITES MESSAGES 9 AND 10.		CLEARWTQ WRITEMSG	25.9.4 25.9.3

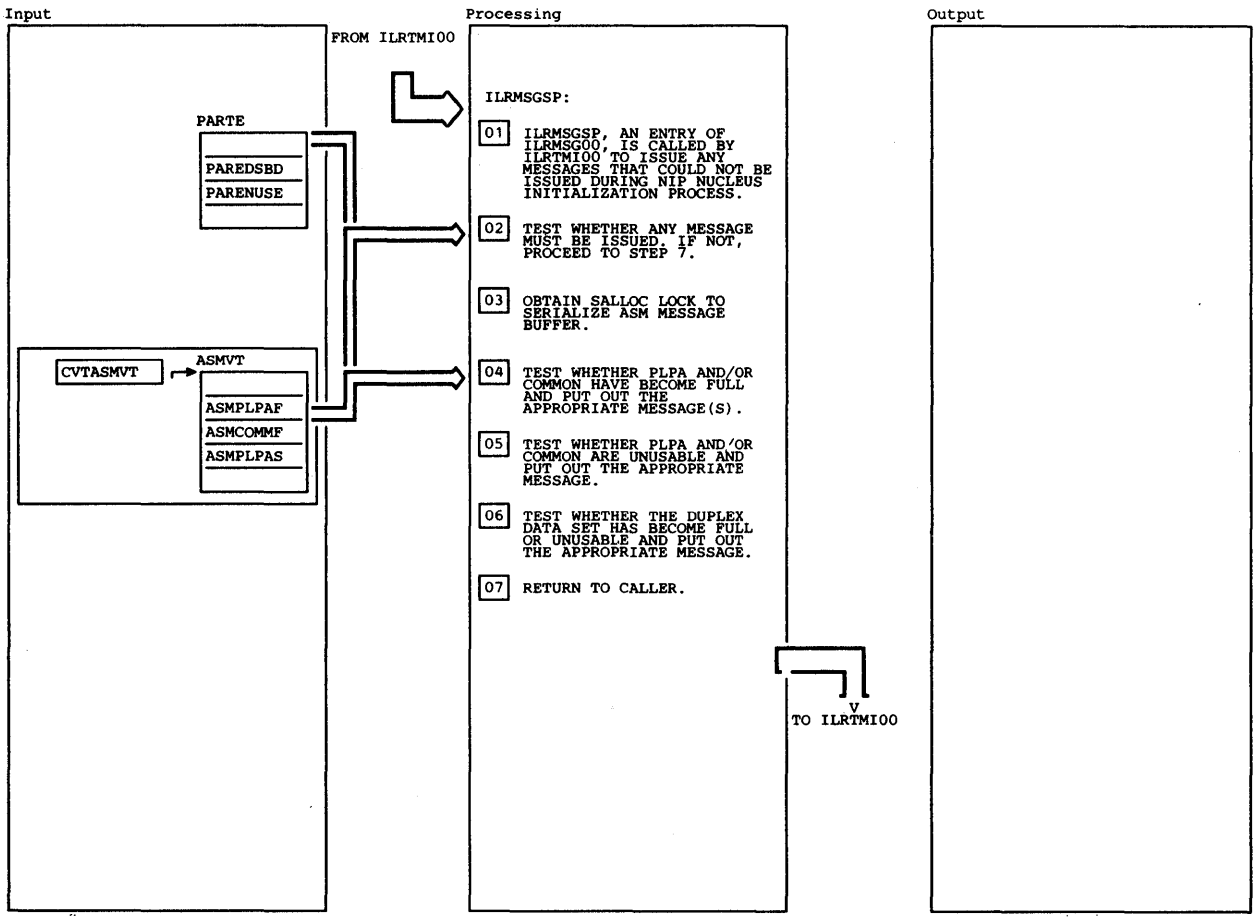
Diagram 25.9 ILRMSG00 (Part 1 of 2)



Notes	Routine	Label	Ref
07 IF THE DATA SET THAT HAS BECOME UNUSABLE IS THE DUPLEX DATA SET, THEN CALL THE SUBROUTINE TO TERMINATE THE SYSTEM IF PLPA/COMMON PAGE DATA SET IS UNUSABLE (I.E. EITHER THE PAREDSBD FLAG OR THE PARENUSE FLAG IS ON) OR IF BOTH PLPA AND COMMON PAGE DATA SETS ARE FULL (I.E. BOTH ASMPLPAF AND ASMCOMMF ARE ON). OTHERWISE, SET THE MESSAGE NUMBER TO 7, TURN OFF THE DUPLEX OPTION FLAG (ASMDUPLX) AND TURN ON THE DUPLEX SUSPENDED FLAG (ASMNODPX). CALL THE SUBROUTINE TO BUILD MESSAGE 7 AND WRITE IT TO THE OPERATOR.		TERMSYS WRITEMSG	25.9.2 25.9.3
08 IF THE DATA SET THAT HAS BECOME UNUSABLE IS A LOCAL PAGE DATA SET AND IT IS THE LAST AVAILABLE LOCAL DATA SET (PARTLCNT=1) THEN THE SUBROUTINE TO TERMINATE THE SYSTEM IS CALLED. IF IT IS NOT THE LAST AVAILABLE LOCAL DATA SET, THEN SET MESSAGE NUMBER TO 9. DECREMENT THE UNRESERVED AVAILABLE LOCAL SLOT COUNT (ASMBKSLT) BY THE NUMBER OF SLOTS MADE AVAILABLE BY THIS DATA SET (PARESLTA). DECREMENT THE LOCAL PAGE DATA SET COUNT (PARTLCNT) BY ONE. CALL THE SUBROUTINE TO BUILD MESSAGE 9 AND WRITE IT TO THE OPERATOR.		TERMSYS WRITEMSG	25.9.2 25.9.3
09 RETURN TO CALLER OF ILRMSG00.			

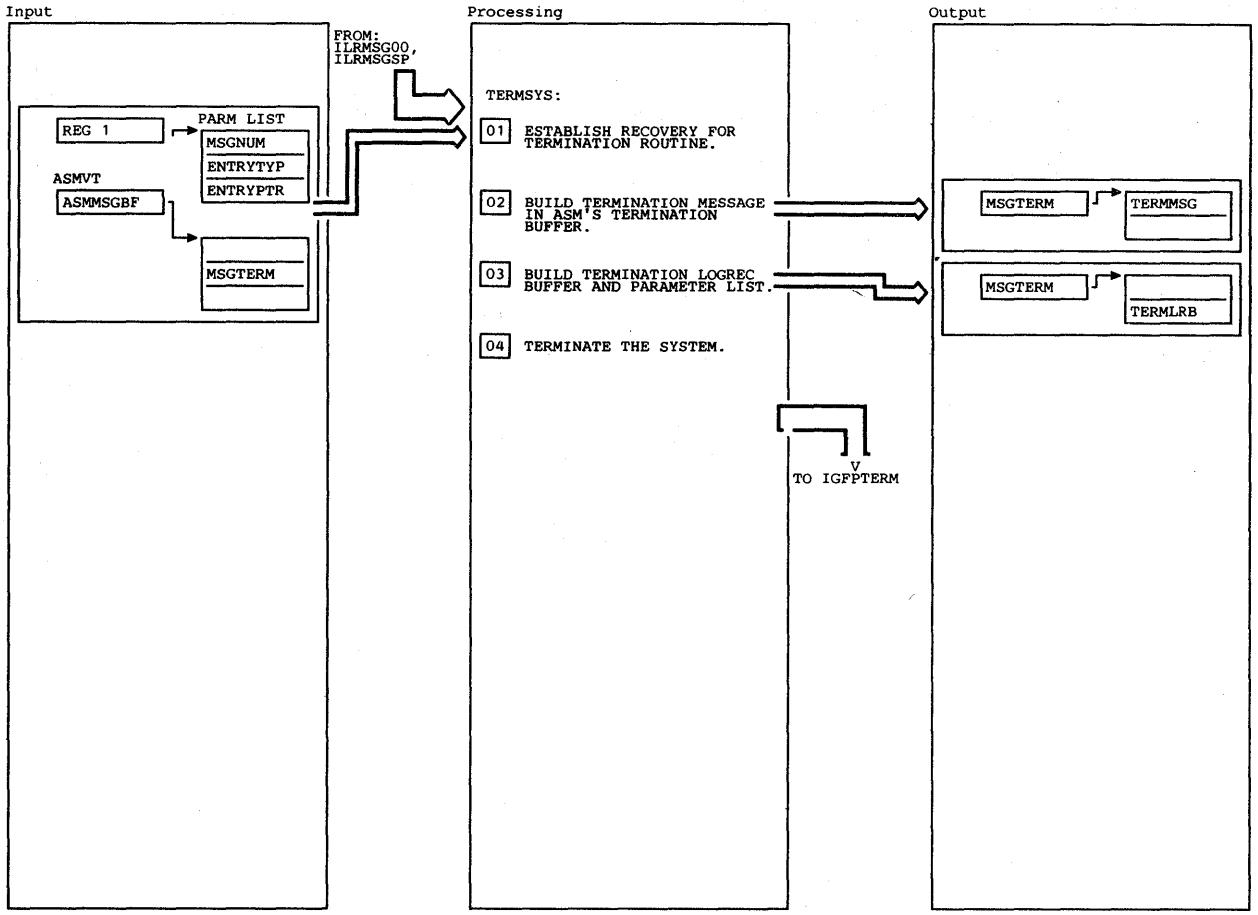
Notes	Routine	Label	Ref

Diagram 25.9 ILRMSG00 (Part 2 of 2)



Notes	Routine	Label	Ref
01 MESSAGES ISSUED BY ILRMSG00 DURING NIP ARE NEVER RECEIVED BY THE OPERATOR. ILRTMI00, WHICH IS GIVEN CONTROL AFTER NIP, CALLS ILRMSGSP TO DETERMINE AND RE-ISSUE THESE MESSAGES.			
02 TO TEST WHETHER ANY MESSAGES MUST BE ISSUED, CHECK THE FOLLOWING BITS: ASMPLPAF, ASMCOMMF, ASMNODPX, THE PAREDSBD AND PARENUSE FLAGS OF PLPA PARTE AND COMMON PARTE. IF NONE OF THESE BITS ARE ON, NO MESSAGE MUST BE ISSUED, SO PROCEED TO STEP 7.			
03 IF ANY MESSAGES MUST BE WRITTEN, THE SALLOC MUST BE OBTAINED TO SERIALIZE ASM'S MESSAGE BUFFER.			
04 IF PLPA IS FULL (ASMPLPAF) AND SPILLED TO COMMON (ASMPLPAS), CALL THE SUBROUTINE TO BUILD MESSAGE 5 AND WRITE IT TO THE OPERATOR. IF COMMON IS FULL (ASMCOMMF), THEN CALL THE SUBROUTINE TO BUILD MESSAGE 6 AND WRITE IT TO THE OPERATOR. IF BOTH PLPA AND COMMON ARE FULL, THEN CALL THE SUBROUTINE TO BUILD MESSAGE 10 AND WRITE IT TO THE OPERATOR. MESSAGE 10 WILL REFER TO COMMON IF PLPA SPILLED TO COMMON. OTHERWISE, IT WILL REFER TO PLPA.		WRITEMSG WRITEMSG WRITEMSG	25.9.3
05 IF EITHER PLPA OR COMMON ARE UNUSABLE (PAREDSBD OR PARENUSE IS ONE), CALL THE SUBROUTINE TO BUILD MESSAGE 9 AND WRITE IT TO THE OPERATOR. CALL SUBROUTINE AGAIN TO BUILD MESSAGE 10 AND WRITE IT TO THE OPERATOR.		WRITEMSG WRITEMSG	25.9.3
06 IF THE DUPLEX DATA SET HAS BECOME FULL OR NOT USABLE (PAREDSBD OR PARENUSE IS ON), CALL THE SUBROUTINE TO BUILD MESSAGE 7 AND WRITE IT TO THE OPERATOR.		WRITEMSG	25.9.3
07 RETURN TO CALLER.			

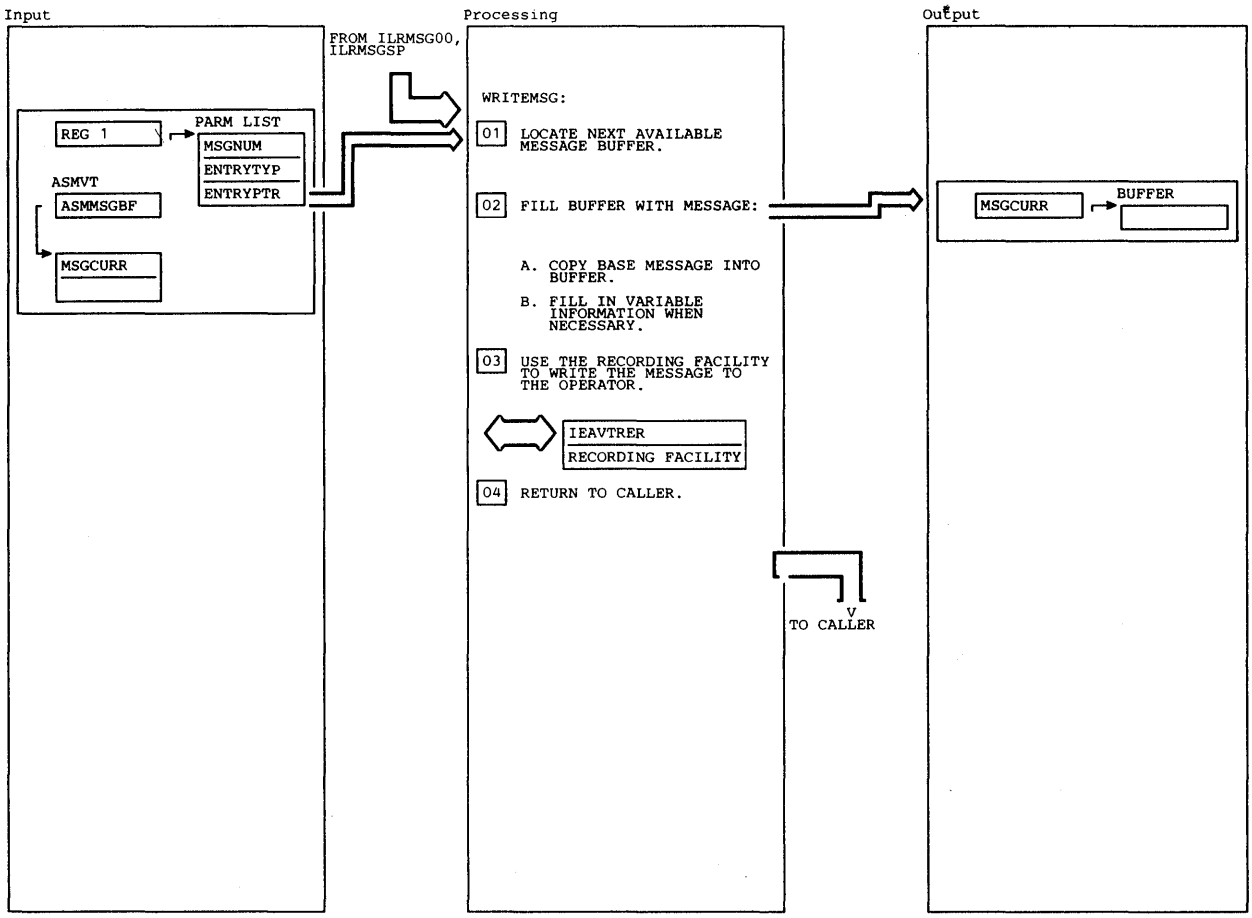
Diagram 25.9.1 ILRMSGSP (Part 1 of 1)



Notes	Routine	Label	Ref
01 RECOVERY IS ESTABLISHED (SETFRR) FOR THE TERMINATION ROUTINE TO GIVE ASM A SECOND CHANCE WHEN THE SYSTEM MUST BE TERMINATED. THE RECOVERY ROUTINE (ILRMSG01) WILL NOT ATTEMPT TO USE MCH (MACHINE CHECK HANDLER) TO TERMINATE THE SYSTEM, BUT WILL SIMPLY DO A LPSW (LOAD PSW).			
02 THE BASE TERMINATION MESSAGE IS PLACED IN THE TERMINATION BUFFER. THE VARIABLE INFORMATION IS THEN FILLED IN. THIS VARIABLE INFORMATION INCLUDES THE DATA SET TYPE (PLPA, COMMON, ETC.) AND THE VOLID OF THE DATA SET.			
03 THE SYSTEM TERMINATION LOGREC BUFFER IS INITIALIZED WITH ONE OF ASM'S WAIT STATE CODES. THE WAIT STATE IS X'02E' IF THE DSBD FLAG OR THE NUSE FLAG OF THE PARTE IS SET. OTHERWISE, THE WAIT STATE IS X'03C'. THE PARAMETER LIST IS ALSO INITIALIZED WITH THE REAL ADDRESS OF THE MESSAGE AND THE REAL ADDRESS OF THE LRB (LOGREC BUFFER).			
04 THE MACHINE CHECK HANDLER TERMINATION ROUTINE IS BRANCH ENTERED TO TERMINATE THE SYSTEM.	IGFPTERM	IGFPTERM	

Notes	Routine	Label	Ref

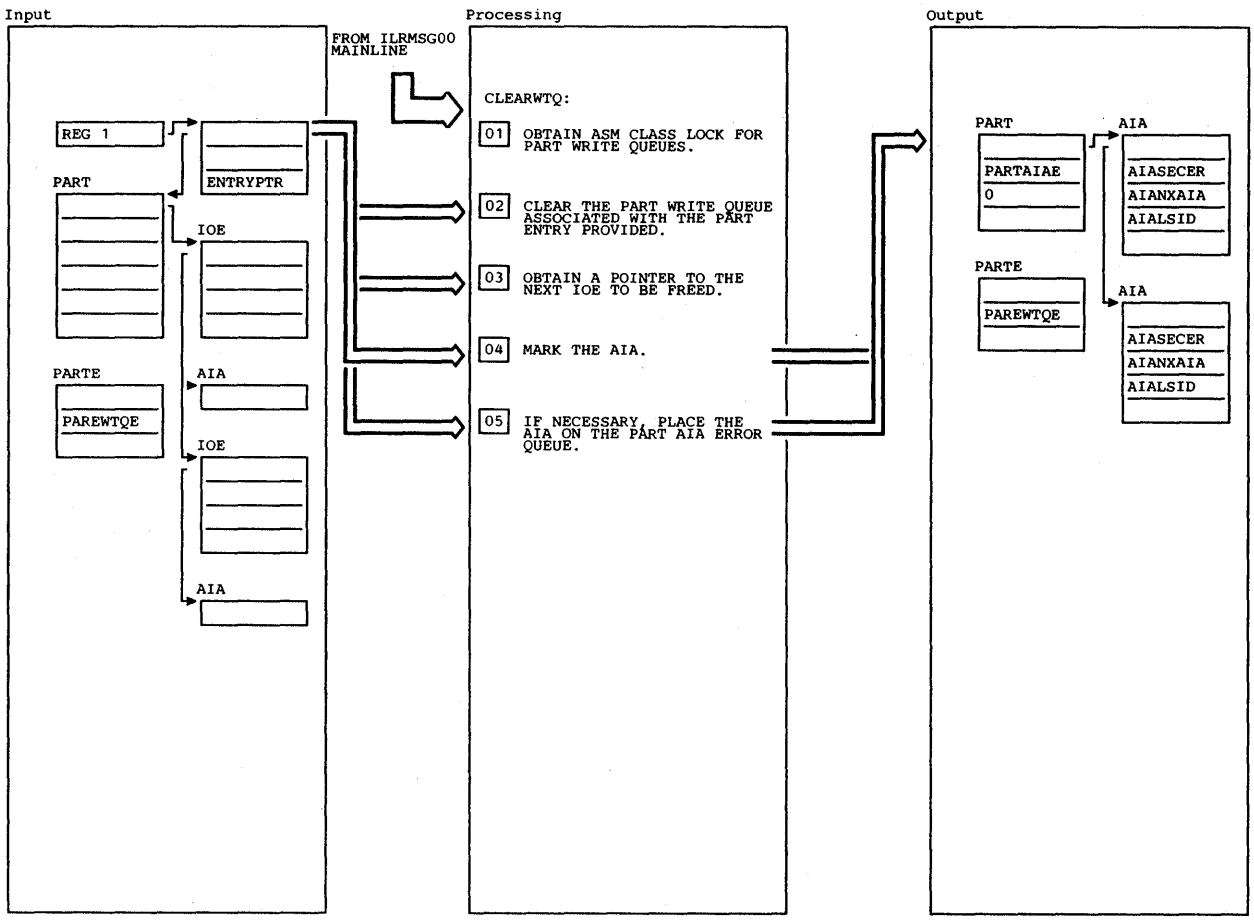
Diagram 25.9.2 TERMSYS (Part 1 of 1)



Notes	Routine	Label	Ref
01 ASM'S MESSAGE BUFFER AREA IS POINTED TO BY ASMMSGBF. THE FIRST WORD OF THE HEADER IS THE POINTER TO THE CURRENT BUFFER (I.E. THE BUFFER LAST USED). THIS POINTER IS UPDATED TO POINT TO THE NEXT AVAILABLE BUFFER. IF THE CURRENT BUFFER HAPPENS TO BE THE LAST BUFFER THEN THE NEXT AVAILABLE BUFFER IS THE FIRST BUFFER. THE NEW CURRENT BUFFER IS THE ONE TO BE USED.			
02 TO PLACE THE MESSAGE IN THE BUFFER: A. COPY THE BASE MESSAGE INTO THE BUFFER. THIS IS ALL THAT NEEDS TO BE DONE FOR MESSAGES 5 AND 6 SINCE THEY ARE CONSTANT. B. FOR MESSAGES 7, 9 AND 10, FILL IN THE VARIABLE INFORMATION SUCH AS THE DATA SET TYPE (PLPA, COMMON, LOCAL OR SWAP), THE VOLUME ID OF THE DATA SET, AND WHETHER THE DATA SET IS FULL OR BAD (THE DATA SET IS BAD IF EITHER THE DSBD FLAG OR THE NUSE FLAG IS ON, OTHERWISE THE DATA SET IS FULL).			
03 USE THE WTO OPTION OF RECORD TO WRITE THE MESSAGE TO THE OPERATOR. RETURN TO CALLER.	RECORD		

Notes	Routine	Label	Ref

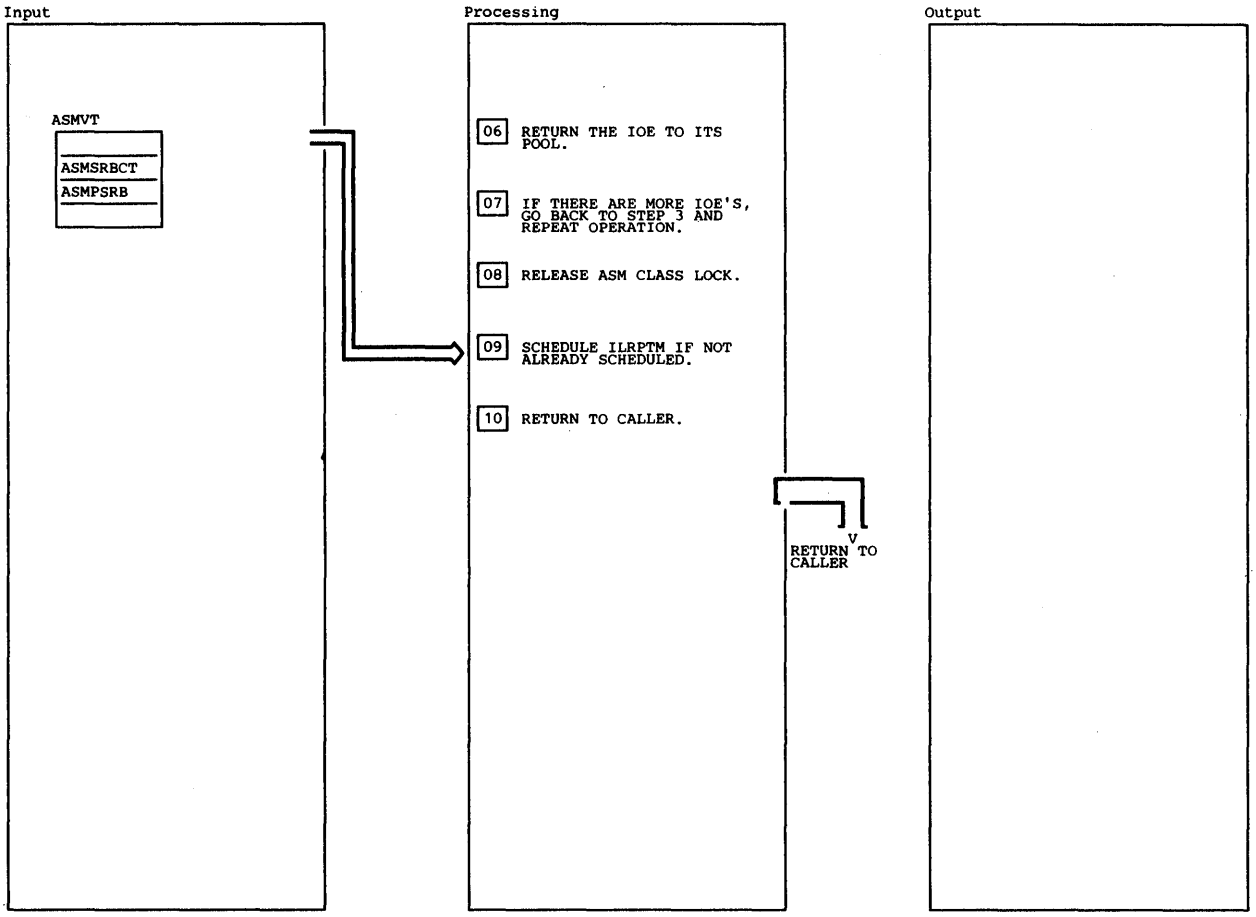
Diagram 25.9.3 WRITEMSG (Part 1 of 1)



Notes	Routine	Label	Ref
01 THE ASM CLASS LOCK IS OBTAINED IN ORDER TO SERIALIZE THE PART WRITE QUEUES.			
02 SAVE THE POINTER TO THE FIRST IOE ON THE PART WRITE QUEUE ASSOCIATED WITH THE PART ENTRY PROVIDED. ZERO THE FORWARD AND BACKWARD POINTERS OF THE PART WRITE QUEUE.			
03 OBTAIN A POINTER TO THE IOE THAT IS TO BE FREED.			
04 OBTAIN A POINTER TO THE AIA FROM THE IOE. IF THE PART ENTRY PROVIDED IS THE PART ENTRY FOR PLPA OR COMMON, THEN ZERO THE LSID FIELD IN THE AIA. IF THE PART ENTRY IS THE DUPLEX PART ENTRY, THEN TURN ON THE SECONDARY WRITE ERROR FLAG IN THE AIA.			
05 IF THE AIA COUNT OF OUTSTANDING WRITE OPERATIONS FOR A DUPLEXED WRITE OPERATION IS EQUAL TO TWO (INDICATING THIS IS THE FIRST REQUEST RETURNING FROM IOS), THEN DECREMENT THAT COUNT BY ONE. OTHERWISE, USE COMPARE AND SWAP LOGIC TO PLACE THAT AIA ON THE PART AIA ERROR QUEUE TO BE RETURNED TO RSM.			

Notes	Routine	Label	Ref

Diagram 25.9.4. CLEARWTQ (Part 1 of 2)



Notes	Routine	Label	Ref
06 USE ILRGMA TO RETURN THE IOE TO ITS POOL.	ILRGMA		
07 IF THERE ARE MORE IOE'S TO BE FREED, THEN GO BACK TO STEP 3 AND REPEAT THE PROCESS FOR THE NEXT IOE.			
08 THE ASM CLASS LOCK USED TO SERIALIZE THE PART WRITE QUEUES IS RELEASED.			
09 IF ILRPTM IS NOT ALREADY SCHEDULED, THEN SCHEDULE IT.			
10 RETURN TO ILRMSG00.			

Notes	Routine	Label	Ref

Diagram 25.9.4 CLEARWTQ (Part 2 of 2)

VIO Control

VIO Control coordinates and synchronizes all ASM processing required to support VIO data sets. ASM treats each VIO data set as a Logical Group (LG) of 4096-byte pages. The Virtual Block Processor (VBP) requests assignment of a LG each time a new VIO data set is created. ASM assigns a four-byte Logical Group Number (LGN) for each logical group requested by VBP. ASM provides a journaling facility for logical groups that allows VBP to direct saving the current contents of a VIO data set, for later recovery if necessary. For each journaled VIO data set ASM assigns a unique value called the 'S' symbol. All requests for services on a VIO data must be made using either the LGN or the eight byte 'S' symbol assigned to the data set. Each page within a logical group is identified by an eight-byte Logical Page Identifier (LPID). The LPID consists of the LGN followed by a four byte Relative Page Number (RPN). The LGN is assigned by ASM, the RPN is assigned by VBP.

The four central control blocks for VIO Control processing are the LGVT (Logical Group Vector Table), ASMHD (ASM Header), LGE (Logical Group Entry), and the ASPCT (Auxiliary Storage Page Correspondence Table).

The LGVT resides in SQA and contains a small header section plus an eight-byte entry (LGVTE) for each LG. The LGVTE contains the address of the ASCB for the address space to which LG is assigned and a pointer to the LGE for the LG.

The ASMHD is the focal point of VIO Control processing. An ASMHD exists for each active address space. It resides in SQA and contains paging I/O control information and VIO Control information. The VIO Control information includes a pointer to the SRB used to schedule SRB Controller and a queue header for the LGE queue.

An LGE exists for each LG assigned to the address space. The LGEs are allocated from SQA and reside on a single-threaded queue based in the ASMHD. The LGE controls all processing of a logical group. It includes control information for the LGE, a process queue containing ACEs (ASM Control Elements) and AIAs (ASM I/O Areas) representing all work in progress or waiting to be executed. The LGE also contains a pointer to the ASPCT.

The ASPCT contains an LSID (Logical Slot Identifier) for each VIO data set page written to auxiliary storage. The ASPCT also has a header with additional control information for the LG. An ASPCT exists for each LG and resides in user private area storage. For further information about the ASPCT, see "Diagnostic Aids" in Volume 7.

VIO Control consists of four central routines:

- ILRPOS — Page Operations Starter.
- ILRGOS — Group Operations Starter.
- ILRSRBC — SRB Controller.
- ILRVIOCM — VIO Completion.

VIO Control also includes a special Job Termination Resource Manager (ILRJTERM).

Page Operations Starter

I/O Control (ILRPAGIO) calls ILRPOS whenever a paging request for a VIO page is received from RSM. It can also be called by the Transfer Page Routine (ILRTRPAG), an entry point in ILRPOS entered from RSM.

VIO Page Requests

ILRPAGIO sends a chain of AIAs to ILRPOS. These AIAs may be for different VIO data sets. ILRPOS tests to determine if a paging operation is pending or if a group operation is pending or in progress for the LGs on which the paging is to be done. If there is a group operation pending or in progress, the paging request is in error and the error AIA is returned to ILRPAGIO. If there is a paging operation pending for this LG, ILRPOS queues the AIA to the LGE Process Queue for later processing.

If there is no paging operation pending or group operation pending or in progress, ILRPOS locates the LSID corresponding to the VIO LPID and queues the input AIA to the ASMVT staging queue (ASMSTAGQ). The LSID is located by finding the Logical-to-Physical Mapping Entry (LPME) in the ASPCT via the RPN portion of the LPID. The LPME address is put into the AIA. The LPME contains the LSID corresponding to the LPID. On a page-out operation, ILRPOS frees the LSID in the LPME. On a page-in, ILRPOS moves the LSID into the AIA.

The ILRESTRT entry point of ILRPOS handles any VIO paging requests that are queued for later processing. The SRB Controller (ILRSRBC) calls ILRESTRT whenever it finds unstarted paging requests on the LGE process queue.

Transfer Page Requests

RSM initiates a Transfer Page request by calling the ILRTRPAG entry point of ILRPOS. ILRTRPAG builds an ASM Control Element (ACE) by copying into it the information in the ACA (ASM Control Area) that RSM passes it. ILRTRPAG then calls the main entry point of ILRPOS. If a paging operation is pending for the Logical Group the Transfer Page

Request is being made against, ILRPOS queues the ACE to the LGE Process Queue for later processing. If the request can be started immediately, ILRPOS calls the ILRTRANS entry point of ILRPOS to process the request.

The ILRTRANS entry point of ILRPOS handles any Transfer Page requests that are queued for later processing. The SRB Controller (ILRSRBC, also part of VIO Control) calls ILRTRANS whenever it finds unstarted transfer requests on the LGE Process Queue.

Group Operations Starter

ILRGOS accepts the following group requests from VBP: ASSIGN LG, SAVE LG/LGN, ACTIVATE LG, and RELEASE LG. An ACA is the input parameter list. ILRGOS always does an ASSIGN operation immediately. SAVE, ACTIVATE, and RELEASE are started immediately only if no other operations are pending or in progress for the Logical Group.

ASSIGN LGN

For an ASSIGN request, ILRGOS assigns a new LGN, builds a LGE and an ASPCT, and returns the LGN to VBP.

SAVE, ACTIVATE, and RELEASE

For these requests ILRGOS moves the input information from the AIA into an ACE, and then queues the ACE to the LGE Process Queue to prevent any other group operation from starting until this operation completes. If any group operations are in progress or pending, the ACE is marked work-pending. Otherwise, ILRGOS calls the appropriate group operator (see Section 4, "VIO Group Operators") to process the request.

The Release LG operator (ILRRLG, one of the VIO Group Operators) calls the ILRFRELG entry point of ILRGOS to free the LGE and make the LGVTE available. For that Logical Group, ILRFRELG dequeues the LGE from the ASM Header Queue, returns the LGVTE to the available queue, and frees the LGE.

SRB Controller

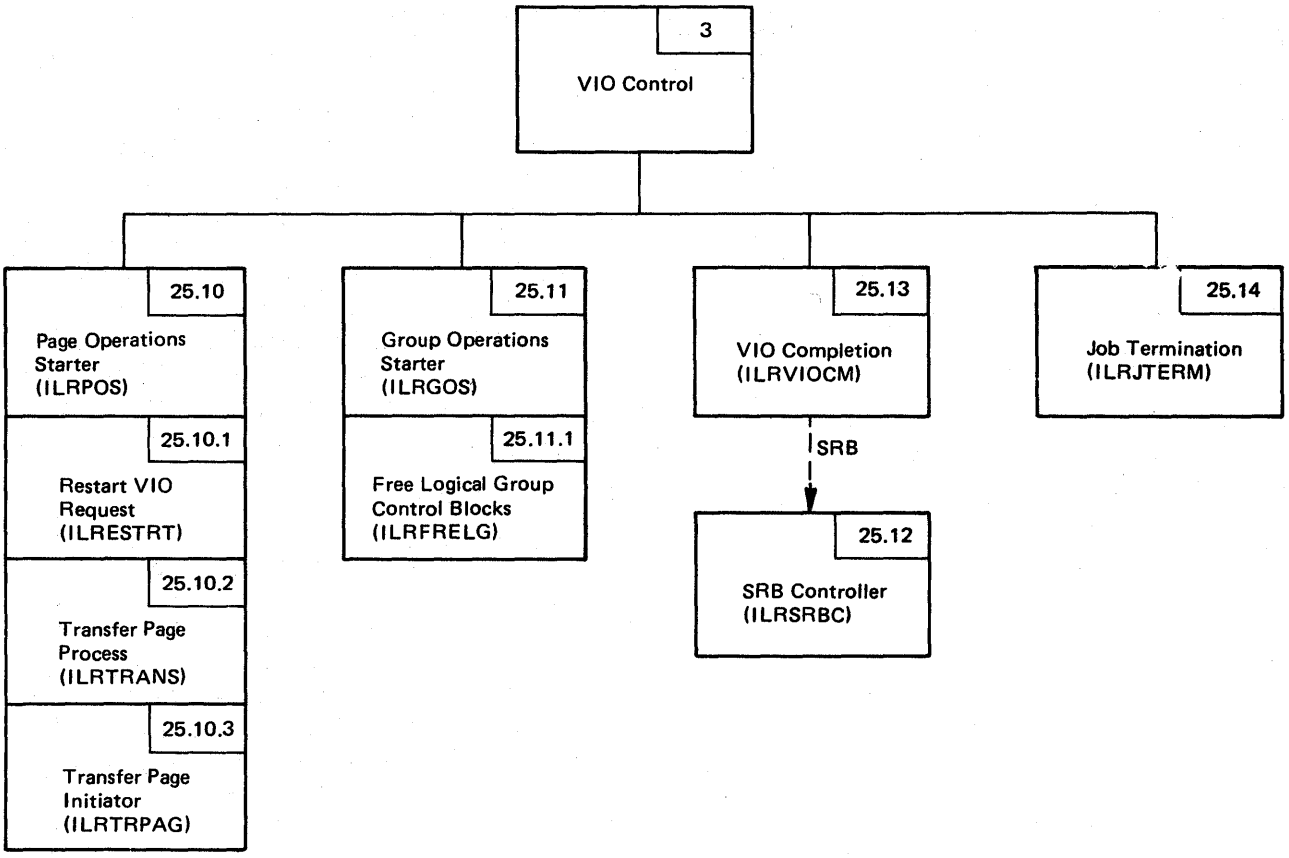
An SRB scheduled by VIO Completion or by ILRGOS causes the SRB Controller (ILRSRBC) to be dispatched in the address space for which a page or group operation is pending. ILRSRBC finds the pending work via the LGE queue based in the ASM Header (ASMHD), determines which work can be started, separates the startable work into group operation and page operation chains, and starts the work by posting ILRPOS, by calling the appropriate group operator, or by calling the ILRTRANS or by ILRESTRT entries in ILRPOS.

VIO Completion

I/O Control passes control to VIO Completion (ILRVIOCM) whenever a VIO paging operation is completed. ILRVIOCM processes one AIA as input, dequeues it from the LGE Process Queue, and returns it to I/O Control. For a page-out, ILRVIOCM stores the newly-assigned LSID in the ASPCT. For a page-in, ILRVIOCM sets error flags, if necessary. If any more work is pending on the LGE Process Queue, ILRVIOCM schedules an SRB for ILRSRBC to start the work prior to return to I/O Control.

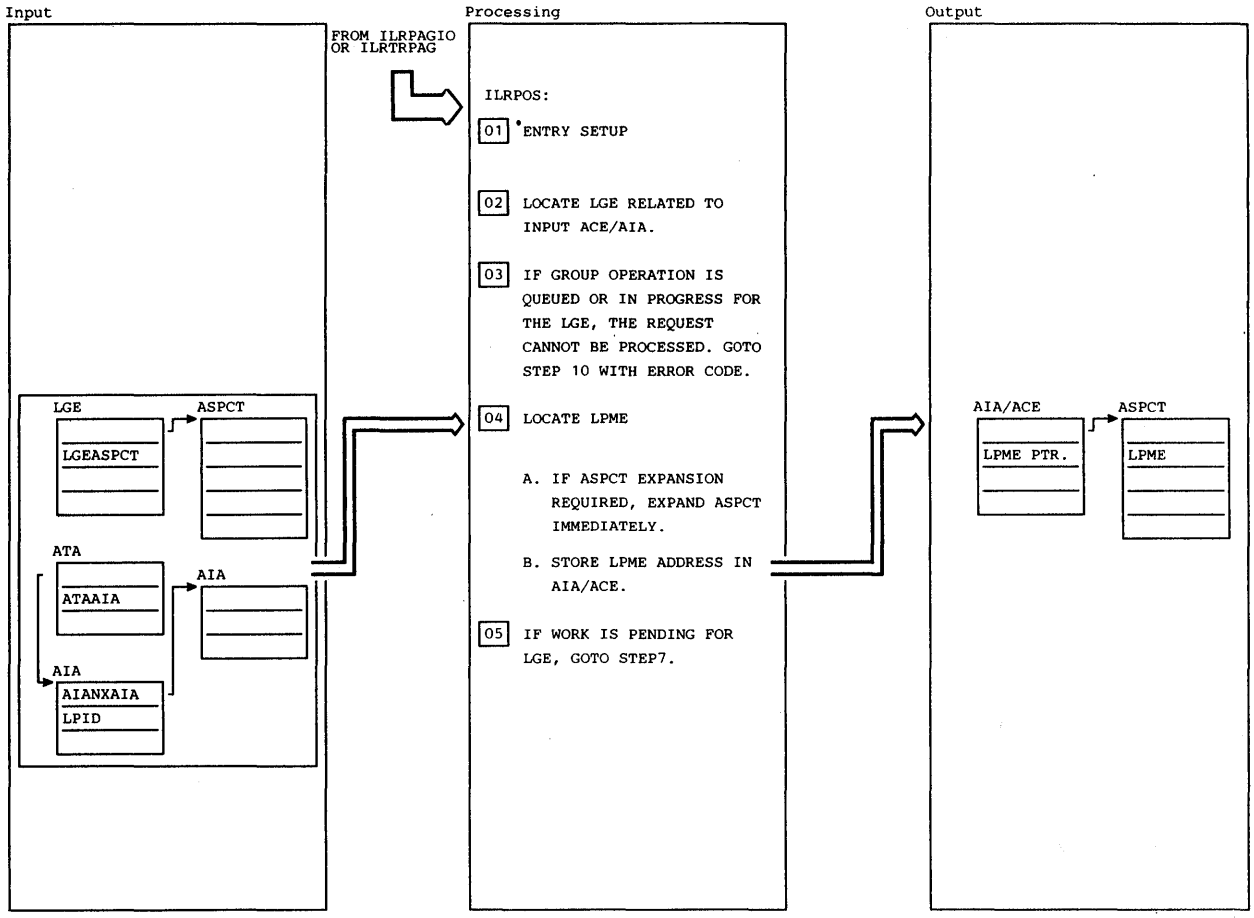
Job Termination Resource Manager

The initiator's job termination module (IEFSD166) calls the Job Termination Resources Manager (ILRJTERM) to deactivate any VIO data sets still active at job deletion time. ILRJTERM searches each LGE process queue for a RELEASE LG ACE. If a RELEASE ACE is not queued for an LG, ILRJTERM obtains one, intializes it, and queues it. ILRJTERM then schedules ILRSRBC to start the RELEASE operations.



25.x. — Module
25.x.y. — Entry point in module 25.x.

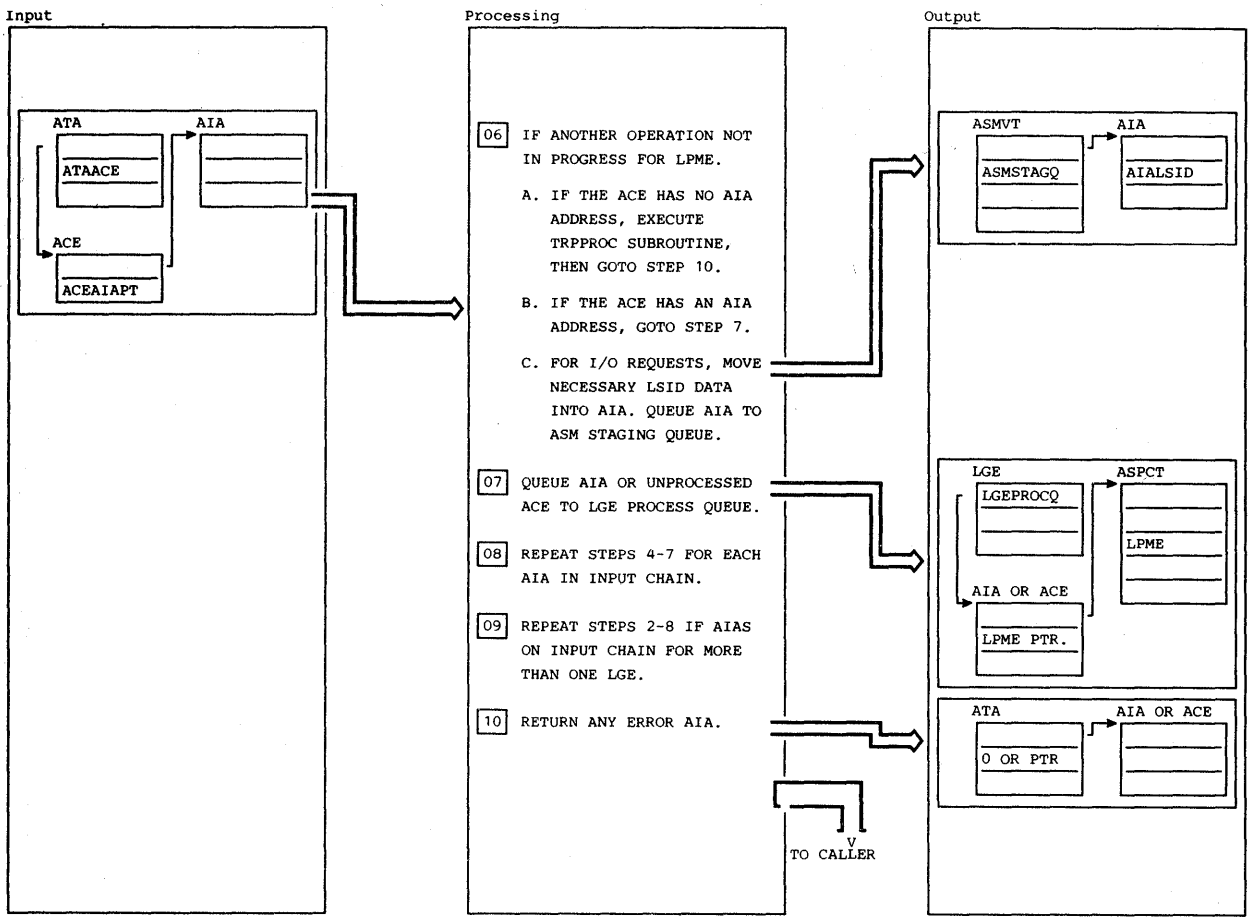
Figure 2-59. VIO Control Overview



Notes	Routine	Label	Ref
01 PAGE OPERATIONS STARTER (POS) RECEIVES CONTROL FROM I/O CONTROL, OR FROM THE ILRTRPAG SECONDARY ENTRY POINT FOR TRANSFER PAGE REQUESTS. INPUT IS A SINGLE ACE (ILRTRPAG) OR A STRING OF ONE OR MORE AIAS (I/O CONTROL). ILRPOS ATTEMPTS TO START ALL OPERATIONS IMMEDIATELY. AIAS THAT CAN BE STARTED IMMEDIATELY ARE RETURNED TO I/O CONTROL. ACE (TRANSFER PAGE) IS PROCESSED COMPLETELY IF STARTABLE IMMEDIATELY. OTHER AIAS AND ACES ARE PUT ON THEIR PROCESS QUEUES FOR LATER PROCESSING. ASM LOCK OF CURRENT ADDRESS SPACE IS OBTAINED. FOR RECOVERY, ILRIOFRR RECOVERY ROUTINE HANDLES ERRORS OCCURRING IN ILRPOS (ALL ENTRIES).			
02 THE LGE IS FOUND VIA THE LPID IN THE INPUT ACE/AIA.			
03 EITHER OF THESE CONDITIONS WILL PREVENT PROCESSING OF THE OPERATION. PAGE OPERATIONS ARE TREATED AS ERRORS IN THIS CASE			

Notes	Routine	Label	Ref
TO PREVENT INTERLOCK SITUATIONS FROM ARISING IF ASPCT EXPANSION IS REQUIRED FOR THE LG.			
04 WHILE LOCATING THE LPME VIA THE RPN OF THE LPID, THE ASPCT MAY REQUIRE EXPANSION.			
A. THE RPN LEADS TO AN LPME THAT DOES NOT YET EXIST IN THE ASPCT. ASPCT EXPANSION IS PERFORMED IMMEDIATELY WHILE ALL NECESSARY LOCKS ARE HELD. DELAY OF THIS PROCESSING WOULD CAUSE A POTENTIAL LOCAL LOCK INTERLOCK SITUATION.			
B. ONCE THE RPN LEADS TO AN EXISTING LPME, THE LPME ADDRESS IS PLACED IN THE AIA/ACE FOR USE BY OTHER VIO CONTROLLER ROUTINES.			
05 IF WORK IS PENDING FOR THE LGE, NEW WORK CANNOT BE STARTED BECAUSE IT MAY BE FOR THE SAME PAGE FOR WHICH WORK IS PENDING.			

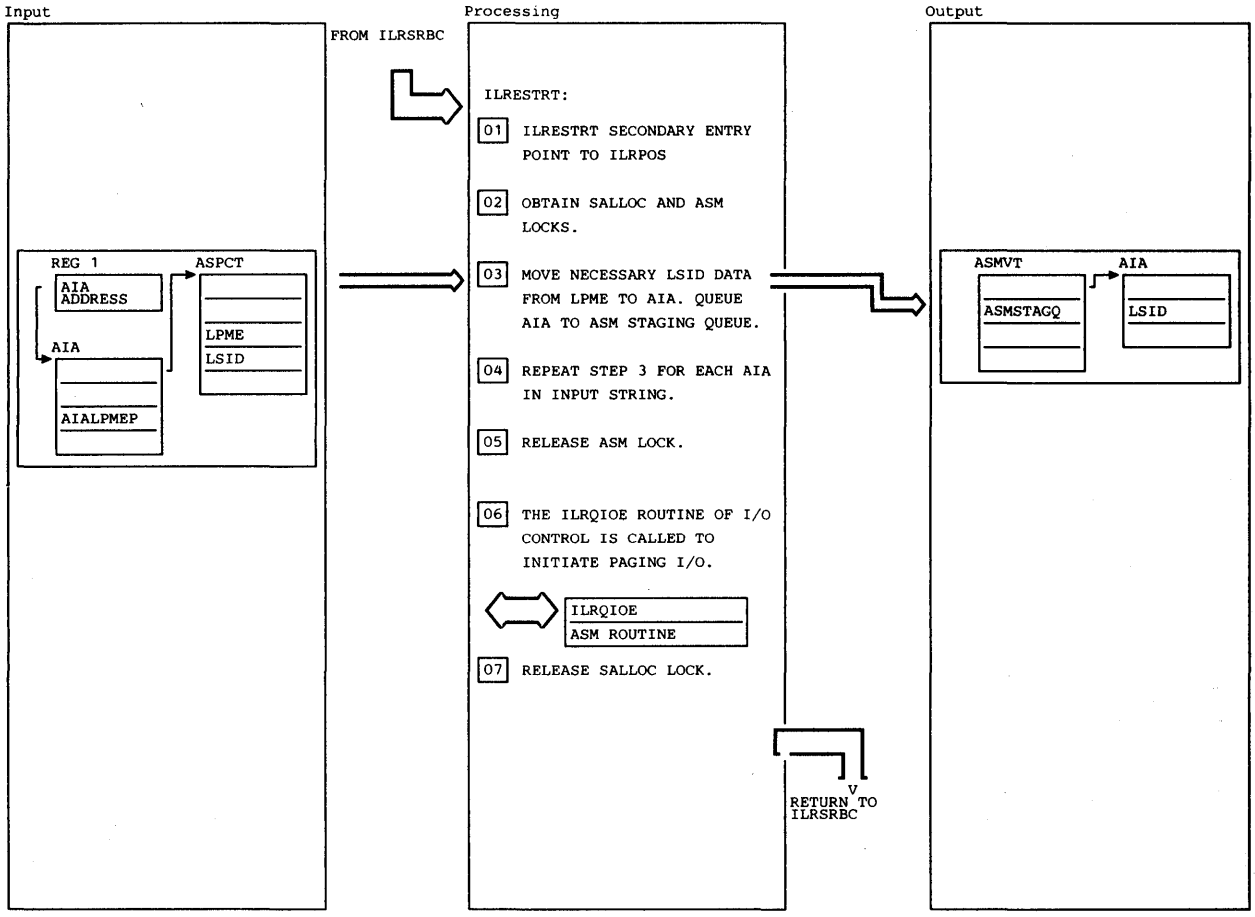
Diagram 25.10 ILRPOS (Part 1 of 2)



Notes	Routine	Label	Ref
06 AT THIS POINT, LPME IS KNOWN AND IS PROCESSED UNLESS THE LPME IS ALREADY IN PROGRESS. IN THIS CASE, THE AIA/ACE MUST BE QUEUED TO THE PROCESS QUEUE TO BE HANDLED LATER.			
A. IF THE INPUT IS A SINGLE ACE, THE TRANSFER PAGE REQUEST IS PROCESSED IMMEDIATELY BY THE SUBROUTINE. THE ACE IS FREED.		TRPPROC	25.11.2
B. IF THE TRP ACE HAS AN AIA ADDRESS, THE TRANSFER PAGE OPERATION CANNOT BE STARTED UNTIL I/O REPRESENTED BY THE AIA COMPLETES.			
C. FOR PAGING I/O AIAS, MOVE NECESSARY DATA FROM LPME TO AIA. LSID MOVED FOR PAGE-IN REQUESTS, LSID FREED (BY ILRFRSL1) AND/OR CLEARED FOR PAGE-OUTS. THE LPME PTR IS SAVED IN AIA AND AIA QUEUED TO ASM STAGING QUEUE TO BE PROCESSED BY THE ILRQIOE SUBROUTINE OF I/O CONTROL.	ILRFRSLT	ILRFRSL1	

Notes	Routine	Label	Ref
07 ALL AIAS WILL BE QUEUED AND ANY UNPROCESSED ACES QUEUED TO PROVIDE SYNCHRONIZATION OF ALL OPERATIONS FOR LG.			
08 EACH AIA IS PROCESSED SEPARATELY. WHEN A NEW LGID IS ENCOUNTERED IN AN LGE, THE WORK PENDING FLAG IS SET ON THE CURRENT LGE IF ALL AIAS WERE NOT QUEUED TO THE STAGING QUEUE.			
09 AIAS MAY BE PASSED FOR MULTIPLE LGIDS (LOGICAL GROUPS), BUT ALL LSIDS MUST BE IN THE CURRENT ADDRESS SPACE.			
10 ANY AIAS STARTABLE IMMEDIATELY HAVE BEEN QUEUED TO THE STAGING QUEUE. AIAS NOT QUEUED TO STAGING QUEUE WILL BE STARTED LATER BY ILRSRBC. IN ERROR CONDITIONS, THE AIA/ACE IS RETURNED TO THE CALLER.			

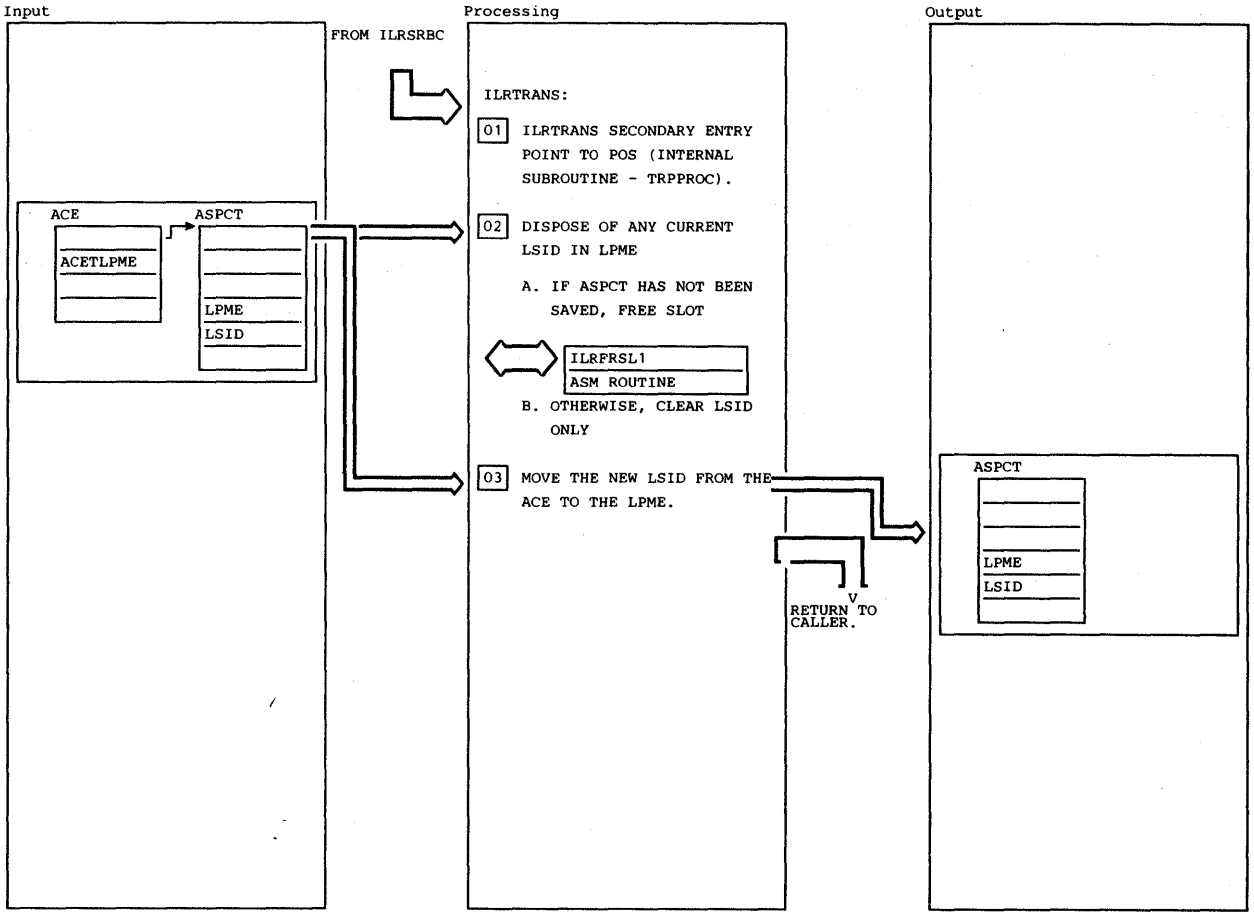
Diagram 25.10 ILRPOS (Part 2 of 2)



Notes	Routine	Label	Ref
01 THE RESTART ENTRY POINT OF POS RECEIVES CONTROL FROM SRBC WHENEVER UNSTARTED I/O REQUESTS (AIAS) ARE FOUND ON A PROCESS QUEUE. THE PROPER LSID INFORMATION IS PLACED IN THE AIAS AND THE AIAS ARE QUEUED TO THE STAGING QUEUE FOR I/O CONTROL TO START THE I/O OPERATION. INPUT MAY BE A STRING OF ONE OR MORE AIAS. THIS ENTRY POINT USES SUBROUTINES COMMON TO MAINLINE ILRPOS PROCESSING.			
02 THE SALLOC LOCK IS REQUIRED TO PROVIDE A SAVE AREA IF THE FREE SLOT ROUTINE HAS TO BE CALLED. IT IS ALSO REQUIRED FOR THE CALL TO ILRQIOE. THE ASM CLASS LOCK FOR THE CURRENT ADDRESS SPACE IS REQUIRED TO SERIALIZE LPME PROCESSING.			
03 THE LPME ADDRESS IN THE AIAS IS PROCESSED. LSID IS MOVED INTO THE AIA FOR PAGE-IN REQUESTS. LSID IS FREED (BY ILRFRSL1) AND/OR CLEARED FOR PAGE OUTS.	ILRFRSLT	ILRFRSL1	

Notes	Routine	Label	Ref
04 A STRING OF AIAS MAY BE PASSED AS INPUT. THEY DO NOT HAVE TO BE FOR THE SAME LOGICAL GROUP, BUT MUST BE ONLY FOR LOGICAL GROUPS IN THE CURRENT ADDRESS SPACE.			
05 THE ASM LOCK IS RELEASED IN ORDER THAT THE SALLOC LOCK (LOWER IN HIERARCHY) IS THE ONLY LOCK HELD AT ENTRY TO ILRQIOE. THE ASM LOCK IS NO LONGER NEEDED BY THIS ROUTINE.			
06 THIS ENTRY OF I/O CONTROL REQUIRES THE SALLOC LOCK.	ILRPAGIO	ILRQIOE	
07 THE SALLOC LOCK IS FREED BEFORE RETURNING TO ILRSRBC WHO HAS NO FURTHER LOCK REQUIREMENTS.			

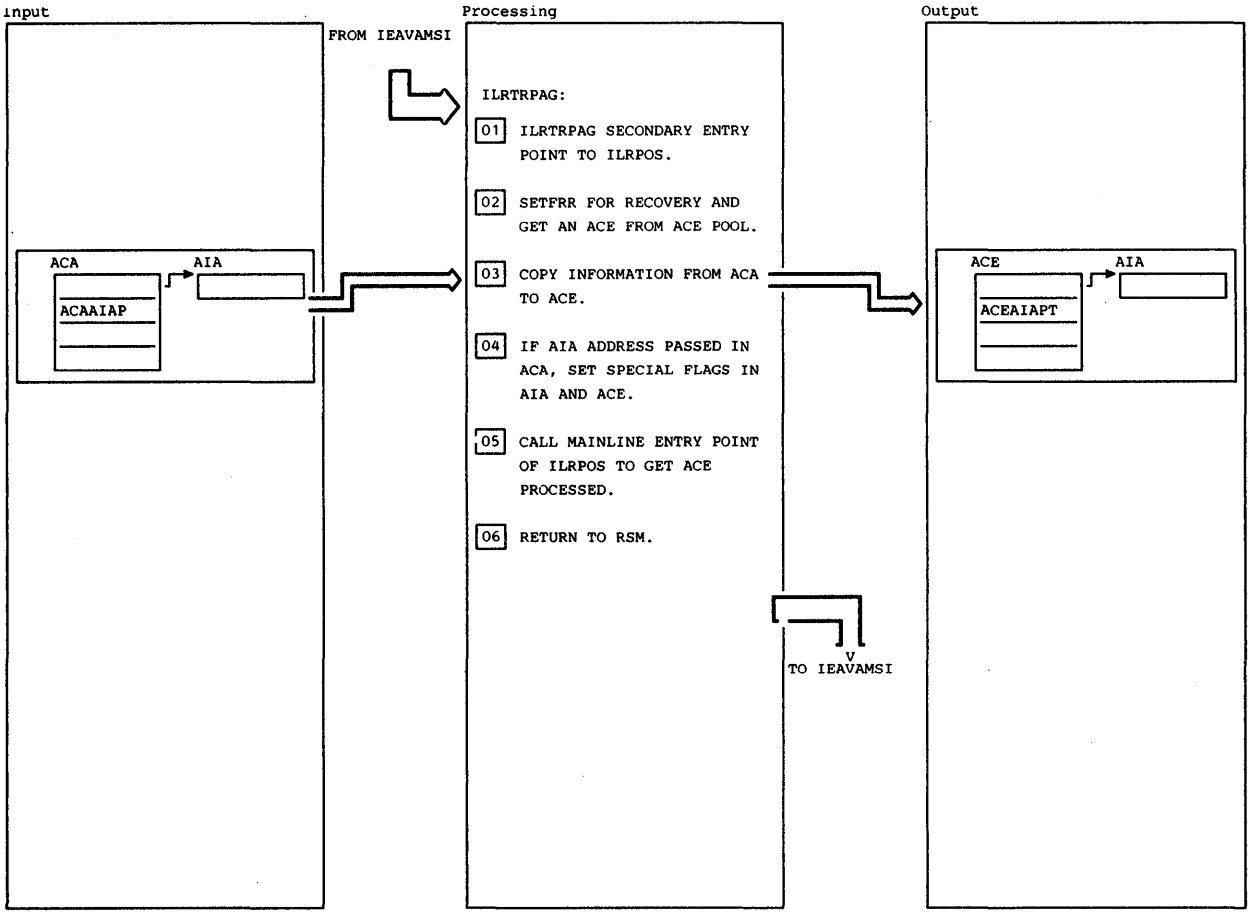
Diagram 25.10.1 ILRESTRT (Part 1 of 1)



Notes	Routine	Label	Ref
01 ILRTRANS SUBROUTINE/ENTRY OF ILRPOS IS CALLED BOTH INTERNALLY BY ILRPOS AND EXTERNALLY BY ILRSRBC WHENEVER A TRANSFER PAGE OPERATION IS REQUIRED. INPUT IS A SINGLE ACE CONTAINING A SOURCE LSID AND A TARGET LPME ADDRESS.			
02 THE INPUT ACE POINTS TO THE LPME TO PROCESS.			
A. ANY VALID LSID IN THE LPME IS FREED (BY ILRFRSL1) IF THE ASPCT HAS NOT BEEN SAVED OR THE CURRENT LPME HAS ALREADY BEEN RELEASED AT LEAST ONCE AFTER THE LAST SAVE.	ILRFRSLT	ILRFRSL1	
B. IF THE SLOT IDENTIFIED BY THE LSID MUST BE SAVED FOR A FUTURE ACTIVATE, THE LSID IS SET TO ZERO BUT THE ASSOCIATED SLOT IS NOT FREED.			
03 THIS COMPLETES THE TRANSFER PAGE OPERATION. THE LSID MOVED INTO THE LPME WAS FORMERLY ASSIGNED TO A VIO WINDOW PAGE BY ASM AS THE RESULT OF A NON-VIO DIRECTED PAGE-OUT OPERATION.			

Notes	Routine	Label	Ref

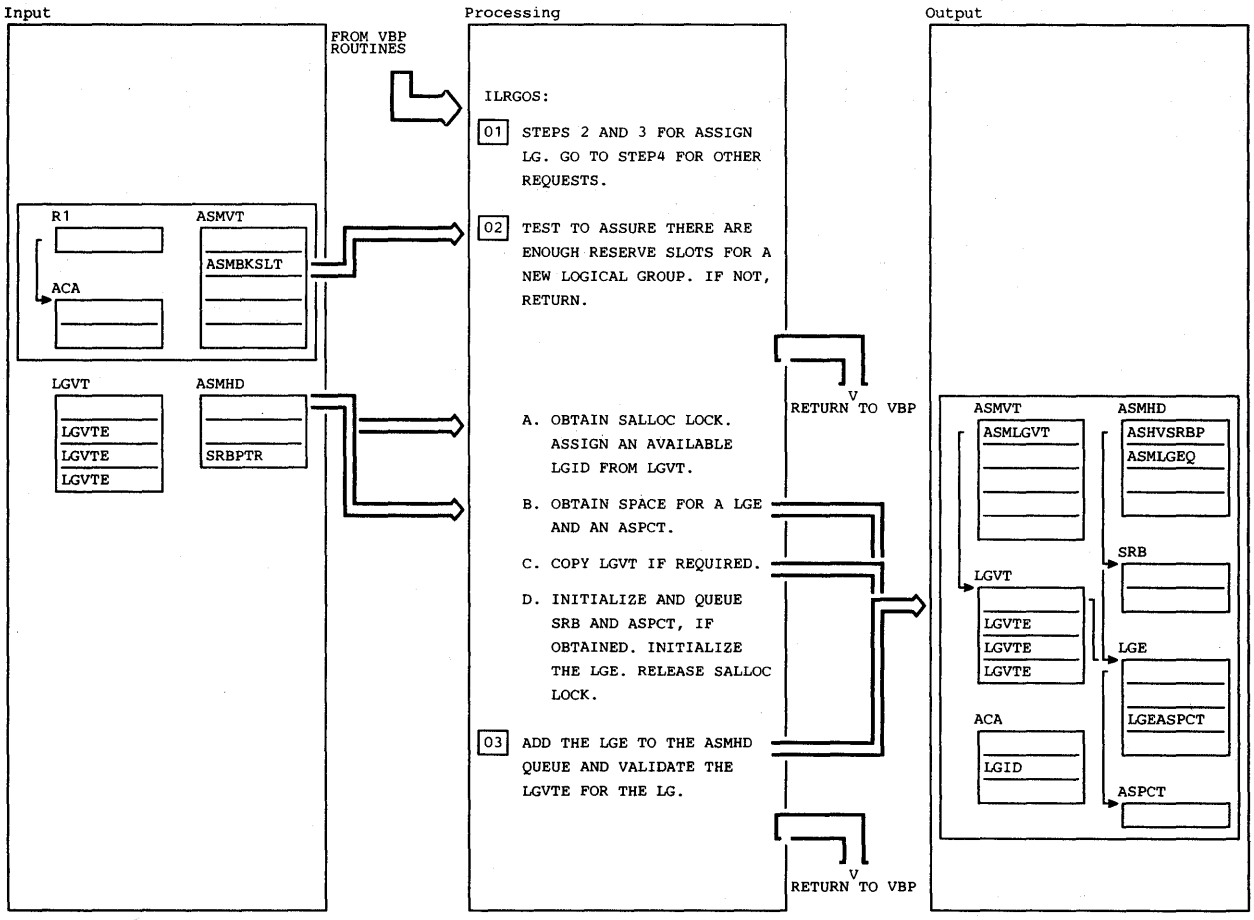
Diagram 25.10.2. ILRTRANS (Part 1 of 1)



Notes	Routine	Label	Ref
01 THIS IS AN INTERNAL ENTRY POINT TO ILRPOS USED BY RSM WHEN REQUESTING A TRANSFER PAGE OPERATION VIA THE ILRCALL MACRO. THIS ROUTINE WILL BUILD AN ACE AND PASS IT TO MAINLINE ILRPOS FOR PROCESSING.			
02 THE RECOVERY ENVIRONMENT IS CREATED FOR ASM VIA A SETFRR SINCE THIS IS AN EXTERNAL ENTRY TO ASM. THE ACE IS OBTAINED FROM THE COMMON ACE POOL, BASED IN THE ASMVT.			
03 THE INFORMATION PASSED IN THE ACA IS COPIED TO THE ACE. THE ACA IS ONLY A PARAMETER LIST OWNED BY RSM THAT MUST BE RETURNED ON EXIT EVEN IF THE OPERATION HAS NOT COMPLETED.			
04 IF RSM SUPPLIED A NON-ZERO AIA ADDRESS IN THE ACA, A SOURCE LSID IS NOT YET AVAILABLE. THE SOURCE LSID WILL BE AVAILABLE WHEN THE PAGE-OUT OPERATION IDENTIFIED BY THE AIA COMPLETES.			

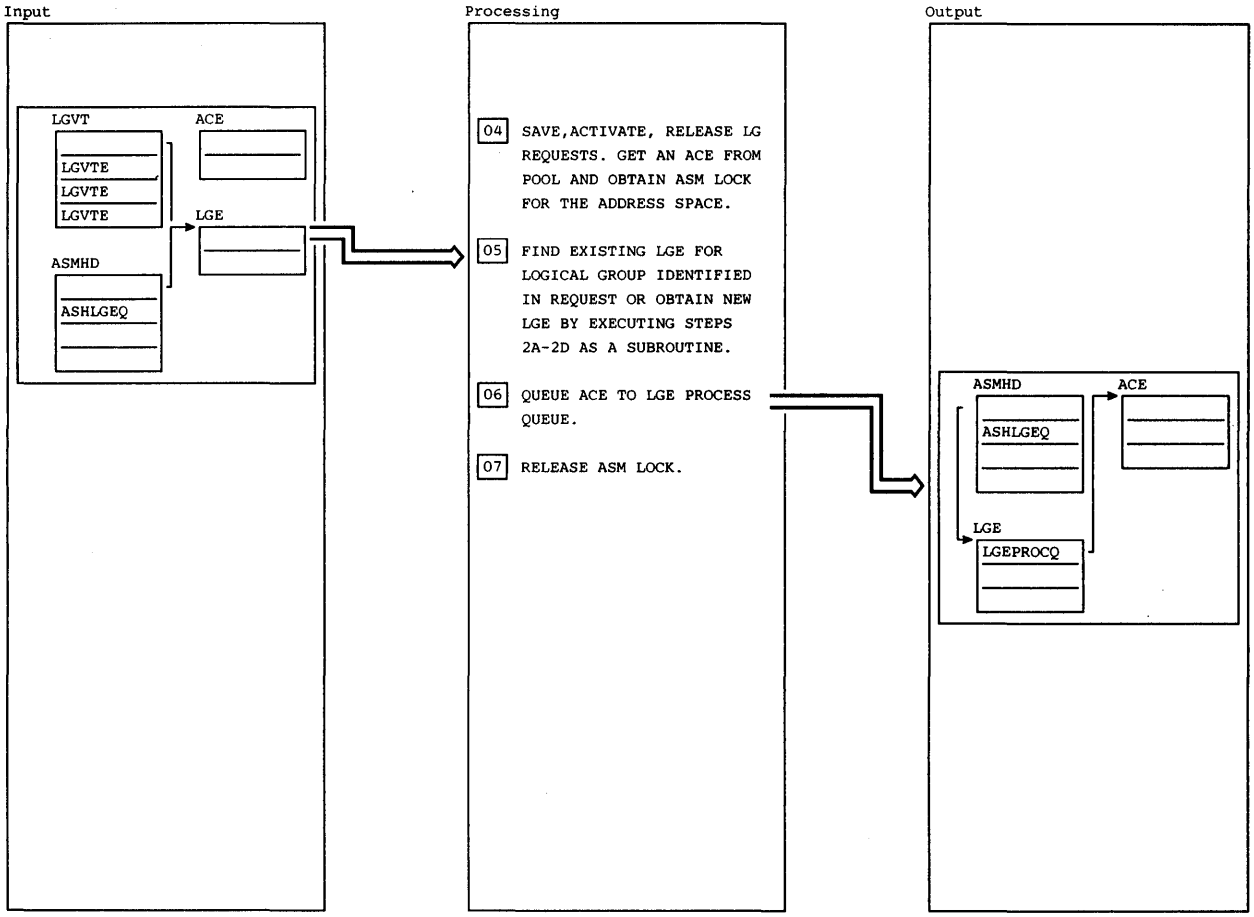
Notes	Routine	Label	Ref
THEREFORE, THIS AIA MUST BE MARKED AS A VIO AIA FOR PROCESSING BY VIO COMPLETION. THE ACE IS ALSO MARKED TO INDICATE THAT IT IS WAITING FOR I/O TO COMPLETE.			
05 MAINLINE ILRPOS CODE IS CALLED, PASSING THE ADDRESS OF THE ACE TO BE PROCESSED IN THE ATA. THE ATA (ASM TRACKING AREA) IS USED TO RECORD INFORMATION NEEDED FOR RECOVERY. UPON RETURN, IF ERRORS OCCURRED, THE ACE IS FREED AND STEP 4 BACKED OUT IF EXECUTED. IF NO ERRORS OCCURRED, ILRPOS MAINLINE HAS DISPOSED OF THE ACE.	ILRPOS	ILRPOS	
06 NO SPECIAL RETURN INFORMATION IS PASSED BACK TO RSM EXCEPT A RETURN CODE.			

Diagram 25.10.3 ILRTRPAG (Part 1 of 1)



Notes	Routine	Label	Ref
01 THE GROUP OPERATIONS STARTER (GOS) ALWAYS RECEIVES CONTROL FROM VBP (VIRTUAL BLOCK PROCESSOR) VIA AN ILRCALL MACRO INSTRUCTION. AN ACA IS THE INPUT PARAMETER LIST. GOS EXECUTES ALL ASSIGN LG REQUESTS IMMEDIATELY. SAVE, ACTIVATE AND RELEASE MAY OR MAY NOT BE STARTED IMMEDIATELY. FOR ASSIGN AND RELEASE, THE LOCAL LOCK IS HELD ON ENTRY AND SETFRR IS ISSUED FOR RECOVERY. AN ESTAE IS USED FOR SAVE AND ACTIVATE REQUESTS BECAUSE NO LOCKS ARE HELD AT ENTRY. ILRGOS01 (BOTH AN FRR AND ESTAE RECOVERY ROUTINE) HANDLES ERRORS OCCURRING IN ILRGOS.			
02 IF ENOUGH SLOTS ARE NOT AVAILABLE, AN ERROR RETURN CODE IS PASSED BACK TO VBP.			
A. THE SALLOC LOCK IS NEEDED FOR THE SQA GETMAIN. IT ALSO SERIALIZES LGVT EXPANSION AND SRB CREATION, IF REQUIRED. AN LGID IS TAKEN FROM A LGVTE ON THE LGVT AVAILABLE QUEUE. IF NO LGVTE IS AVAILABLE, THE LGVT MUST BE EXPANDED BY BEING COPIED INTO A LARGER STORAGE AREA.			
B. SQA SPACE IS OBTAINED FOR AN LGE, FOR AN SRB THIS IS FIRST ASSIGN FOR ADDRESS SPACE, AND FOR A LGVT IF REQUIRED IN STEP2A. LSQA SPACE IS OBTAINED FOR AN ASPCT.	IEAVGM00 IEAVGM00	GLBRANCH CRBRANCH	
C. ANY CODE THAT REFERENCES A LGE THRU THE LGVT MUST NOT SAVE THE POINTER TO THE LGVTE. IF THE LGVT MUST BE COPIED TO BE EXPANDED, THE POINTER WILL BE CHANGED.			
D. SALLOC IS RELEASED AFTER THE CONTROL BLOCKS HAVE BEEN BUILT AND QUEUED. IF STEP 5 IS EXECUTOR OF THESE STEPS, THE ASPCT WILL BE OBTAINED BY THE GROUP OPERATOR.			
03 THIS WHOLE OPERATION IS SERIALIZED BY THE ASM LOCK FOR THE ADDRESS SPACE.			

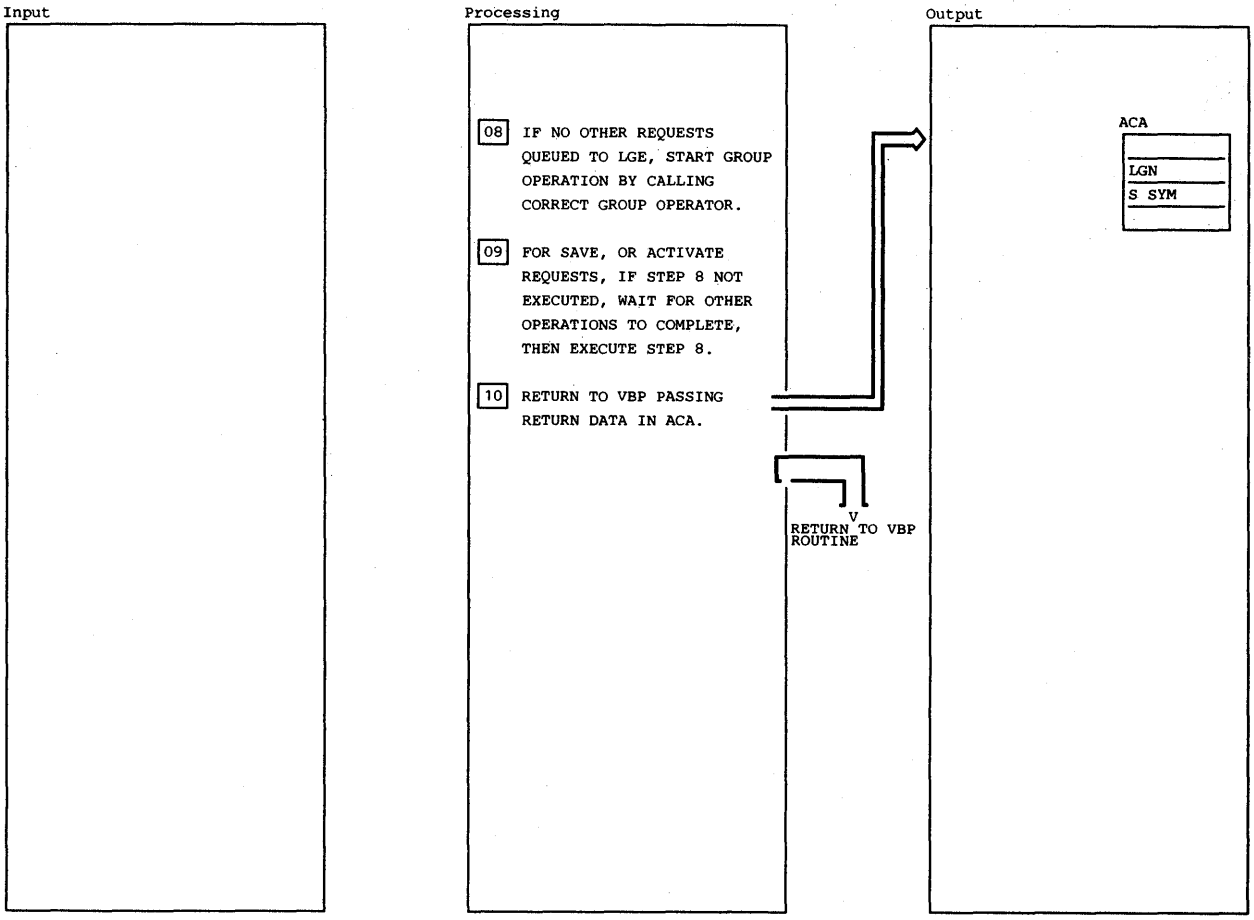
Diagram 25.11 ILRGOS (Part 1 of 3)



Notes	Routine	Label	Ref
04 THE ACE IS OBTAINED BEFORE THE LOCK IN ORDER TO ALLOW EXPANSION OF THE ACE POOL IF NECESSARY.			
05 IF REQUEST IS FOR RELEASE 'S' SYMBOL, LGE MAY NOT EXIST DUE TO A WARM START. FOR ACTIVATE REQUESTS, AN LGE IS ASSUMED TO NEVER EXIST. A NEW LGID AND LGE ARE CREATED IN THESE CASES.			
06 THE ACE IS QUEUED IN ANTICIPATION OF ASYNCHRONOUS COMPLETION OF GROUP OPERATIONS AND TO PREVENT ANY OTHER OPERATION FROM STARTING UNTIL THIS OPERATION COMPLETES.			
07 THE ASM LOCK IS RELEASED BEFORE CALLING ANY GROUP OPERATORS BECAUSE THEY ARE IN PAGEABLE LPA. PAGE FAULTS MUST NOT OCCUR WHILE A GLOBAL LOCK IS HELD.			

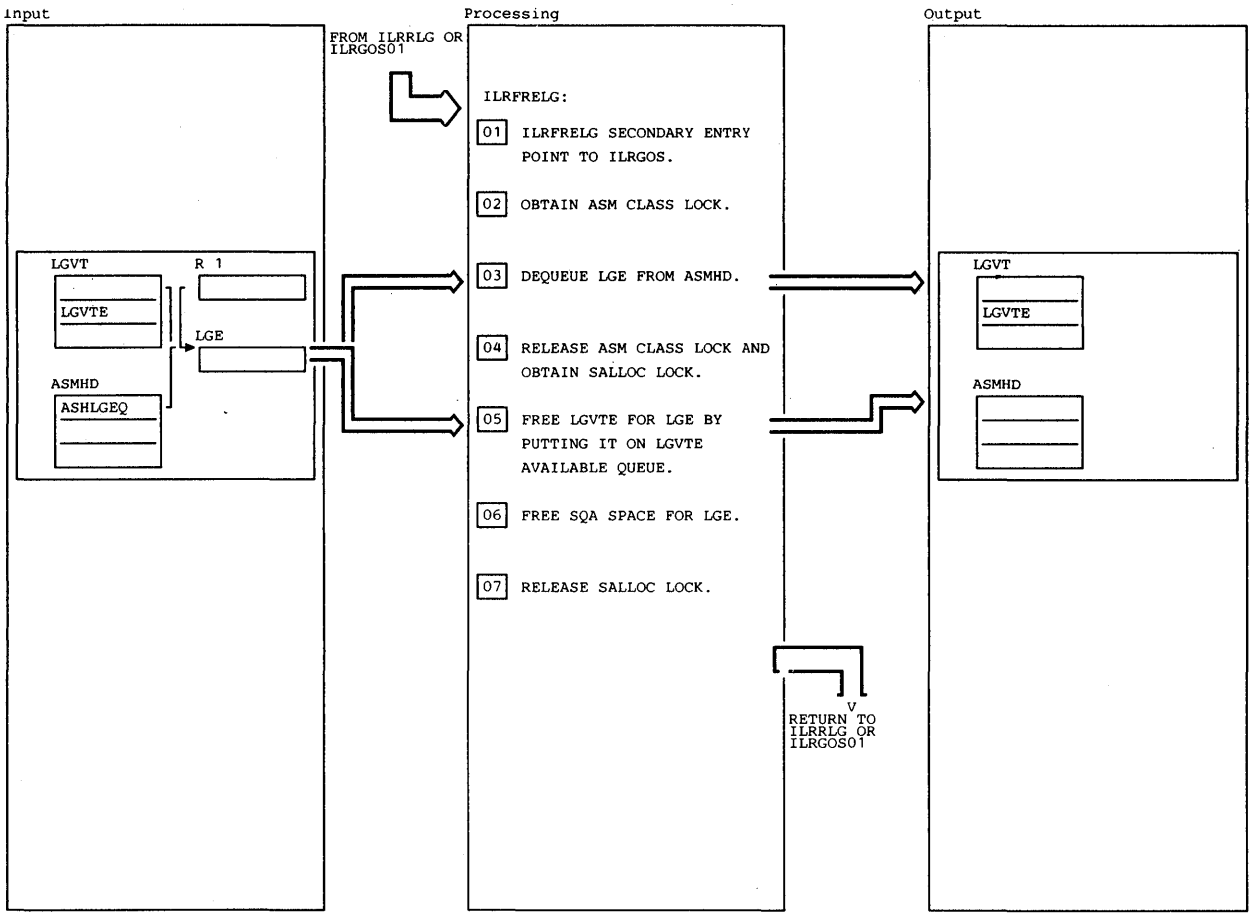
Notes	Routine	Label	Ref

Diagram 25.11 ILRGOS (Part 2 of 3)



Notes	Routine	Label	Ref
08 IF THE OPERATION CANNOT BE STARTED IMMEDIATELY THE LGE IS MARKED WORK PENDING.	ILRSAV ILRACT ILRRLG	ILRSAV ILRACT ILRRLG	
09 ILRGOS WAITS ON AN ECB IN THE ACE THAT WILL BE POSTED BY SRB CONTROLLER WHEN ALL CURRENTLY QUEUED WORK ON THE PROCESS QUEUE IS COMPLETE. IF OR WHEN THE GROUP OPERATOR CAN BE CALLED, GOS ALLOCATES A VSAM BUFFER FOR THE OPERATION VIA A COUNT. IF NO BUFFERS ARE AVAILABLE, ILRGOS WAITS FOR ONE TO BECOME AVAILABLE. AT THIS TIME THE SAVE OR ACTIVATE GROUP OPERATOR IS CALLED. UPON RETURN, THE VSAM BUFFER IS RETURNED TO THE GENERAL POOL AND ANY GOS ROUTINE WAITING UNDER ANOTHER TCB FOR THE BUFFER IS POSTED. NOTE THESE ACTIONS ARE TAKEN ONLY FOR SAVE OR ACTIVE REQUESTS. NO WAIT OR VSAM BUFFER MANAGEMENT IS NECESSARY FOR RELEASE LG REQUESTS.			
10 INFORMATION IN THE ACE IS MOVED TO THE ACA AND THE ACE DEQUEUED AND FREED WHEN THE OPERATION IS COMPLETE.			

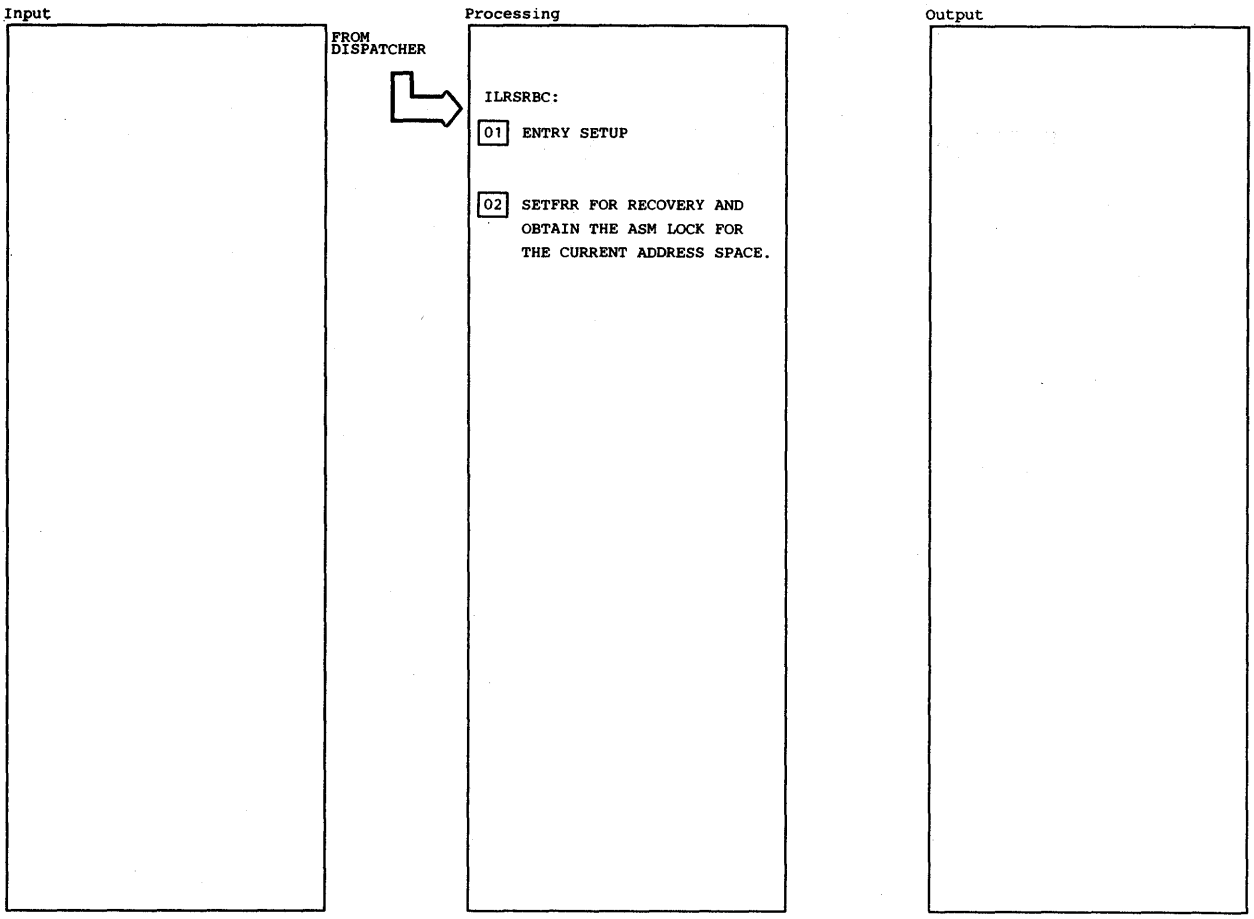
Diagram 25.11 ILRGOS (Part 3 of 3)



Notes	Routine	Label	Ref
01 THE ILRFRELG ENTRY POINT IS CALLED BY ILRRLG AND ILRGOS01 TO FREE THE LGE SPACE AND MAKE THE LGID AVAILABLE (BY PUTTING ITS ASSOCIATED LGVTE ON THE AVAILABLE QUEUE) FOR THE LOGICAL GROUP BEING RELEASED. THIS ENTRY POINT IS REQUIRED BECAUSE GLOBAL LOCKS ARE REQUIRED TO PERFORM THESE FUNCTIONS. ILRRLG IS IN PAGEABLE LPA AND CANNOT HOLD GLOBAL SPIN LOCKS.			
02 THE ASM CLASS LOCK OF THE ADDRESS SPACE SERIALIZES THE QUEUE OF LGES BASED IN THE ASMHD.			
03 THE LGE QUEUE IS SEARCHED FOR THE INPUT LGE WHICH IS THEN DEQUEUED.			
04 THE ASM CLASS LOCK IS NO LONGER REQUIRED AND THE SALLOC IS REQUIRED TO SERIALIZE THE FREEING OF THE LGVTE AND THE CALLING OF FREEMAIN.			

Notes	Routine	Label	Ref
05 THE LGVTE IS FREED BY QUEUEING IT TO THE LGVTE AVAILABLE QUEUE IN THE LGVT HEADER AND PLACING THE LGID IN THE LGVTE.			
06 THE SPACE USED FOR THE LGE IS FREED VIA THE GLOBAL BRANCH ENTRY POINT TO FREEMAIN.	IEAVGM00	GLBRANCH	
07 THE SALLOC LOCK IS FREED BEFORE RETURNING.			

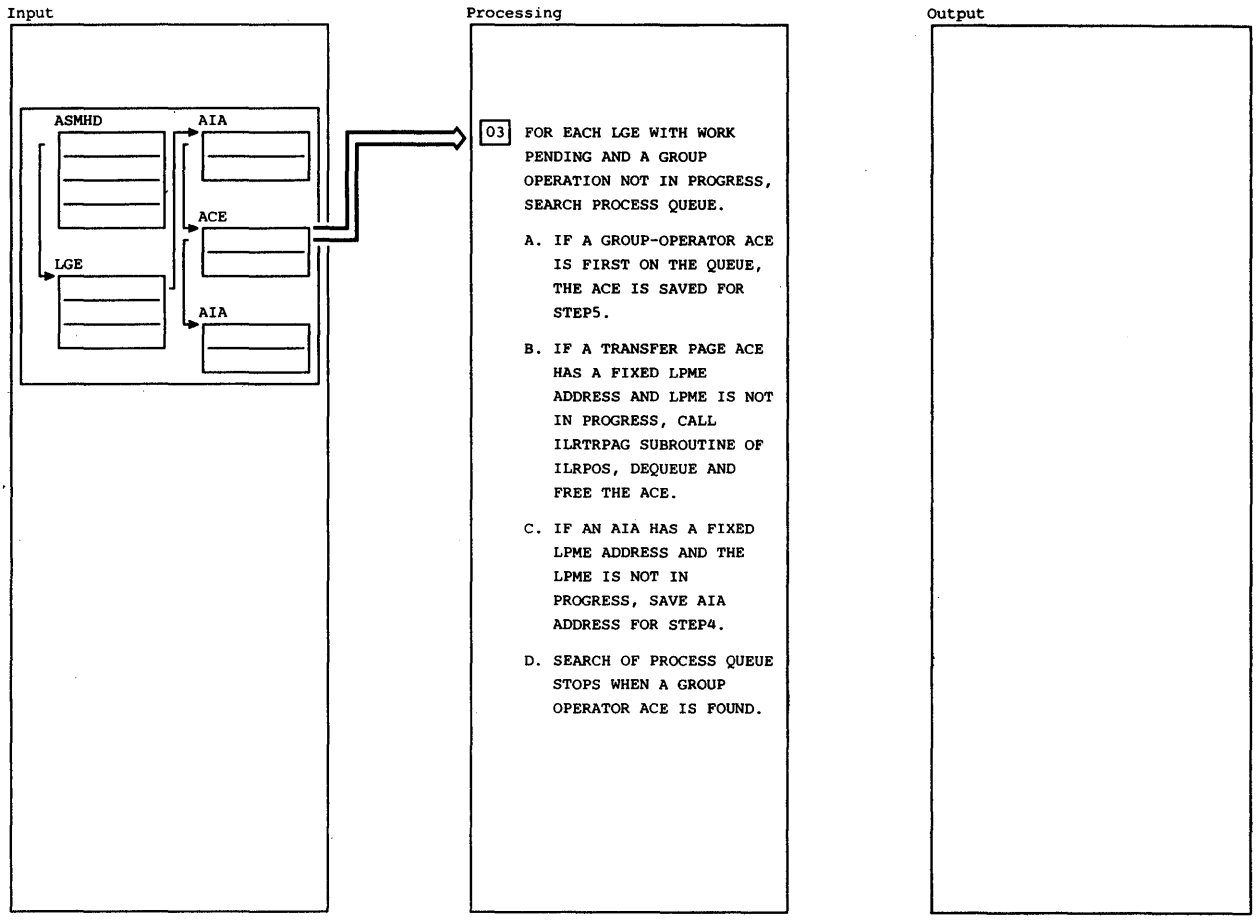
Diagram 25.11.1 ILRFRELG (Part 1 of 1)



Notes	Routine	Label	Ref
01 THE SRB CONTROLLER (ILRSRBC) ALWAYS RECEIVES CONTROL FROM THE DISPATCHER AS THE RESULT OF A SCHEDULE BY ILRGOS OR ILRVIOC. EACH LGE PROCESS QUEUE FOR THE CURRENT ADDRESS SPACE IS SEARCHED FOR WORK THAT CAN BE STARTED. PAGE OPERATIONS ARE STARTED BY CALLING THE RESTART ENTRY POINT (ILRESTRT) OF ILRPOS. GROUP OPERATIONS, EXCEPT FOR RELEASE LG AND DEACTIVATE, ARE STARTED BY POSTING THE ECB WAITED ON BY ILRGOS. FOR RELEASE LG, ILRRLG IS CALLED. DEACTIVATE IS PROCESSED BY SRB CONTROLLER.			
02 THE ASM LOCK FOR THE CURRENT ADDRESS SPACE IS OBTAINED AFTER DOING A SETFRR TO ESTABLISH THE RECOVERY ENVIRONMENT. ILRSRB01 RECOVERY ROUTINE HANDLES ERRORS OCCURRING IN ILRSRBC. THE SRB IS MADE AVAILABLE REUSE. EACH LGE QUEUED TO THE ASMHD THAT HAS WORK PENDING AND NO GROUP OPERATION IN PROGRESS IS PROCESSED IN STEPS 3 AND 4.			

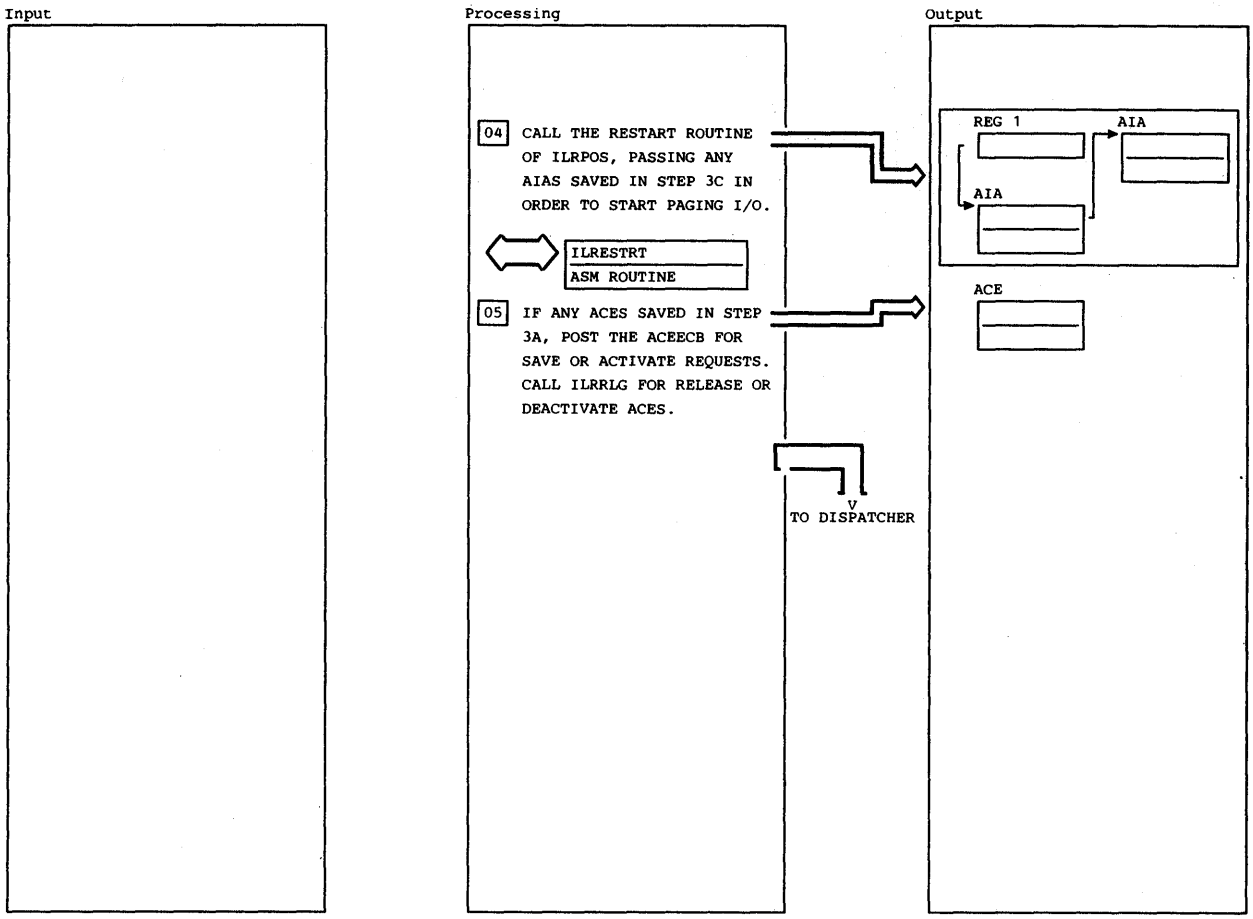
Notes	Routine	Label	Ref

Diagram 25.12 ILRSRBC (Part 1 of 3)



Notes	Routine	Label	Ref
03 THE PROCESS QUEUE IS SEARCHED FOR EACH LGE QUEUED TO THE ASMHD OF THE CURRENT ADDRESS SPACE. IF NO WORK IS PENDING OR A GROUP OPERATION IS IN PROGRESS, THE LGE IS SKIPPED AS THERE IS NO STARTABLE WORK.			
A. THE ACE IS SAVED UNTIL THE ASM LOCK IS RELEASED. GROUP OPERATORS ARE PAGEABLE AND MUST NOT BE CALLED WHILE HOLDING THE ASM LOCK.			
B. TRANSFER PAGE REQUESTS MAY BE PROCESSED IMMEDIATELY WITH THE LOCK HELD.	ILRPOS	ILRTRANS	
C. AIAS TO BE PROCESSED ARE SAVED FOR A SINGLE CALL TO ILRESTRT. THIS IS DONE BECAUSE RESTART MUST BE ENTERED WITHOUT THE ASM LOCK. IF I/O IS PENDING FOR A TRP ACE, THE IN-PROGRESS FLAG IS SET. IF LPME IN-PROGRESS, ACE OVERRIDE FLAG OVERRIDES THE IN-PROGRESS FLAG.			
D. THIS ACTION ALLOWS FULL SERIALIZATION OF THE LOGICAL GROUP, ASSURING THAT OPERATIONS ARE PERFORMED IN THE ORDER RECEIVED.			

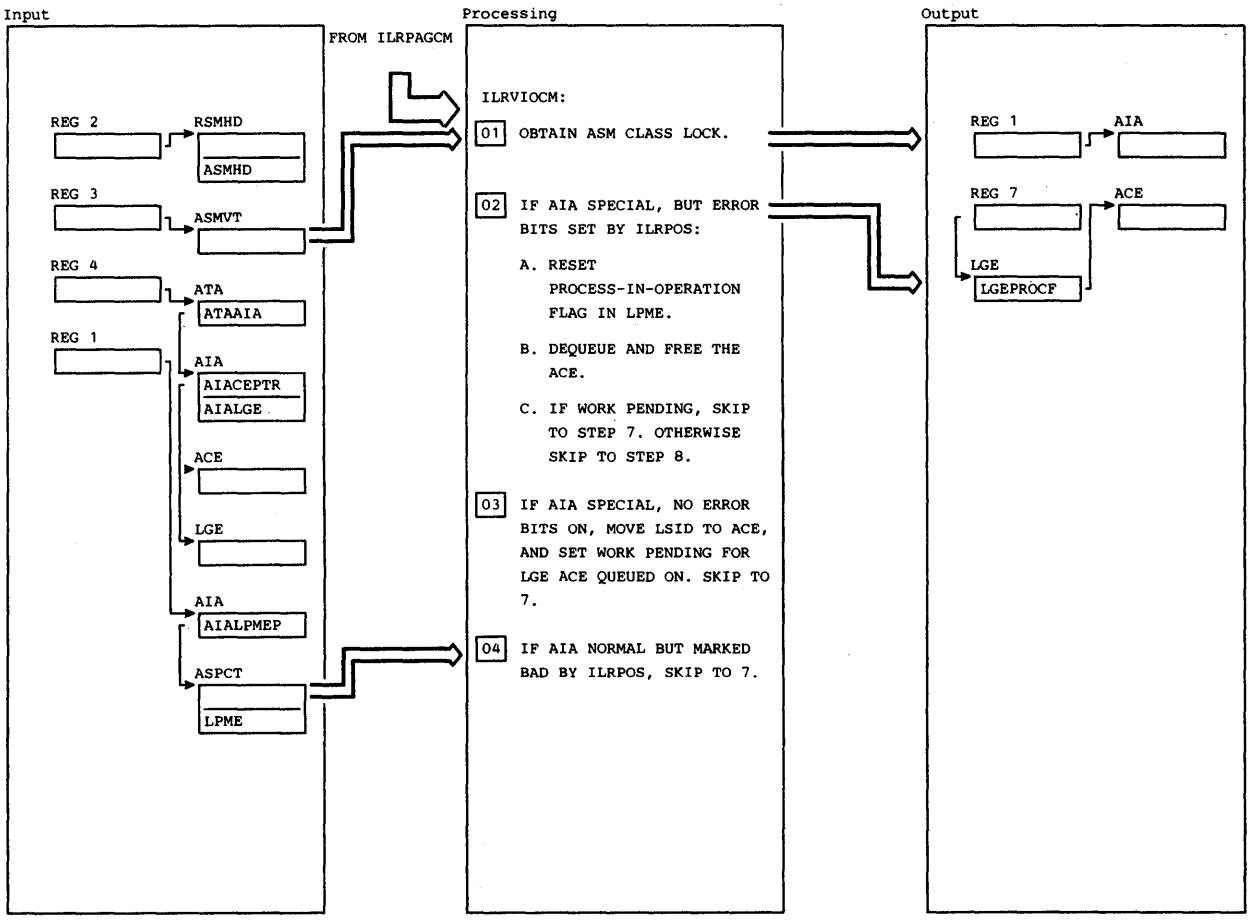
Diagram 25.12 ILRSRBC (Part 2 of 3)



Notes	Routine	Label	Ref
04 THE RESTART ROUTINE WILL PERFORM FINAL AIA PROCESSING BEFORE PASSING THE AIAS TO ILRQIOE TO INITIATE PAGING I/O.	ILRPOS	ILRESTRT	
05 GROUP OPERATIONS READY TO BE STARTED ARE STARTED BY POSTING THE ECBS ILRGOS IS WAITING ON. ILRGOS THEN STARTS THE OPERATION BY CALLING THE GROUP OPERATOR. IF THE ACE IS FOR A RELEASE LG REQUEST, THE GROUP OPERATOR IS CALLED. DEACTIVATE ACES ARE PROCESSED ONLY BY SRB CONTROLLER. THEY WERE CREATED BY ILRJTERM DURING JOB DELETION PROCESSING.	IEAVSY50 ILRRLG	IEAOPT02 ILRRLG	

Notes	Routine	Label	Ref

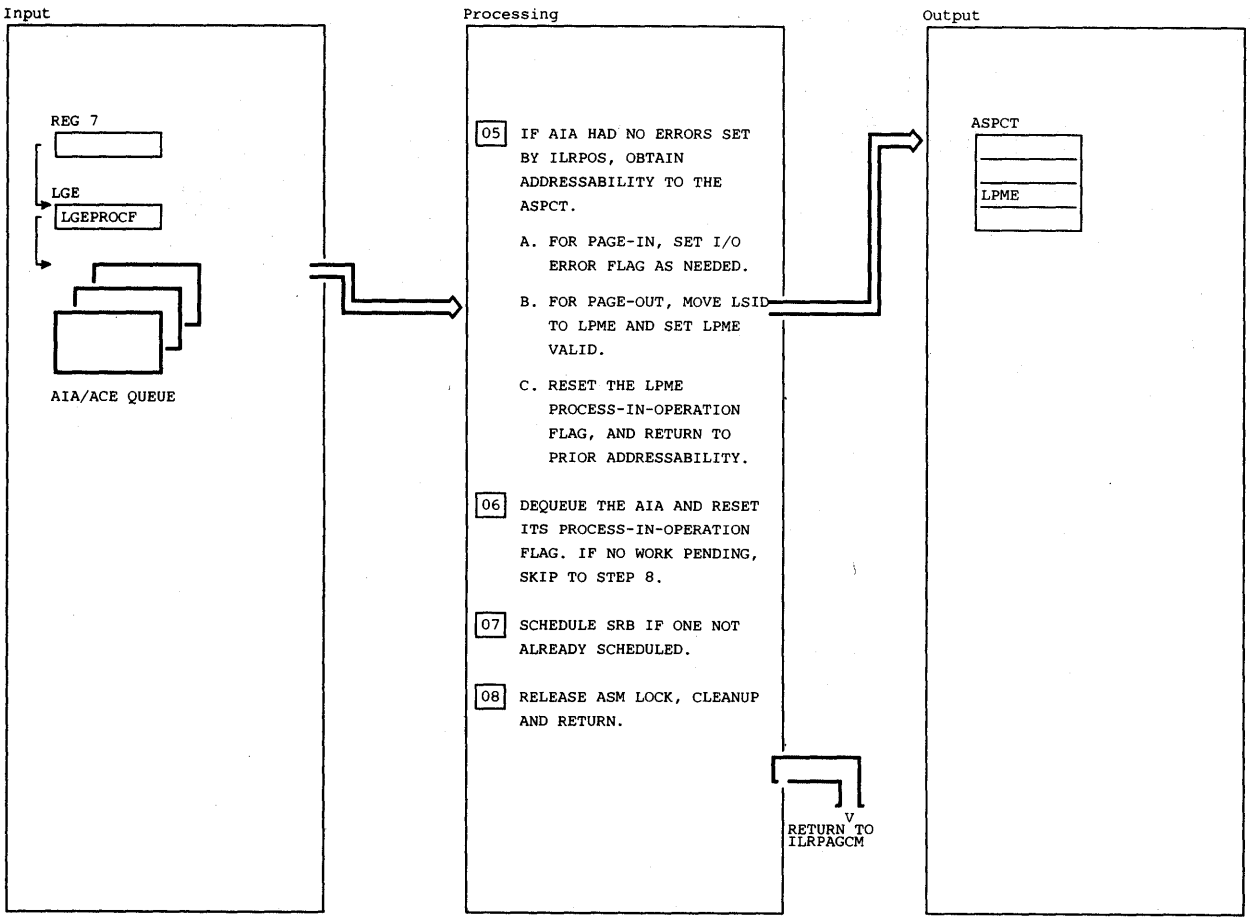
Diagram 25.12 ILRSRBC (Part 3 of 3)



Notes	Routine	Label	Ref
01 VIO COMPLETION ALWAYS RECEIVES CONTROL FROM PAGE COMPLETION. ONE AIA IS RECEIVED AS INPUT. THE AIA IS PROCESSED, DEQUEUED FROM ITS PROCESS QUEUE, AND RETURNED TO PAGE COMPLETION. UPON ENTRY, ENTRY INFORMATION IS RECORDED IN ATA FOR USE DURING RECOVERY. ILRIOFRR RECOVERY ROUTINE HANDLES ERRORS OCCURRING IN ILRVIOCM. THE ASM CLASS LOCK IS OBTAINED. THE POINTER TO THE AIA IS FOUND IN THE ATA, AND THE ASCB POINTER IS FOUND IN RSMHD.			
02 THIS IS A SPECIAL CASE CREATED BY ILRPOS WHEN TRANSFER PAGE WAS REQUESTED, AND THE SOURCE LSID HAD NOT YET BEEN DETERMINED DUE TO A PAGE-OUT IN PROGRESS. THE 'SPECIAL' AIA PROCESSED HERE AND IN STEP 3 IS THAT PAGE OUT.			
A. OBTAIN ADDRESSABILITY TO THE ASPCT BY AN INTERNAL TRAS (TRANSFER ADDRESS SPACE) MACRO. RESET PROCESS-IN-OPERATION FLAG IN LPME AND TRAS BACK.			

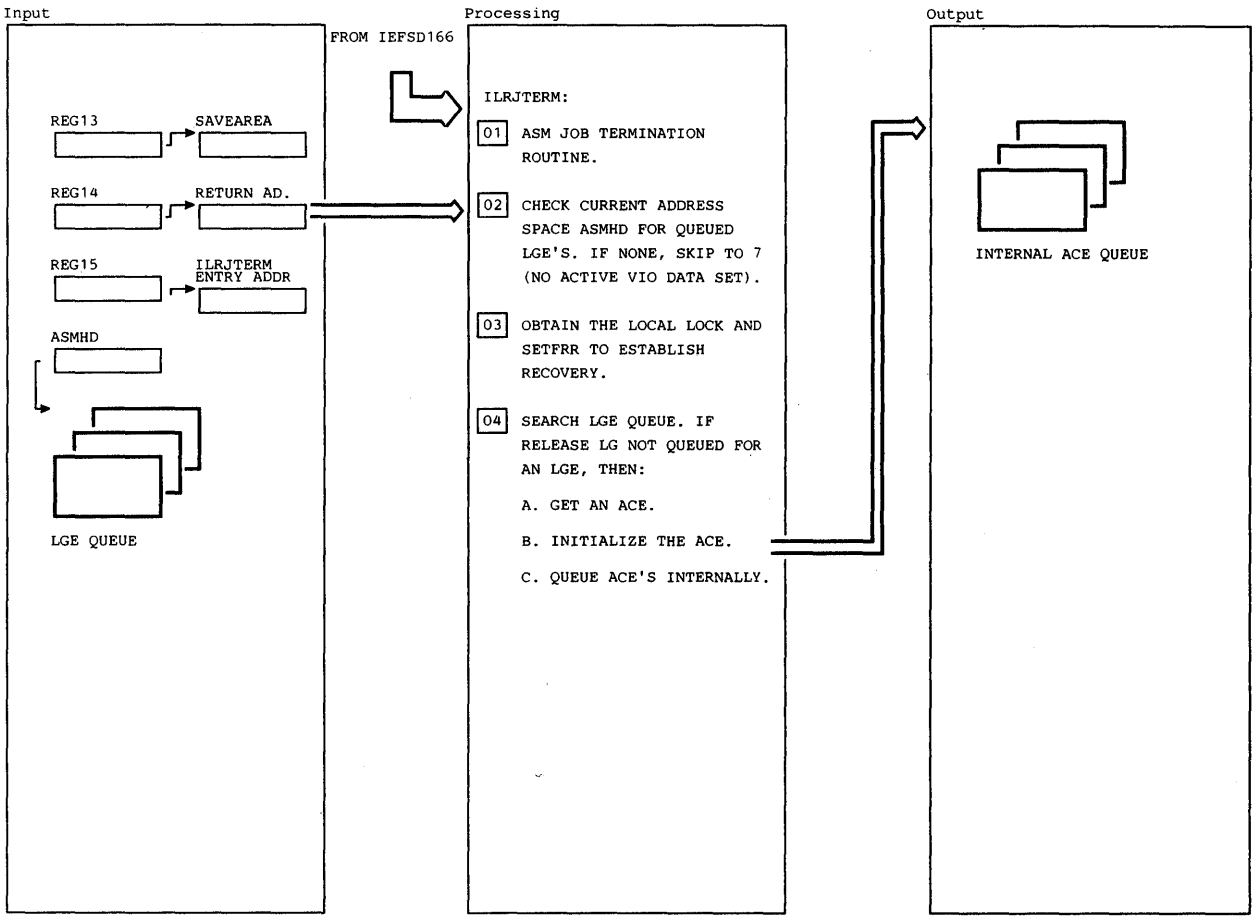
Notes	Routine	Label	Ref
B. DEQUEUE AND FREE THE ACE RELATED TO THE SPECIAL AIA WITH ERRORS.	ILRGMA		
03 FOR A 'SPECIAL' AIA WITH NO ERROR BITS ON, MOVE LSID FROM AIA TO ACE, SET WORK PENDING FLAG IN LGE AND SCHEDULE SRB FOR ILRSRBC.			
04 IF AIA NORMAL BUT ERROR BITS ON, DEQUEUE IT AND SCHEDULE SRB FOR ILRSRBC IF REQUIRED.			

Diagram 25.13 ILRVIOCM (Part 1 of 2)



Notes	Routine	Label	Ref
05 FOR NORMAL AIA WITH NO ERRORS: ADDRESSABILITY TO THE ASPCT IS OBTAINED BY AN INTERNAL TRAS (TRANSFER ADDRESS SPACE) MACRO THAT LOADS THE SEGMENT TABLE ORIGIN ADDRESS FOR THE ADDRESS SPACE CONTAINING THE ASPCT. A. PAGE-IN AIA: IF PERMANENT READ ERRORS OCCUR, THE LPME IS SO FLAGGED TO GIVE TRUE ERROR CODES FOR ANY FUTURE REQUESTS FOR THIS PAGE. B. PAGE-OUT AIA: SINCE THIS ROUTINE DOES NOT RECEIVE PAGE-OUT AIAS WITH I/O ERRORS, THE LSID CAN SIMPLY BE MOVED FROM THE AIA TO THE LPME. C. IN EITHER CASE, THE LPME PROCESS-IN-OPERATION FLAG IS RESET AND ADDRESSABILITY TO THE ADDRESS SPACE AT ENTRY IS RESTORED.			
06 THE AIA IS DEQUEUED FROM ITS LGE PROCESS QUEUE BECAUSE ONCE IT IS RETURNED TO I/O CONTROL (WHO RETURNS IT TO RSM) RSM WILL FREE THE PCB/AIA FOR REUSE. THE AIA PROCESS-IN-OPERATION FLAG IS RESET.			
07 IF ANY WORK IS PENDING FOR THE LGE PROCESSED SRB CONTROLLER IS SCHEDULED TO ATTEMPT TO START THE WORK.	ILRSRBC	ILRSRBC	
08 THE ASM LOCK IS RELEASED, AND THE INPUT AIA IS RETURNED TO ILRPAGCM VIA A POINTER IN THE ATA.	ILRPAGCM	ILRPAGCM	

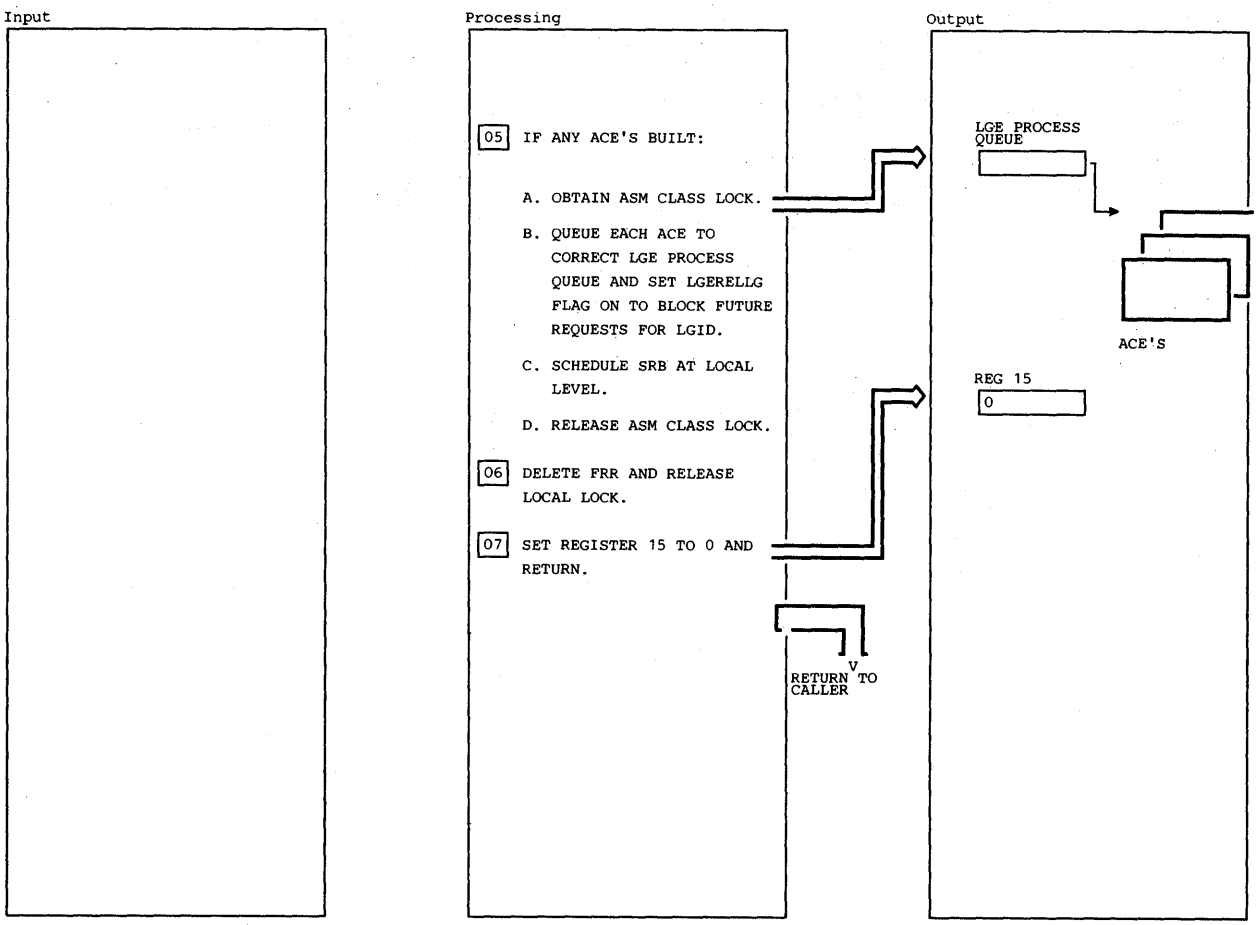
Diagram 25.13 ILRVIOCM (Part 2 of 2)



Notes	Routine	Label	Ref
01 JOB TERMINATION PROCESSING RECEIVES CONTROL FROM THE INITIATOR JOB DELETION MODULE (IEFSD166) WHICH CALLS THIS ROUTINE ON ALL JOB TERMINATIONS.			
02 NO LOCKS HELD,KEY0, SUPERVISOR STATE. IHAPSA CONTAINS A POINTER TO THE ASCB, WHICH IN TURN CONTAINS A POINTER TO THE RSMHD. THE RSMHD CONTAINS THE ASMHD WHICH WILL LOCATE ANY QUEUED LGE'S. THE NORMAL SITUATION IS THAT NO LGE'S ARE QUEUED FROM ASMHD, AND CONTROL IS RETURNED IMMEDIATELY TO IEFSD166. (SKIP TO 7).			
03 IF AN LGE WAS FOUND, THEN THE LOCAL LOCK IS OBTAINED IN ORDER TO SERIALIZE THE LGE QUEUE AGAINST ASSIGN AND RELEASE LG, AND TO ALLOW SETFRR COVERAGE. ILRJTM01, ANOTHER ENTRY POINT, IS THE RECOVERY ROUTINE INDICATED ON THE SETFRR.			
04 IF RELEASE LG NOT QUEUED	ILRGMA		

Notes	Routine	Label	Ref
(LGERELLG='0'B), AN ACE IS REQUIRED VIA ILRGMA FROM THE ACE POOL IN ASMVT. THE ACE OPCODE, LGID, LGE PTR, ETC. FIELDS ARE INITIALIZED, AND THE ACE IS KEPT ON AN INTERNAL QUEUE FOR PROCESSING IN STEP 5.			

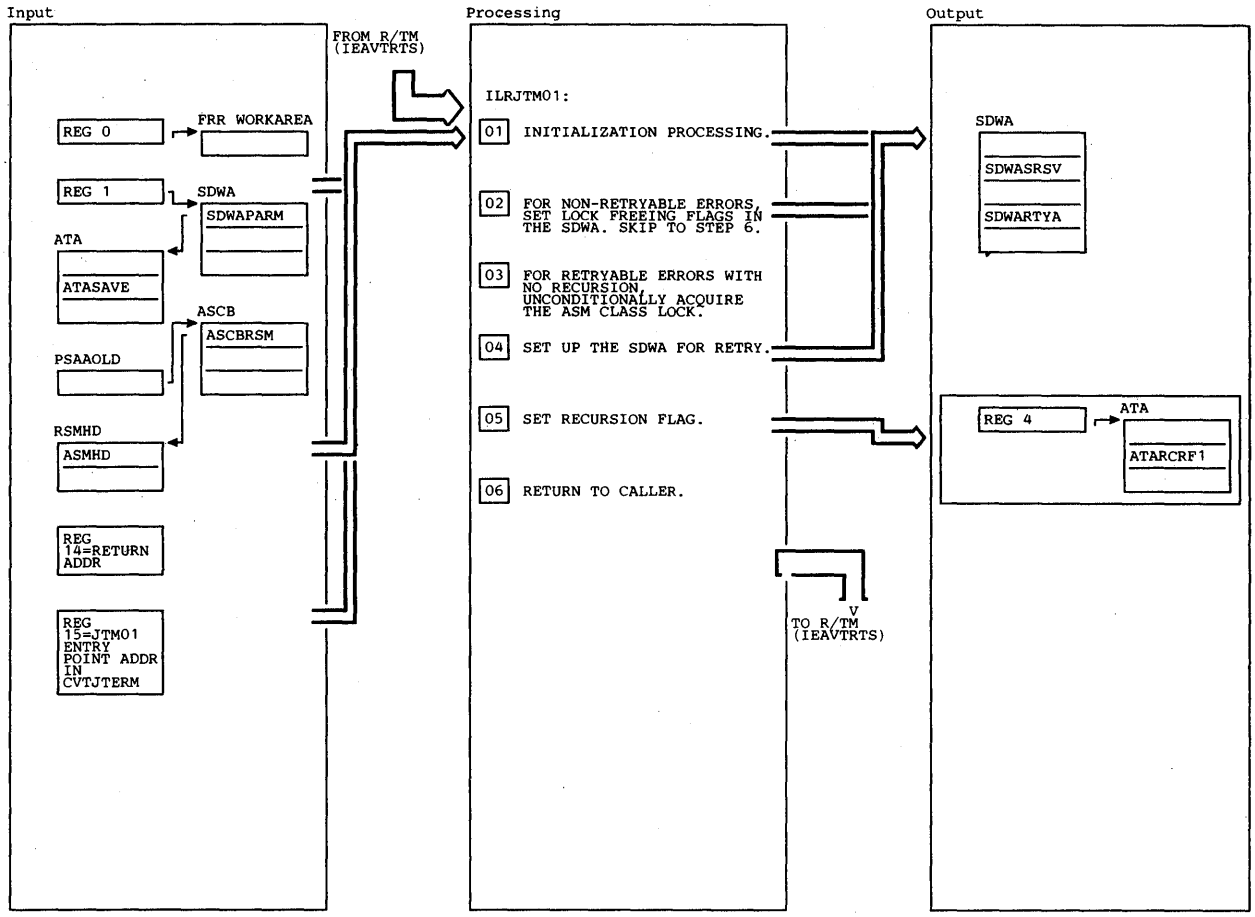
Diagram 25.14 ILRJTERM (Part 1 of 2)



Notes	Routine	Label	Ref
05 ONCE THE SEARCH IS COMPLETE, IF ANY ACE'S HAVE BEEN BUILT IN STEP 4, THE ASM CLASS LOCK IS OBTAINED, THE ACE'S ARE QUEUED ON TO THE CORRECT LGE PROCESS QUEUE AND LGERELLG IS SET. IF AN SRB WAS NOT YET SCHEDULED, (ASHSCHED='0'B) THEN IT IS SCHEDULED AND ASHSCHED IS SET. THE ASM CLASS LOCK IS RELEASED.			
06 FRR IS DELETED AND LOCAL LOCK RELEASED.			
07 RETURN CODE (REG 15) IS ALWAYS SET TO 0 FOR COMPLETENESS - CALLER DOES NOT NEED TO CHECK.			

Notes	Routine	Label	Ref

Diagram 25.14 ILRJTERM (Part 2 of 2)



Notes	Routine	Label	Ref
01 ILRJTM01 IS THE RECOVERY ROUTINE FOR ILRJTERM. PLACE NECESSARY POINTERS IN REGISTERS FOR THIS ROUTINES PROCESSING AND AS RETRY REGISTERS. COPY THE MODULE CSECT, AND RECOVERY ROUTINE IDS INTO THE SDWA.			
02 NON-RETRYABLE ERRORS ARE RESTART, DAT, AND RECURSION. PERCOLATION CAUSES MEMORY TERMINATION. INDICATE THAT RTM IS TO FREE THE LOCAL AND ASM LOCKS ACQUIRED BY ILRJTERM.	SETRP		
03 THE ASM LOCK IS REQUIRED AT ILRJTERM'S RETRY POINT.	SETLOCK		
04 REGISTER 13 IS LOADED FROM THE ATA. ILRJTM01 STORES ITS REGISTERS INTO THE SDWA (SDWASRSV) TO BE USED AS RETRY REGISTERS. ILRJTERM'S RETRY POINT, ILRCRTR1, IS PLACED IN THE SDWA. SDWA RECORDING AND RETRY INDICATORS ARE SET.	SETRP		
05 THE RECURSION FLAG, ATARCRF1, IS TURNED ON TO PREVENT A RETRY FOR ANOTHER ERROR IN ILRJTERM.			

Notes	Routine	Label	Ref

Diagram 25.14.1 ILRJTMO1 (Part 1 of 1)

VIO Group Operators

The VIO Group Operators perform all processing necessary to create, save, restore, and delete a logical group (LG) and its associated ASPCT. The three basic operators are SAVE, ACTIVATE, and RELEASE. Two other routines that assist the operators are the Task Mode Release routine and the VSAM Interface routine.

The SAVE, ACTIVATE, and RELEASE operators execute in the address space to which an LG is assigned. The SAVE and ACTIVATE operators are invoked only by ILRGOS. The RELEASE operator can be invoked either by ILRGOS or by the SRB Controller.

Task Mode Release processing occurs in the Master Scheduler address space as an extension of RELEASE processing when the LG being released has been previously saved. Task Mode Release gets control via a POST by RELEASE whenever a saved copy of an LG exists on SYS1.STGINDEX. This processing occurs asynchronously to processing in the address space owning the LG because VBP requires no return data. It also prevents unnecessary delays in normal job deletion processing.

The VSAM Interface routine is a service routine, used by SAVE, ACTIVATE, and Task Mode Release to access the SYS1.STGINDEX data set. This data set is used to save copies of ASPCTs for journaled logical groups.

Note that a fourth group operation can be requested by VBP. This is the ASSIGN LG operation. Processing of this request occurs within ILRGOS, as described in Chapter 3, "VIO Control".

SAVE Operator

The SAVE Operator saves active VIO ASPCTs on SYS1.STGINDEX. ILRGOS passes control to the SAVE Operator (ILRSAV) with an ACE as input. The ACE contains either an 'S' symbol or an LGID, and a pointer to the LGE.

If the input ACE contains an LGID, ILRSAV is processing a previously unsaved logical group. ILRSAV flags each valid LPME as saved and increases the saved slot counter in the ASPCT. ILRSAV then calls ILRVSAMI to write the ASPCT to SYS1.STGINDEX. After ILRVSAMI returns, ILRSAV flags the ASPCT as saved, copies the 'S' symbol (assigned by ILRGOS) from the ASPCT to the ACE, and returns to ILRGOS.

If the input ACE contains an 'S' symbol, ILRSAV is processing a previously saved logical group. ILRSAV calls ILRVSAMI to retrieve and erase the previously saved copy of the ASPCT. After ILRVSAMI returns, ILRSAV flags each valid LPME as saved, frees the unneeded storage for the old ASPCT, and updates the appropriate counters. ILRSAV then calls ILRVSAMI to write the ASPCT to SYS1.STGINDEX. After ILRVSAMI returns, ILRSAV flags the ASPCT as saved and flags it as having no slots released after the save. ILRSAV then copies the 'S' symbol (assigned by ILRGOS) from the ASPCT to the ACE and returns to ILRGOS.

ACTIVATE Operator

The ACTIVATE Operator (ILRACT) retrieves a saved ASPCT from SYS1.STGINDEX and rebuilds it in the current address space's LSQA.

ILRGOS passes control to ILRACT with an ACE as input. The input ACE contains 'S' symbol and a pointer to the newly-created LGE. During an ACTIVATE request, there is never an active ASPCT for the VIO data set being activated.

ILRACT calls ILRVSAMI to retrieve the ASPCT from SYS1.STGINDEX. After ILRVSAMI returns, ILRACT copies the new LGN from the LGE (both built by ILRGOS) into the ASPCT, copies the retrieved ASPCT from I/O buffers to LSQA storage, frees the I/O buffers, stores the LSQA address of the ASPCT in the LGE, and returns to ILRGOS.

RELEASE Operator

The RELEASE Operator (ILRRLG), along with the Task Mode Release Operator (described in the next section), releases paging slots back to the system and erases saved ASPCTs from SYS1.STGINDEX. ILRRLG posts Task Mode Release only if the LG being released has been previously saved.

ILRGOS or ILRSRBC passes control to ILRRLG with an ACE as input. The ACE contains either an 'S' symbol or an LGID, and, if the VIO data set is active, a pointer to the LGE.

If the ACE contains an LGID, the data set is active. ILRRLG releases valid LPMEs frees the LSQA storage used for the ASPCT, and calls ILRFRELG (entry point of ILRGOS) to free the LGE storage and mark the LGVTE as available. ILRRLG also updates the appropriate slot counters, then returns control to either ILRGOS or ILRSRBC.

If the ACE contains an 'S' symbol, the ASPCT has been saved and the data set may or may not be active. If the ASPCT is not active, ILRRLG sets the inactive flag in the input ACE. In either case, ILRSAV adds the ACE to the head of the release queue in the ASMVT, issues a POST to start Task Mode Release processing, and updates the

appropriate slot counters. If the ASPCT is active, ILRRLG calls ILRFRELG (entry point of ILRGOS) to free the LGE storage and mark the LGVTE available.

Task Mode Release Operator

The Task Mode Release Operator (ILRTMRLG) has two responsibilities: to call Task Mode Initialization (ILRTMI00) to complete ASM initialization, and to complete the release processing for a saved ASPCT.

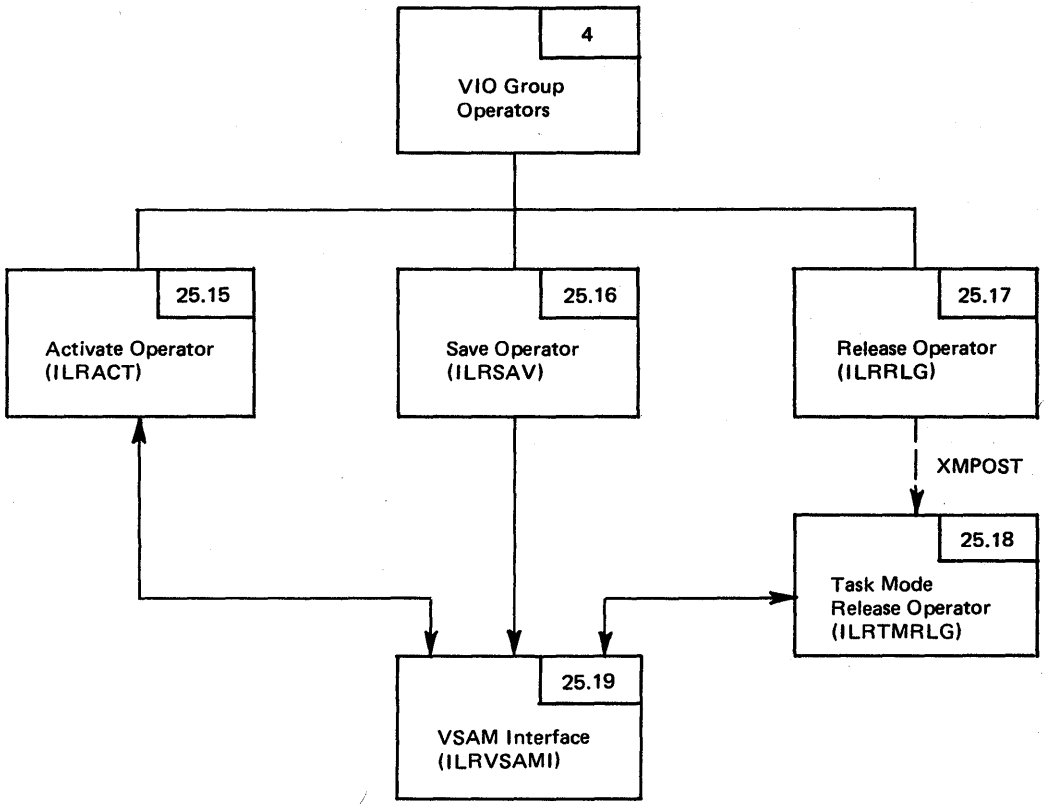
ILRTMRLG runs under the ASM TCB in the Master Scheduler address space established during system initialization. The Master Scheduler attaches ILRTMRLG. ILRTMRLG establishes the recovery environment via an ESTAE, initializes pointer, then loads the Task Mode Initialization routine (ILRTMI00). Upon return, ILRTMRLG deletes ILRTMI00, posts Master Scheduler Initialization, and issues a wait on the ECB in the ASMVT.

ILRRLG posts this ECB to start Task Mode Release processing. ILRTMRLG processes a queue of ACEs, each representing a LG whose ASPCT is saved on SYS1.STGINDEX. ILRTMRLG calls ILRVSAMI to retrieve and erase the saved copy of the ASPCT on SYS1.STGINDEX. After ILRVSAMI returns ILRTMRLG frees all slots assigned in the ASPCT. After it has processed all the ACEs, ILRTMRLG waits on its ECB for more work.

VSAM Interface

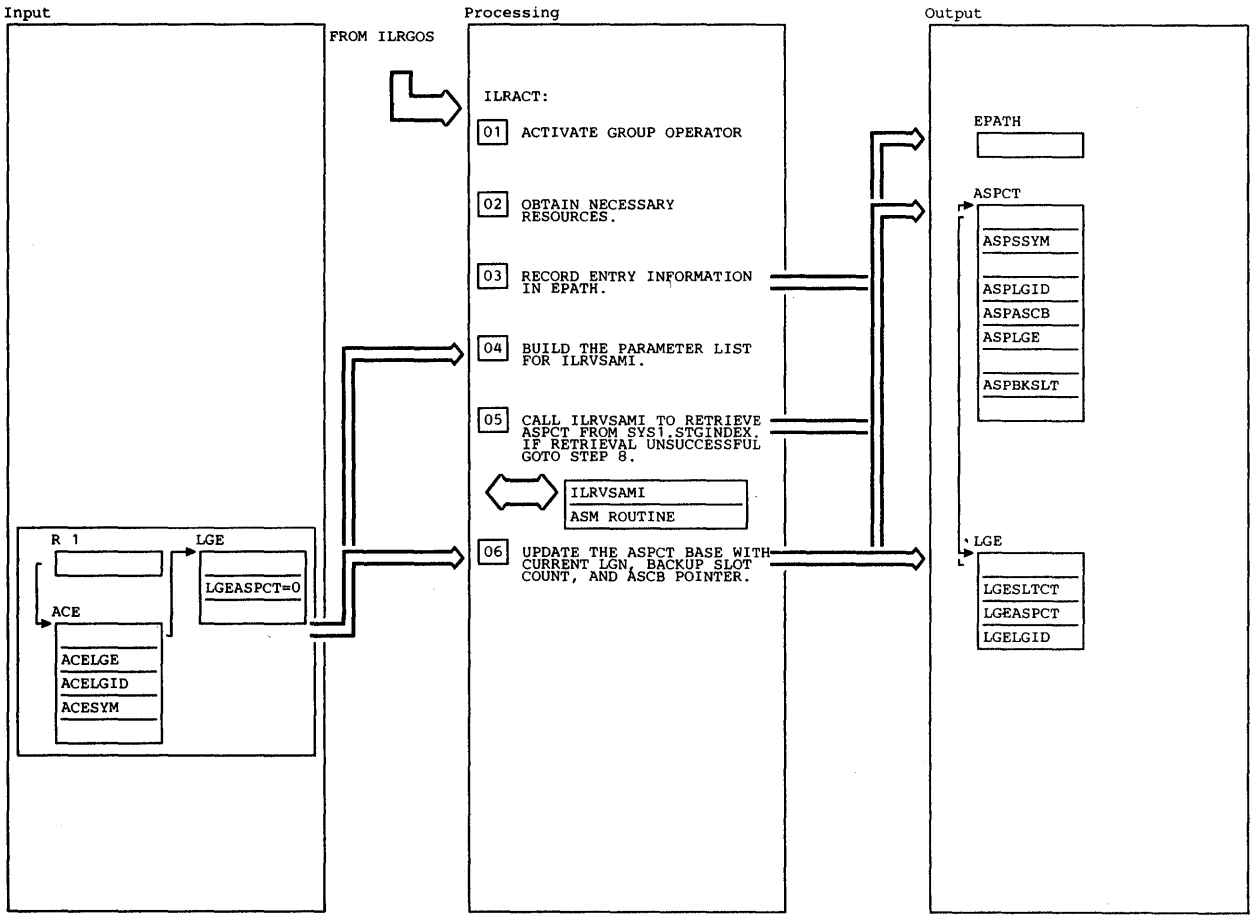
The VSAM Interface routine (ILRVSAMI) supplies all the necessary functions the VIO Group operators require for access to SYS1.STGINDEX. There are separate subroutines for each function required by each operator. The functions are:

- GETASPCT — retrieve an ASPCT from SYS1.STGINDEX.
- PUTASPCT — write an ASPCT to SYS1.STGINDEX.
- RETERASE — retrieve and then erase an ASPCT from SYS1.STGINDEX.



25.x. — Module
25.x.y. — Entry point in module 25.x.

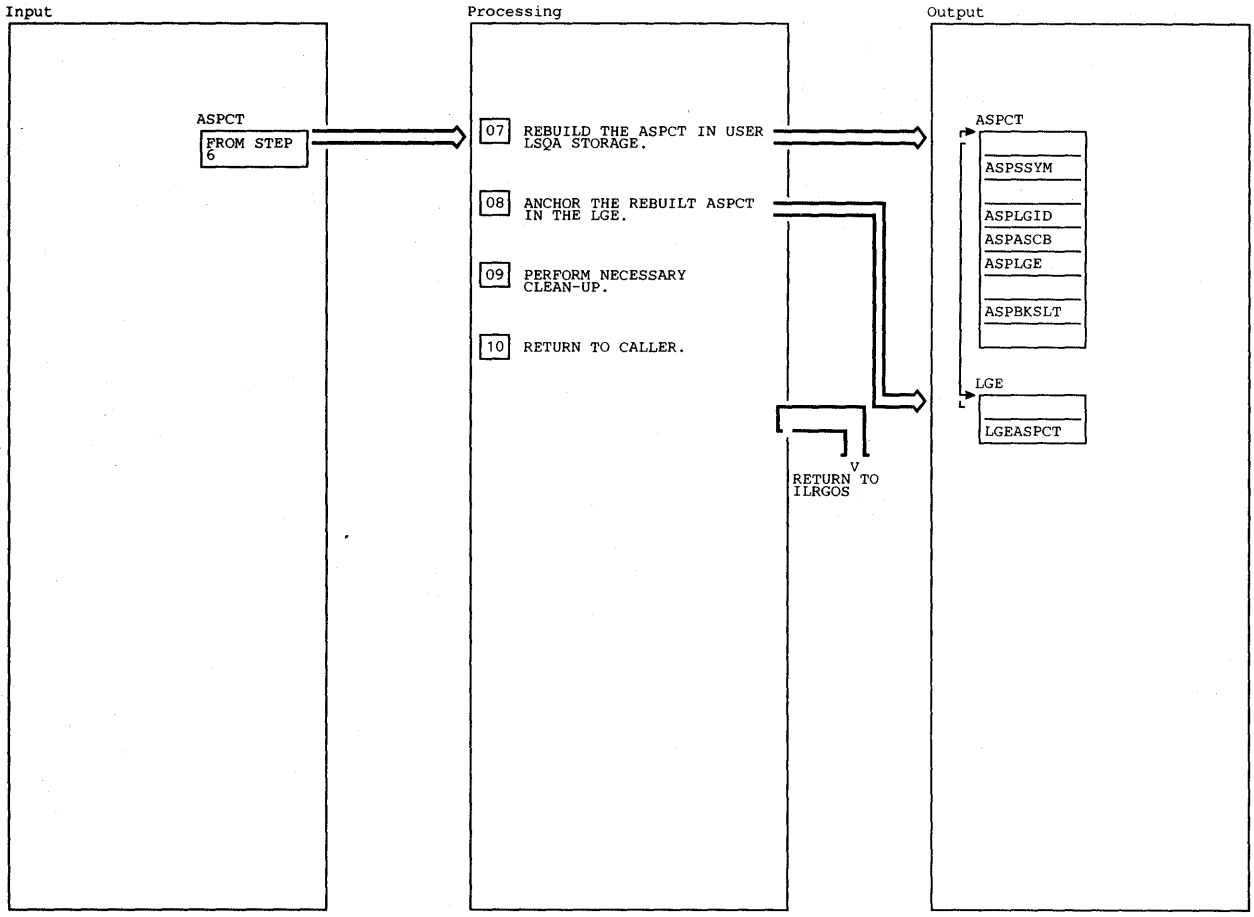
Figure 2-60. VIO Group Operators Overview



Notes	Routine	Label	Ref
01 THE ACTIVATE GROUP OPERATOR (ILRACT) IS RESPONSIBLE FOR RETRIEVING A SAVED ASPCT FROM SYS1.STGINDEX AND REBUILDING IT IN THE CURRENT ADDRESS SPACE LSQA. ILRGMA IS USED TO OBTAIN A WORKAREA FROM ASM'S POOL OF WORKAREA. FOR RECOVERY PURPOSES, ILRGOS01 RECOVERY ROUTINE HANDLES ERRORS OCCURRING IN ILRACT AND ITS PATH THROUGH ILRVSAMI.			
02 IF ILRGMA IS UNSUCCESSFUL IN OBTAINING A WORKAREA, THE RETURN CODE IS SET TO 28, AND ILRACT RETURNS TO ILRGOS.	ILRGMA		
03 ILRACT'S CSECT IDENTIFIER IS SET IN THE EPATH. THE POINTER TO THE WORKAREA IS STORED IN THE EPATH.			
04 STORE THE ADDRESS OF THE EPATH IN THE WORKAREA PARAMETER LIST FOR ILRVSAMI. STORE THE ADDRESS OF ACESYM IN THE PARAMETER LIST, AND SET THE REQUEST OP CODE TO A 01.			
05 CALL ILRVSAMI TO RETRIEVE THE ASPCT FROM SYS1.STGINDEX. IF ILRVSAMI WAS UNSUCCESSFUL, SAVE THE RETURN CODE IN THE WORKAREA AND SKIP TO STEP NO. 9.	ILRVSAMI	ILRVSAMI	
06 THE RETRIEVED ASPCT BASE IS NOW UPDATED. THE LGE (ASPLGID) IS COPIED FROM THE LGE(LGELGID). THE ASCB POINTER (ASPASCB) IS INITIALIZED FROM PSAAOLD. THE POINTER TO THE LGE(ASPLGE) IS INITIALIZED THE ADDRESS OF THE LGE(ACELGE). THE NUMBER OF SLOTS REQUIRED TO BACK THE VIO DATASET (ASPBKSLT) IS CALCULATED BY DIVIDING THE MAXIMUM RPN(ASPMAXPN) BY THE CURRENT VALUE OF ILRSLOTV.			

Notes	Routine	Label	Ref

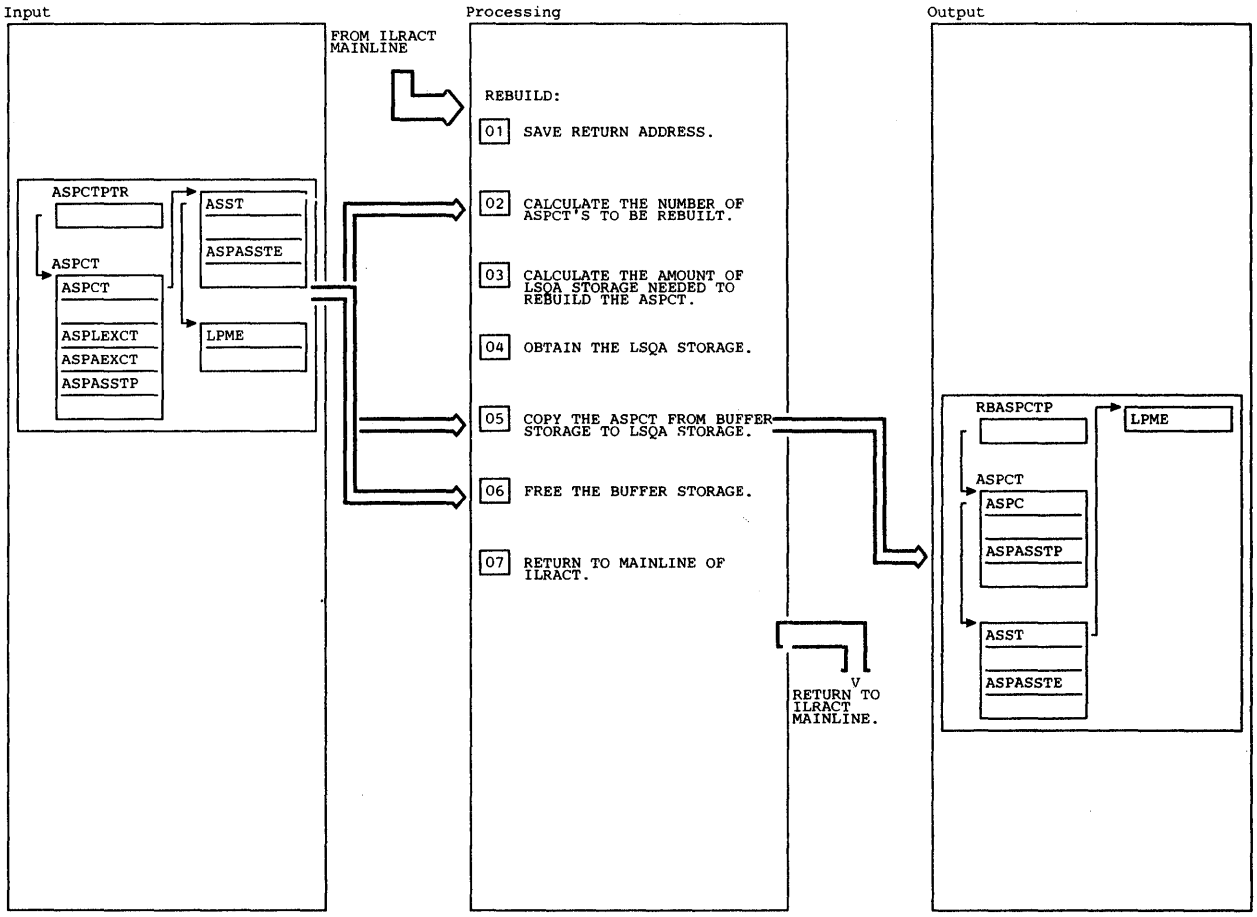
Diagram 25.15 ILRACT (Part 1 of 2)



Notes	Routine	Label	Ref
07 THE INTERNAL SUBROUTINE REBUILD IS CALLED TO COPY THE ASPCT IN BUFFER(S) STORAGE TO LSQA. A. SAVE THE RETURN CODE IN WORKAREA: IF IT WAS NON-ZERO SKIP TO STEP 9.		REBUILD	25.18.1
08 THE LSQA STORAGE IS ANCHORED IN THE LGE (LGEASPCT), AND THE NUMBER OF SLOTS ACTUALLY USED BY THE ASPCT(ASPSAVCT) IS COPIED TO THE LGE(LGESLTCT).			
09 SAVE THE INTERNAL WORKAREA RETURN CODE IN THE USERS REG 15. USE ILRGMA TO RETURN THE WORKAREA TO ITS POOL.	ILRGMA		
10 RESTORE REGISTERS AND RETURN TO ILRGOS.			

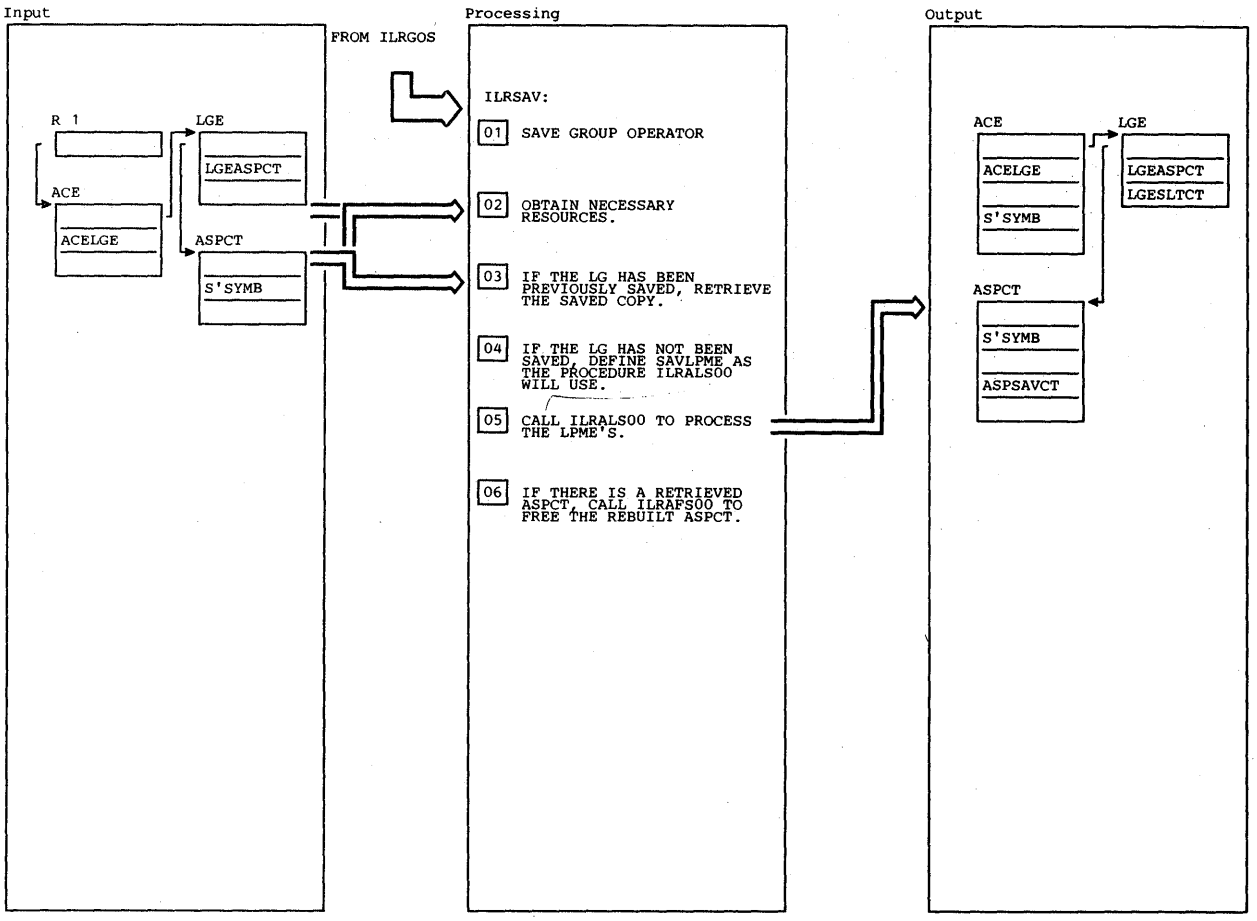
Notes	Routine	Label	Ref

Diagram 25.15 ILRACT (Part 2 of 2)



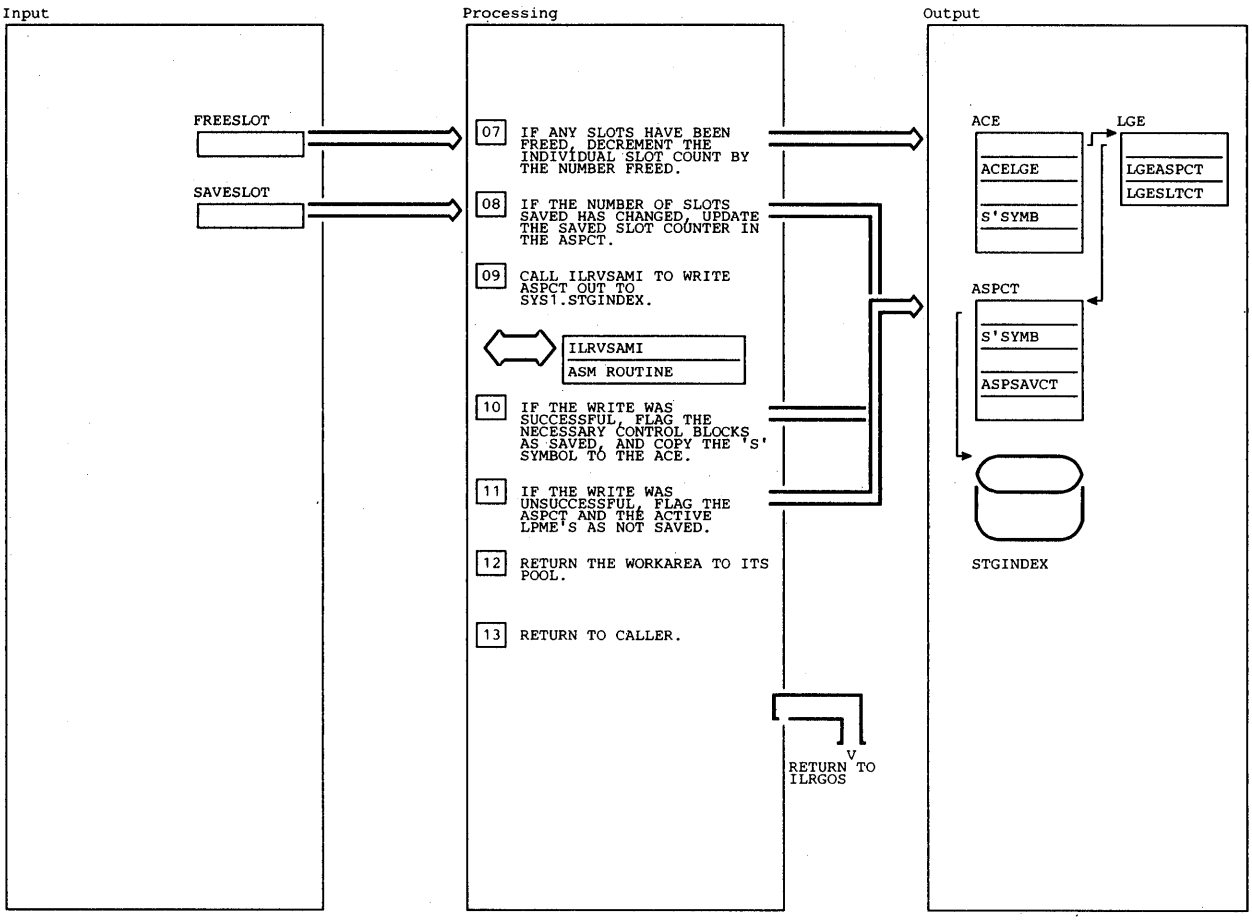
Notes	Routine	Label	Ref
01 SAVE REGISTER 14 IN WORKAREA.			
02 THE NUMBER OF ASPCT'S TO BE BUILT IS THE NUMBER OF ASST EXTENSIONS (ASPAEXCT) PLUS THE NUMBER OF LPME EXTENSIONS (ASPLEXCT) PLUS ONE FOR THE ASPCT BASE.			
03 THE AMOUNT OF LSQA SPACE NEEDED IS THE NUMBER OF ASPCT'S TO BE REBUILT TIMES THE LENGTH OF AN ASPCT.			
04 ISSUE A BRANCH ENTRY GETMAIN TO OBTAIN THE LSQA SPACE.	GETMAIN		
A. IF UNSUCCESSFUL IN OBTAINING THE LSQA SPACE, LOAD REG 1 WITH POINTER TO ASPCT BASE AND SET REG 0 TO LENGTH OF BUFFER. CALL ILRAFS00 TO FREE THE RETRIEVED ASPCT. IF ILRAFS00 WAS UNSUCCESSFUL ISSUE AN 087 ABEND. SET THE RETURN CODE TO 28, AND RETURN TO MAINLINE.	GETMAIN ILRAFS00 ABEND		
B. SAVE AND RECORD IN EPATH, POINTER TO STORAGE.			
05 COPY THE ASPCT BASE FROM THE BUFFER TO LSQA STORAGE. IF THERE ARE NO EXTENSIONS (ASPAEXCT=0) THEN SKIP TO STEP 6.			
A. UPDATE THE LSQA POINTER BY LENGTH OF AN ASPCT.			
B. STORE THE LSQA POINTER IN THE BASE ASPCT (ASPASSTP(I)).			
C. COPY THE ASST EXTENSION FROM BUFFER TO LSQA STORAGE.			
D. UPDATE THE LSQA POINTER BY LENGTH OF AN ASST EXTENSION.			
E. STORE LSQA POINTER IN ASST EXTENSION (ASPASSTE(I)).			
F. COPY THE LPME EXTENSION FROM BUFFER TO LSQA STORAGE.			
G. UPDATE THE LSQA POINTER BY LENGTH OF LPME EXTENSION.			
H. ANYMORE LPME EXTENSIONS (I=ASPLEXCT), THEN GOTO STEP E.			
I. ANYMORE ASST EXTENSIONS (I=ASPAEXCT), THEN GOTO STEP B.			
06 IF THERE ARE NO EXTENSIONS, ISSUE A FREEMAIN FOR THE ASPCT BASE BUFFER STORAGE. IF THERE ARE EXTENSIONS THE LOAD REG 1 WITH POINTER TO ASPCT BASE, SET REG 0 TO LENGTH OF BUFFER. CALL ILRAFS00 TO FREE THE BUFFER STORAGE. IF ILRAFS00 WAS UNSUCCESSFUL ISSUE AN 087 ABEND.	FREEMAIN ILRAFS00 ABEND		
07 RETURN TO MAINLINE ILRACT.			

Diagram 25.15.1 REBUILD (Part 1 of 1)



Notes	Routine	Label	Ref
01 THE SAVE GROUP OPERATOR (ILRSAV) IS RESPONSIBLE FOR SAVING VIO ASPCTS ON SYS1.STGINDEX. IF THE INPUT ACE CONTAINS AN LGID, ILRSAV WILL BE PROCESSING ONLY THE ACTIVE ASPCT. IF THE ACE CONTAINS AN 'S' SYMBOL, ILRSAV WILL BE PROCESSING THE ACTIVE AND THE PREVIOUSLY SAVED ASPCTS. FOR RECOVERY, ILRGOS01 RECOVERY ROUTINE HANDLES ERRORS OCCURRING IN ILRSAV AND ITS PATH THROUGH ILRVSAMI.			
02 USE ILRGMA TO OBTAIN A WORKAREA FROM ASM'S POOL. IF ILRGMA IS UNSUCCESSFUL, SET THE RETURN CODE TO 28, AND RETURN TO CALLER. SAVE THE POINTERS TO THE ASMHD FROM INPUT REG 2, THE ASMVT FROM INPUT REG 3, THE EPATH FROM INPUT REG 4, AND ACE FROM INPUT REG 1. GET POINTERS TO LGE FROM ACELGE, AND ASPCT FROM LGEASPCT. RECORD THE POINTER TO THE WORKAREA IN THE EPATH.	ILRGMA		
03 IF THE ASPCT IS FLAGGED AS HAVING BEEN SAVED (ASPSAVED=1), THEN CALL INTERNAL SUBROUTINE SAVEDASP TO RETRIEVE THE SAVED ASPCT.		SAVEDASP	25.15.1
04 IF THE LG HAS NOT BEEN SAVED, DEFINE SAVELPME AS THE PROCEDURE ILRALS00 WILL USE TO FLAG THE LPME AS SAVED.			
05 ZERO THE FREESLOT AND SAVESLOT COUNTERS. LOAD REG 0 WITH THE ADDRESS OF THE ACTIVE ASPCT BASE. CALL ILRALS00 TO PROCESS ALL ACTIVE LPME'S BY CALLING THE ROUTINE DEFINED TO IT IN STEP 2 (BY SAVEDASP) OR STEP 3.	ILRALS00		25.15.2 25.15.3
06 IF THERE IS A RETRIEVED ASPCT (RETASPCT.=0), LOAD REG 0 WITH THE LENGTH OF A VSAM BUFFER, LOAD REG 1 WITH ADDRESS OF THE RETRIEVED ASPCT BASE. CALL ILRAFS00 TO FREE THE BUFFERS. IF THERE IS A NON-ZERO RETURN CODE FROM ILRAFS00 ISSUE AN 087 ABEND.	ILRAFS00 ABEND		

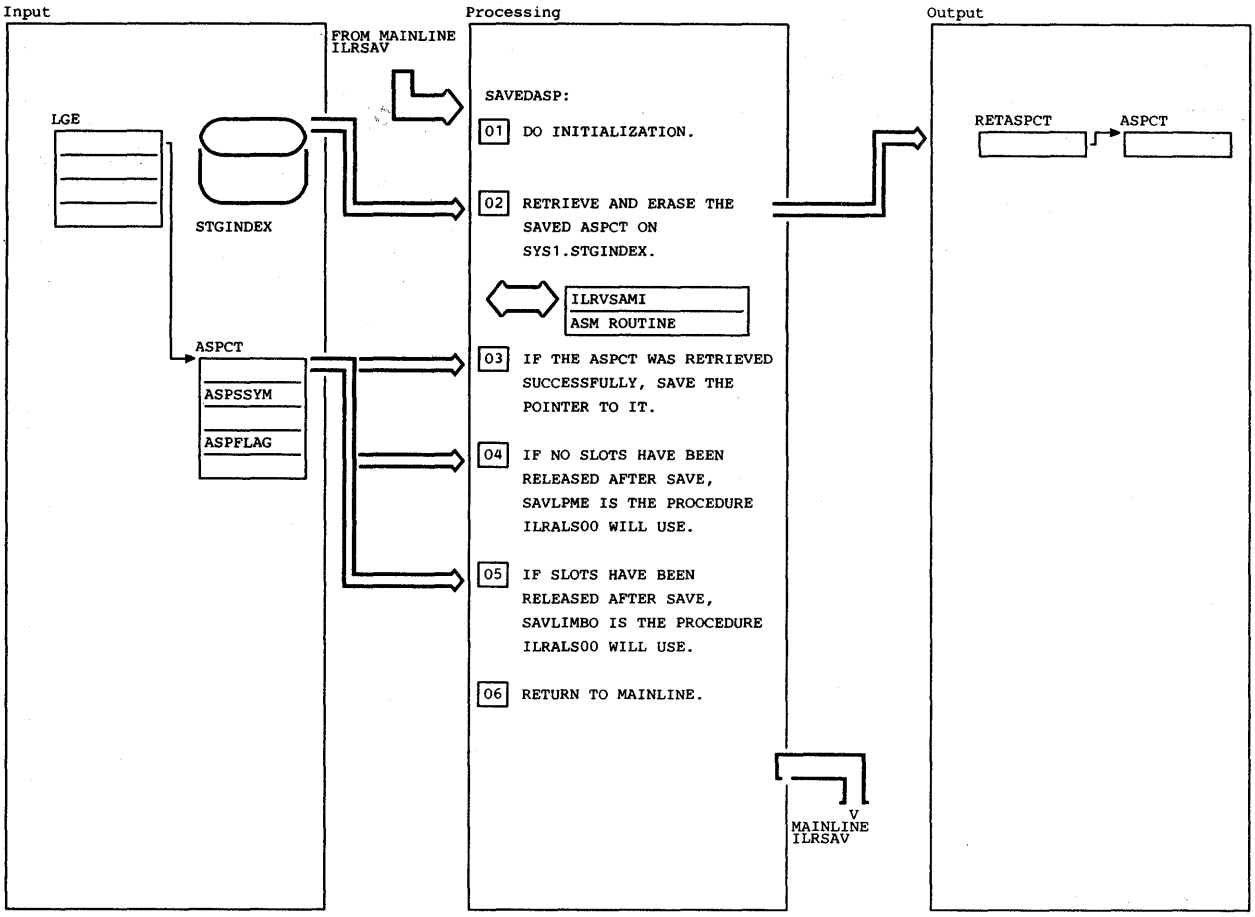
Diagram 25.16 ILRSAV (Part 1 of 2)



Notes	Routine	Label	Ref
07 IF THE FREESLOT COUNTER IS NON-ZERO DUE TO SAVLIMBO PROCESSING, THEN SUBTRACT THE FREESLOT COUNTER FROM THE LGESLTCT (ILRGOS WILL USE THIS VALUE TO UPDATE THE INDIVIDUAL SLOT COUNT IN THE ASCB.).			
08 IF THE SAVESLOT COUNTER IS NON-ZERO DUE TO SAVLIMBO PROCESSING, THEN UPDATE THE SAVED SLOT COUNT IN THE ASPCT (ASPSAVCT) BY THE SAVESLOT COUNTER.			
09 STORE THE POINTERS TO THE EPATH, AND THE ASPCT BASE IN THE WORKAREA PARAMETER LIST FOR ILRVSAMI. SET THE REQUEST OP-CODE TO 02. CALL ILRVSAMI TO WRITE THE ASPCT OUT TO STGINDEX.	ILRVSAMI	ILRVSAMI	
10 IF THERE IS A ZERO RETURN CODE FROM ILRVSAMI, FLAG THE ASPCT AS SAVED (ASPSAVED=1), AND AS HAVING NO SLOTS RELEASED AFTER SAVE (ASPSAVRP=0). COPY THE 'S' SYMBOL TO THE ACE (ACESYM=ASPSSYM).			
11 IF THERE IS A NON-ZERO RETURN CODE FROM ILRVSAMI, FLAG THE ASPCT AS NOT SAVED (ASPSAVED=0). DEFINE UNSAVLPM AS THE PROCEDURE ILRALS00. CALL ILRALS00 TO FLAG ALL ACTIVE LPME'S AS NOT SAVED. ZERO THE SAVE SLOT COUNTER (ASPSAVCT) IN THE ASPCT.	ILRALS00		25.15.4
12 USE ILRGMA TO RETURN TO RETURN THE WORKAREA TO ITS POOL.	ILRGMA		
13 RESTORE REGISTERS AND RETURN TO CALLER.			

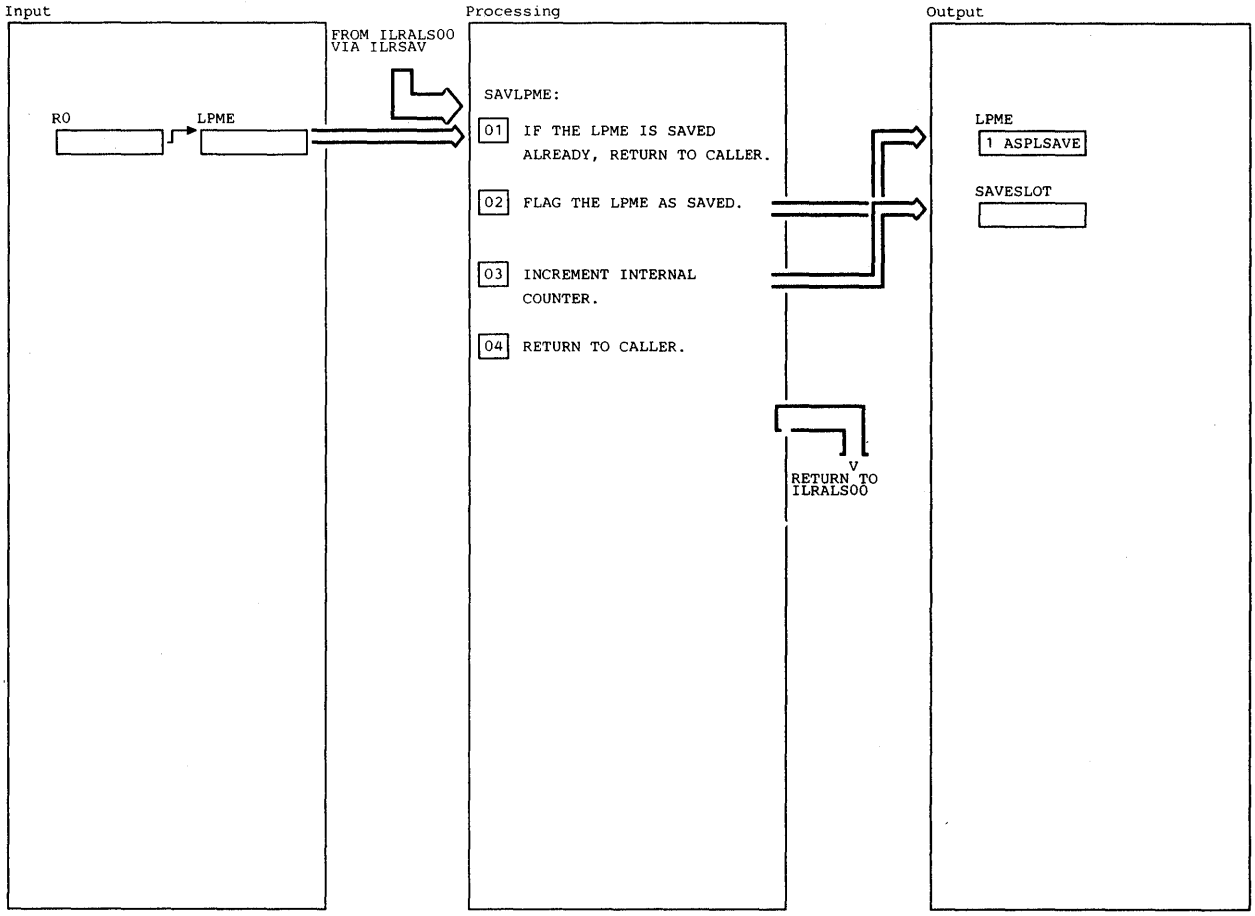
Notes	Routine	Label	Ref

Diagram 25.16 ILRSAV (Part 2 of 2)



Notes	Routine	Label	Ref
01 SAVE THE RETURN ADDRESS. GET THE ADDRESS OF THE 'S' SYMBOL (ASPSSYM) IN THE ASPCT.			
02 STORE POINTERS TO THE EPATH AND THE 'S' SYMBOL IN THE PARAMETER LIST FOR ILRVSAMI. SET THE REQUEST OP-CODE TO 03 (RETRIEVE AND ERASE). CALL ILRVSAMI TO RETRIEVE AND ERASE THE SAVED ASPCT ON SYS1.STGINDEX.	ILRVSAMI	ILRVSAMI	
03 IF THERE IS A ZERO RETURN CODE, SAVE THE POINTER TO THE RETRIEVED ASPCT IN THE WORKAREA (RETASPCT). IF THERE IS A NON-ZERO RETURN CODE FROM ILRVSAMI, TURN THE SAVED FLAG (ASPSAVED) OFF IN THE ASPCT.			
04 IF NO SLOTS HAVE BEEN RELEASED AFTER SAVE (ASPSAVRP=0), THEN DEFINE SAVLPME AS THE PROCEDURE ILRALS00 WILL USE TO FLAG THE LPME AS SAVED.			
05 IF SLOTS HAVE BEEN RELEASED AFTER SAVE (ASPSAVRP=1), THEN DEFINE SAVLIMBO AS THE PROCEDURE ILRALS00 WILL USE TO PROCESS THE SLOTS ACCORDING TO THE LPME FLAGS.			
06 RETURN TO MAINLINE.			

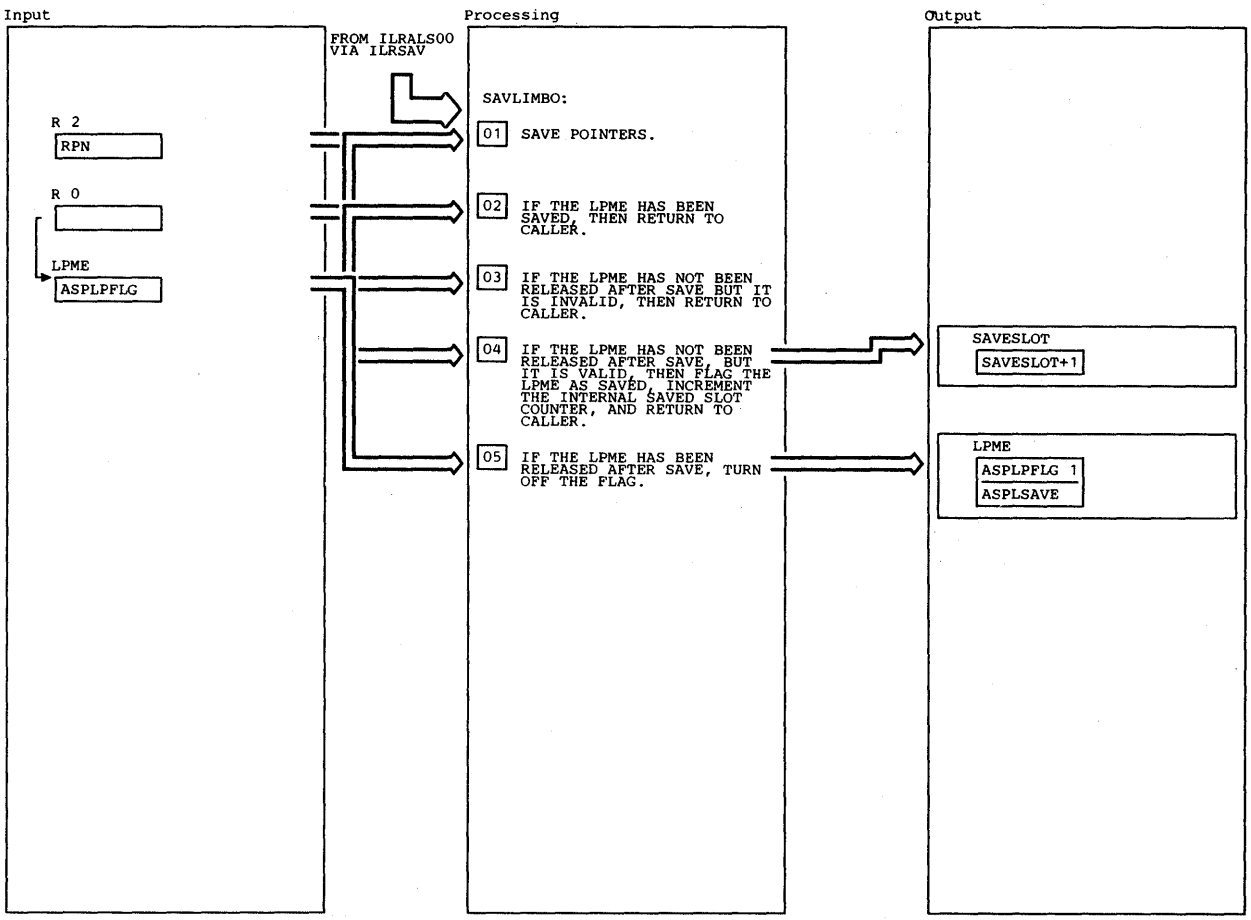
Diagram 25.16.1 SAVEDASP (Part 1 of 1)



Notes	Routine	Label	Ref
01 IF THE LPME IS FLAGGED AS SAVED (ASPLSAVE=1),THEN GOTO STEP 4.			
02 TURN THE SAVED FLAG (ASPLSAVE) ON IN THE LPME.			
03 INCREMENT THE SAVESLOT COUNTER BY ONE.			
04 RETURN TO CALLER.			

Notes	Routine	Label	Ref

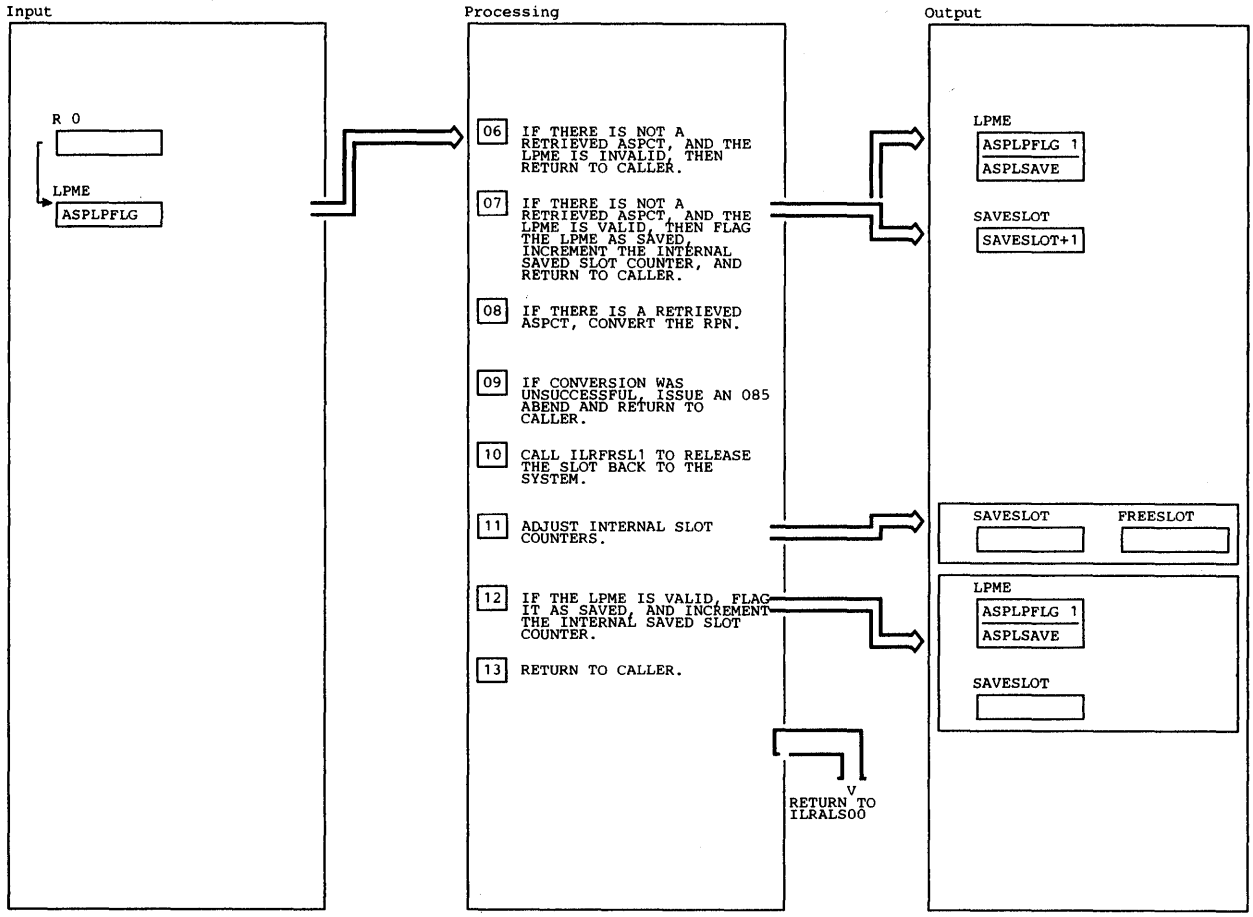
Diagram 25.16.2 SAVLPME (Part 1 of 1)



Notes	Routine	Label	Ref
01 FOR A SAVE LG REQUEST, THIS PROCEDURE PROCESSES THE LPMES OF THE PREVIOUSLY SAVED ASPCT IN WHICH SOME SLOTS WERE RELEASED AFTER SAVE. SAVE THE RETURN ADDRESS, THE POINTER TO THE LPME AND THE CORRESPONDING RPN, IN THE WORKAREA.			
02 IF THE LPME SAVED FLAG (ASPLSAVE) IS ON, THEN SKIP TO STEP 13.			
03 IF THE RELEASED AFTER SAVE FLAG (ASPLSVRP) AND THE VALID FLAG (ASPLVALD) ARE OFF IN THE LPME, THEN SKIP TO STEP 13.			
04 IF THE RELEASED AFTER SAVE FLAG (ASPLSVRP) IS OFF AND THE VALID FLAG (ASPLVALD) IS ON IN THE LPME, THEN FLAG THE LPME AS SAVED (ASPLSAVE=1), INCREMENT THE SAVESLOT COUNTER AND SKIP TO STEP 13.			
05 IF THE RELEASED AFTER SAVE FLAG (ASPLSAVE) IS ON, THEN TURN IT OFF.			

Notes	Routine	Label	Ref

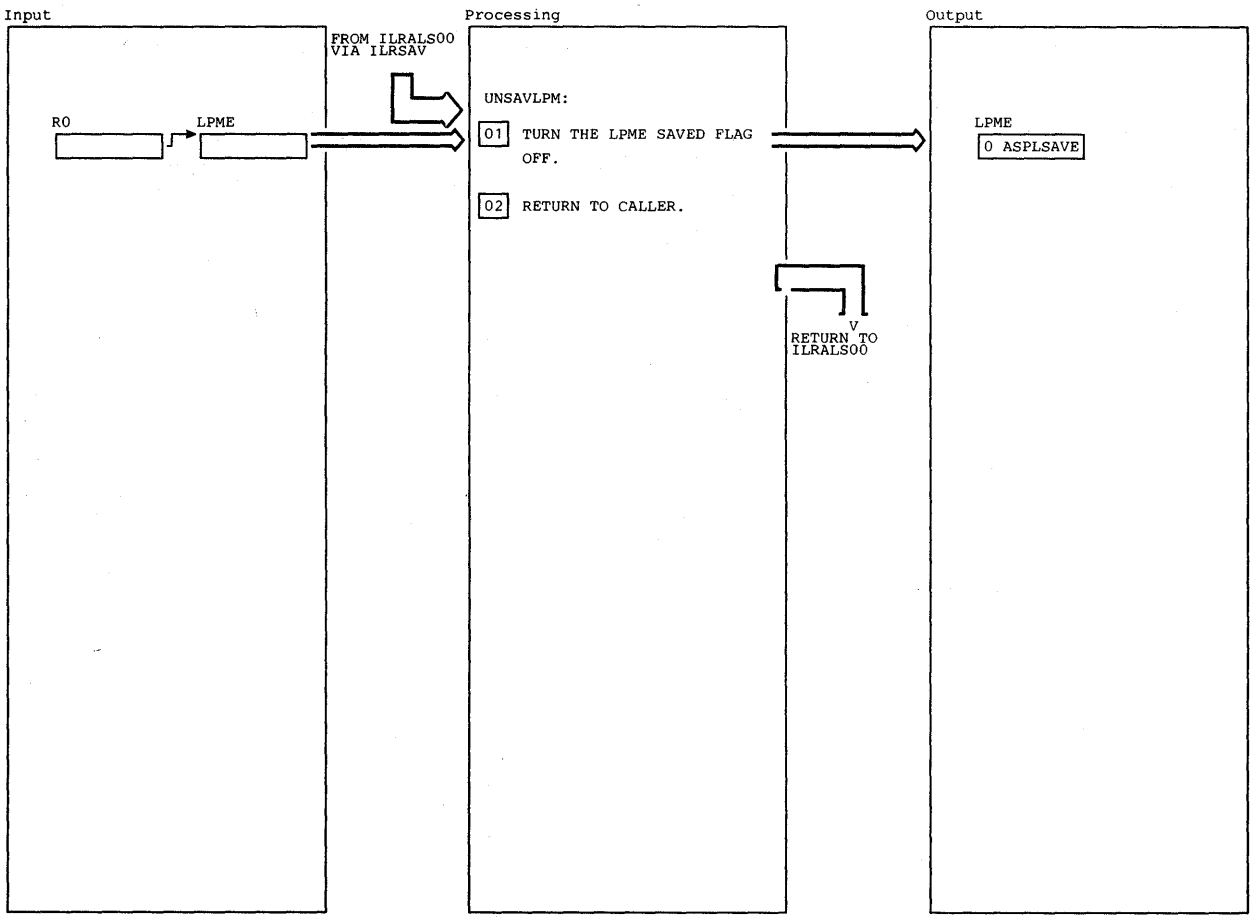
Diagram 25.16.3 SAVLIMBO (Part 1 of 2)



Notes	Routine	Label	Ref
06 IF THERE IS NOT A RETRIEVED ASPCT (RETASPCT=0), AND THE LPME VALID FLAG (ASPLVALD) IS OFF, THEN SKIP TO STEP 13.			
07 IF THERE IS NOT A RETRIEVED ASPCT (RETASPCT=0), AND THE LPME VALID FLAG (ASPLVALD) IS ON, THEN FLAG THE LPME AS SAVED (ASPLSAVE=1), INCREMENT THE SAVESLOT COUNTER, AND SKIP TO STEP 13.			
08 IF THERE IS A RETRIEVED ASPCT (RETASPCT¬=0), CONVERT THE RPN TO AN LPME.			
09 IF THE CONVERSION FAILS, ISSUE AN 085 ABEND FOR RECORDING AND THEN SKIP TO STEP 13.			
10 SAVE THE POINTER TO THE ACTIVE LPME, CALL ILRFRSL1 TO FREE THE SLOT FOR THIS LPME, RESET LPMEPTR BACK TO THE ACTIVE LPME.	ILRFLSLT	ILRFRSL1	
11 DECREMENT THE SAVESLOT COUNTER BY ONE, AND INCREMENT THE FREESLOT COUNTER BY ONE.			
12 IF THE LPME VALID FLAG IS ON (ASPLVALD=1), THEN FLAG THE LPME AS SAVED (ASPLSAVE=1), AND INCREMENT THE SAVESLOT COUNTER.			
13 RETURN TO CALLER.			

Notes	Routine	Label	Ref

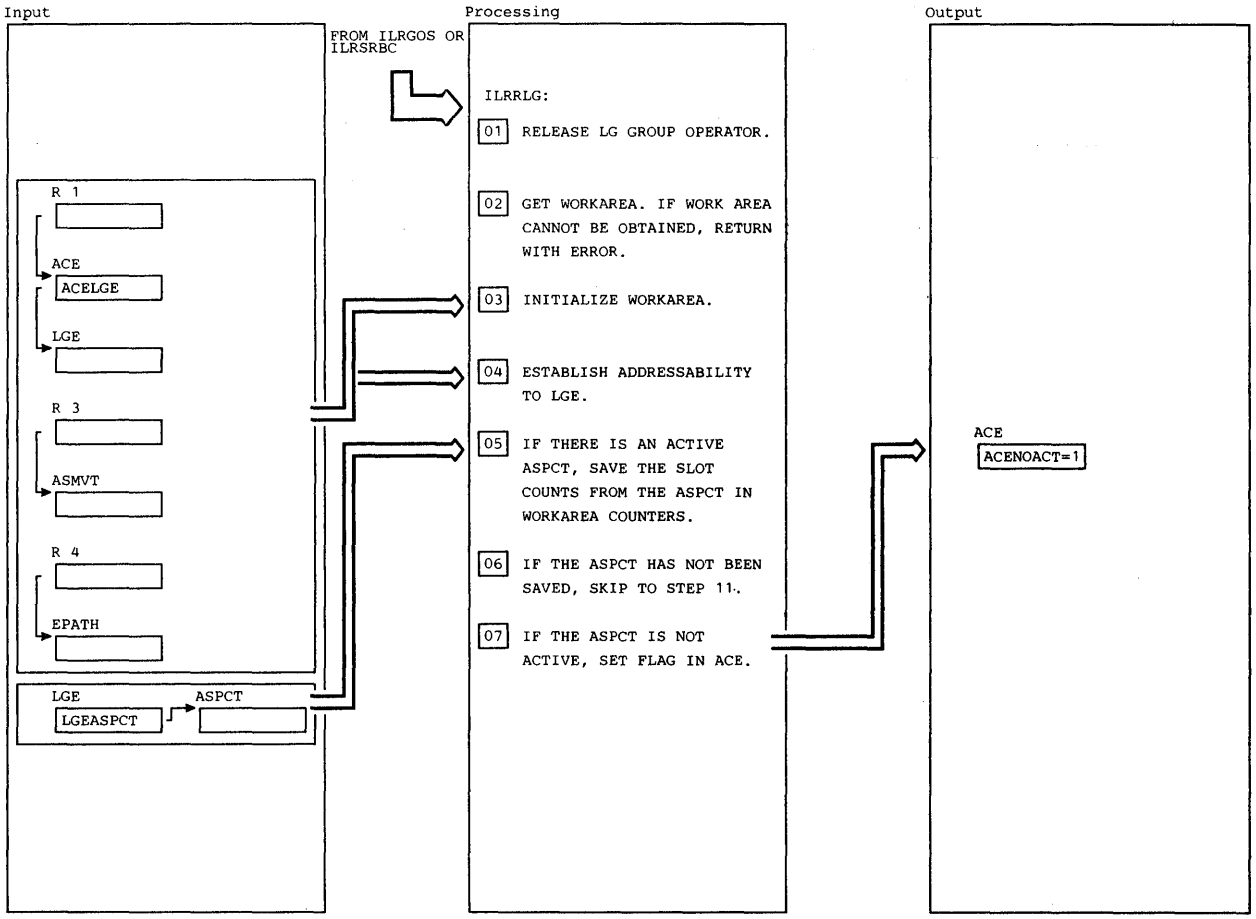
Diagram 25.16.3. SAVLIMBO (Part 2 of 2)



Notes	Routine	Label	Ref
01 TURN THE SAVED FLAG (ASPLSAVE) OFF IN THE LPME.			
02 RETURN TO ILRALS00.			

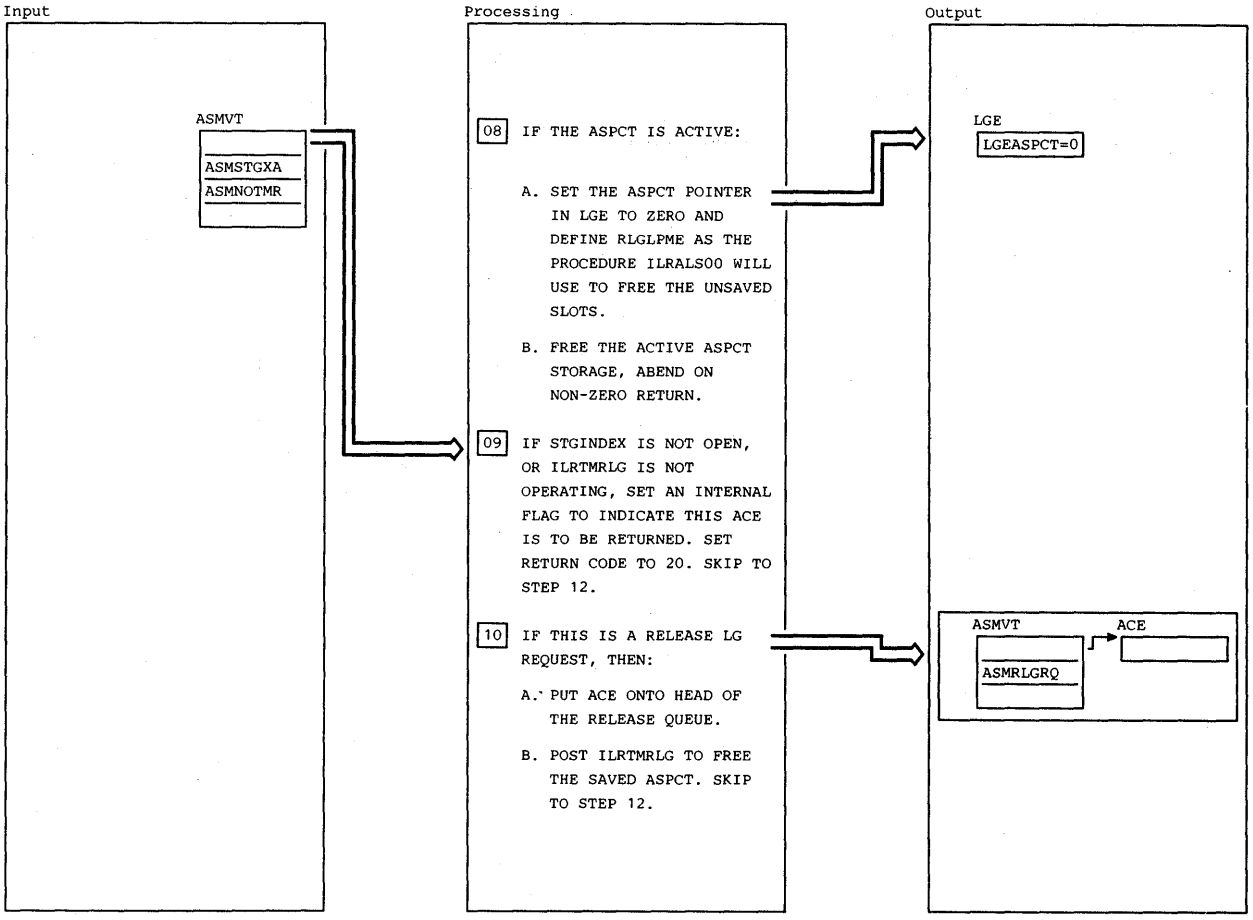
Notes	Routine	Label	Ref

Diagram 25.16.4 UNSAVLPM (Part 1 of 1)



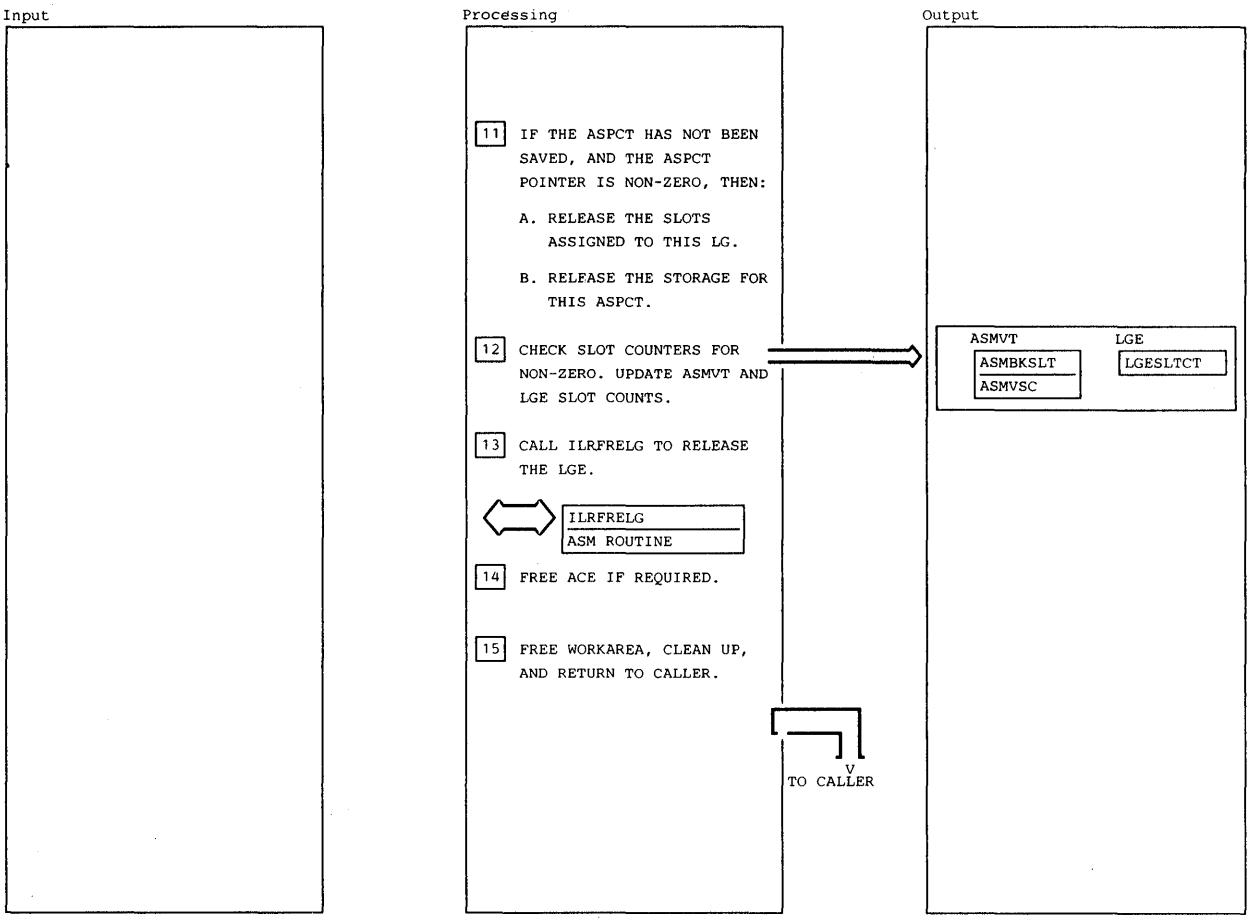
Notes	Routine	Label	Ref
01 THE RELEASE LOGICAL GROUP OPERATOR (ILRRLG) TOGETHER WITH ILRTMRLG ARE RESPONSIBLE FOR RELEASING SLOTS OF THIS LG, FREEING THE ACTIVE ASPCT, AND ERASING THE SAVED ASPCT FROM SYS1.STGINDEX. FOR RECOVERY, ILRGOS01 RECOVERY HANDLES ERRORS OCCURRING IN ILRRLG.			
02 RECORD ENTRY IN EPATH. USE ILRGMA TO OBTAIN A WORKAREA FROM ASM'S POOL. IF ILRGMA IS UNSUCCESSFUL (REG1=0), SET THE RETURN CODE TO 28, AND RETURN TO CALLER.	ILRGMA		
03 INITIALIZE THE SAVESLOT, BACKSLOT, AND FREESLOT WORKAREA COUNTERS TO ZERO.			
04 GET THE POINTER TO THE LGE FROM THE ACE (ACELGE).			
05 IF THE LGEASPCT POINTER IS NON-ZERO (ACTIVE ASPCT), STORE THE LGEASPCT POINTER IN EPATH, SAVE THE ASPBKSLT COUNT IN BACKSLOT AND THE ASPSAVCT COUNT IN SAVESLOT.			
06 IF THIS LOGICAL GROUP HAS NOT BEEN SAVED (ACEUSYM=OFF), PROCEED TO STEP 11.			
07 IF LGEASPCT IS ZERO, SET ACENOACT=ON IN ACE.			

Diagram 25.17 ILRRLG (Part 1 of 3)



Notes	Routine	Label	Ref
08 IF LGEASPCT IS NONZERO:			
A. PUT POINTER TO ASPCT IN REGISTER 0. SET LGEASPCT TO ZERO. CALL ILRALS00.	ILRALS00		25.16.1
B. PUT POINTER TO ASPCT IN REG 1, LENGTH OF ASPCT IN REGISTER 0. SET THE EPATH ASPCT POINTER TO ZERO. CALL ILRAFS00 TO FREE THE ASPCT STORAGE. IF REG 15 NOT 0, SAVE REGISTERS FOR RECOVERY AND ISSUE AN 087 ABEND.	ILRAFS00		
09 IF ASMSTGXA=0 OR ASMNOTMR=1, THEN SET AN INTERNAL FLAG INDICATING THAT THE ACE IS TO BE FREED. SET THE RETURN CODE TO 20 AND PROCEED TO STEP 12.			
10 IF ASMSTGXA=1 AND ASMNOTMR=0 AND ACEOP=ACERELLG, THEN:			
A. SET THE FORWARD POINTER IN THE ACE TO ZERO. COMPARE AND SWAP THE ACE ONTO ASMRLGRQ.			
B. POST ILRTMRLG VIA THE ASMTMECB FIELD IN ASMVT. ILRTMRLG WILL RELEASE THE SAVED ASPCT ON SYS1.STGINDEX.		IEAOPT01	

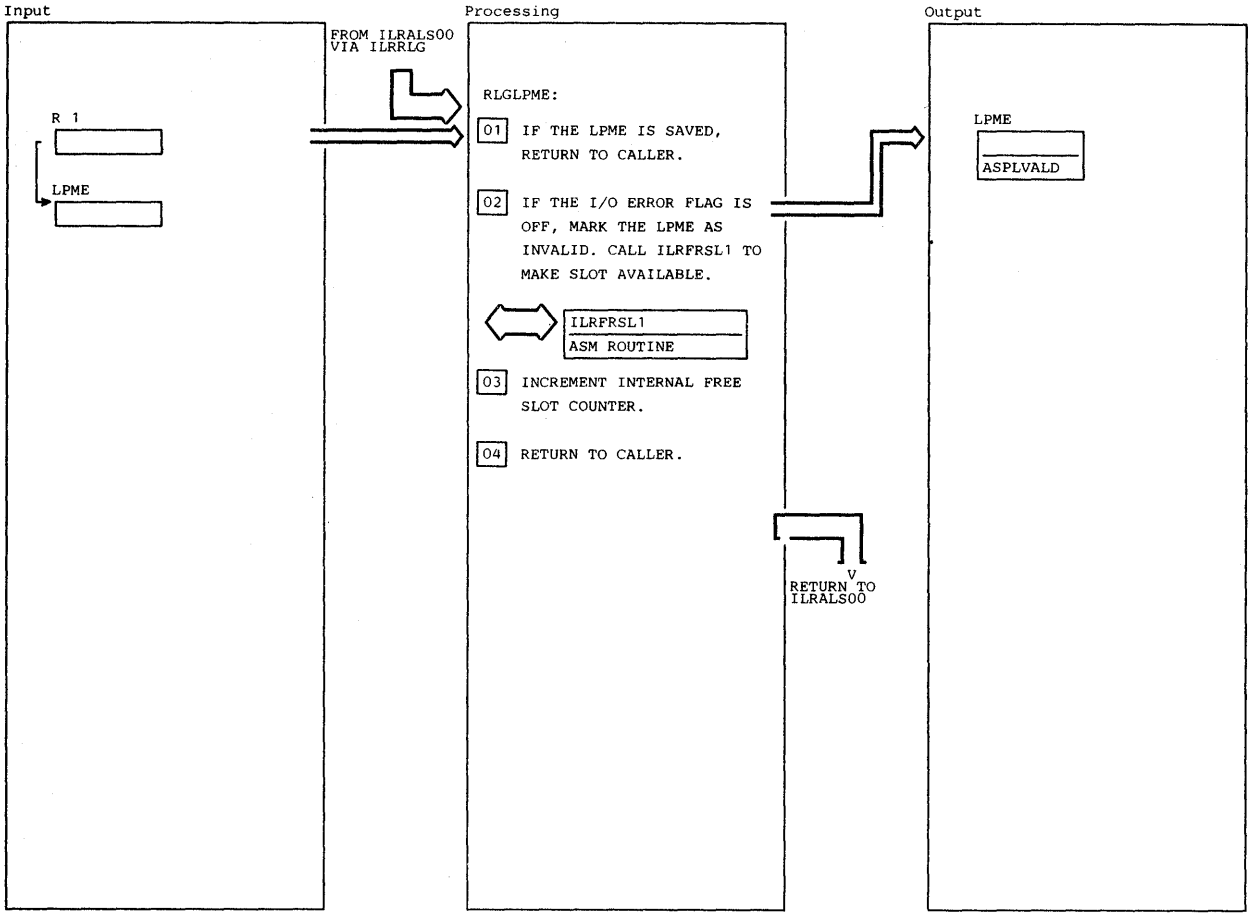
Diagram 25.17 ILRRLG (Part 2 of 3)



Notes	Routine	Label	Ref
11 IF ACEUSYM FLAG WAS OFF, THE LGEASPCT IS NON-ZERO, THEN:			
A. LOAD REG 0 WITH POINTER TO ASPCT AND CALL ILRALS00 (WHICH USES RLGLPME) TO RELEASE THE SLOTS ASSIGNED TO THIS LG.	ILRALS00		25.16.1
B. LOAD REG 1 WITH POINTER TO ASPCT AND SET REG 0 TO LENGTH OF THE ASPCT. SET THE EPATH POINTER OF THE ACTIVE ASPCT TO ZERO, AND CALL ILRAFS00 TO FREE THE STORAGE USED FOR THIS ASPCT. IF THERE IS A NON-ZERO RETURN CODE FROM ILRAFS00, SAVE REGISTERS FOR RECOVERY, AND ISSUE AN 087 ABEND.	ILRAFS00		
12 IF THE BACKSLOT COUNTER IS NON-ZERO, ADD IT BACK INTO THE AVAILABLE SLOT COUNT (ASMBKSLT). IF THE FREESLOT COUNTER IS NON-ZERO, SUBTRACT IT FROM, ASMVSC AND LGESLTCT. IF THE SAVESLOT COUNTER IS NON-ZERO, SUBTRACT IT FROM LGESLTCT.			

Notes	Routine	Label	Ref
13 SET THE LGE POINTERS TO THE ACE TO ZERO. LOAD REG 1 WITH POINTER TO LGE AND CALL ILRFRELG TO RELEASE THE LGE.	ILRGOS	ILRFRELG	
14 IF ACEUSYM FLAG WAS OFF OR THIS IS A DEACTIVATE REQUEST (ACEOP=ACEDEACT) OR STORAGE INDEX CLOSED (ASMSTGXA=0) OR ILRTMRLG IS NOT OPERATING (ASMNOTMR=ON) THEN SET THE ACE POINTER IN THE EPATH TO ZERO, LOAD REG 1 WITH POINTER TO ACE, AND CALL ILRGMA TO FREE THE ACE.	ILRGMA		
15 SET THE WORK AREA POINTER AND THE RLG BIT IN EPATH TO ZERO. INVOKE ILRGMA TO FREE THE WORKAREA RESTORE REGISTERS AND RETURN TO CALLER.	ILRGMA		

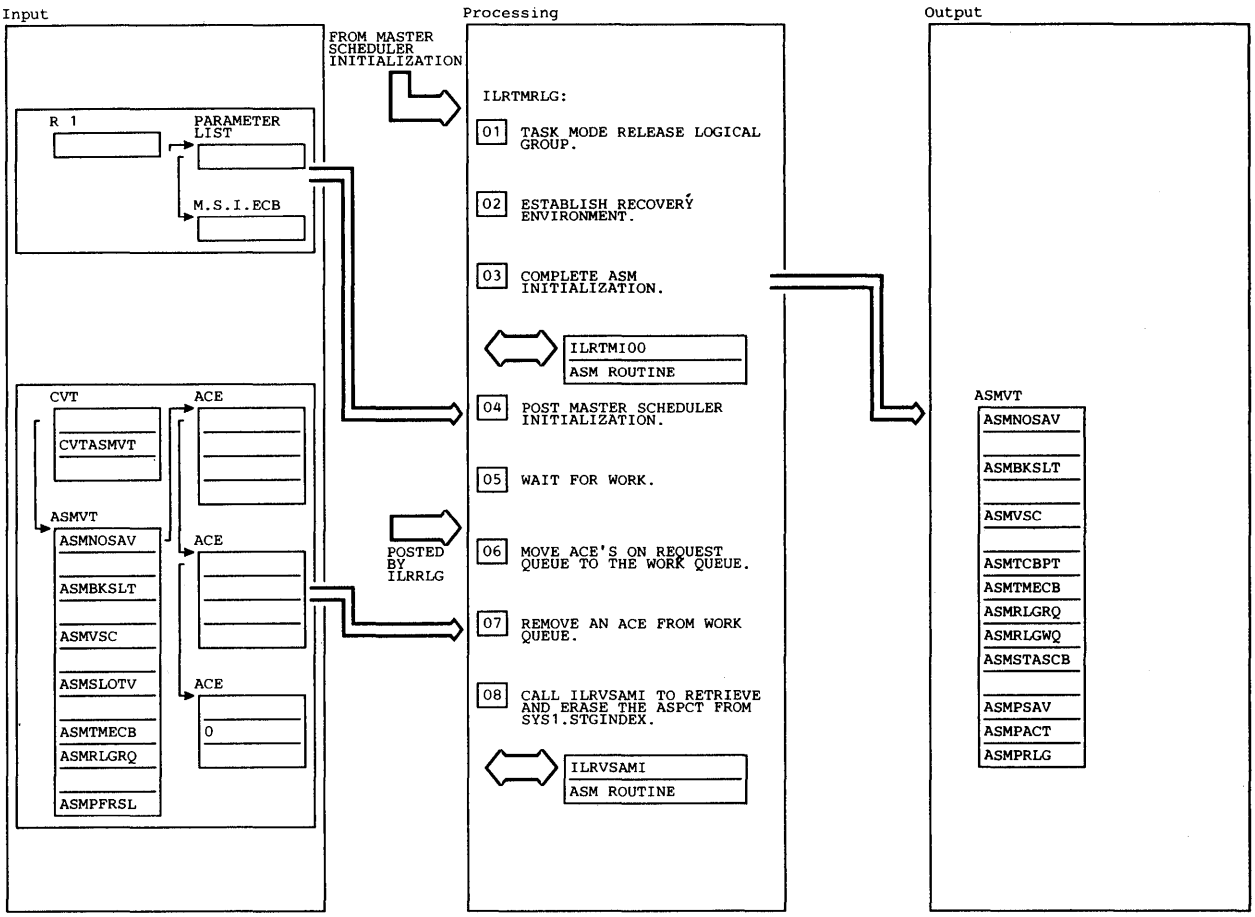
Diagram 25.17 ILRRLG (Part 3 of 3)



Notes	Routine	Label	Ref
01 IF THE LPME HAS BEEN SAVED (ASPLSAVE=1), THEN GOTO STEP 4. THIS LPME WILL BE PROCESSED IN THE MASTER SCHEDULER ADDRESS SPACE BY ILRTMRLG.			
02 IF ASPLIOER=OFF, MARK THIS LPME AS INVALID (ASPLVALD).	ILRFRSLT	ILRFRSL1	
03 INCREMENT THE INTERNAL FREED SLOT COUNTER BY 1. THIS COUNTER IS USED TO UPDATE THE VIO SLOT COUNT IN THE ASMVT (ASMVSC) AND THE VIO SLOT COUNT IN THE ASCB (ASCBVSC).			
04 RETURN TO CALLER.			

Notes	Routine	Label	Ref

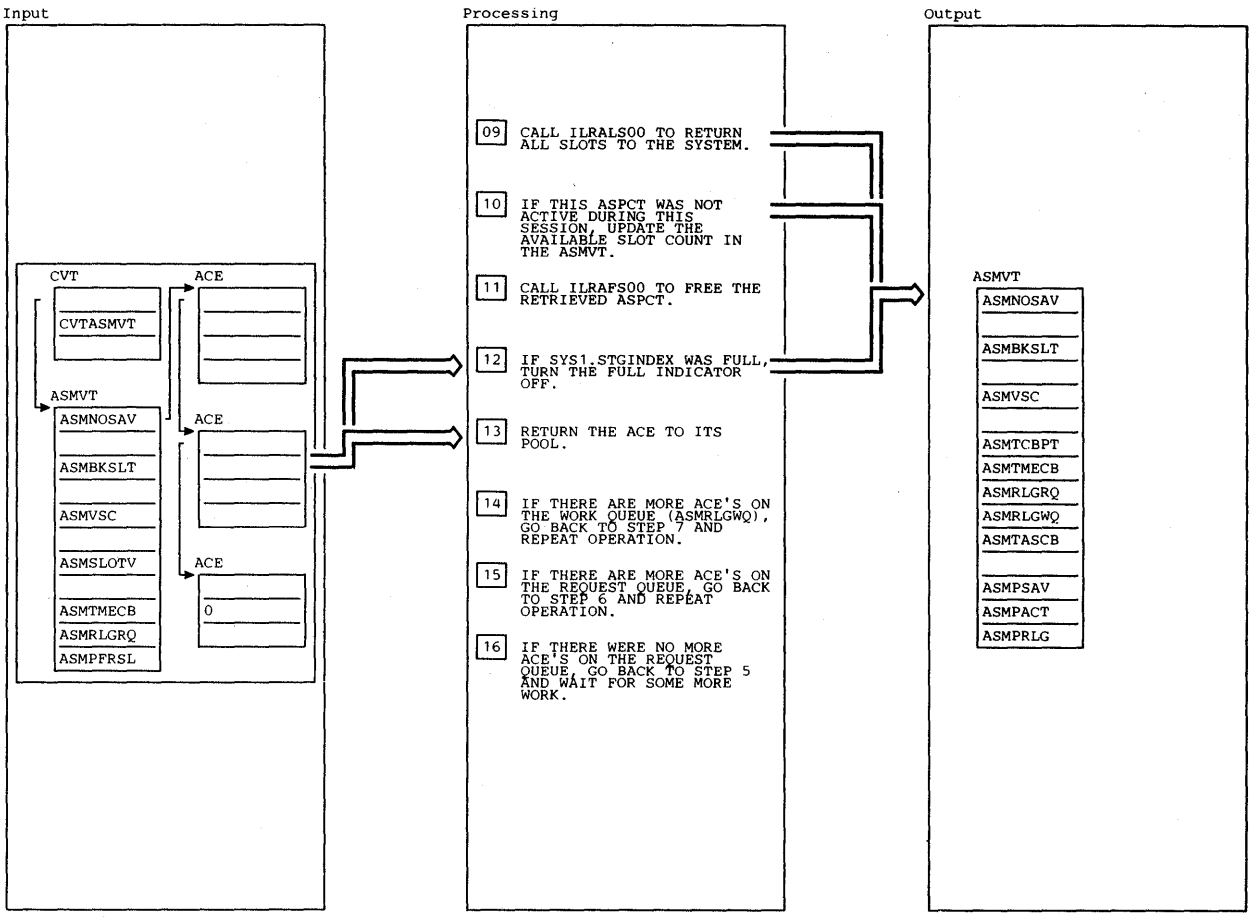
Diagram 25.17.1 RLGLPME (Part 1 of 1)



Notes	Routine	Label	Ref
01 ENTRY IS FROM MASTER SCHEDULER INITIALIZATION ROUTINE IEEMB860. ILRTMRLG HAS TWO SEPARATE RESPONSIBILITIES: TO CALL ILRTMI00 TO COMPLETE ASM'S INITIALIZATION AND TO COMPLETE THE RELEASE LOGICAL GROUP PROCESSING FOR A SAVED ASPCT. ISSUE AN UNCONDITIONAL GETMAIN TO OBTAIN SPACE FOR A WORKAREA. FOR RECOVERY, ILRTMI01 RECOVERY ROUTINE HANDLES ERRORS OCCURRING IN ILRTMRLG AND ITS PATH THROUGH ILRVSAMI.	GETMAIN		
02 ISSUE AN ESTAE TO ESTABLISH RECOVERY ENVIRONMENT.	ESTAE		
03 INITIALIZE CERTAIN FIELDS IN THE ASMVT. CALL ILRTMI00 IN ORDER TO COMPLETE INITIALIZATION OF ASM.			
A. PLACE THE ENTRY POINTS FOR ILRACT, ILRRLG, AND ILRSAV INTO THE ASMVT. ALSO PUT THE ADDRESS OF THE CURRENT TCB AND CURRENT ASCB INTO THE ASMVT.			
B. ISSUE A LOAD OF ILRTMI00 TO OBTAIN ITS ENTRY POINT ADDRESS. CALL ILRTMI00 TO COMPLETE ASM INITIALIZATION. SAVE ILRTMI00'S RETURN CODE. ISSUE A DELETE OF ILRTMI00 TO FREE THE STORAGE IT OCCUPIED.	ILRTMI00 LOAD DELETE	ILRTMI00	
04 ISSUE A POST ON THE MASTER SCHEDULER INITIALIZATION'S ECB TO INDICATE THAT ASM INITIALIZATION IS COMPLETE.	POST		
05 ISSUE A WAIT ON THE ASMTMECB IN THE ASMVT. ILRRLG WILL POST THIS ECB WHEN THERE IS WORK FOR ILRTMRLG TO DO.	WAIT		

Notes	Routine	Label	Ref
06 COMPARE AND SWAP THE ACE'S ON THE REQUEST QUEUE (ASMRLGRQ) TO THE WORK QUEUE (ASMRLGWQ).			
07 REMOVE AN ACE FROM THE HEAD OF THE WORK QUEUE (ASMRLGWQ).			
08 STORE THE POINTERS TO THE EPATH AND TO THE 'S' SYMBOL IN THE ACE INTO THE WORK AREA PARAMETER LIST FOR ILRVSAMI. SET THE REQUEST OP-CODE TO 04 - RETRIEVE AND ERASE FOR RELEASE. PLACE A POINTER TO THE PARAMETER LIST INTO REGISTER 1, THEN CALL ILRVSAMI. IF ILRVSAMI COULD NOT RETRIEVE THE ASPCT, SKIP TO STEP 13.			

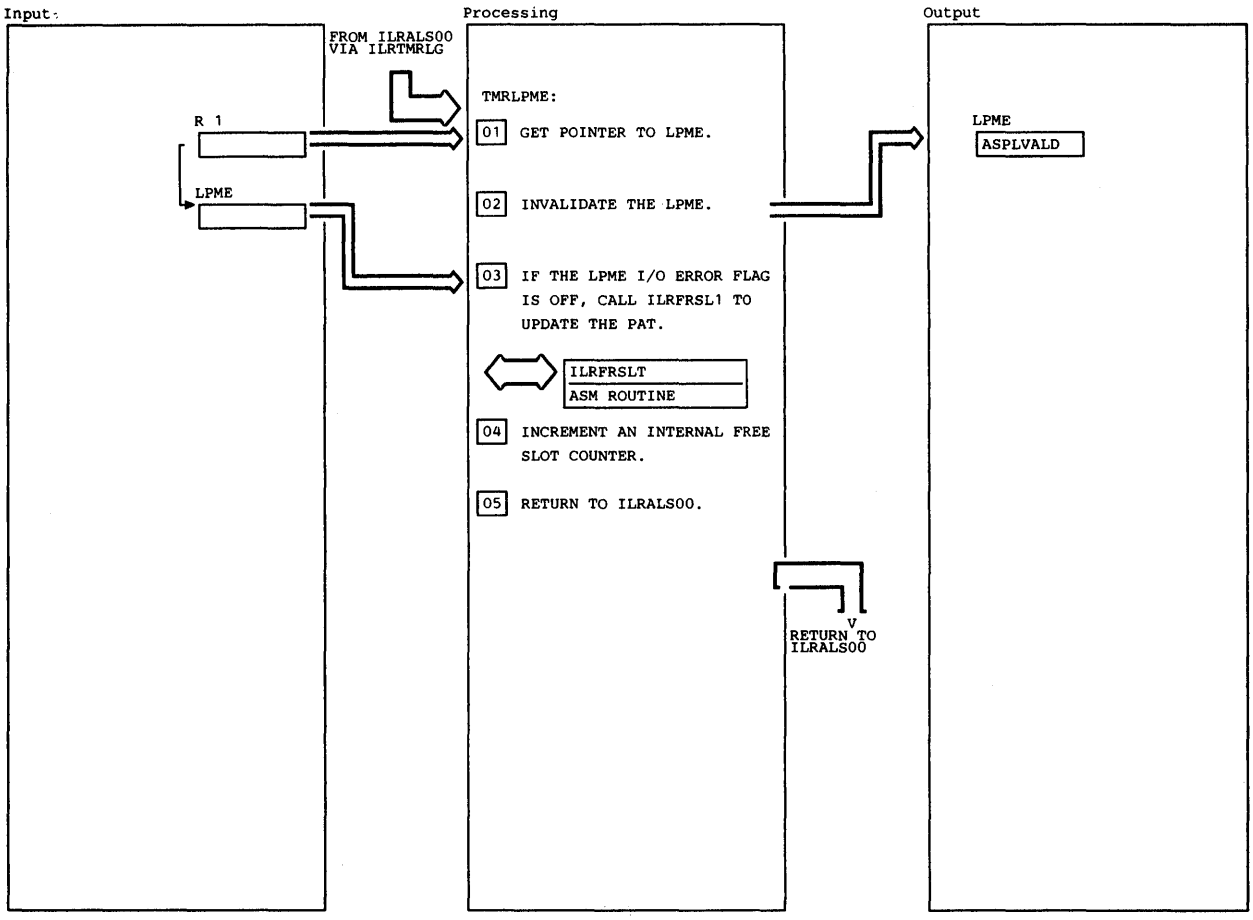
Diagram 25.18 ILRTMRLG (Part 1 of 2)



Notes	Routine	Label	Ref
09 DEFINE TMRLPME AS THE PROCEDURE ILRALS00 WILL USE. LOAD REG 0 WITH POINTER TO THE RETRIEVE ASPCT BASE. CALL ILRALS00 TO RELEASE THE SLOTS ASSIGNED TO THIS ASPCT. UPON RETURN FROM ILRALS00, THE NUMBER OF SLOTS FREED IS DECREMENTED FROM THE VIO SLOT COUNT (ASMVSC).	ILRALS00		25.17.1
10 IF THE ACENACT FLAG IS ON, THIS ASPCT WAS NOT ACTIVE SINCE THE LAST WARM START IPL. THE NUMBER OF SLOTS USED TO BACK UP THIS DATA SET IS CALCULATED BY DIVIDING THE MAXIMUM NUMBER OF SLOTS THAT COULD BE ALLOCATED TO THIS DATA SET BY THE ILRSLOTV CONSTANT. THE RESULT IS USED TO INCREMENT THE AVAILABLE SLOT COUNT (ASMBKSLT).			
11 OBTAIN THE LOCAL LOCK SINCE ILRAFS00 NEEDS IT WHILE FREEING SPACE. LOAD REG 1 WITH THE POINTER TO THE RETRIEVED ASPCT BASE. SET REG 0 TO THE LENGTH OF AN I/O BUFFER. CALL ILRAFS00 TO FREE THE BUFFER SPACE. RELEASE THE LOCAL LOCK. IF THERE IS A NON-ZERO RETURN CODE FROM ILRAFS00 ISSUE AN 087 ABEND.	ABEND ILRAFS00		
12 IF SYS1.STGINDEX FULL FLAG (ASMNOSAV) IS ON IN THE ASMVT, TURN THE FLAG OFF. THIS WILL ALLOW ASM TO PERFORM SAVE OPERATIONS AGAIN.			
13 USE ILRGMA TO RETURN THE ACE TO ITS POOL.	ILRGMA		
14 IF THE WORK QUEUE (ASMRLGWQ) IS NON-ZERO, THEN GO BACK TO STEP 7 AND REPEAT THE PROCESS FOR THE NEXT ACE ON THE WORK QUEUE.			

Notes	Routine	Label	Ref
15 IF THE REQUEST QUEUE (ASMRLGRQ) IS NON-ZERO THEN GO BACK TO STEP 6 AND REPEAT THE PROCESS FOR THE NEXT GROUP OF ACE'S ON THE REQUEST QUEUE.			
16 IF REQUEST QUEUE (ASMRLGRQ) IS ZERO, GO BACK TO STEP 4 AND WAIT FOR ILRRLG TO SEND SOME MORE WORK.			

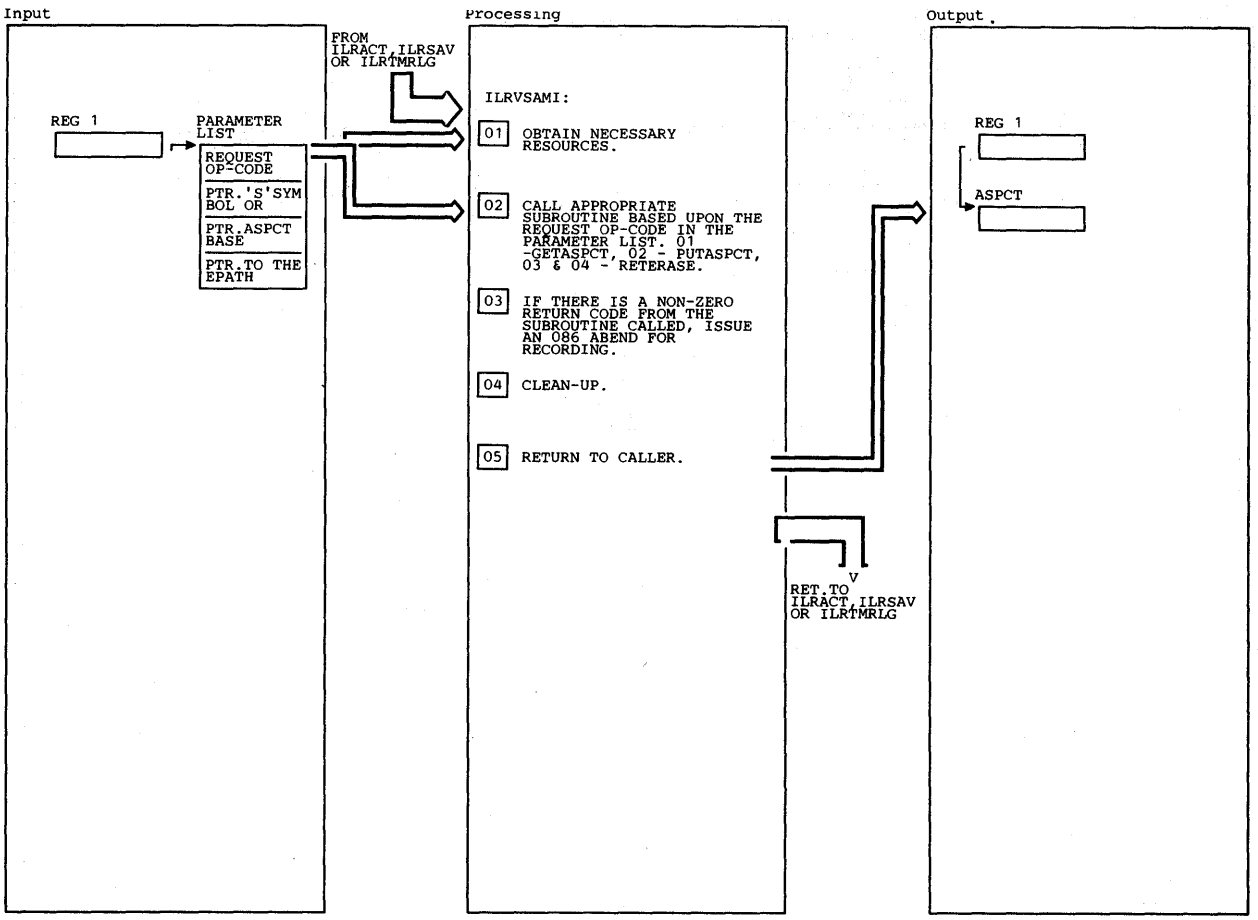
Diagram 25.18 ILRTMRLG (Part 2 of 2)



Notes	Routine	Label	Ref
01 OBTAIN THE POINTER TO THE LPME FROM REG 1.			
02 TURN OFF THE LPME VALID FLAG (ASPLVALD).			
03 IF THE LPME I/O ERROR FLAG (ASPLIOER) IS OFF, THEN LOAD REG 1 WITH THE LSID TO BE FREE AND CALL ILRFRSL1. ILRFRSL1 WILL UPDATE THE APPROPRIATE BIT IN THE PAT MAP AND SLOT AVAILABLE COUNT (PARESLTA) OF THE APPROPRIATE PART ENTRY, THUS MAKING THE SLOT AVAILABLE FOR FURTHER USE.	ILRFRSLT	ILRFRSL1	
04 INCREMENT AN INTERNAL FREE SLOT COUNTER BY 1. THIS COUNT IS USED LATER TO UPDATE THE VIO SLOT COUNT IN THE ASMVT.			
05 RETURN TO ILRALS00.			

Notes	Routine	Label	Ref

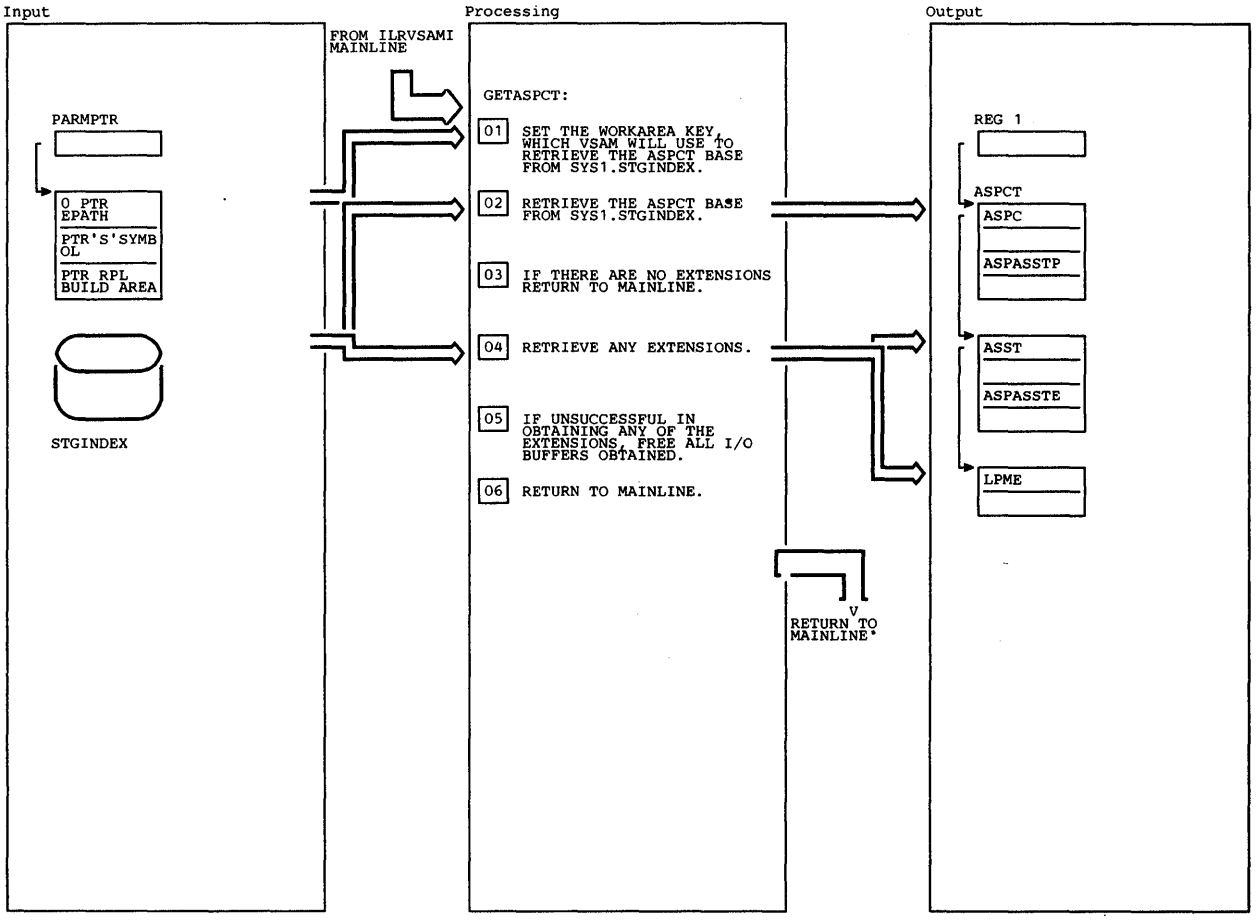
Diagram 25.18.1 TMRLPME (Part 1 of 1)



Notes	Routine	Label	Ref
01 USE ILRGMA TO OBTAIN A WORKAREA AND RPL BUILD AREA. IF ILRGMA CANNOT GET A WORKAREA, RETURN TO CALLER WITH A RETURN CODE OF 28. SAVE POINTER TO PARAMETER LIST IN WORKAREA, GET POINTER TO EPATH FROM INPUT PARAMETER LIST. RECORD ENTRY INFORMATION IN EPATH. RECORD POINTER TO WORKAREA IN EPATH. RECOVERY FOR ILRVSAMI IS ESTABLISHED BY ITS CALLER.	ILRGMA		
02 IF THE REQUEST OP-CODE IS: 01 CALL GETASPCT 0 TO RETRIEVE THE ASPCT FROM SYS1.STGINDEX: 02 CALL PUTASPCT TO STORE THE ASPCT ON SYS1.STGINDEX: 03 CALL RETERASE TO RETRIEVE AND ERASE THE ASPCT FROM SYS1.STGINDEX FOR SAVE LG REQUESTS: 04 CALL RETERASE FOR RELEASE LG REQUESTS.		GETASPCT PUTASPCT RETERASE	25.19. 1 25.19. 2 25.19. 3
03 IF THERE IS A NON-ZERO RETURN CODE FROM WHICHEVER SUBROUTINE WAS CALLED, ISSUE AN 086 ABEND.	ABEND		
04 SET THE EPATH POINTER TO THE WORKAREA TO ZERO. USE ILRGMA TO RETURN THE WORKAREA AND RPL BUILD AREA TO ITS POOL. SET THE ENTRY INFORMATION IN THE EPATH TO ZERO.	ILRGMA		
05 RESTORE REGISTERS AND RETURN TO CALLER.			

Notes	Routine	Label	Ref

Diagram 25.19 ILRVSAMI (Part 1 of 1)



Notes	Routine	Label	Ref
01 COPY THE 'S' SYMBOL INTO THE WORKAREA AND APPEND A FULL WORD OF ZEROS TO THE END, THUS MAKING THE TWELVE BYTE KEY FOR VSAM.			
02 CALL INTERNAL SUBROUTINE GETONE TO RETRIEVE THE ASPCT BASE FROM SYS1.STGINDEX. IF THERE IS A NON-ZERO RETURN CODE AND AN I/O BUFFER WAS OBTAINED ISSUE A FREEMAIN TO FREE THE I/O BUFFER, AND THEN RETURN TO MAINLINE WITH THE RETURN CODE FROM GETONE.	FREEMAIN	GETONE	25.19.5
03 IF THE ASST EXTENSION COUNT (ASPAEXCT) IN THE ASPCT BASE IS ZERO, THEN RETURN TO MAINLINE.			
04 PROCESS EXTENSIONS.			
A. SAVE THE LPME EXTENSION COUNT (ASPLEXCT), AND THE ASST EXTENSION COUNT (ASPAEXCT) IN THE WORKAREA.			
B. ZERO THE LPME AND ASST EXTENSION COUNTS IN THE ASPCT BASE.			
C. ZERO THE ASST EXTENSION POINTERS (ASPASSTP) IN THE ASPCT BASE.			
D. RECORD THE POINTER TO THE ASPCT BASE IN THE EPATH.			
E. INCREMENT THE TWELVE BYTE WORKAREA KEY BY ONE.			
F. CALL INTERNAL SUBROUTINE GETONE TO RETRIEVE AN ASST EXTENSION FROM SYS1.STGINDEX.		GETONE	25.19.5
G. IF THERE IS A NON-ZERO RETURN CODE FROM GETONE AND AN I/O BUFFER WAS OBTAINED, ISSUE A FREEMAIN FOR THIS LAST BUFFER. ZERO THE EPATH POINTER TO THE ASPCT BASE, LOAD REG 1 WITH POINTER TO ASPCT BASE, SET REG 0 TO LENGTH OF BUFFER, CALL ILRAFS00 TO FREE ALL THE BUFFERS. IF THERE IS A NON-ZERO RETURN CODE FROM ILRAFS00, ISSUE AN 087 ABEND. RETURN TO MAINLINE WITH THE RETURN CODE FROM GETONE.	FREEMAIN ILRAFS00 ABEND	GETONE	25.19.5
H. ZERO THE LPME EXTENSION POINTERS (ASPASSTE) IN THE ASST EXTENSION. STORE THE POINTER TO THE ASST EXTENSION IN THE ASPCT BASE (ASPASSTP(BASE)). INCREMENT THE ASST EXTENSION COUNT IN THE ASPCT BASE. DECREMENT THE ASST EXTENSION COUNT IN THE WORKAREA.			
I. INCREMENT THE TWELVE BYTE WORKAREA KEY BY ONE.			
J. CALL INTERNAL SUBROUTINE GETONE TO RETRIEVE AN LPME EXTENSION FROM SYS1.STGINDEX.		GETONE	25.19.5
K. IF THERE WAS A NON-ZERO RETURN CODE FROM GETONE AND AN I/O BUFFER WAS OBTAINED, STORE THE POINTER TO THE LPME I/O BUFFER IN THE ASST EXTENSION (ASPASSTE(ASST)) AND INCREMENT THE LPME EXTENSION COUNT (ASPLEXCT) IN THE ASPCT BASE. LOAD REG 1 WITH POINTER TO ASPCT BASE, ZERO EPATH POINTER TO ASPCT BASE BUFFER, SET REG 0 TO LENGTH OF I/O BUFFER. CALL ILRAFS00 TO FREE I/O BUFFERS. IF THERE IS A NON-ZERO RETURN CODE FROM ILRAFS00, ISSUE AN 087 ABEND. RETURN TO MAINLINE WITH RETURN CODE FROM GETONE.	ABEND		
L. STORE POINTER TO LPME EXTENSION IN ASST EXTENSION (ASPASSTE(ASST)). INCREMENT THE LPME EXTENSION COUNT (ASPLEXCT) IN THE ASPCT.			

Diagram 25.19.1 GETASPCT (Part 1 of 2)

Input

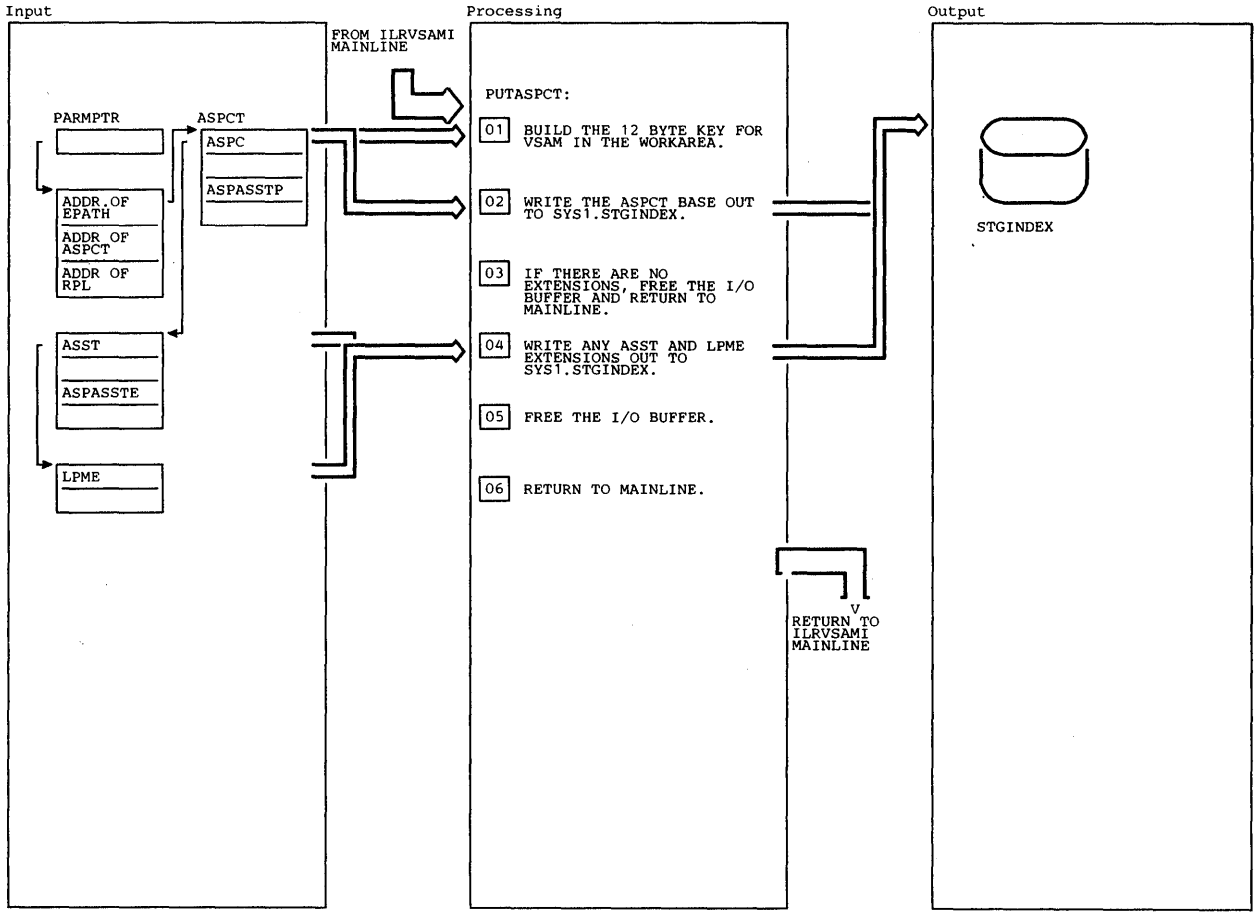
Processing

Output

Notes	Routine	Label	Ref
DECREMENT THE WORKAREA LPME EXTENSION COUNT.			
M. IF THE LPME EXTENSION COUNT IN THE WORKAREA IS ZERO SKIP TO STEP 4-0.			
N. INCREMENT THE ASST EXTENSION ARRAY SUBSCRIPT (ASST). IF WE HAVE NOT REACHED THE END OF THE ARRAY (ASST > ASPNASST) THEN GOTO STEP 4I. IF WE HAVE REACHED THE END OF THE ARRAY, RESET THE SUBSCRIPT TO BEGINNING OF ARRAY (ASST=1).			
O. IF THE WORKAREA ASST EXTENSION COUNT IS ZERO, RETURN TO MAINLINE.			
P. INCREMENT THE ASPCT BASE ARRAY SUBSCRIPT (BASE). IF WE HAVE REACHED END OF ARRAY (BASE > 4) THEN RETURN TO MAINLINE. IF WE HAVE NOT, GOTO STEP 4E.			
05 THIS STEP IS ALL THE ERROR EXITS IN STEP4.			
06 RETURN TO MAINLINE.			

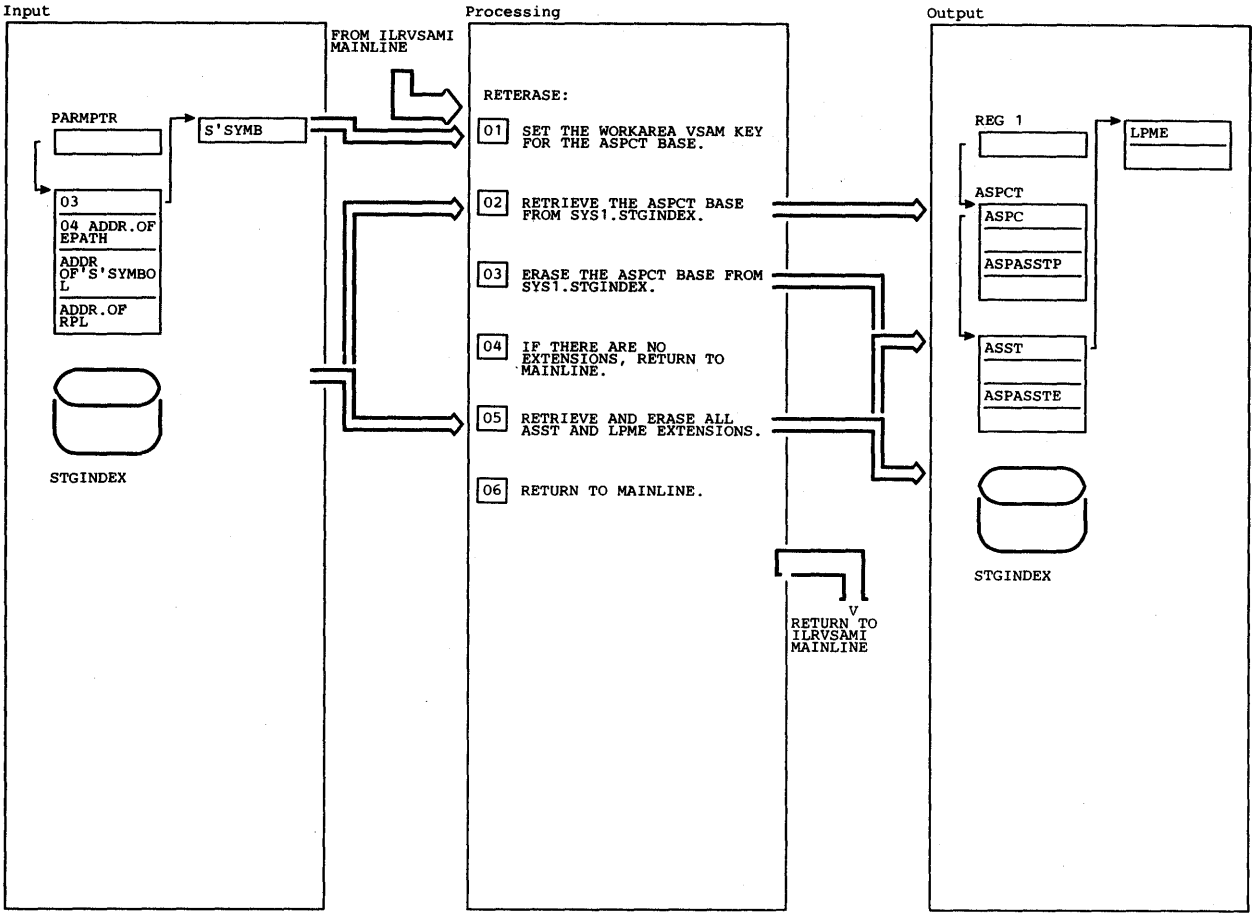
Notes	Routine	Label	Ref

Diagram 25.19.1 GETASPCT (Part 2 of 2)



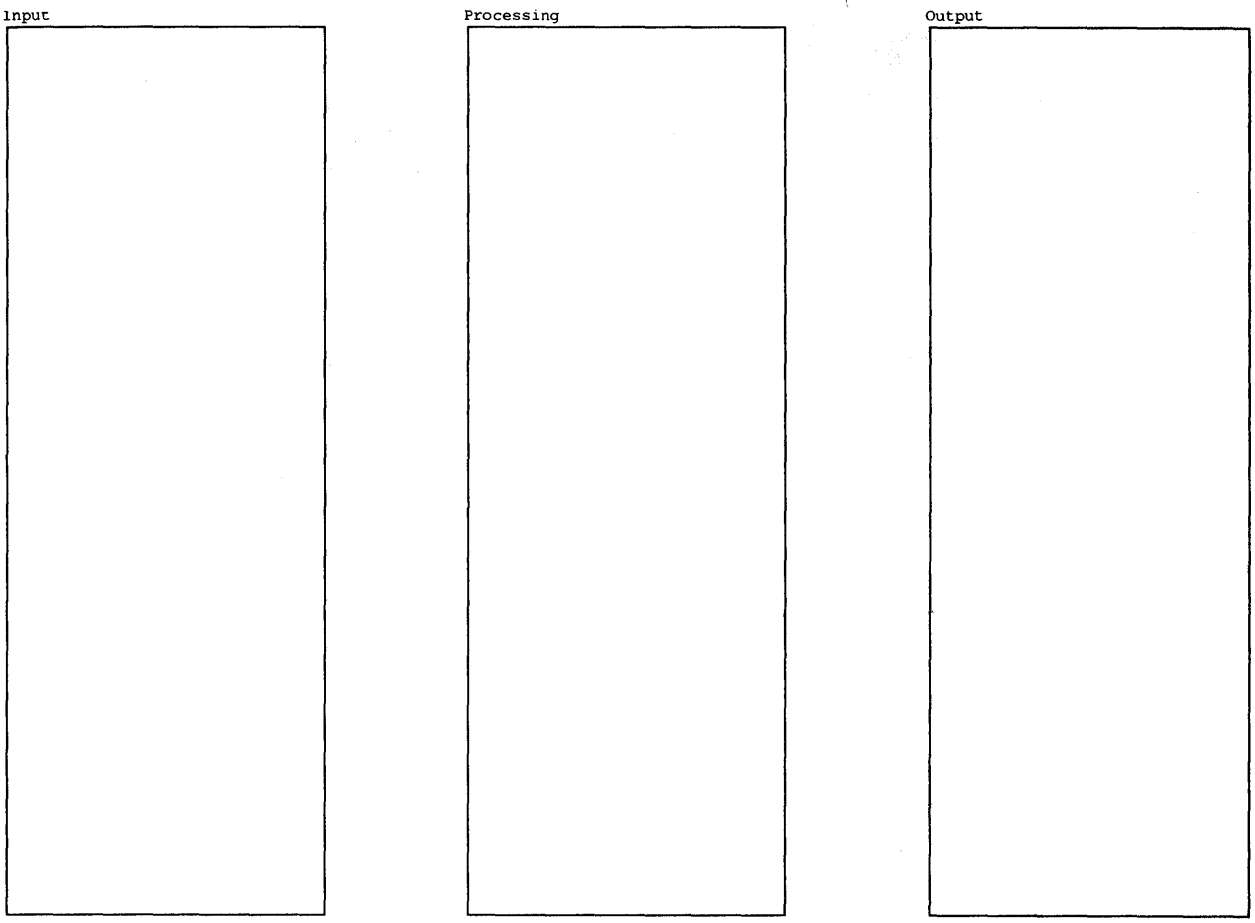
Notes	Routine	Label	Ref
01 COPY THE 12 BYTE VSAM KEY FROM THE BASE ASPCT TO THE WORKAREA KEY.			
02 LOAD REG 1 WITH THE POINTER TO THE ASPCT BASE. CALL INTERNAL SUBROUTINE PUTONE TO WRITE THE ASPCT BASE OUT TO SYS1.STGINDEX. IF THERE IS A NON-ZERO RETURN CODE AND STORAGE WAS OBTAINED FOR AN I/O BUFFER, ISSUE A FREEMAIN TO FREE THE BUFFER, AND RETURN TO MAINLINE WITH THE RETURN CODE FROM PUTONE.	FREEMAIN	PUTONE	25.19.4
03 IF THE ASST EXTENSION COUNT (ASPAEXCT) IS ZERO IN THE ASPCT BASE, ZERO THE EPATH POINTER TO THE I/O BUFFER, ISSUE A FREEMAIN TO FREE THE I/O BUFFER AND RETURN TO MAINLINE.	FREEMAIN		
04 PROCESS EXTENSIONS.			
A. SAVE THE LPME (ASPLEXCT) AND ASST (ASPAEXCT) EXTENSION COUNTS IN THE WORKAREA.			
B. GET POINTER TO AN ASST EXTENSION FROM THE ASPCT BASE (ASPASSTP(BASE)).			
C. COPY THE VSAM KEY FROM THE ASST EXTENSION TO THE WORKAREA KEY.			
D. LOAD REG 1 WITH POINTER TO ASST EXTENSION AND CALL THE INTERNAL SUBROUTINE PUTONE.		PUTONE	25.19.4
E. IF THERE IS A NON-ZERO RETURN CODE FROM PUTONE, GOTO STEP 4M.			
F. DECREMENT THE WORKAREA ASST EXTENSION COUNT.			
G. LOAD REG 1 WITH POINTER TO AN LPME EXTENSION (ASPASSTE(ASST)). COPY THE VSAM KEY FROM THE LPME EXTENSION TO THE WORKAREA. CALL INTERNAL SUBROUTINE PUTONE. IF THERE IS A NON-ZERO RETURN CODE FROM PUTONE, GOTO STEP 4M.		PUTONE	25.19.4
H. DECREMENT THE WORKAREA LPME EXTENSION COUNT.			
I. IF THE WORKAREA LPME EXTENSION COUNT IS ZERO, SKIP TO STEP 4L.			
J. INCREMENT THE ASST ARRAY SUBSCRIPT (ASST).			
K. IF THE SUBSCRIPT J IS NOT GREATER THAN THE NUMBER OF ENTRIES IN THE ARRAY (ASPNASST) GOTO STEP 4G. IF THE END OF THIS ARRAY HAS BEEN REACHED, RESET THE SUBSCRIPT FOR THE NEXT ASST ARRAY.			
L. IF THE ASST EXTENSION COUNT IS NOT ZERO, INCREMENT THE BASE ARRAY SUBSCRIPT (BASE) AND GOTO STEP 4D. IF THE COUNTER IS ZERO, GOTO STEP 5.			
M. IF AN ERROR OCCURS WHILE WRITING ON THE EXTENSIONS, ALL THE RECORDS ALREADY WRITTEN MUST BE ERASED. AN ERASE IS ISSUED AGAINST THE CURRENT RECORD AND THE VSAM KEY IS DECREMENTED BY ONE. THIS PROCESS (ERASE, DECREMENT KEY) CONTINUES UNTIL THE LAST FOUR BYTES OF THE VSAM KEY ARE ZERO.	ERASE		
05 SET THE EPATH POINTER TO THE I/O BUFFER TO ZERO ISSUE A FREEMAIN TO FREE THE I/O BUFFER.	FREEMAIN		

Diagram 25.19.2 PUTASPCT (Part 1 of 1)



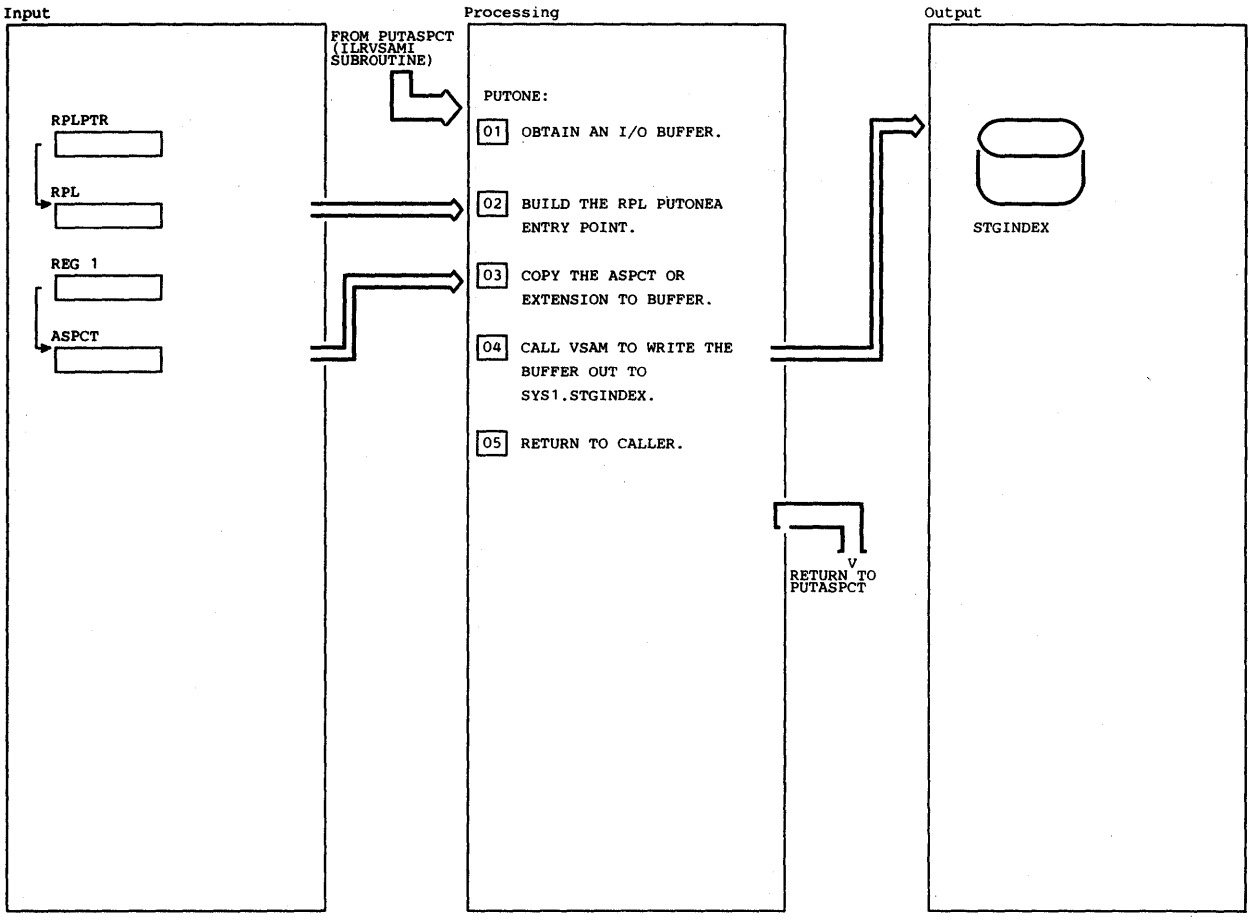
Notes	Routine	Label	Ref
01 THIS SUBROUTINE RETRIEVES AND ERASES THE ASPCT FROM SYS1.STGINDEX. COPY THE 'S' SYMBOL TO THE WORKAREA VSAM KEY. APPEND A FULLWORD OF ZEROS, THUS MAKING THE 12 BYTE KEY NEEDED FOR VSAM RETRIEVED.			
02 CALL INTERNAL SUBROUTINE GETONE TO RETRIEVE THE ASPCT BASE FROM SYS1.STGINDEX. IF THERE IS A ZERO RETURN CODE, GOTO STEP 3. IF THERE WAS A NON-ZERO RETURN CODE, SAVE THE RETURN CODE. IF STORAGE WAS OBTAINED, ISSUE A FREEMAIN TO FREE THE I/O BUFFER. LOAD THE SAVED RETURN CODE, AND RETURN TO MAINLINE.	FREEMAIN	GETONE	25.19.5
03 ISSUE AN ERASE FOR THE ASPCT BASE. IF THERE IS A ZERO RETURN CODE, GOTO STEP 4. IF THERE IS A NON-ZERO RETURN CODE, ISSUE A SHOWCB TO DETERMINE THE TYPE OF ERROR. IF RETRY IS POSSIBLE GO BACK AND REISSUE THE ERASE. SET THE INTERNAL RETURN CODE TO 4.	ERASE SHOWCB		
04 IF THE ASST EXTENSION COUNT (ASPAEXCT) IS ZERO, RETURN TO MAINLINE.			
05 ASST AND LPME EXTENSIONS:			
A. SAVE THE ASST (ASPAEXCT) AND LPME (ASPLEXCT) EXTENSION COUNTS IN THE WORKAREA. ZERO THE ASST AND LPME EXTENSION COUNTS IN THE ASPCT BASE. ZERO THE POINTERS (ASPASSTP) TO THE ASST EXTENSIONS IN THE ASPCT BASE. RECORD THE POINTER TO THE BUFFER ASPCT BASE IN THE EPATH.			
B. SET THE WORKAREA KEY FOR AN ASST EXTENSION. CALL INTERNAL SUBROUTINE GETONE TO RETRIEVE THE ASST EXTENSION FROM SYS1.STGINDEX. IF THERE IS A ZERO RETURN CODE FROM GETONE SKIP TO STEP 5.	GETONE		25.19.5
C. IF THERE WAS A NON-ZERO RETURN CODE FROM GETONE, AN I/O BUFFER WAS OBTAINED, AND THIS IS A RELEASE REQUEST(04), MAKE THE BUFFER LOOK LIKE AN ASST EXTENSION, AND GOTO STEP 5D. IF THIS WAS NOT A RELEASE REQUEST AND STORAGE WAS OBTAINED, FREE THE ASST EXTENSION BUFFER. CALL ILRAFS00 TO FREE THE BUFFERS. IF THERE IS A NON-ZERO RETURN CODE FROM ILRAFS00, ISSUE AN 087 ABEND AND RETURN TO MAINLINE.	FREEMAIN ILRAFS00 ABEND		
D. ZERO THE ASST EXTENSION ARRAY (ASPASSTE) OF POINTERS TO THE LPME EXTENSIONS. STORE THE POINTER TO THE ASST EXTENSION IN THE ASPCT BASE ARRAY (ASPASSTP(I)). INCREMENT THE ASST EXTENSION COUNT (ASPAEXCT) IN THE ASPCT BASE. DECREMENT THE WORKAREA ASST EXTENSION COUNT.			
E. ISSUE AN ERASE OF THE ASST EXTENSION. IF THERE IS A NON-ZERO RETURN CODE FROM ERASE, ISSUE A SHOWCB TO DETERMINE THE TYPE OF ERROR. IF RETRY IS POSSIBLE, GO BACK AND REISSUE THE ERASE. IF RETRY IS IMPOSSIBLE, SET THE INTERNAL RETURN CODE TO 4.	ERASE SHOWCB		
F. SET THE WORKAREA KEY FOR AN LPME EXTENSION. CALL INTERNAL SUBROUTINE GETONE TO RETRIEVE AN LPME EXTENSION.		GETONE	25.19.5
G. IF THERE IS A ZERO RETURN CODE FROM GETONE, SKIP TO STEP 5S.			
H. IF THERE IS A NON-ZERO RETURN CODE AND THIS IS A RELEASE REQUEST, SKIP TO STEP 5O.			
I.-IF THERE IS A NON-ZERO RETURN			

Diagram 25.19.3 RETERASE (Part 1 of 2)



Notes	Routine	Label	Ref
CODE, THIS IS NOT A RELEASE REQUEST, AND STORAGE WAS NOT OBTAINED SKIP TO STEP 5K.			
J. IF THERE WAS A NON-ZERO RETURN CODE, THIS IS NOT A RELEASE REQUEST, AND STORAGE WAS OBTAINED, STORE THE POINTER TO THE LPME BUFFER IN THE ASST EXTENSION ARRAY (ASPASSTE(J)). ZERO THE EPATH POINTER TO THE I/O BUFFER AND INCREMENT THE LPME EXTENSION COUNT (ASPLEXCT) IN THE ASPCT BASE.			
K. ISSUE AN ERASE OF AN ASST EXTENSION. DECREMENT THE ASST EXTENSION COUNT. SET THE ASST EXTENSION CARRAY SUBSCRIPT (J) FOR BEGINNING OF ARRAY.			
L. INCREMENT THE WORKAREA KEY FOR THE NEXT LPME EXTENSION. ISSUE AN ERASE OF AN LPME EXTENSION. DECREMENT THE LPME EXTENSION COUNT IN WORKAREA. INCREMENT THE ASST EXTENSION SUBSCRIPT. IF THE LPME EXTENSION COUNT IS ZERO, SET THE RETURN CODE TO 20 AND RETURN TO MAINLINE.	ERASE		
M. IF THE ASST EXTENSION ARRAY SUBSCRIPT HAS NOT REACHED THE END OF THE ARRAY GOTO STEP 5M. IF IT HAS REACHED THE END OF THE ARRAY, INCREMENT THE WORKAREA KEY FOR THE NEXT ASST EXTENSION AND GOTO STEP 5L.			
N. IF STORAGE WAS OBTAINED, ISSUE A FREEMAIN, TO FREE THE LPME EXTENSION BUFFER. DECREMENT THE LPME EXTENSION COUNT.	FREEMAIN		
O. IF THE LPME EXTENSION COUNT IS ZERO, GOTO STEP 5I.			
P. IF THE ASST EXTENSION ARRAY SUBSCRIPT HAS NOT REACHED THE END OF THE ARRAY, GOTO STEP 5F.			
Q. LOAD INTERNAL RETURN CODE AND RETURN TO MAINLINE.			
R. STORE THE POINTER TO THE LPME EXTENSION BUFFER IN THE ASST EXTENSION ARRAY (ASPASSTE(J)). ZERO THE EPATH POINTER TO THE I/O BUFFER. INCREMENT THE LPME EXTENSION COUNT IN THE BASE (ASPLEXCT). DECREMENT THE LPME EXTENSION COUNT.			
S. ISSUE AN ERASE OF AN LPME EXTENSION. IF THERE IS A ZERO RETURN CODE FROM ERASE GOTO STEP 5P.			
T. IF THERE IS A NON-ZERO RETURN CODE FROM ERASE, ISSUE A SHOWCB TO DETERMINE THE ERROR. IF A RETRY IS POSSIBLE GO BACK TO STEP 5T TO REISSUE THE ERASE. IF RETRY IS NOT POSSIBLE, SET THE INTERNAL RETURN CODE TO 4 AND GOTO STEP 5P.	SHOWCB		
06 RETURN TO MAINLINE.			

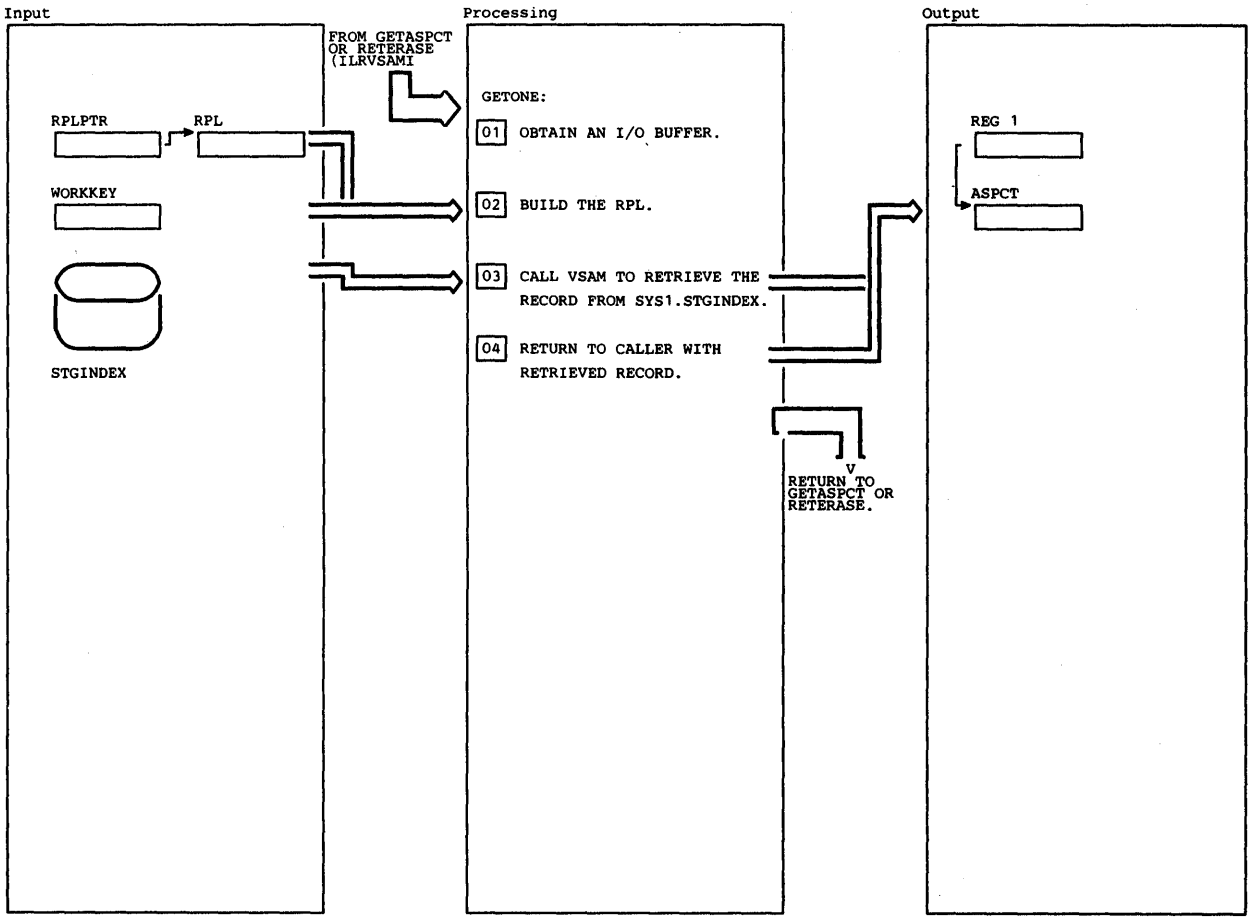
Diagram 25.19.3 RETERASE (Part 2 of 2)



Notes	Routine	Label	Ref
01 ISSUE A GETMAIN TO OBTAIN A 2K I/O BUFFER. IF THERE IS A ZERO RETURN CODE FROM GETMAIN SAVE THE POINTER TO THE BUFFER AND RECORD THE POINTER IN THE EPATH. IF THERE IS A NON-ZERO RETURN CODE FROM GETMAIN, SET THE RETURN CODE TO 28, AND RETURN TO CALLER.	GETMAIN		
02 ISSUE A GENCB TO BUILD THE RPL FOR A VSAM PUT REQUEST. IF THERE IS A NON-ZERO RETURN CODE FROM GENCB, SET THE RETURN CODE TO 20 AND RETURN TO CALLER.	GENCB		
03 PUTONEA ENTRY POINT. COPY THE ASPCT BASE, OR ASST EXTENSION OR LPME EXTENSION TO THE GETMAINED I/O BUFFER.			
04 ISSUE A VSAM PUT TO WRITE THE I/O BUFFER OUT TO SYS1.STGINDEX. IF THERE IS A NON-ZERO RETURN CODE FROM PUT, ISSUE A SHOWCB TO DETERMINE THE ERROR. IF WE CAN RETRY, GO BACK AND REISSUE THE PUT. IF STGINDEX IS FULL, SET	PUT SHOWCB		

Notes	Routine	Label	Ref
THE RETURN CODE TO 24, OTHERWISE, SET THE RETURN CODE TO 20. RETURN TO CALLER.			
05 SET THE RETURN CODE TO ZERO AND RETURN TO CALLER.			

Diagram 25.19.4 PUTONE (Part 1 of 1)



Notes	Routine	Label	Ref
01 OBTAIN THE LOCAL LOCK AND ISSUE A GETMAIN FOR A 2K BUFFER TO BE USED FOR VSAM I/O. RELEASE THE LOCAL LOCK. IF THERE IS A NON-ZERO RETURN CODE FROM GETMAIN, SET THE RETURN CODE TO 28 AND RETURN TO CALLER. IF THERE IS A ZERO RETURN CODE, SAVE THE POINTER TO THE GETMAINED AREA AND RECORD IT IN THE EPATH. IF THE INTERNAL RPL BUILT FLAG IS ON SKIP TO STEP 3.	SETLOCK GETMAIN		
02 ISSUE A GENCB TO BUILD AN RPL FOR A GET REQUEST FROM VSAM. IF THERE IS A NON-ZERO RETURN CODE FROM GENCB, SET THE RETURN CODE TO 20 AND RETURN TO CALLER. IF THERE WAS A ZERO RETURN CODE SET THE INTERNAL RPL BUILT FLAG.	GENCB		
03 ISSUE A VSAM GET PASSING THE RPL. IF THERE IS A NON-ZERO RETURN CODE FROM GET, ISSUE A SHOWCB TO DETERMINE THE TYPE OR ERROR. IF IT WAS RECORD NOT FOUND, SET THE RETURN CODE TO 08 AND RETURN TO CALLER. IF THE ERROR WAS ONE OF THE RETRYABLE ERRORS, GO BACK AND REISSUE THE GET. IF IT WAS NOT ONE OF THE RETRY ERRORS, SET THE RETURN CODE TO 20 AND RETURN TO CALLER.	SHOWCB		
04 SET RETURN CODE TO 0, AND RETURN TO CALLER.			

Diagram 25.19.5 GETONE (Part 1 of 1)

Recovery

ASM Recovery provides the mechanism to handle any errors that occur during normal ASM processing. Errors are classified into two groups. First, there are the errors in mainline processing that are detected during normal execution. These errors, sometimes referred to as determinate errors, normally do not prevent continuation of the ASM process in progress. The errors are recorded in SYS1.LOGREC and mainline processing resumes.

The second group of errors are the unexpected, or indeterminate errors. ASM Recovery itself first detects these errors. ASM Recovery attempts to determine the severity of the error in terms of the extent of damage to ASM control blocks and/or code and to the process in progress at the time of the error. Appropriate action is then taken. Possible actions that may be taken include recording the error with module identification and appropriate ASM status information, clean-up of ASM resources where possible, converting the error to a failure indication such as a return code to the caller of ASM, and terminating a task or address space if necessary.

For recovery purposes, ASM code has been divided into functional areas. Each recovery routine has primary responsibility for the mainline code it covers.

The functional areas of recovery are:

- I/O Control Modules and Page Operations Starter (ILRPOS)
- Swap Modules (ILRSWAP, ILRSWPDR)
- I/O Subsystem — front end (ILRPTM, ILRSRT)
- I/O Subsystem — back end (ILRCMP)
- Group Operations Starter (ILRGOS) and VIO Group Operators
- SRB Controller (ILRSRBC)
- Task Mode Release Processing (ILRTMRLG)
- Message Module (ILRMSG00)
- Address Space Termination (ILRTERMR)
- Job Termination (ILRJTERM)
- Page Expansion (ILRPGEXP)
- Special I/O to Page Data Sets (ILRPREAD)

The ASM recovery environment is established via the SETFRR or ESTAE macro. The task mode release processing recovery environment is established during system initialization and is always present. Issuance of the SETFRR or ESATE macro is held to a minimum to allow maximum recovery coverage with minimum overhead. Recovery environments are established only at external entry points to ASM.

Mainline ASM processing is tracked via the new ASM Tracking Area (ATA) and the Recovery Audit Trail Area (EPATH). The ATA is mapped onto the 24-byte area returned by the SETFRR macro. The module establishing the recovery environment dynamically obtains the EPATH. The ATA and EPATH will contain module, CSECT, and entry point data in addition to other data required for error recovery processing.

I/O Control Modules and Page Operation Starter (ILRPOS)

The I/O control FRR (ILRIOFRR) is the routine RTM calls whenever an error is encountered during ASM's swap processing, initial page processing, or page completion processing. This FRR is placed on the current stack if:

- ILRSWAP, ASM's swap controller, has been called by RSM;
- ILRPAGIO, ASM's page I/O controller, has been called by RSM or by ILRSWAP on a swap out request;
- ILRTRPAG (entry point in ILRPOS), ASM's transfer page routine, has been called by RSM;
- ILRPAGCM, ASM's page completion controller, has been called by the I/O subsystem for notification of I/O completion, or by the VIO SRB Controller and the front end of the I/O subsystem to handle errors;
- ILRSWPDR, ASM's swap driver, has been scheduled to start I/O to a swap data set.

The FRR consists of a mainline router and recovery subroutines for each of the ASM functions covered. The mainline receives control from RTM on an error. At this time, the SDWA contains information about the error, such as error type (program check, machine check, etc.), registers and PSW at the time of the error, and information about the mode of the system at the time of the error. The SDWA also contains the address of the ATA, the ASM tracking area mapped to the six-word parameter area provided by SETFRR. The mainline of the FRR uses the tracking information in the ATA to determine which ASM function was in control at the time of the error and then gives control to the recovery subroutine for this function. The mainline first performs common verifications and set up for the recording of the error. The functions identified in the mainline of this FRR include:

- ILRQIOE
- ILRSLSQA

- SWAPCOMP subroutine of ILRPAGCM
- ILRVIOCM
- PAGECOMP subroutine of ILRPAGCM
- ILRPOS
- ILRPAGIO
- ILRPAGCM
- ILRSWAP
- ILRTRPAG (entry point in ILRPOS)
- ILRSWPDR

Each recovery subroutine attempts recovery and/or clean-up of its resources and, if retry is desired, places the retry address in the SDWA. In the cases of ILRSLSQA, ILRSWAP, and ILRSWPDR, the subroutine calls one of the entries in ILRSWP01 to do the recovery. Some common clean-up is also performed in the mainline. The mainline completes the set-up for retry if retry has been requested.

Recovery for ILRQIOE begins by validity-checking the ASMSTAGQ and the AIA checkpointed in the ATA. If this AIA is valid, it is marked with the indeterminate error flag and queued to the AIA error queue in the PART. Any work already queued to the temporary write queques is then queued to the PART write queues. Finally, Part Monitor (ILRPTM) is scheduled if it is not already scheduled. The retry point is set to return to the caller.

Recovery for the SWAPCOMP subroutine of ILRPAGCM begins by validity-checking the SARWAITQ and the queue of remaining AIAs checkpointed in the ATA. If the AIA in the ATA is valid and is for a swap-in request, it is marked with the indeterminate error flag and queued to the internal PIOPQ (the internal queue of AIAs to be given to module IEAVPIOP). The ASMIORQC count is also increased. If the AIA is valid and is for a swap-out request, the AIA is marked with the indeterminate error flag and queued to the ASHCAPQ. Because an indeterminate error has occurred, the address space is then terminated. Retry is not attempted.

Recovery for ILRVIOCM begins by a 'TRAS' back to the current address space. The AIA checkpointed in the ATA is then validity checked. This AIA, or its related ACE, is dequeued from the LGEPROCQ. The SRB Controller is scheduled and the ASM class lock is freed, if held. Retry is not attempted. Recovery processing is completed by recovery for the PAGECOMP subroutine of ILRPAGCM.

Recovery for the PAGECOMP subroutine of ILRPAGCM begins by a 'TRAS' back to the current address space. On a 'TRAS' error, the address space involved is terminated. The in-process AIA queue, pointed to by the AIA checkpointed in the ATA, is validity checked. If the AIA in the ATA is valid, it is marked with the indeterminate error flag and queued to the internal PIOPQ (the internal queue of AIA's to be given to module IEAVPIOP). The ASMIORQC count is also increased. Retry is attempted when Recovery determines that ILRSLSQA can be called.

Recovery for ILRPOS begins by validity-checking the AIA/ACE checkpointed in the ATA. If it is a valid AIA, it is dequeued from the ASMSTAGQ, the ASMIORQR count is decreased, and the AIA is marked with the indeterminate error flag. If the ASM lock is held, the LGEPROCQ is validity checked and the AIA/ACE is dequeued from it. The ASM lock is freed if held. The retry point is set so that return is to the caller.

Recovery for ILRPAGIO begins by validity-checking the AIA checkpointed in the ATA. If this AIA is valid, it is marked with the indeterminate error flag and its address is placed in the work area for return to RSM. If the AIA is on the ASMSTAGQ, it is dequeued and the ASMIORQR count is decreased. The retry point is set to call ILRQIOE.

Recovery for ILRTRPAG entry point of ILRPOS begins by validity-checking the ACE checkpointed in the ATA. If it is a valid ACE, it is returned to its cell pool. If there is a related AIA, it is disconnected from the ACE. Retry is not attempted.

Recovery for ILRPAGCM is contained in the clean-up processing performed for all routines on the ILRPAGCM path. This clean-up consists of first placing any unprocessed AIAs on the AIA error queue in the PART and then scheduling Part Monitor (ILRPTM) to process these AIAs. The clean-up then attempts to return any completed AIAs to RSM either by retrying at the call to IEAVPIOP (if retry is permitted) or by calling IEAVPIOP directly.

Swap Modules (ILRSWAP, ILRSWPDR)

ILRIOFRR passes control to swap recovery (ILRSWP01) to process errors that occur in ASM's swapping path. This module has three entry points: ILRSWP01 for swap driver recovery, ILRCSWAP for

front end swap processor recovery, and ILRCSLSQ for swap LSQA processor recovery.

Recovery for ILRSWPDR
Swap driver recovery processes all errors that occur in ASM's swap driver, ILRSWPDR, as non-retryable. All swap recovery routines receive control from ILRIOFRR. The control blocks associated with the error are validity-checked. Unprocessed swap sets are returned to work queues for future processing. Resources such as the IORB are freed. If a SARTE has been checkpointed by swap driver, it is unlocked. Swap driver's SRB is rescheduled to ensure continuity for ASM swap processing. There is special processing that will rebuild one IORB and chain it to the SARTE when a SARTE's last IORB fails validity checking or when processing an ASM-issued '084' abend. A SARTE (swap data set) is unusable without an IORB.

Recovery for ILRSWAP
Swap processor recovery processes all errors which occur in ASM's swap processor, ILRSWAP. The ASMHD swap queue is validity checked. Since the swap request being processed has not yet been merged with other swap requests on ASM's internal queues, a retry is set up into ILRSWAP to return this swap request to RSM.

Recovery for ILRSLSQA
Swap LSQA recovery processes all errors that occur in ASM's swap LSQA processor, ILRSLSQA. It is convenient to identify three stages of swap processing: AIAs on the ASMHD swap queue, essentially just starting swap processing; AIAs on the SART wait queue, grouped for SWAP processing and containing assigned LSIDs; and single AIAs connected to SCCWs ready to be sent to the swap driver for I/O processing.

Swap LSQA recovery identifies the stage of processing at the time of error and validity-checks the control blocks and queues being processed. The swap AIAs are returned to the appropriate queues or to RSM via address space termination if the error precludes the successful completion of the swap request. Unused SCCWs are returned to the SARTE SCCW available queue. The retry address and registers are set up in the SDWA at a point in ILRSLSQA where swap AIAs returned to queues by recovery are reprocessed.

I/O Subsystem — Front End (ILRPTM, ILRSRT)

ILRSRT01 is the FRR for both ILRPTM and ILRSRT. It is made active by ILRPTM and it remains active until ILRSRT and ILRPTM processing is complete and ILRPTM deletes it. The major objective of this FRR is to get rid of invalid or loop-causing control blocks that ILRPTM and ILRSRT were using at the time of the error that caused ILRSRT01 to be invoked. Another objective is to restructure the environment so that any remaining requests will be properly processed by ILRPTM and ILRSRT.

I/O Subsystem — Back End (ILRCMP)

ILRCMP01 is the recovery routine for ILRCMP, the I/O completion routine. I/O completion consists of four entry points — ILRCMPDI, the DIE exit; ILRCMPAE, the abnormal end appendage; ILRCMPNE, the normal end appendage; and ILRCMP, the termination routine. The recovery routine attempts to clean up whatever resources have been checkpointed in the ATA and force reprocessing for any requests not yet attempted. For ILRCMP, the termination routine, the SRB is scheduled so that ILRCMP can complete its processing. For the other entry points (ILRCMPDI, ILRCMPNE and ILRCMPAE), percolation causes the IOS FRR to get control and force a X'45' to the termination routine.

Group Operation Starter (ILRGOS) and VIO Group Operators

ILRGOS01 is the recovery routine for ILRGOS and its paths to VSAM; ILRGOS calls the group operators ILRSAV, ILRRLG, and ILRACT, which call ILRVSAMI. ILRGOS01 serves as an ESTAE for Save and Activate requests and an FRR for Release Logical Group and Assign requests. It only retries for record-only abends. For all other errors, the resources are freed and the error percolated. Eventually ILRJTERM will clean up at job termination and ILRTERMR will clean up at address space termination.

If the error occurred during an Assign request, any storage obtained on behalf of the request is freed. Since no ACE is created for the Assign request, there is no trace of the request once recovery has completed. Percolation to VBP allows VBP to take care of the ACA.

If the error occurred during an Activate request, any storage obtained on behalf of the request is freed. The ACE is returned to the pool and there is no trace of the request once recovery has

completed. Percolation to VBP allows VBP to take care of the ACA.

If the error occurred during a Release Logical Group request, the work-pending flag in the LGE is turned off. ILRSRBC does not look at this LGE because the work-pending flag is off. ILRJTERM does not process this LGE because the Release Logical Group flag in the LGE remains on. ILRTERMR gets control at memory termination and cleans up resources for the ASPCT. If the ASPCT had been saved, a Release Logical Group request is queued to ILRTMRLG's request queue in the ASMVT.

If the error occurred during a Save request, all LPME's are marked as unsaved if EPAUNSAV is on or if the save flag in the ASPCT is off. This allows slots to be freed up during later clean-up processing. If the save flag is off in the ASPCT, the 'S' symbol is set to zero so that a later release request for the LGE will be honored. The ACE is dequeued from the LGE so that there is no trace of this Save request. The work-pending flag in the LGE is turned off if there is no remaining work on the LGE. The Save-request-queued flag in the LGE is turned off if there are no more Save requests queued for this LGE. The group-operations-in-process flag in the LGE is turned off. The ACA is then returned to the available pool. Further processing can be done for this LGE and ILRJTERM issues a Deactivate request for this LGE at job termination in order to clean up.

SRB Controller (ILRSRBC)

SRB Controller Recovery (ILRSRB01) processes all errors that occur in ASM's SRB Controller, ILRSRBC, or in either of the two ILRPOS subroutines (ILRESTRT and ILRTRANS), or in ILRRLG when called by ILRSRBC. The internal queues of AIAs and ACEs, and the ASM Header LGE queue are validity-checked. Startable AIAs and group operation ACEs are set up for reprocessing by the SRB controller. The ATA, which checkpoints critical ASM control blocks, is copied into the SWDA. Resources such as ILRSRBC's and ILRRLG's workarea cells are freed. SRB controller's SRB is rescheduled to ensure continuity for ASM processing in the address space.

The only 'non-retryable' error is one that causes truncation of the ASM Header LGE queue. Because the extent of damage cannot be ascertained and ASM cannot handle future requests for the missing LGES, the address space is terminated.

Task Mode Release Processing (ILRTMRLG)

ILRTMI01, an ESTAE established in ILRTMRLG, is basically recovery for two mainline functions — processing in ILRTMI00 to complete ASM initialization and processing in ILRTMRLG to erase saved ASPCTs from SYS1.STGINDEX and release the slots assigned to the ASPCTs.

If ILRTMI01 is entered due to a failure in ILRTMI00, retry is attempted at the next logical process in ILRTMI00. If, however, RTM does not pass an SDWA to ILRTMI01, then the system is in serious condition since it cannot get 512 bytes of storage (the size of an SDWA) when very few or no other system functions are concurrently executing. In this case ILRTMI01 percolates, causing the Master Scheduler Initialization Task to terminate the IPL.

If the error occurred while the ILRTMRLG main function was executing, every effort is made to keep the task for ILRTMRLG from being terminated, since this task is initiated only once per IPL. Retry in this case is always into ILRTMRLG where it will get the next work element (ACE) off its queue, if there is one, or go into its normal wait, waiting for more work to be queued.

If the error occurred in ILRVSAMI (called by ILRTMRLG), the retry is made into ILRTMRLG, unless it is a record-only abend situation, in which case the retry is made into ILRVSAMI.

Message Module (ILRMSG00)

ILRMSG01 gets control when an error occurs in the ILRMSG00 system termination subroutine. ILRMSG01 loads a wait state PSW.

Address Space Termination (ILRTERMR)

TERMFRR is the recovery routine for ILRTERMR. If ILRTERMR got an error while working on a queue it calls the appropriate queue verification routine. If retry is possible, TERMRFRR attempts to retry at the next retry point in the module.

Job Termination Resource Manager (ILRJTERM)

ILRJTM01 is the recovery FRR for ILRJTERM. The error is recorded in SYS1.LOGREC. If the error is retryable, a retry is requested at a point in ILRJTERM where an SRB for ILRSRBC is scheduled to the address space owning the VIO data set.

Page Expansion (ILRPGEXP)

ESTAER is the recovery routine for ILRPGEXP. It gets control on errors from ILRPGEXP mainline, ILROPS00 or ILRPREAD. If the error occurred while reading or writng ILRTPARB a message is sent to the operator. In all cases control blocks are cleaned up and freed.

Special I/O to Page Data Sets (ILRPREAD)

ESTAEXIT gets control if an error occurs while ILRPREAD is trying to read or write the ILRTPARB. ESTAEXIT frees storage obtained from SQA and returns to RTM. RTM will then give control to ESTAER, the ESTAE routine for ILRPGEXP.

Recovery Service Routine Module (ILRFRR01)

The FRR service routine module contains routines used by the other ASM recovery routines. There are three types of service routines contained in this module: 1) queue verfication routines, 2) control block verification routines, and 3) a PURGEDQ resource manager termination routine. The verification routines verify (and correct, when possible) queues and control blocks that might have been affected by an error that occurred during ASM's processing. This prevents an invalid queue or control block from possibly causing another error during later ASM processing. The following queues have been identified for verification in the case of an error: the ASM staging queue (ASMSTAGQ), and LGE process queue (LGEPROCQ), the SART wait queue (SARWAITQ), a queue of AIA's, a queue of swap AIAs, a queue of SCCWs, a queue of PCCWs, the RSM local I/O queue (RMSLIOQ — a queue of PCBs), a queue of ACEs, and a queue of IOEs. The following control blocks have been identified for verification: the AIA, the ACE, the LGE, the PCB, the SCCW, the PCCW, the IOE, and the IORB-IOSB-SRB combination.

The queue verification routines all have similar methods of operation. They all use the general supervisor queue verification routines to actually verify and correct the queue. Thus, each ASM queue verification routine initializes the parameter list for the general queue verifier with those parameters applicable to its particular queue. The appropriate queue verifier entry point is then called. There is one queue verifier for each type of queue verified: 1) a single-threaded, single-headed queue, 2) a single-threaded, double-headed queue, and 3) a double-threaded, double-headed queue.

The PCB/AIA verification routine begins by checking that the PCB/AIA can be referenced. Then the storage pointed to by PCBASCB checked to see if it can be referenced. Finally, AIAOP is tested for the correct operations code (X'00').

The ACE verification routine begins by checking that the ACE can be referenced. Then the storage pointed by ACELGE is tested to check that it is a valid LGE. Finally, a test is made to check that the LGID in the ACE matches the LGID in the LGE.

The LGE verification routine begins by checking that the LGE can be referenced. Then the value of LGELGID is tested to ensure that it is less than the maximum LGVMAXLG. Finally, a test is made to check that LGVTE indexed by the LGID does point to the LGE.

The SCCW verification routine begins by checking that the SCCW can be referenced, is in the nucleus buffer area, and contains the SCCW identifier. Then tests are made to check that SCCWSEEK contains the seek command code and that SCCWSSEC contains the set-sector command code.

The PCCW verification routine begins by checking that the PCCW can be referenced, is in the nucleus buffer area, and contains the PCCW identifier. Then, tests are made to check that PCCWSRCH contains the TIC command code.

The IOE verification routine begins by checking that the IOE can be referenced. Then the value of IOEAIA is tested. If it is non-zero, the AIA verification routine is used to check that the storage is a valid AIA.

The IORB-IOSB-SRB verification routine begins by checking that the IORB-IOSB-SRB combination can be referenced. Then the IORB storage is checked for the IORB identifier and to ensure that IORPARTE points to a valid PARTE. Finally, the IOSB storage is checked for the correct driver ID and the correct ASID. If all verifications are successful, the constant fields in the IORB-IOSB-SRB are refreshed.

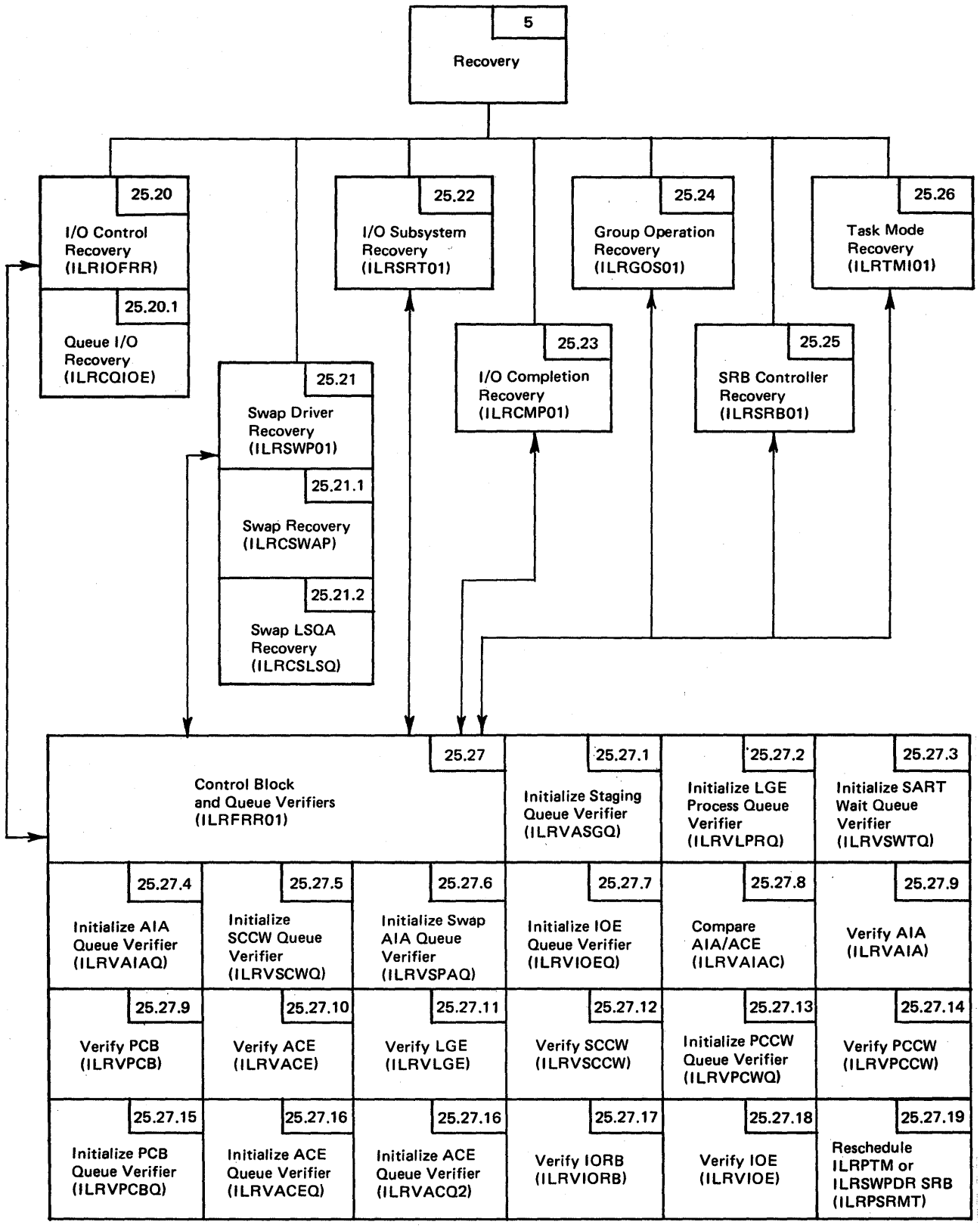
The PURGEDQ resource manager termination routine protects the SRBs for Part Monitor and Swap Driver by rescheduling them if they are ever purged.

Type of Recovery	Recovery Routine	Code Covered
FRR	ILRCMP01	ILRCMP
	ILRGOS01[1]	ILRGOS
	ILRGOS01[1]	ILRRLG
	ILRIOFRR	ILRPAGCM
	ILRIOFRR	ILRPAGIO
	ILRIOFRR	ILRPOS
	ILRCQIOE (ILRIOFRR entry)	ILRQIOE (ILRPAGIO entry)
	ILRIOFRR	ILRVIOCM
	ILRJTM01[2]	ILRJTERM
	ILRMSG01[2]	ILRMSG00
	ILRSRB01	ILRSRBC
	ILRSRT01	ILRPTM
	ILRSRT01	ILRSRT
	ILRSWP01	ILRSWPDR
	ILRCSLSQ (ILRSWP01 entry)	ILRSLSQA (ILRSWAP entry)
	ILRCSWAP (ILRSWP01 entry)	ILRSWAP
	TERMRFRR[2]	ILRTERMR
ESTAE	ESTAER[2]	ILRPGEXP
	ESTAEXIT[2]	ILRPREAD
	ILRGOS01[1]	ILRACT and ILRVSAMI
	ILRGOS01[1]	ILRSAV and ILRVSAMI
	ILRTMI01	ILRTMRLG and ILRVSAMI

[1] ILRGOS01 is both an FRR and an ESTAE.

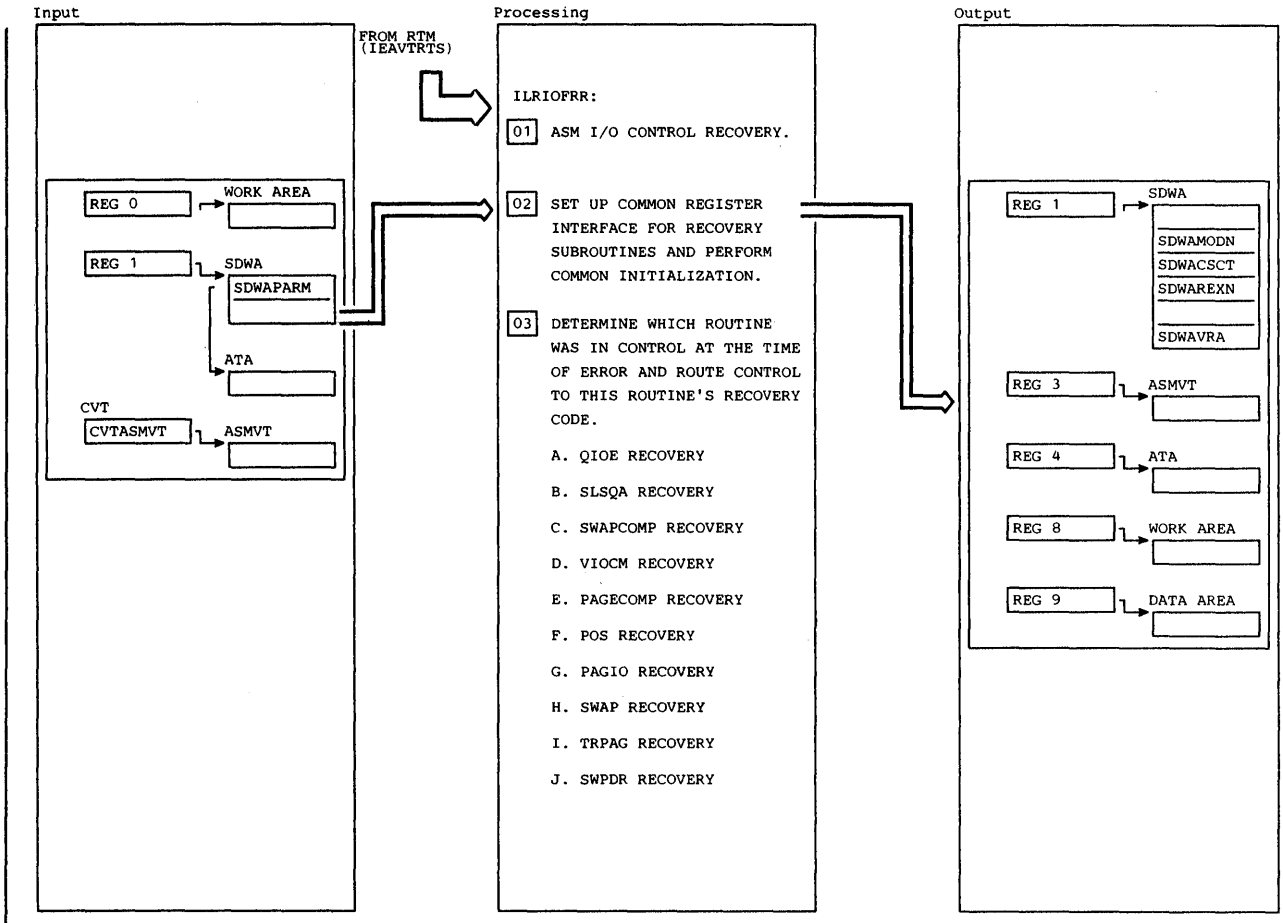
[2] An alternate entry within the module, for which it provides recovery. The MO is with this module's MO, not with the group of recovery routine MOs.

Figure 2-60A. Recovery Routines



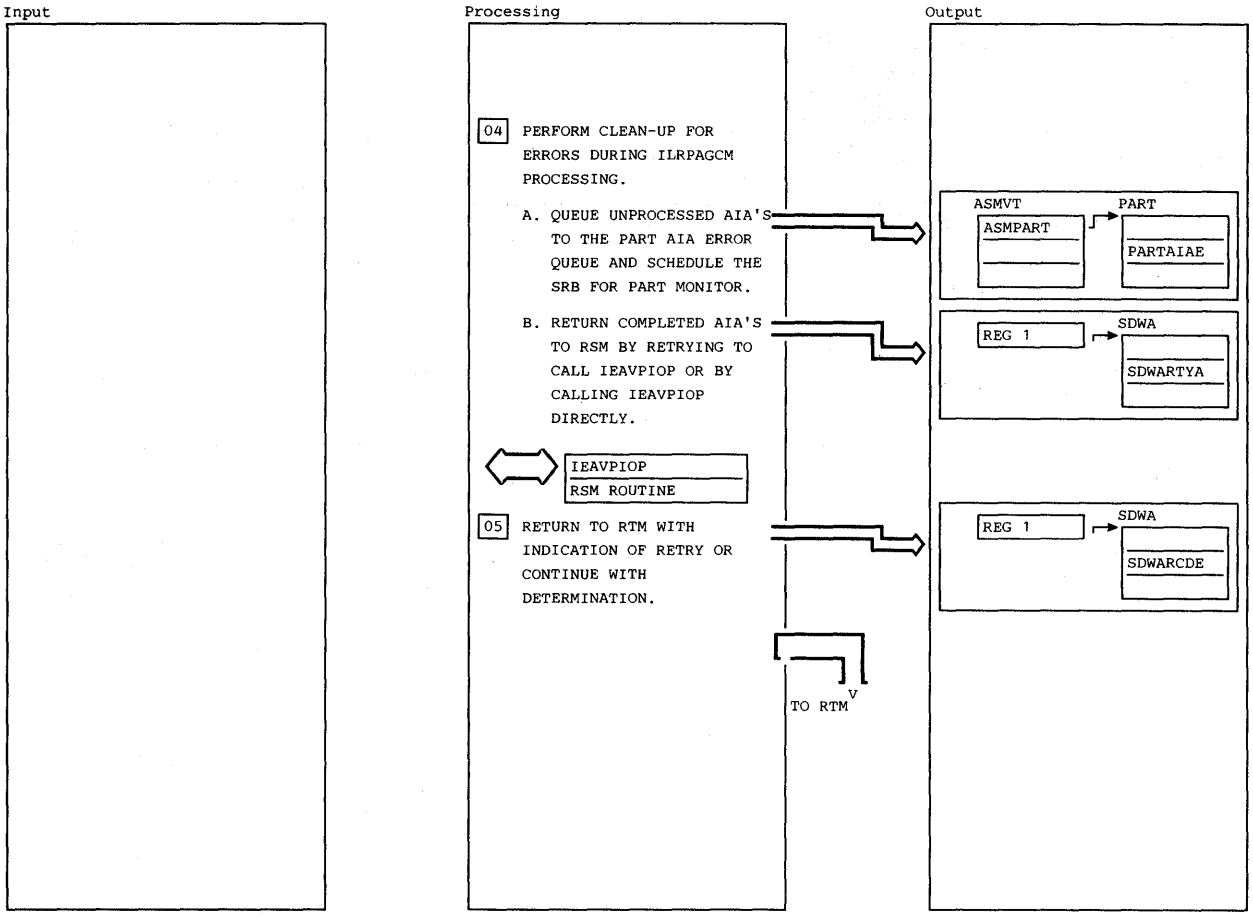
25.x. — Module
25.x.y. — Entry point in module 25.x.

Figure 2-61. Recovery Overview



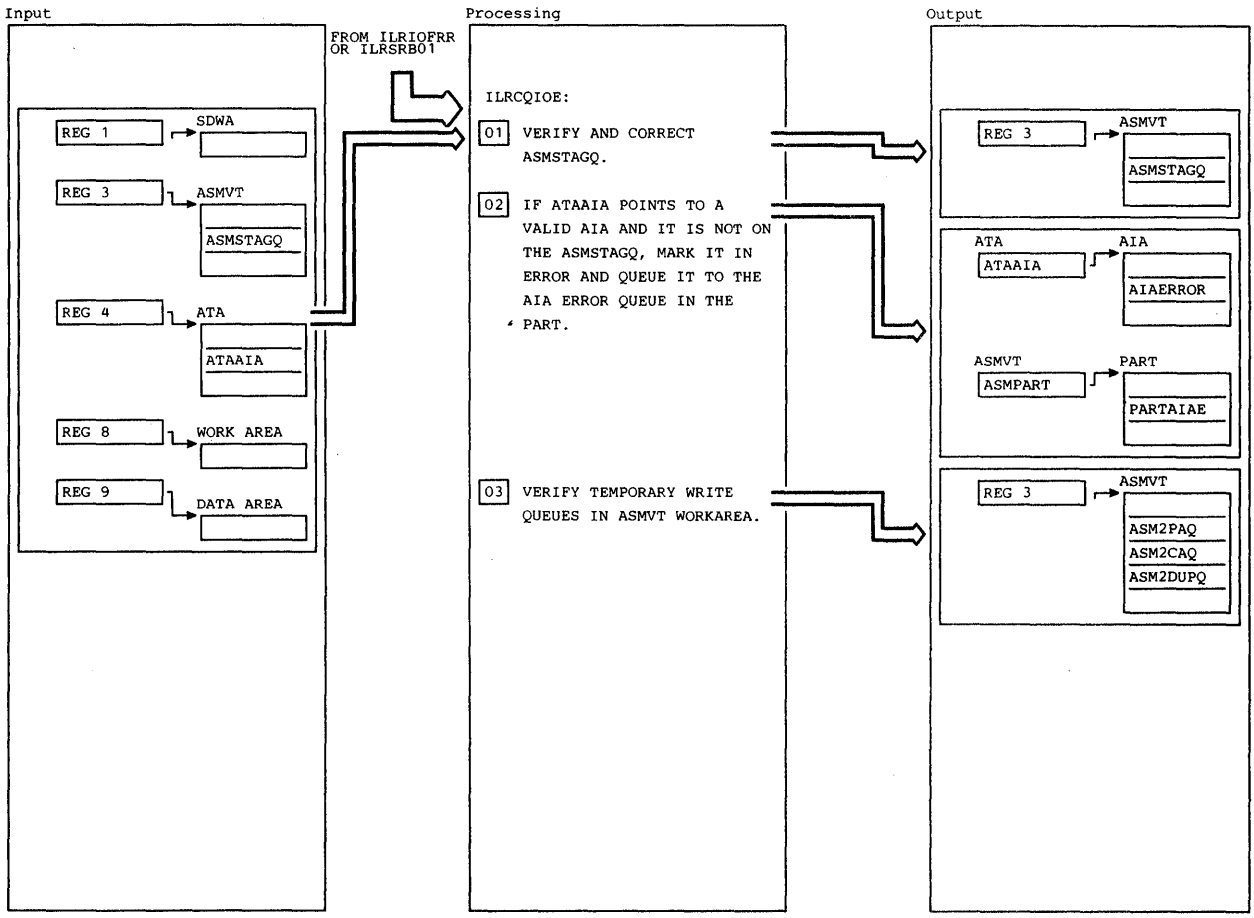
Notes	Routine	Label	Ref
01 ILRIOFRR IS CALLED BY RTM ANYTIME AN ERROR OCCURS DURING ASM'S SWAP PROCESSING, INITIAL PAGE PROCESSING, AND PAGE COMPLETION PROCESSING. ILRIOFRR CALLS ILRSWP01 IF THE ERROR OCCURRED DURING SWAP PROCESSING. OTHERWISE, ILRIOFRR CALLS INTERNAL SUBROUTINES.			
02 PLACE NECESSARY POINTERS IN REGISTERS TO STANDARDIZE THE INTERFACE TO THE RECOVERY SUBROUTINE. INITIALIZE THE PARAMETERS FOR RECORDING AND COPY THE FRR PARAMETER AREA, THE ATA, INTO THE VARIABLE RECORDING AREA SO THAT THE ATA AT THE TIME OF THE ERROR IS RECORDED. ALSO PERFORM VERIFICATION OF COMMON QUEUES.			
03 THE FRR USES THE SECTION FLAGS IN THE ATA TO DETERMINE WHICH ROUTINE WAS IN CONTROL AT THE TIME OF THE ERROR. THE FLAG IS TURNED ON WHEN THE ROUTINE IS ENTERED, AND OFF WHEN THE ROUTINE EXITS. THE SECTION FLAGS ARE TESTED IN THE REVERSE OF THE ORDER IN WHICH THE ROUTINES ARE CALLED TO DETERMINE WHICH ROUTINE WAS 'CURRENT' WHEN THE ERROR OCCURRED.			
A. INVOKE ILRQIOE RECOVERY.		ILRCQIOE	25.20.1
B. INVOKE ILRSLSQA RECOVERY.	ILRSWP01	ILRCSLSQ	25.21.2
C. INVOKE SWAPCOMP RECOVERY.		RECSCOMP	25.20.2
D. INVOKE ILRVIOCM RECOVERY.		RECVIOCM	25.20.3
E. INVOKE PAGECOMP RECOVERY.		RECPCOMP	25.20.4
F. INVOKE ILRPOS RECOVERY.		RECPOS	25.20.5
G. INVOKE ILRPAGIO RECOVERY.		RECPAGIO	25.20.6
H. INVOKE ILRSWAP RECOVERY.	ILRSWP01	ILRCSWAP	25.21.1
I. INVOKE ILRTRPAG RECOVERY.		RECTRPAG	25.20.7
J. INVOKE ILRSWPDR RECOVERY.	ILRSWP01	ILRSWP01	25.21

Diagram 25.20 ILRIOFRR (Part 1 of 2)



Notes	Routine	Label	Ref
04 PERFORM COMMON CLEAN-UP IF RETRY HAS NOT BEEN REQUESTED OR IF RETRY IS NOT ALLOWED. A. COLLECT ANY UNPROCESSED AIA'S FROM THE ATA AND FROM THE ASMVT WORKAREA FOR ILRPAGCM. IF THERE ARE ANY, PLACE THEM ON THE PART AIA ERROR QUEUE AND SCHEDULE THE SRB FOR PART MONITOR, IF NOT ALREADY SCHEDULED, TO PROCESS THESE AIA'S. B. IF THERE ARE ANY COMPLETED AIA'S TO BE RETURNED TO RSM, ATTEMPT TO RETRY TO RETURN THEM TO RSM. ALSO SET THE ILRPAGCM RECURSION FLAG. IF RETRY IS NOT ALLOWED, OR IF THE ILRPAGCM RECURSION FLAG IS SET, ATTEMPT TO RETURN THE AIA'S TO RSM FROM FRR. THIS IS NOT ALWAYS DONE BECAUSE SUCH A CALL MIGHT CAUSE A RECURSIVE PROBLEM.	IEAVPIOP	IEAVPIOP	
05 RETRY IS ATTEMPTED UNLESS PROHIBITED BY RTM OR UNLESS INFORMATION NECESSARY FOR RETRY IS NOT AVAILABLE.			

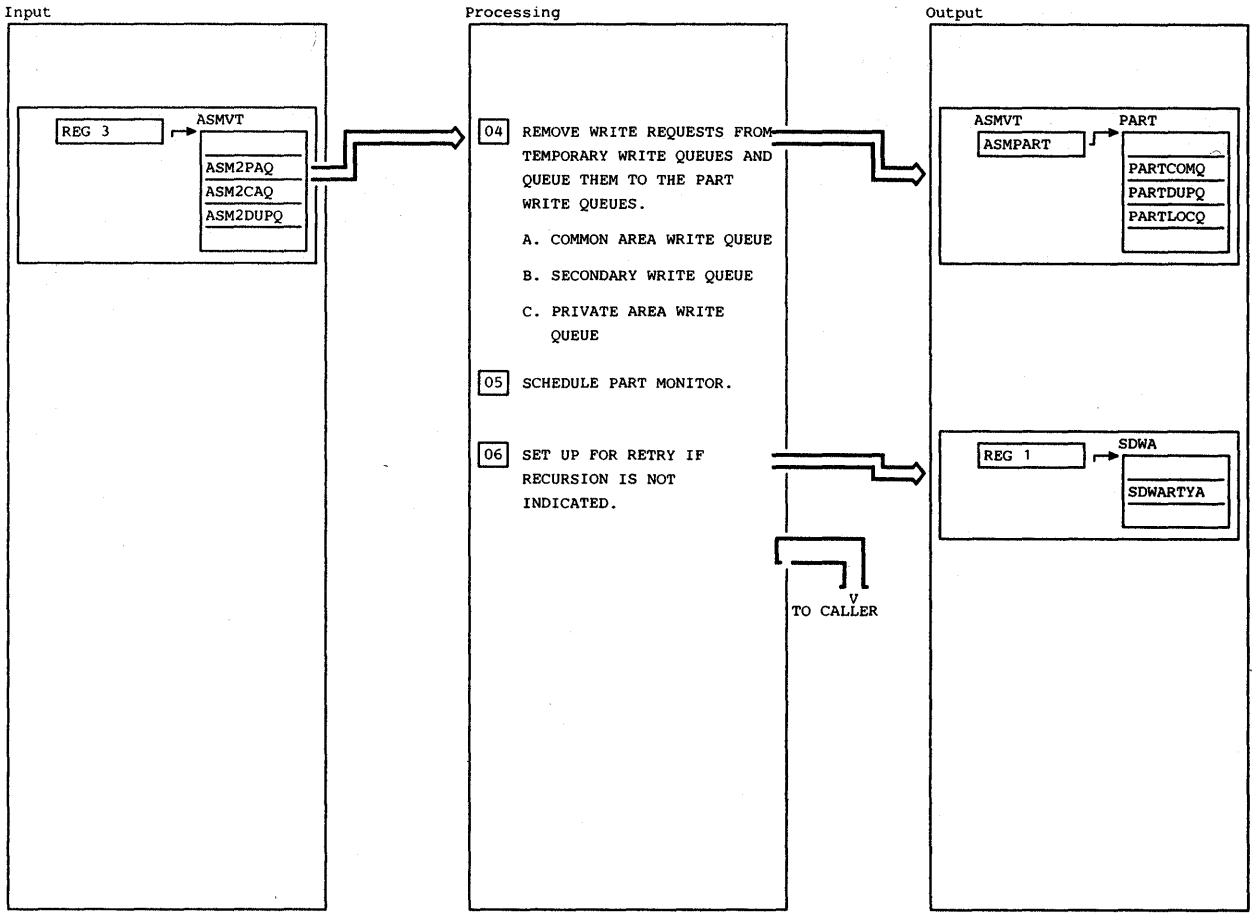
Diagram 25.20 ILRIOFRR (Part 2 of 2)



Notes	Routine	Label	Ref
01 VERIFY AND RECONSTRUCT THE ASMSTAGQ.	ILRFRR01	ILRVASGQ	25.27.1
02 VERIFY THAT ATAAIA POINTS TO A VALID AIA. IF IT DOES, A FURTHER CHECK IS MADE TO SEE IF THIS AIA IS STILL ON THE ASMSTAGQ WAITING TO BE PROCESSED (IF IT WAS, ITS ADDRESS WOULD BE IN ASMSTAGF). IF IT IS NOT ON THE ASMSTAGQ (ALREADY BEING PROCESSED), THE ERROR FLAG IN THE AIA IS TURNED ON AND THE AIA IS QUEUED TO THE ERROR AIA QUEUE IN THE PART, PARTAIAE.	ILRFRR01	ILRVAIA	25.27.9
03 VERIFY AND CORRECT THE TEMPORARY WRITE QUEUES IN THE ASMVT WORKAREA.	ILRFRR01	ILRVIOEQ	25.27.7

Notes	Routine	Label	Ref

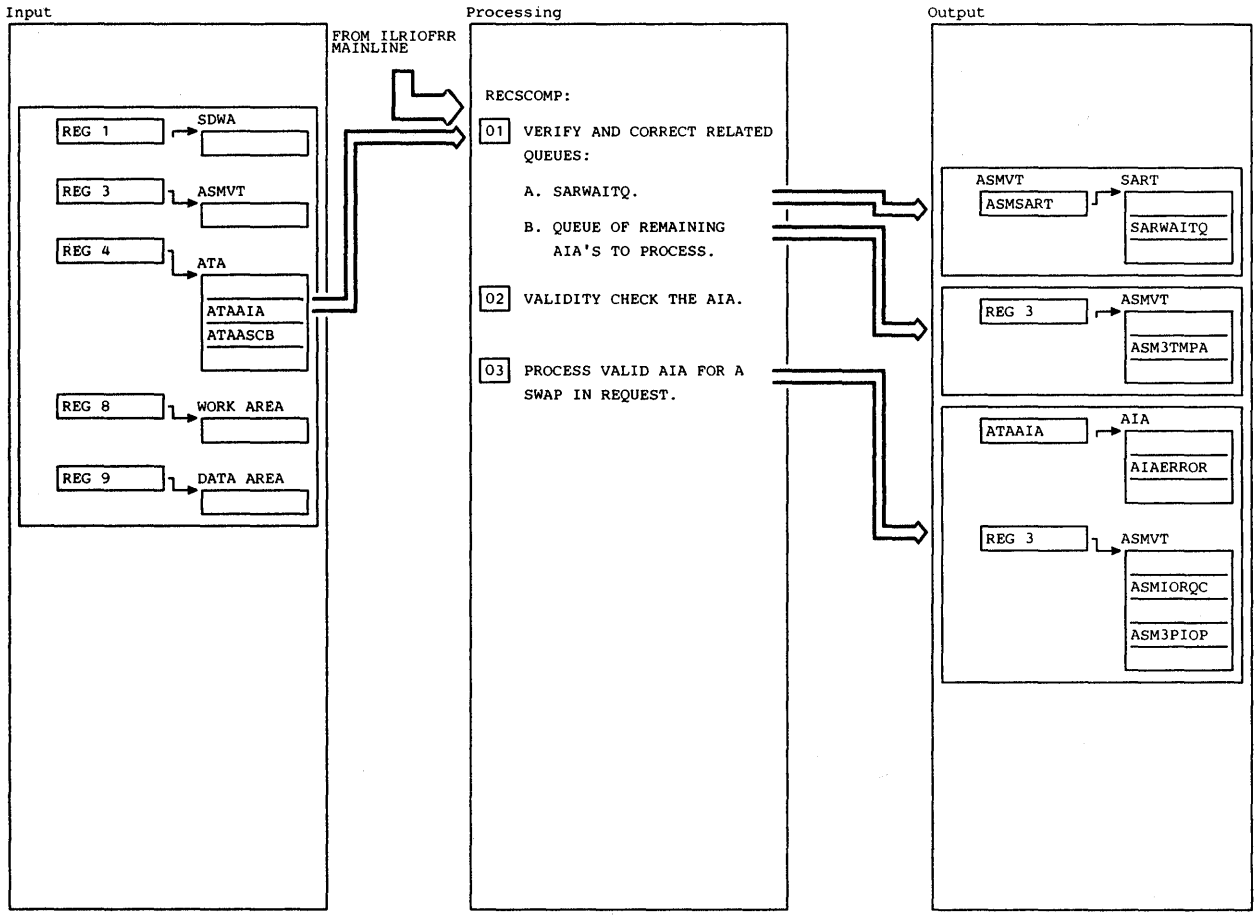
Diagram 25.20.1 ILRCQIOE (Part 1 of 2)



Notes	Routine	Label	Ref
04 IF THERE ARE ANY REQUESTS ON THE TEMPORARY WRITE QUEUES, THE ASM LOCK THAT SERIALIZES THE PART QUEUES MUST BE OBTAINED IF IT IS NOT ALREADY HELD. THE WRITE REQUESTS ARE THEN QUEUED TO THE APPROPRIATE PART WRITE QUEUE. THE ASM CLASS LOCK IS FREED - IT IS FREED EVEN IF IT HAD ALREADY BEEN HELD SINCE RTM DOES NOT FREE LOCKS ON RETRY. SERIALIZATION IS MAINTAINED SINCE THE SALLOC LOCK IS STILL HELD.			
05 PART MONITOR IS SCHEDULED, IF IT ISN'T ALREADY SCHEDULED, TO HANDLE ANY REQUESTS THAT MIGHT HAVE BEEN QUEUED.			
06 IF THE ILRQIOE RECURSION INDICATOR IS NOT SET, THE RETRY ADDRESS IN THE SDWA, SDWARTYA, IS SET TO THE ADDRESS IN ILRQIOE AT WHICH 'RETURN TO THE CALLER' IS PERFORMED. THE ILRQIOE RECURSION INDICATOR IS ALSO SET.			

Notes	Routine	Label	Ref

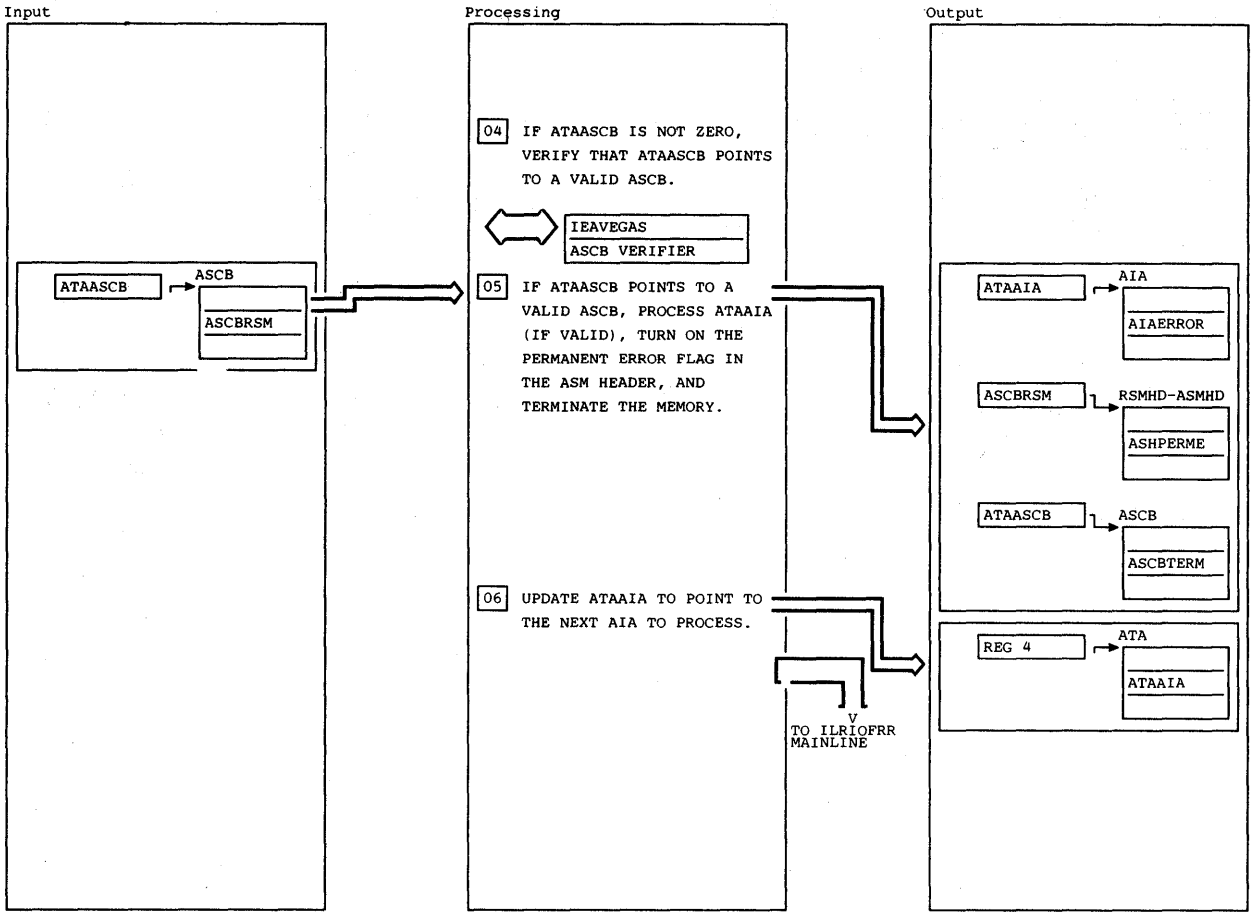
Diagram 25.20.1 ILRCQIOE (Part 2 of 2)



Notes	Routine	Label	Ref
01 VERIFY AND CORRECT THE QUEUES THAT MIGHT HAVE BEEN AFFECTED BY AN ERROR DURING SWAPCOMP PROCESSING:			
A. THE SARWAITQ.	ILRFRR01	ILRVSWTQ	25.27.3
B. THE QUEUE OF REMAINING AIA'S (POINTED TO BY ASM3TMPA).	ILRFRR01	ILRVSPAQ	25.27.6
02 VALIDITY CHECK THE AIA POINTED TO BY ATAAIA.	ILRFRR01	ILRVAIA	25.27.9
03 IF ATAAIA POINTS TO A VALID AIA FOR A SWAP IN REQUEST (ATAASCB=0), THE INDETERMINATE ERROR FLAG, AIAERROR, IS TURNED ON, THE AIA IS PLACED ON THE INTERNAL QUEUE OF AIA'S TO BE RETURNED TO RSM (IF NOT ALREADY THERE), AND THE COUNT OF COMPLETED REQUESTS IN THE ASMVT, ASMIORQC, IS INCREMENTED. SET UP FOR ADDRESS SPACE TERMINATION BY PUTTING THE ASCB ADDRESS IN ATAASCB AND TURNING ON THE SWAP IN FLAG.			

Notes	Routine	Label	Ref

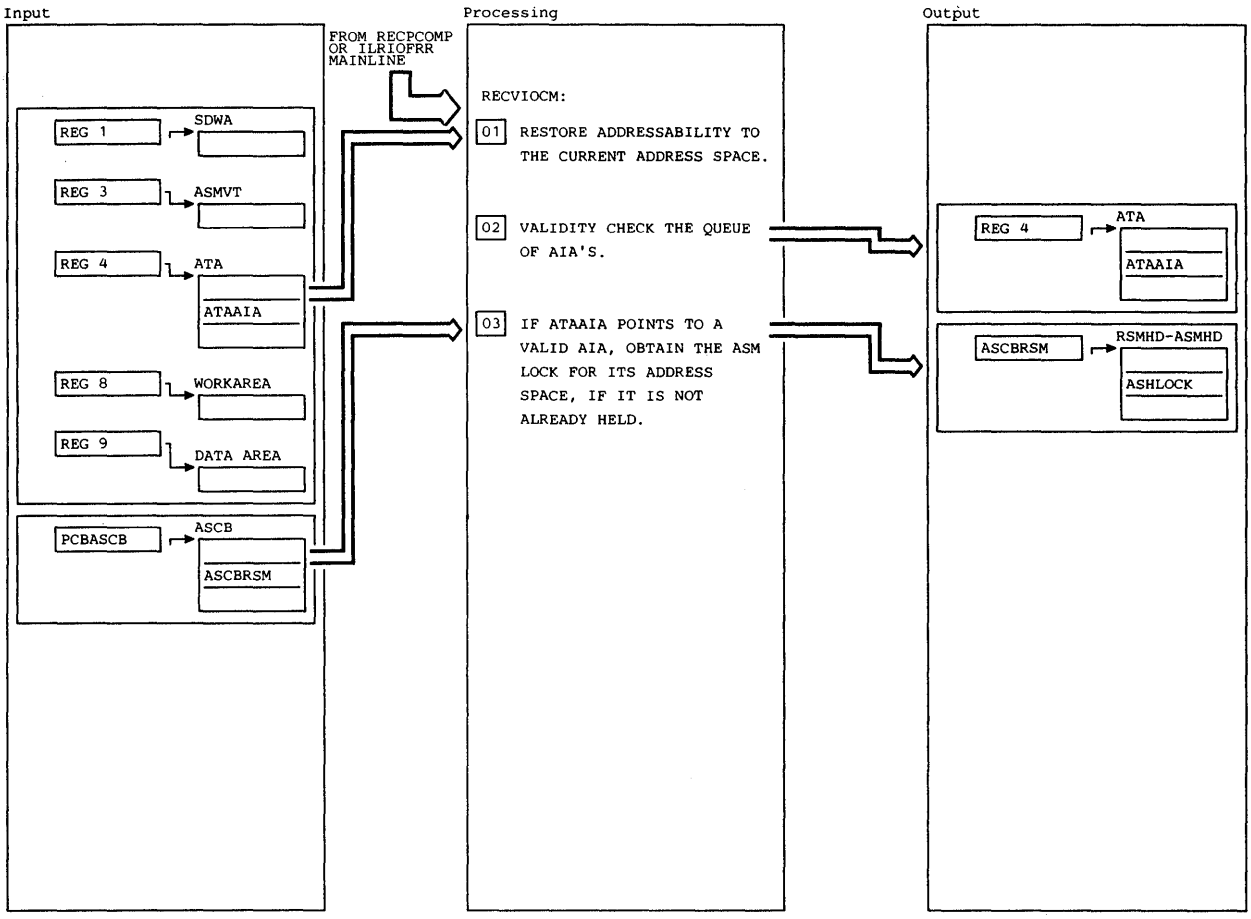
Diagram 25.20.2 RECSCOMP (Part 1 of 2)



Notes	Routine	Label	Ref
04 IF ATAASCB IS NOT ZERO, VERIFY THAT ATAASCB POINTS A VALID ASCB.	IEAVEGAS	IEAVEGAS	
05 IF ATAASCB POINTS TO A VALID ASCB:			
A. IF IT IS A SWAP OUT REQUEST, VALIDITY CHECK THE ASHCAPQ. THEN IF ATAAIA POINTS TO A VALID AIA, TURN ON THE INDETERMINATE ERROR FLAG, AIAERROR, AND PLACE THE AIA ON THE ASHCAPQ. (ASHCAPQ IS A QUEUE OF COMPLETED SWAP-OUT REQUESTS FOR AN ADDRESS SPACE. WHEN ALL THE REQUESTS COMPLETE, THEY ARE RETURNED TO RSM.) TURN ON THE PERMANENT ERROR FLAG, ASHPERME, TO PREVENT SWAPCOMP FROM EVER CALLING IEAVSWPC NORMALLY.	ILRFRR01	ILRVSPAQ	25.27.6
B. IF A SWAP-IN REQUEST,SET THE RSMFAIL FLAG.			
C. BECAUSE AN INDETERMINATE ERROR HAS OCCURRED, TERMINATE	MEMTERM		

Notes	Routine	Label	Ref
THE ADDRESS SPACE.			
06 UPDATE ATAAIA TO POINT TO THE NEXT AIA (IN ASM3TMPA). THE REMAINING AIA'S WILL BE CLEANED-UP BY THE PROCESSING IN THE MAINLINE OF ILRIOFRR.			

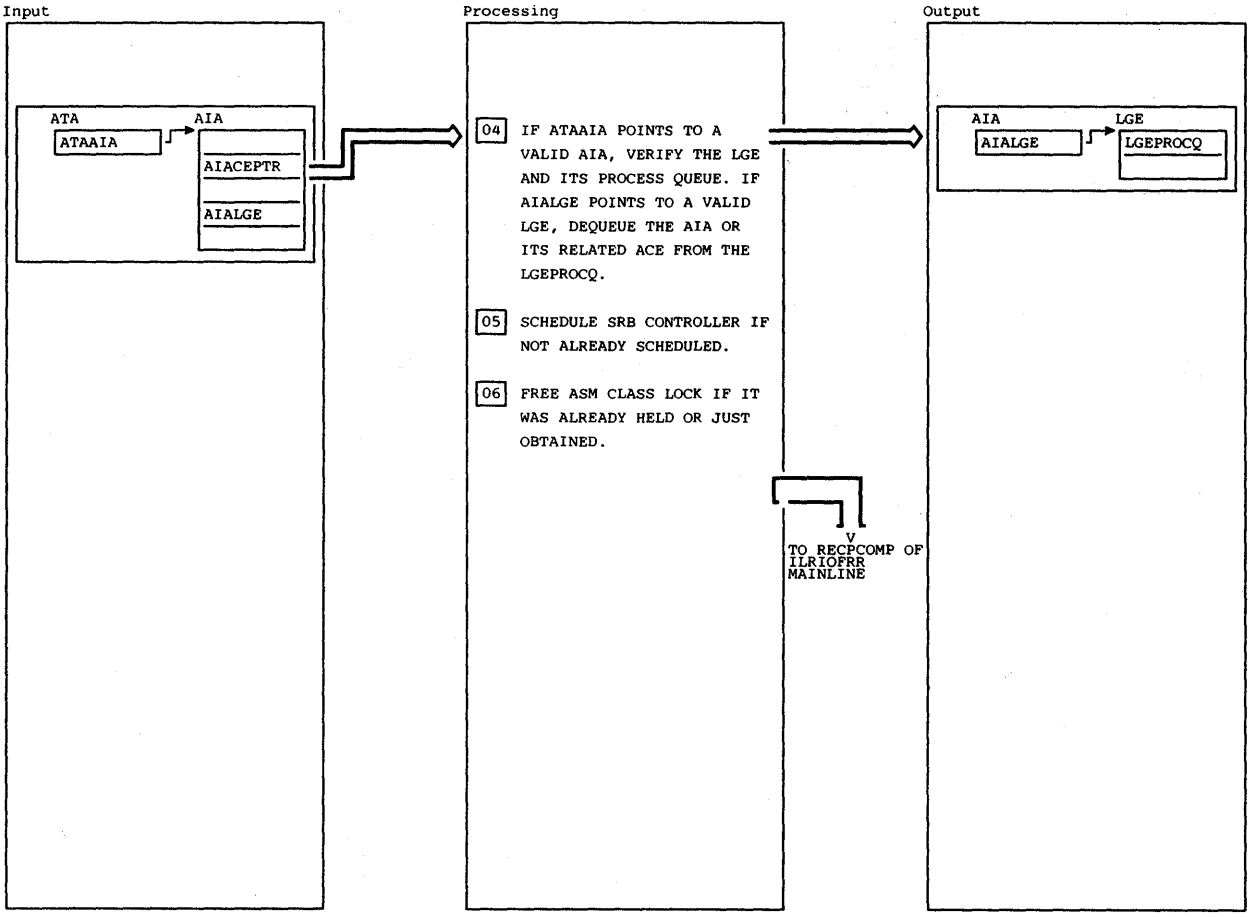
Diagram 25.20.2 RECSCOMP (Part 2 of 2)



Notes	Routine	Label	Ref
01 ISSUE A TRAS BACK TO RESTORE THE STOR (SEGMENT TABLE ORIGIN REGISTER) OF THE CURRENT MEMORY. THIS IS NECESSARY IF THE ERROR OCCURRED DURING A TRAS OPERATION FOR ALL ERRORS EXCEPT A DAT ERROR (RESTORE HAS BEEN DONE BY RTM).			
02 VALIDITY CHECK THE QUEUE OF AIA'S BEING PROCESSED, STARTING WITH ATAAIA.	ILRFRR01	ILRVAIAQ	25.27.4
03 IF ATAAIA POINTS TO A VALID AIA (NON-ZERO ADDRESS), USE THE LOCK WORD IN THE ASM HEADER, ASHLOCK, TO OBTAIN THE ASM CLASS LOCK FOR THE RELATED ADDRESS SPACE.			

Notes	Routine	Label	Ref

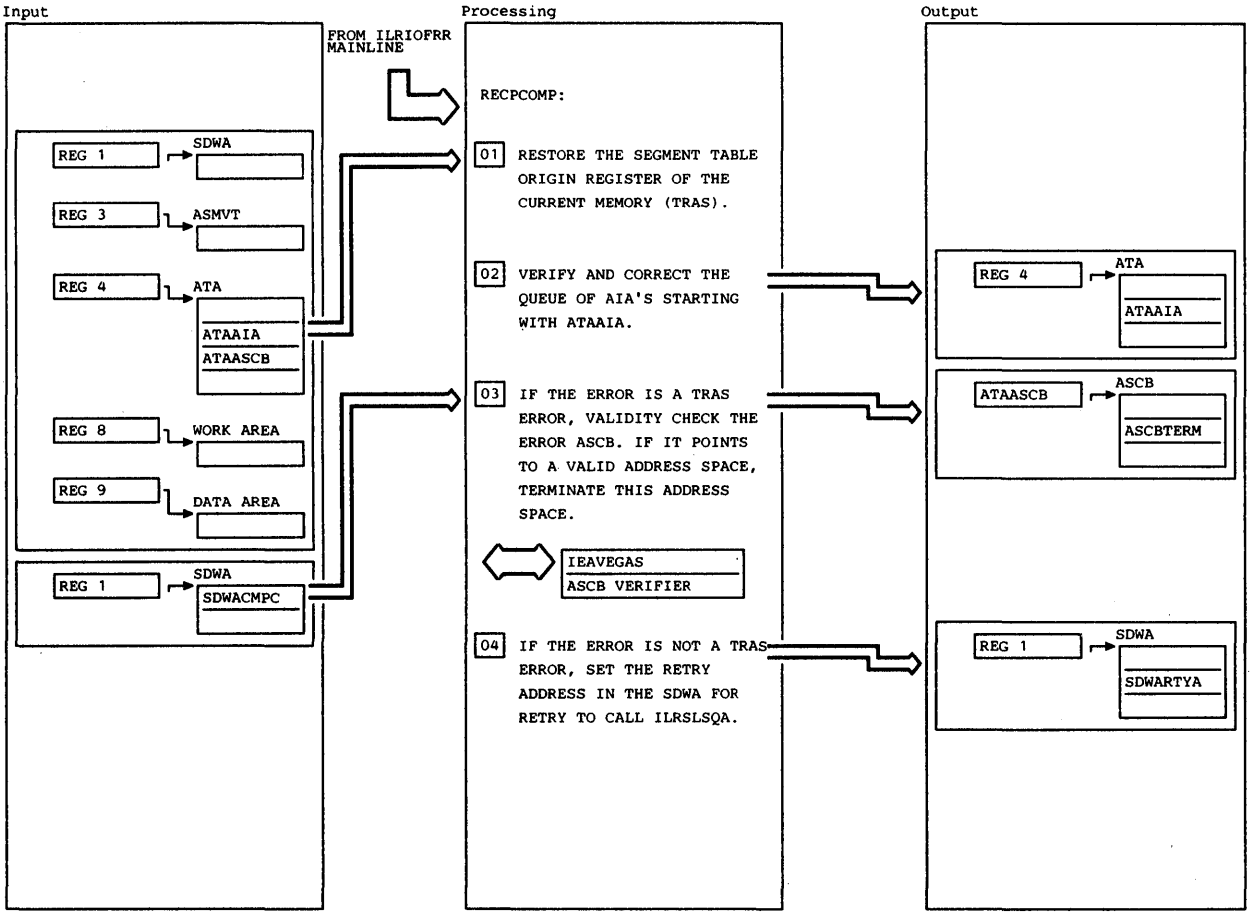
Diagram 25.20.3 RECVIOCM (Part 1 of 2)



Notes	Routine	Label	Ref
04 IF ATAAIA POINTS TO A VALID AIA, THEN AIALGE AND ITS LGEPROCQ ARE VALIDITY CHECKED. IF AIALGE POINTS TO A VALID LGE, THE AIA, OR ITS RELATED ACE (IF AIATRPSP IS ON), IS DEQUEUED FROM THE LGEPROCQ. IF ANY VERIFICATION FAILS, OR IF THE AIA OR ACE WAS NOT FOUND ON THE LGEPROCQ, THE AIA MUST NOT BE RETURNED TO RSM SINCE THE AIA, OR A RELATED ACE, MIGHT STILL BE ON SOME LGEPROCQ WAITING TO BE PROCESSED. SO AN INTERNAL FLAG IS SET TO PREVENT FURTHER PROCESSING OF THIS AIA.	ILRFRR01	ILRVLPRQ	25.27.2
05 SCHEDULE THE SRB CONTROLLER IF IT IS NOT ALREADY SCHEDULED. THIS IS ONLY DONE IF THE ASM LOCK IS HELD.			
06 A TEST IS MADE TO SEE IF THE ASM LOCK IS HELD. THIS TEST IS MADE WHETHER OR NOT ATAAIA POINTS TO A VALID AIA. IF THE ASM LOCK IS HELD, IT IS UNCONDITIONALLY FREED BEFORE CONTROL IS GIVEN TO THE RECPCOMP ROUTINE TO COMPLETE RECOVERY PROCESSING.			

Notes	Routine	Label	Ref

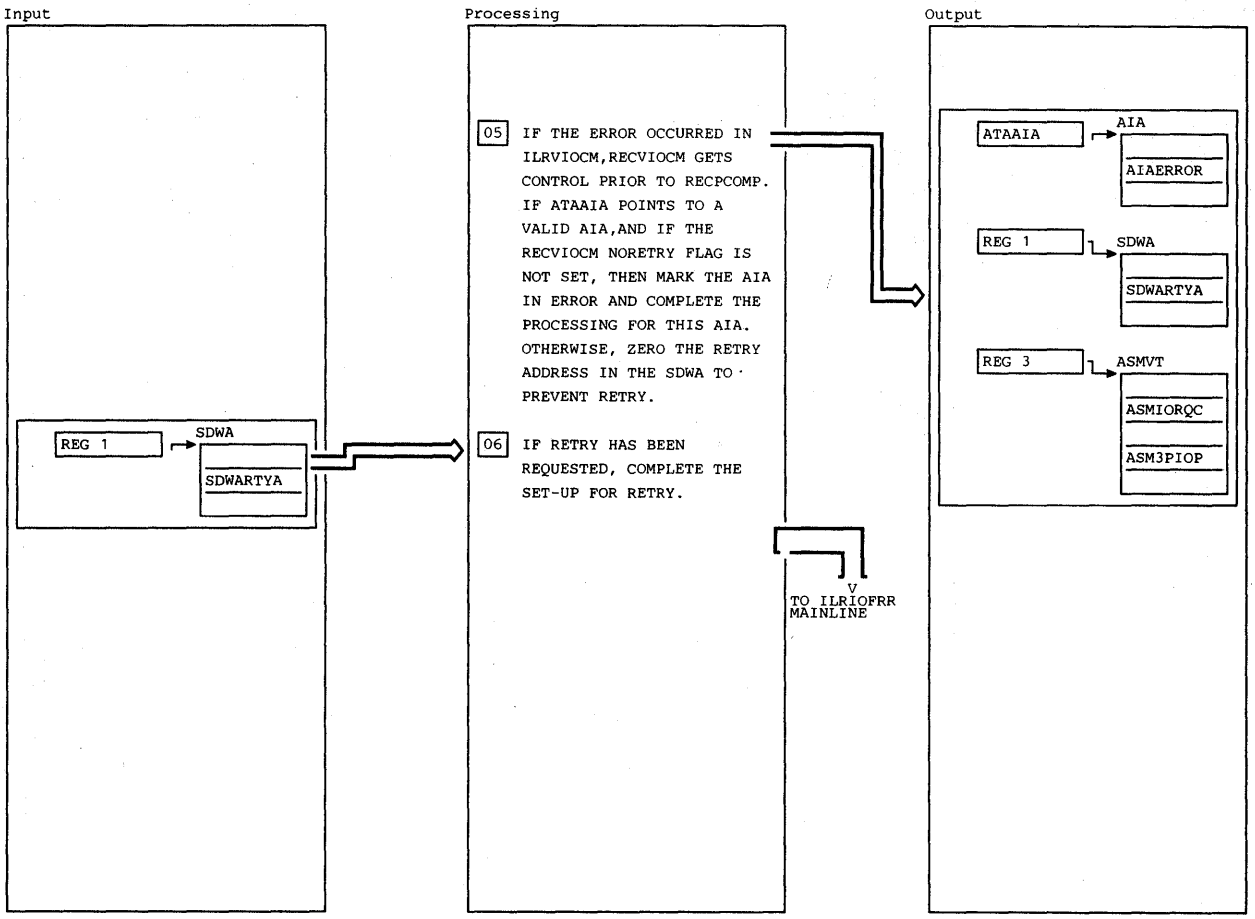
Diagram 25.20.3 RECVIOCM (Part 2 of 2)



Notes	Routine	Label	Ref
01 ISSUE A TRAS BACK TO RESTORE THE SEGMENT TABLE ORIGIN REGISTER (STOR) OF THE CURRENT MEMORY. THIS IS NECESSARY IF THE ERROR OCCURRED DURING A TRAS OPERATION FOR ALL ERRORS EXCEPT A DYNAMIC ADDRESS TRANSLATION (DAT) ERROR (RESTORE HAS BEEN DONE BY RTM).			
02 VERIFY AND CORRECT THE QUEUE OF AIA'S BEING PROCESSED (POINTED TO BY ATAAIA).	ILRFRR01	ILRVAIAQ	25.27.4
03 IF ERROR IS A TRAS ERROR (A DAT ERROR DURING A TRAS OPERATION INDICATED BY A UNIQUE COMPLETION CODE IN THE SDWA), USE MEMTERM TO TERMINATE THE ERROR ADDRESS SPACE (THE ASCB ADDRESS WAS TRACKED IN ATAASCB), IF THE ADDRESS SPACE IS VALID.			
04 FOR NON-TRAS ERRORS, THE MEMORY IS STILL IN PROCESS, SO THE SDWA IS SET FOR RETRY TO THE POINT IN PAGECOMP WHERE THE CALL TO ILRSLSQA IS MADE.			

Notes	Routine	Label	Ref

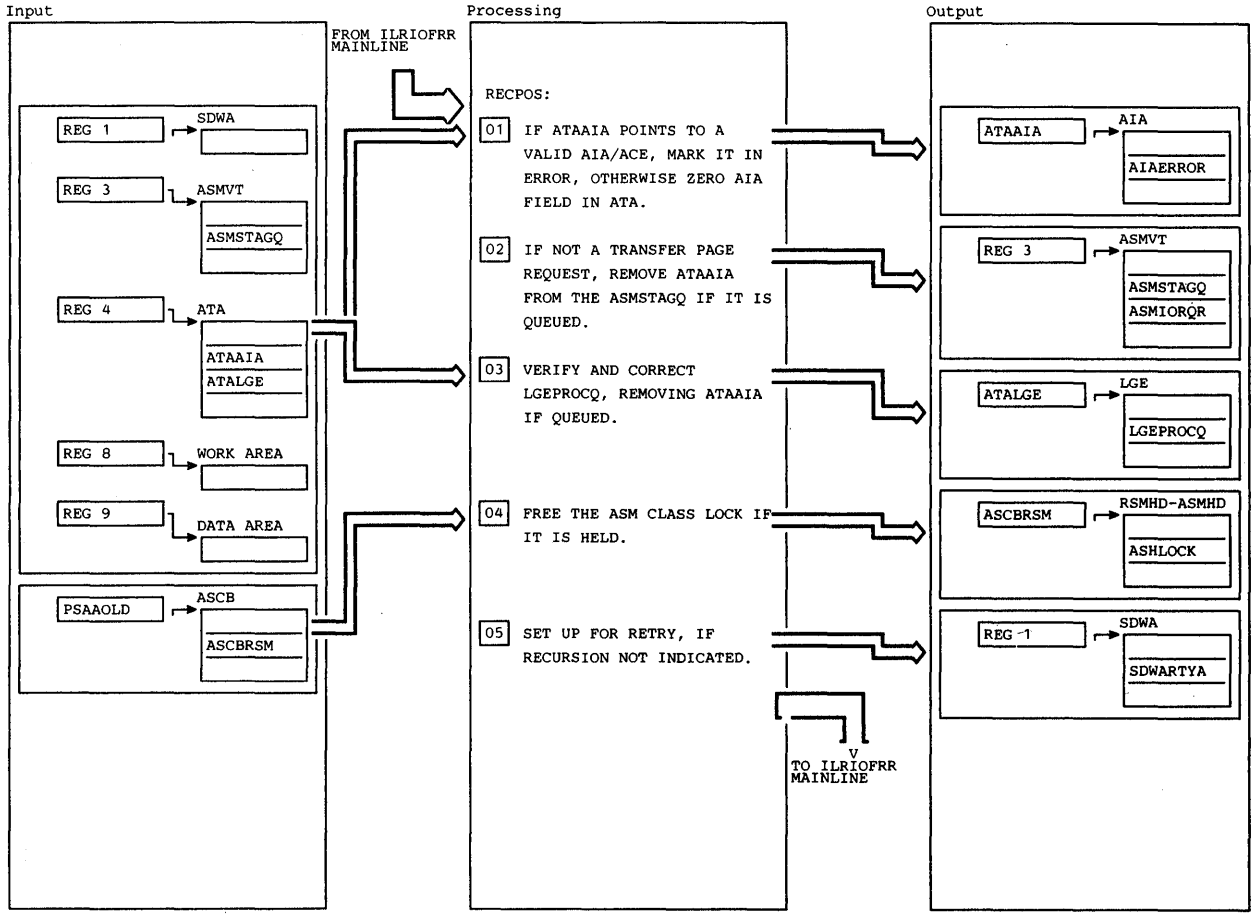
Diagram 25.20.4 RECPCOMP (Part 1 of 2)



Notes	Routine	Label	Ref
05 IF ATAAIA POINTS TO A VALID AIA (NON-ZERO ADDRESS), AND RECVIOCM PROCESSING HAS NOT INDICATED THAT THIS AIA IS NOT TO BE PROCESSED, THE INDETERMINATE ERROR FLAG, AIAERROR, IS TURNED ON AND THE AIA IS PLACED ON THE INTERNAL QUEUE OF AIA'S FOR RSM. THE VALUE OF ATAAIA AND THE ASMIORQC COUNT ARE ALSO UPDATED. OTHERWISE, THE RETRY ADDRESS IN THE SDWA IS ZEROED TO PREVENT RETRY.			
06 IF THE RETRY ADDRESS IN THE SDWA IS NOT ZERO, COMPLETE THE RETRY SET-UP FOR RETRY TO CALL ILRSLSQA.			

Notes	Routine	Label	Ref

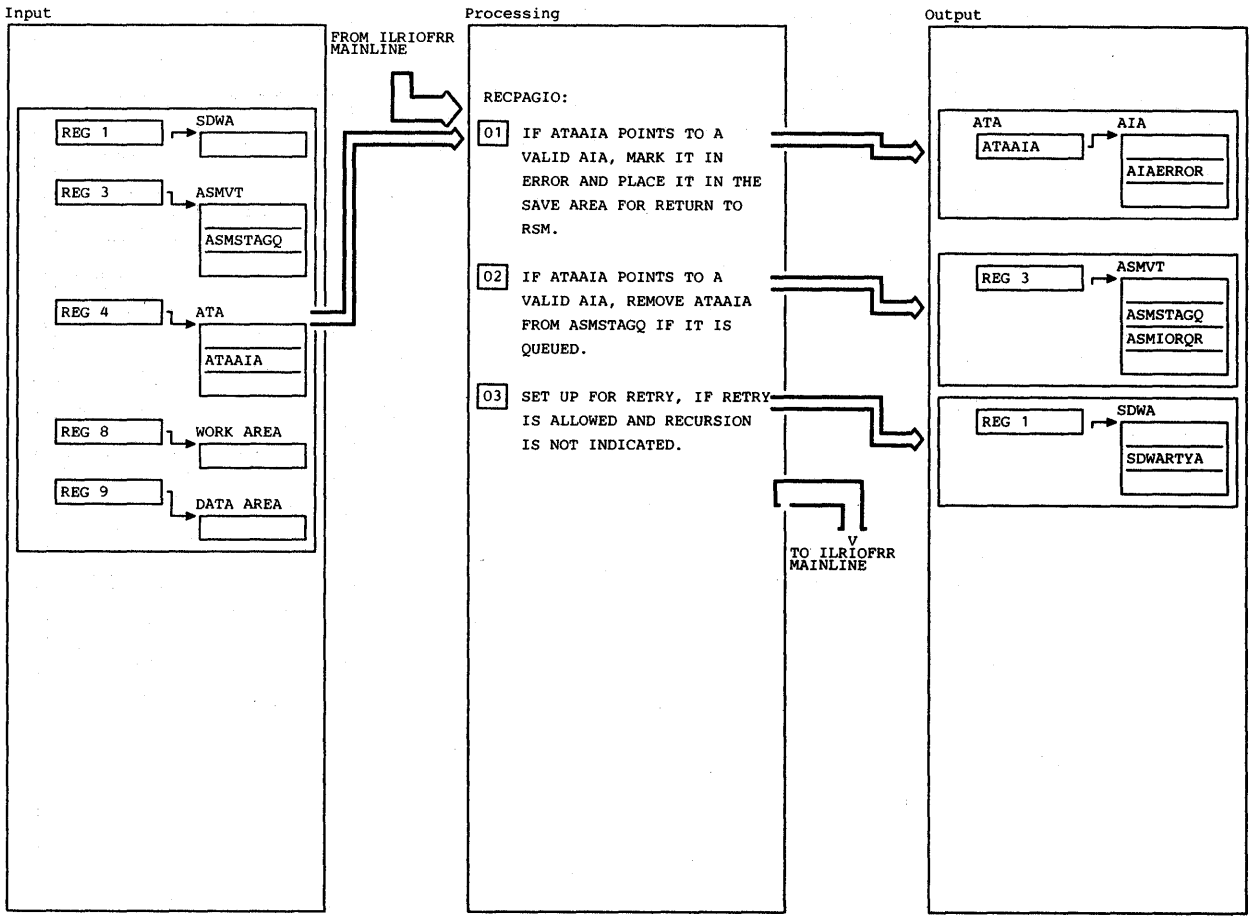
Diagram 25.20.4 RECPCOMP (Part 2 of 2)



Notes	Routine	Label	Ref
01 IF ILRPOS WAS CALLED FOR A TRANSFER PAGE REQUEST, VERIFY THAT ATAAIA POINTS TO A VALID ACE. OTHERWISE VERIFY THAT ATAAIA POINTS TO A VALID AIA, AND THEN TURN ON THE INDETERMINATE ERROR FLAG, AIAERROR. IF ATAAIA DOES NOT POINT TO A VALID AIA OR ACE, THE FIELD IN THE ATA IS ZEROED.	ILRFRR01 ILRFRR01	ILRVACE ILRVAIA	25.27.10 25.27.9
02 IF ILRPOS WAS NOT CALLED FOR A TRANSFER PAGE REQUEST (INDICATED BY THE ATA FLAGS) AND THE AIA POINTED TO BY ATAAIA IS QUEUED TO THE ASMSTAGQ, THE AIA IS DEQUEUED AND THE COUNT OF RECEIVED REQUESTS IN THE ASMVT, ASMIORQR, IS DECREMENTED.			
03 VERIFY THE LGEPROCQ. IF ATALGE IS A VALID LGE DEQUEUE ATAAIA FROM THE LGEPROCQ IF IT IS QUEUED.	ILRFRR01	ILRVLPRQ	25.27.2
04 THE ASM LOCK FOR THE CURRENT ADDRESS SPACE IS FREED, IF IT HAD BEEN HELD BY ILRPOS.			

Notes	Routine	Label	Ref
05 IF THE ILRPOS RECURSION INDICATOR IS NOT SET, THE RETRY ADDRESS IN THE SDWA, SDWARTYA, IS SET TO THE ADDRESS IN ILRPOS AT WHICH 'RETURN TO THE CALLER' IS PERFORMED. THE ILRPOS RECURSION INDICATOR IS ALSO SET. IF THE RECURSION INDICATOR IS ALREADY SET,NO RETRY IS ATTEMPTED.			

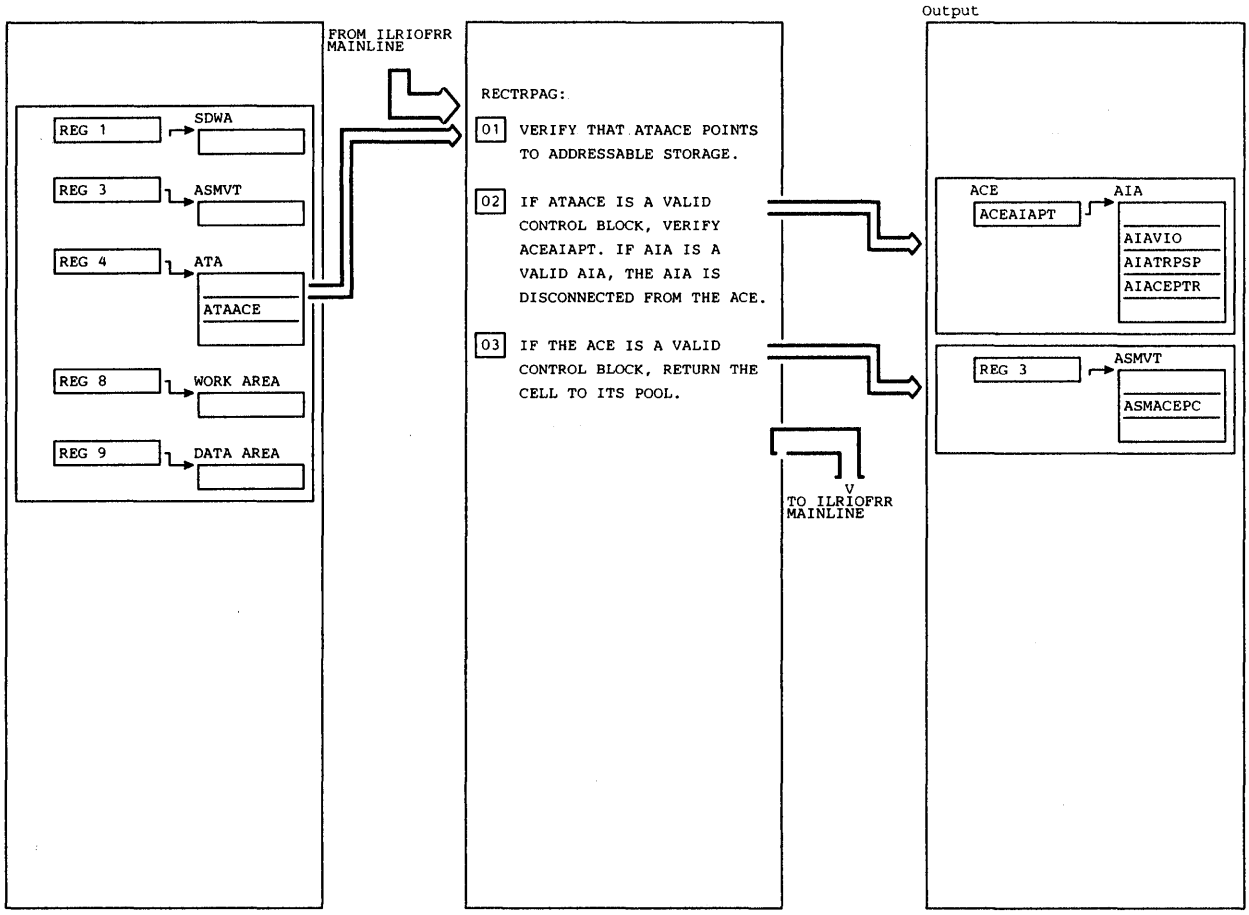
Diagram 25.20.5 RECPOS (Part 1 of 1)



Notes	Routine	Label	Ref
01 VERIFY THAT ATAAIA POINTS TO A VALID AIA. IF IT DOES, THE ERROR FLAG IN THE AIA IS TURNED ON, AND THE VALUE OF ATAAIA IS PLACED IN THE SAVE AREA SO IT WILL BE RETURNED TO RSM. IF ATAAIA DOES NOT POINT TO A VALID AIA, THE VALUE IN THE SAVE AREA REMAINS ZERO TO PREVENT ASM FROM RETURNING AN INVALID ADDRESS TO RSM.	ILRFRR01	ILRVAIA	25.27.9
02 IF THE AIA POINTED TO BY ATAAIA IS QUEUED TO THE ASMSTAGQ, IT IS DEQUEUED AND THE COUNT OF RECEIVED REQUESTS IN THE ASMVT, ASMIORQR, IS DECREMENTED.			
03 IF THE ILRPAGIO RECURSION INDICATOR IS NOT SET,THE RETRY ADDRESS IN THE SDWA, SDWARTYA, IS SET TO THE ADDRESS IN ILRPAGIO AT WHICH THE CALL TO ILRQIOE IS MADE. THE ILRPAGIO RECURSION INDICATOR IS ALSO SET. IF THE RECURSION INDICATOR IS ALREADY SET, NO RETRY IS ATTEMPTED.			

Notes	Routine	Label	Ref

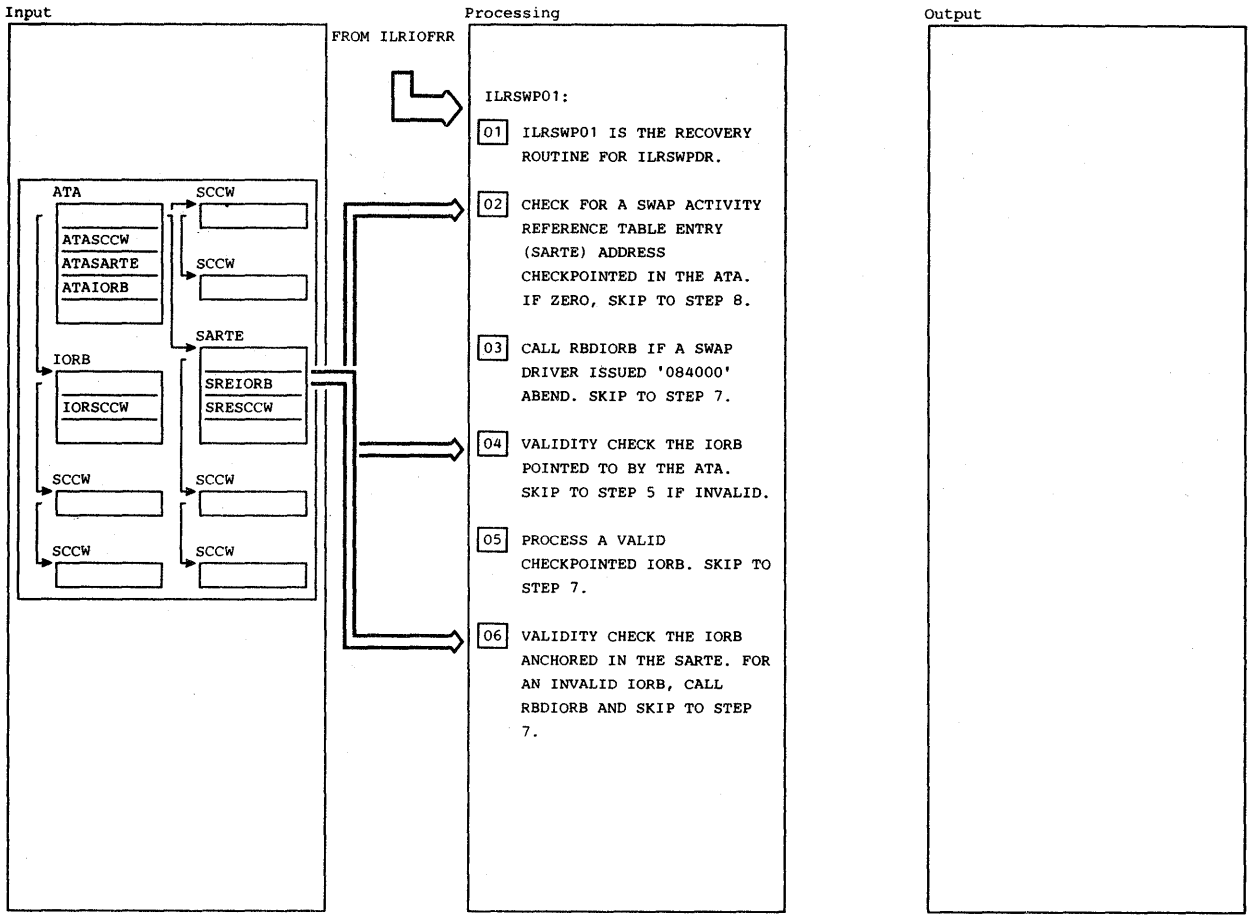
Diagram 25.20.6 RECPAGIO (Part 1 of 1)



Notes	Routine	Label	Ref
01 IF ATAACE IS NON-ZERO, VERIFY THAT THE ENTIRE STORAGE OF THE ACE CAN BE REFERENCED.	IEAVEADV	IEAVEADV	
02 IF ATAACE IS VALID, THE AIA POINTED TO BY ACEAIAPT IS VALIDITY CHECKED. IF IT IS VALID, IT IS DISCONNECTED FROM THE ACE.	ILRFRR01	ILRVAIA	25.27.9
03 IF ATAACE IS VALID, THE CELL USED FOR THE ACE IS RETURNED TO ITS POOL.			

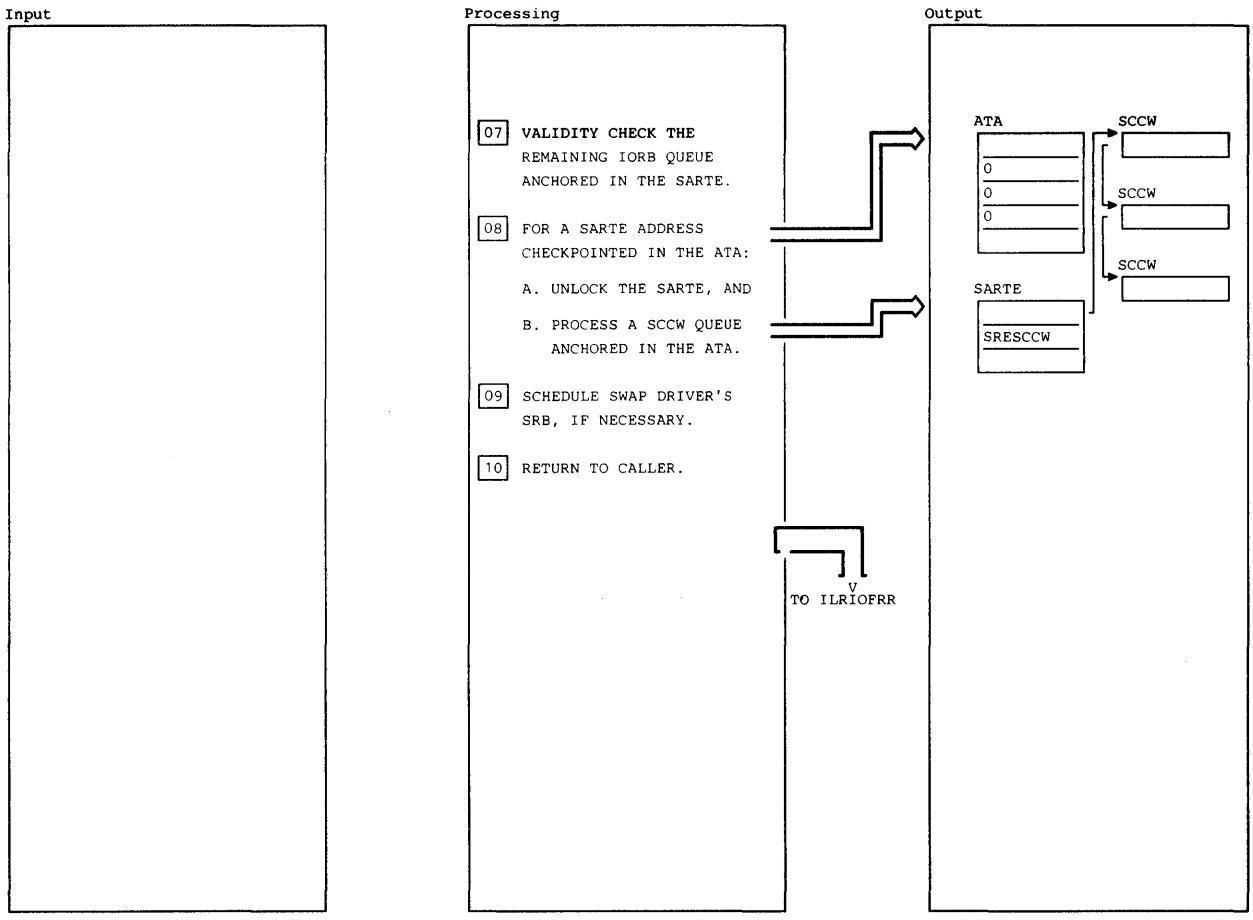
Notes	Routine	Label	Ref

Diagram 25.20.7 RECTRPAG (Part 1 of 1)



Notes	Routine	Label	Ref
01 ILRSWP01 AND ITS TWO ENTRIES HANDLE ERRORS OCCURRING IN ILRSWPDR AND ILRSWAP.			
02 IF A SARTE IS NOT CHECKPOINTED (ATASARTE), THE SWAP DRIVER (ILRSWPDR) IS IN ENTRY OR EXIT PROCESSING. RECOVERY CONSISTS OF RESCHEDULING SWAP DRIVER'S SRB.			
03 AN '084' ABEND WAS ISSUED WHEN SWAP DRIVER FOUND A ZERO IORB ANCHOR IN A SARTE. THE ABEND CODE IS FOUND IN THE SDWA FIELD SDWACMPC. THE RBIORB SUBROUTINE IS CALLED TO ATTEMPT REBUILDING AN IORB.		RBDIORB	25.21.3
04 THE CURRENT IORB IS VALIDITY CHECKED. NOTE THAT IF AN IORB IS NOT CHECKPOINTED, THIS RECOVERY WILL STILL VALIDITY CHECK THE IORB QUEUE ANCHORED IN THE SARTE.	ILRFRR01	ILRVIORB	25.27.17
05 THE QUEUE OF SCCWS ANCHORED IN THE IORB (IORSCCW) IS VALIDITY CHECKED AND MERGED WITH THE QUEUE OF SCCWS ANCHORED IN THE ATA (ATASCCW). BECAUSE SWAP DRIVER MOVES SCCWS INDIVIDUALLY FROM THE ATA TO IORB SCCW QUEUE, A CHECK IS MADE TO INSURE THAT THE ATA SCCW QUEUE DOES NOT CONTAIN DUPLICATE SCCWS. THE IORB IS FREED BY TURNING OFF THE IORB IN USE FLAG (IORUSE) AND THE IORB SCCW QUEUE ANCHOR IS ZEROED.	ILRFRR01	ILRVSCWQ	25.27.5
06 THE IORB ANCHOR IN THE SARTE IS VALIDITY CHECKED TO INSURE AT LEAST ONE VALID IORB EXISTS FOR THIS SARTE. IF THE ANCHOR IORB IS INVALID THE RBDIORB SUBROUTINE IS CALLED TO REBUILD AN IORB.	ILRFRR01	ILRVIORB RBDIORB	25.27.17 25.21.3

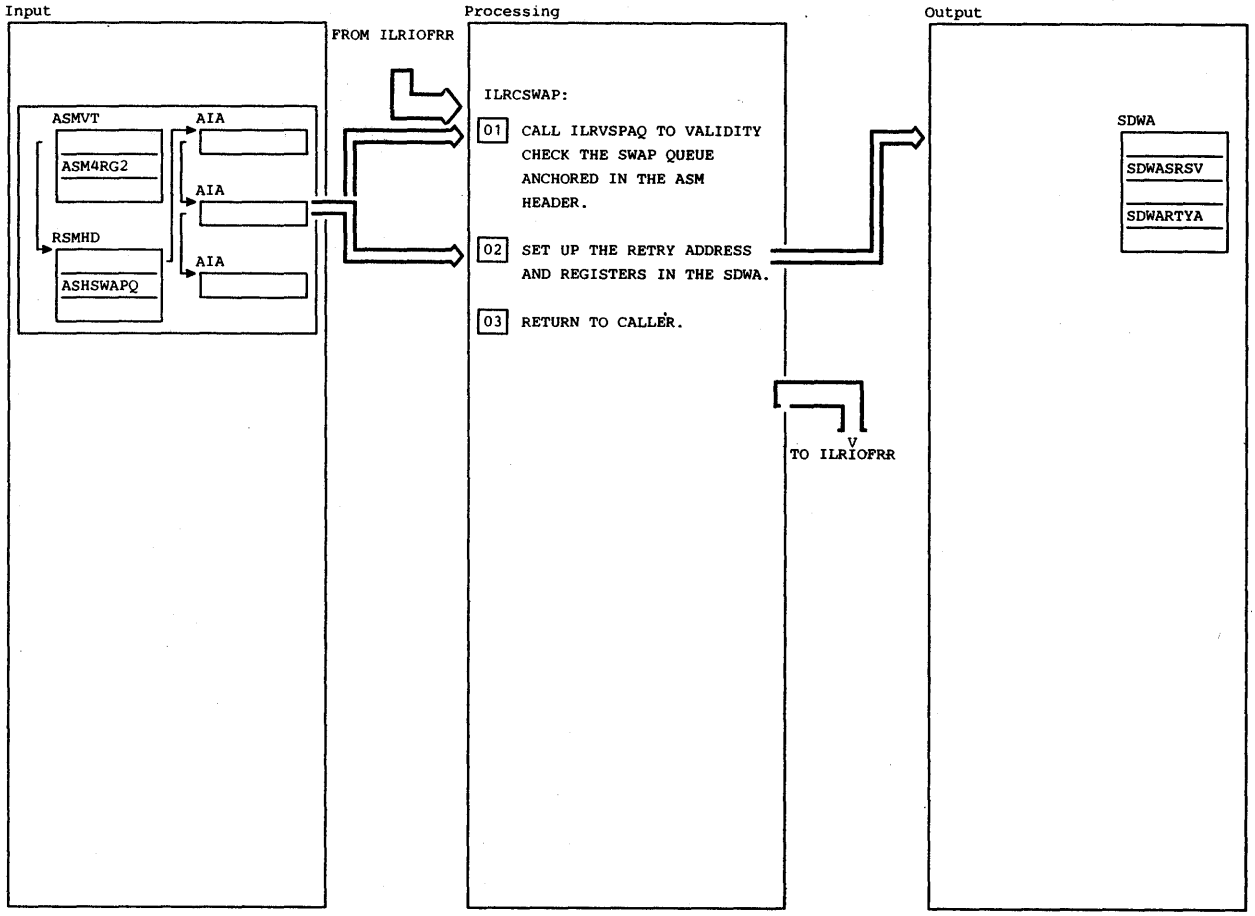
Diagram 25.21 ILRSWPO1 (Part 1 of 2)



Notes	Routine	Label	Ref
07 EACH IORB ON THE SARTE IORB QUEUE IS VALIDITY CHECKED. THE QUEUE IS TRUNCATED IF AN INVALID IORB IS FOUND.	ILRFRR01	ILRVIORB	25.27.17
08 IF THE REBUILD IORB SUBROUTINE WAS CALLED AND WAS UNABLE TO REBUILD AN IORB, THE SARTE ADDRESS WILL HAVE BEEN ZEROED TO PREVENT UNLOCKING THE SARTE. A. SWAP DRIVER'S REDRIVE FLAG (SREDRIVE) IS TURNED OFF AND THE SARTE IS UNLOCKED (SRELOCK) USING COMPARE AND SWAP.			
B. THE ATA SCCW QUEUE IS VALIDITY CHECKED. THE COMMAND CHAINING FLAGS IN THE LAST READ/WRITE CCW OF EACH SCCW ARE CLEARED TO BREAK THE SCCW CHAINING BETWEEN SCCWS. THE ATA SCCW QUEUE IS ADDED TO THE SARTE SCCW WORK QUEUE (SRESCCW) USING COMPARE AND SWAP.	ILRFRR01	ILRVSCWQ	25.27.5
09 IF A SWAP DRIVER SRB IS NOT	SCHEDULE		

Notes	Routine	Label	Ref
CURRENTLY ACTIVE (SARSRBCT=0), THE SRB COUNT IS INCREMENTED USING COMPARE AND SWAP. THE SWAP DRIVER SRB IS RESCHEDULED TO PROCESS WORK LEFT ON THE SARTE SCCW QUEUE BY THIS RECOVERY.			

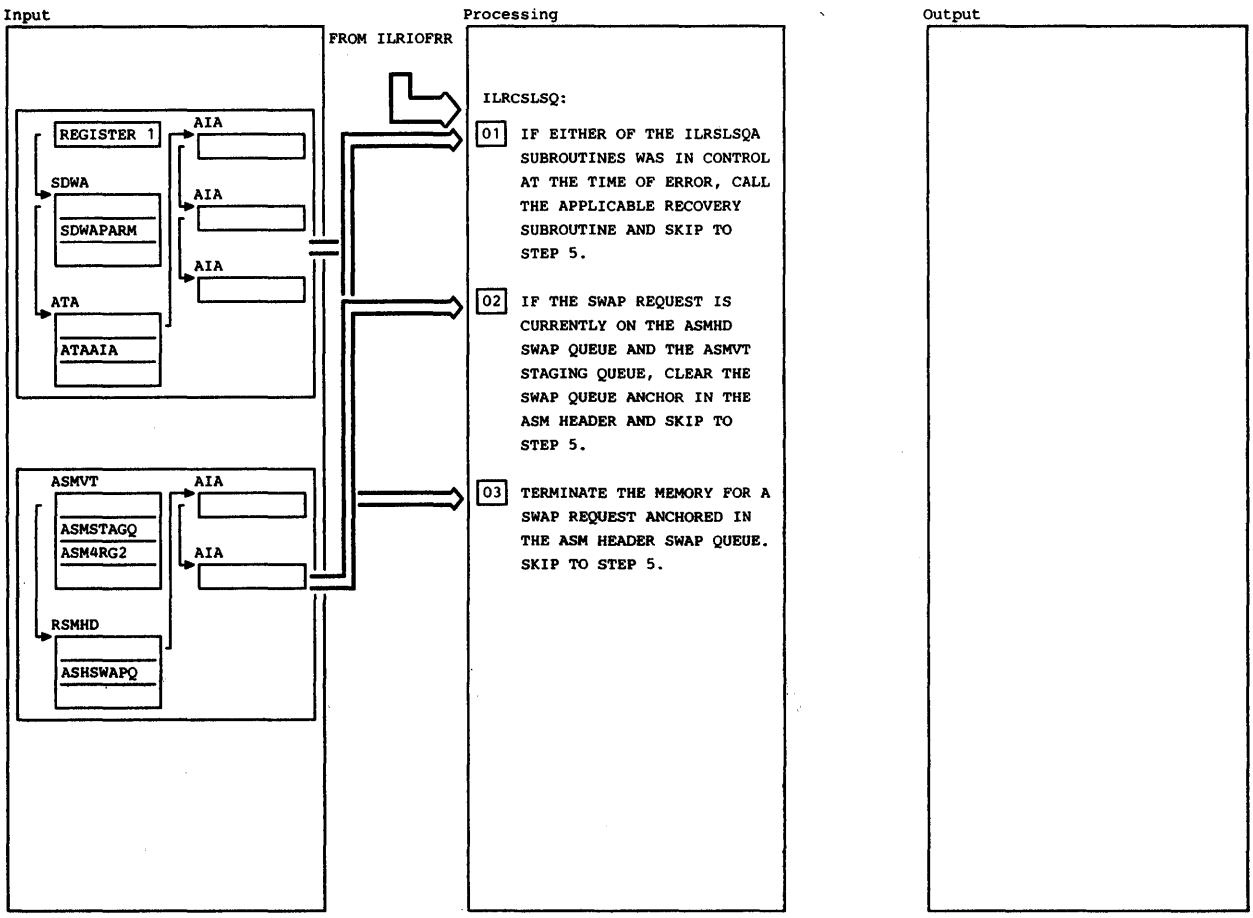
Diagram 25.21 ILRSWPO1 (Part 2 of 2)



Notes	Routine	Label	Ref
01 THE EFFECT OF THIS RECOVERY IS TO PASS SWAP AIAS BACK TO RSM SO THAT FRAMES ASSOCIATED WITH THE AIAS MAY BE FREED PRIOR TO MEMORY TERMINATION. A NON ZERO SWAP QUEUE ANCHORED IN THE ASM HEADER IS VALIDITY CHECKED TO PREVENT A REOCCURRENCE OF THIS ERROR BY RSM. THE FIRST AIA RETURNED TO RSM IS FLAGGED IN ERROR (AIAERROR).	ILRFRR01	ILRVSPAQ	
02 IF THE ERROR IS NOT RECURSIVE, A RECURSION FLAG IS SET IN THE ATA. THE RETRY ADDRESS AND THE REGISTER SAVE AREA ARE INITIALIZED IN THE SDWA. A NON-ZERO RETRY ADDRESS INDICATES TO ILRIOFRR THAT RETRY OF THE ERROR IS REQUESTED. ILRIOFRR WILL COMPLETE INITIALIZATION OF THE SDWA.			

Notes	Routine	Label	Ref

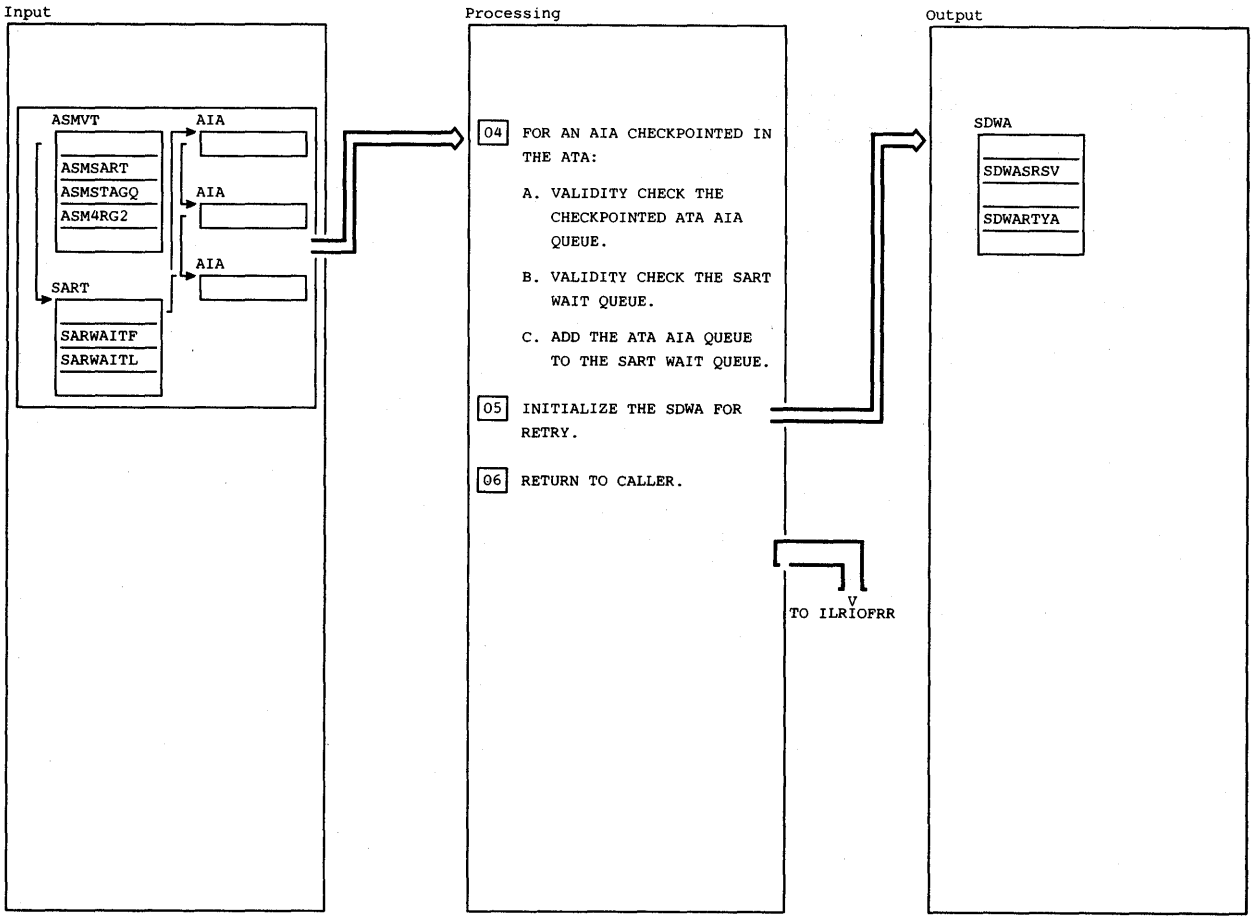
Diagram 25.21.1 ILRCSWAP (Part 1 of 1)



Notes	Routine	Label	Ref
01 THE RECOVERY SUBROUTINES ARE: ASETRCVY FOR ASIGNSET -- SCCWRCVY FOR SCCWPROC. FLAGS IN THE ATA INDICATE THE ROUTINE IN CONTROL AT THE TIME OF ERROR. ERRORS IN ASIGNSET, WITH THE EXCEPTION OF ASM ISSUED ABENDS, ARE TREATED AS MAINLINE ILRSLSQA ERRORS.		ASETRCVY SCCWRCVY	25.21.5 25.21.4
02 A SWAP REQUEST CAN COMPLETE SUCCESSFULLY IF IT IS ALREADY ON THE ASMVT STAGING QUEUE (ASMSTAGQ). THIS SITUATION OCCURS WHEN ILRSLSQA IS MOVING A SWAP REQUEST FROM THE ASMHD SWAP QUEUE TO THE ASMVT STAGING QUEUE.			
03 THE SWAP REQUEST IS NOT CURRENTLY READY FOR I/O PROCESSING. THE MEMORY IS SCHEDULED FOR TERMINATION WITH A SYSTEM X'028' COMPLETION CODE. FOR A SWAP-IN REQUEST, A FLAG IN THE RSM HEADER(RSMFAIL) IS SET TO INDICATE A SWAP-IN FAILURE.	CALLRTM		

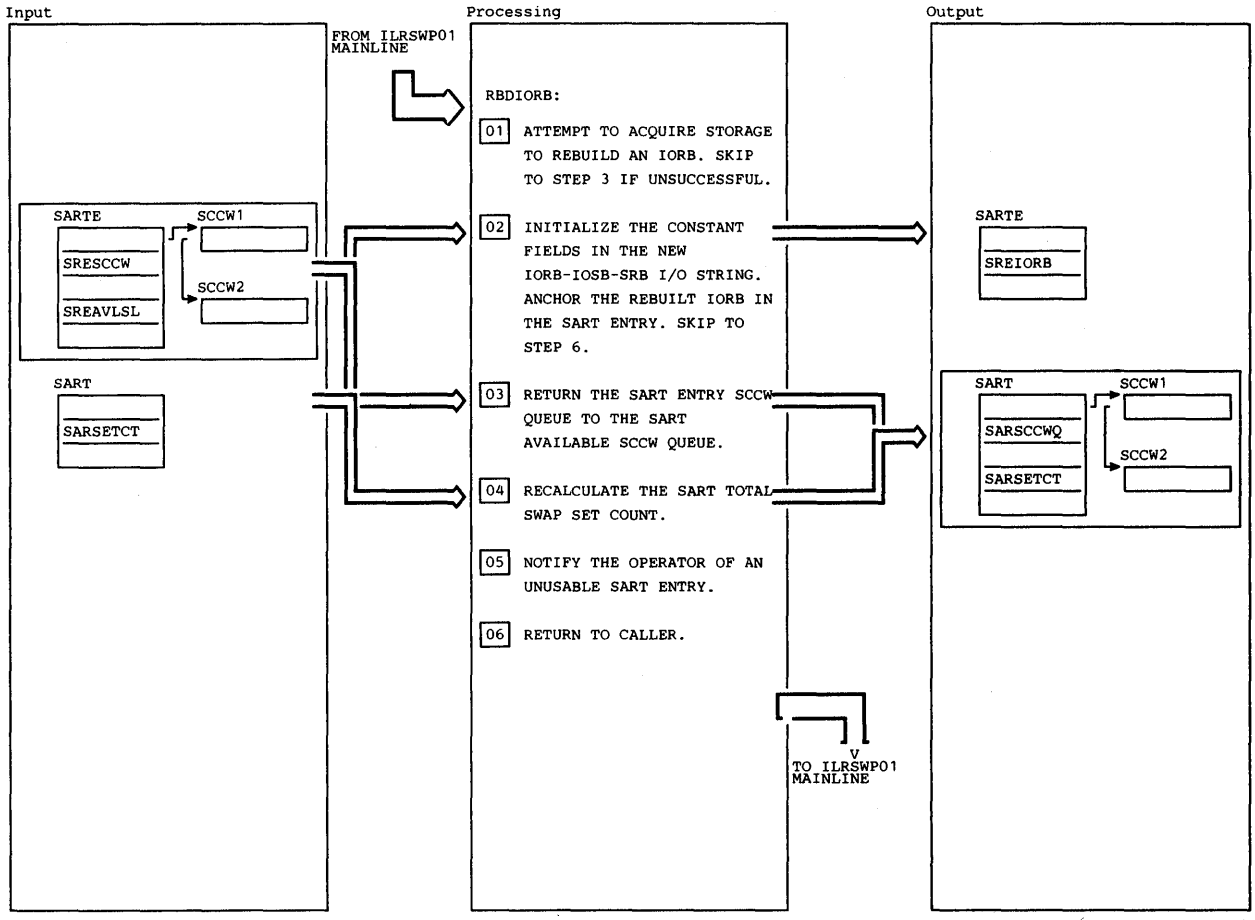
Notes	Routine	Label	Ref

Diagram 25.21.2 ILRCSLSQ (Part 1 of 2)



Notes	Routine	Label	Ref
04 A CHECKPOINTED AIA REPRESENTS A SWAP REQUEST CURRENTLY READY FOR I/O PROCESSING. THE ATA AIA QUEUE CAN BE LEFT ON THE SART WAIT QUEUE FOR REPROCESSING BY ILRSLSQA.			
A. THE CHECKPOINTED AIA QUEUE IS VERIFIED BEFORE ADDING IT TO THE SART WAIT QUEUE.	ILRFRR01	ILRVSPAQ	25.27.6
B. THE SART WAIT QUEUE IS VALIDITY CHECKED TO REMOVE INVALID ELEMENTS CAUSED BY THIS ERROR.	ILRFRR01	ILRVSWTQ	25.27.3
C. IF THE CHECKPOINTED AIA IS VALID, IT IS COMPARED TO EACH ELEMENT ON THE SART WAIT QUEUE. THE AIA IS ADDED TO THE SART WAIT QUEUE IF NOT ALREADY ON THE QUEUE. SERIALIZATION FOR THE WAIT QUEUE IS PROVIDED BY THE SALLOC LOCK, HELD BY THIS RECOVERY ROUTINE ON ENTRY.			
05 FOR ASM ISSUED ABENDS THE ERROR PSW AND REGISTERS (SDWANXT1 AND SDWASRSV) ARE USED FOR RETRY. IF THIS IS NOT A RECURSIVE ERROR (ATARCRF6='0'B), THE RECURSION FLAG IS SET IN THE ATA, AND A RETRY ADDRESS IN ILRSLSQA IS SET IN THE SDWA. AT THE RETRY POINT, LABELLED ILRCRSP2, ILRSLSQA WILL CHECK FOR WORK LEFT ON THE SART WAIT QUEUE BY RECOVERY. THE RSM HEADER REGISTER (REG3), AND ILRSLSQA BASE REGISTER (REG 12) ARE REINITIALIZED IN THE SDWA. THE ATA CHECKPOINTED FIELDS ARE CLEARED.			

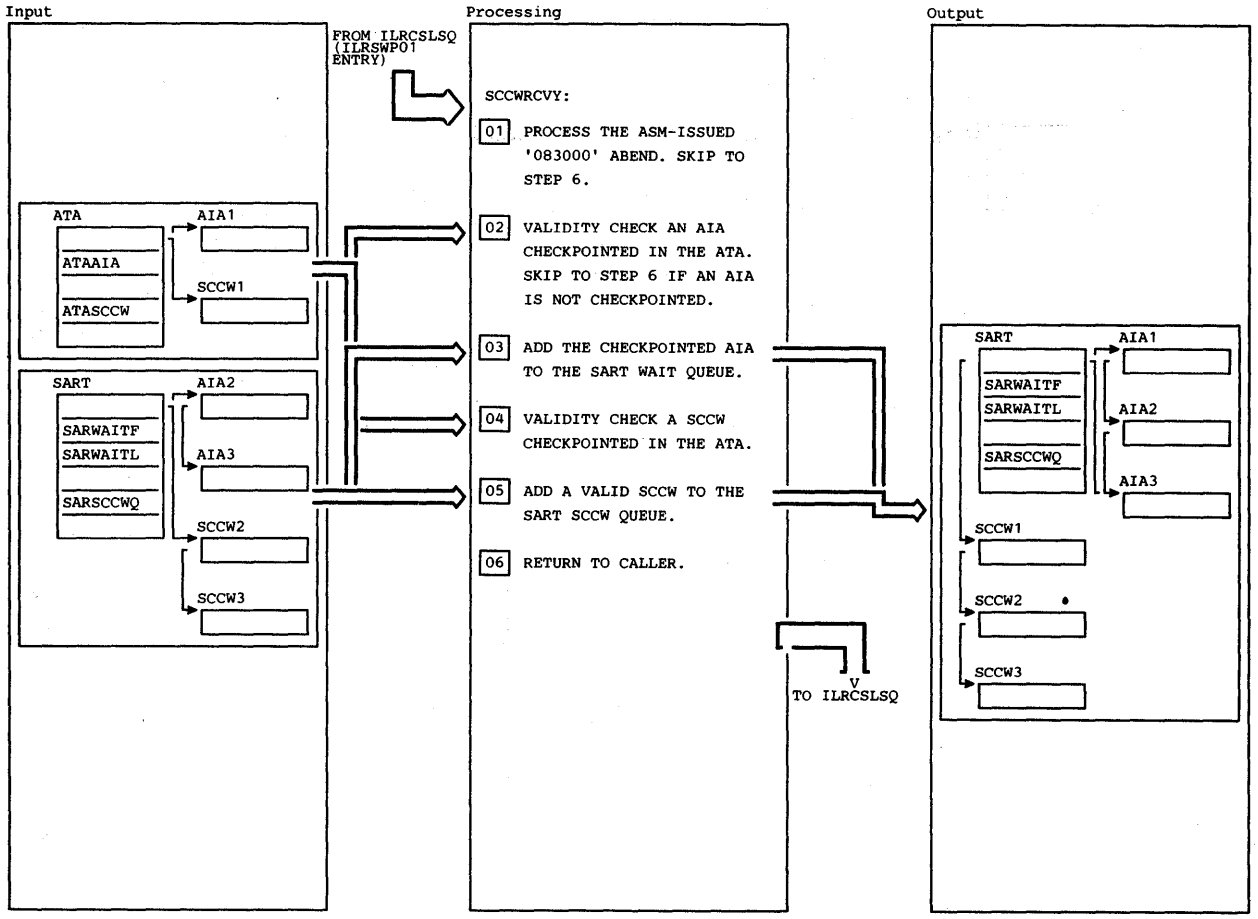
Diagram 25.21.2 ILRCSLSQ (Part 2 of 2)



Notes	Routine	Label	Ref
01 THE SALLOC LOCK IS OBTAINED UNCONDITIONALLY AND A BRANCH ENTRY TO GETMAIN FOR SQA STORAGE IS ISSUED FOR THE LENGTH OF AN IORB-IOSB-SRB I/O STRING.	SETLOCK IEAVGM00		
02 THE ACQUIRED STORAGE IS CLEARED. THOSE FIELDS CHECKED BY ILRVIORB ARE INITIALIZED (IORID, IORSWAP, IORPARTE, IOSDVRID, AND IOSMISID). ILRVIORB IS CALLED TO INITIALIZE CRITICAL IORB-IOSB-SRB FIELDS.	ILRFRR01	ILRVIORB	25.27.17
03 A SART ENTRY IS UNUSABLE SINCE THE IORB CANNOT BE REBUILT. THE SCCW WORK QUEUE (SRESCCW) IS VALIDITY CHECKED AND RETURNED TO THE SART AVAILABLE SCCW QUEUE. THE AIAS ANCHORED IN THESE SCCWS ARE LOST.	ILRFRR01	ILRVSCWQ	25.27.5
04 THE COUNT OF AVAILABLE SWAP SETS ON THIS SARTE IS SET TO ZERO. THE SART TOTAL SWAP SET COUNT IS RECALCULATED AS THE SUM OF EACH USABLE SARTE AVAILABLE SWAP SET COUNT.			

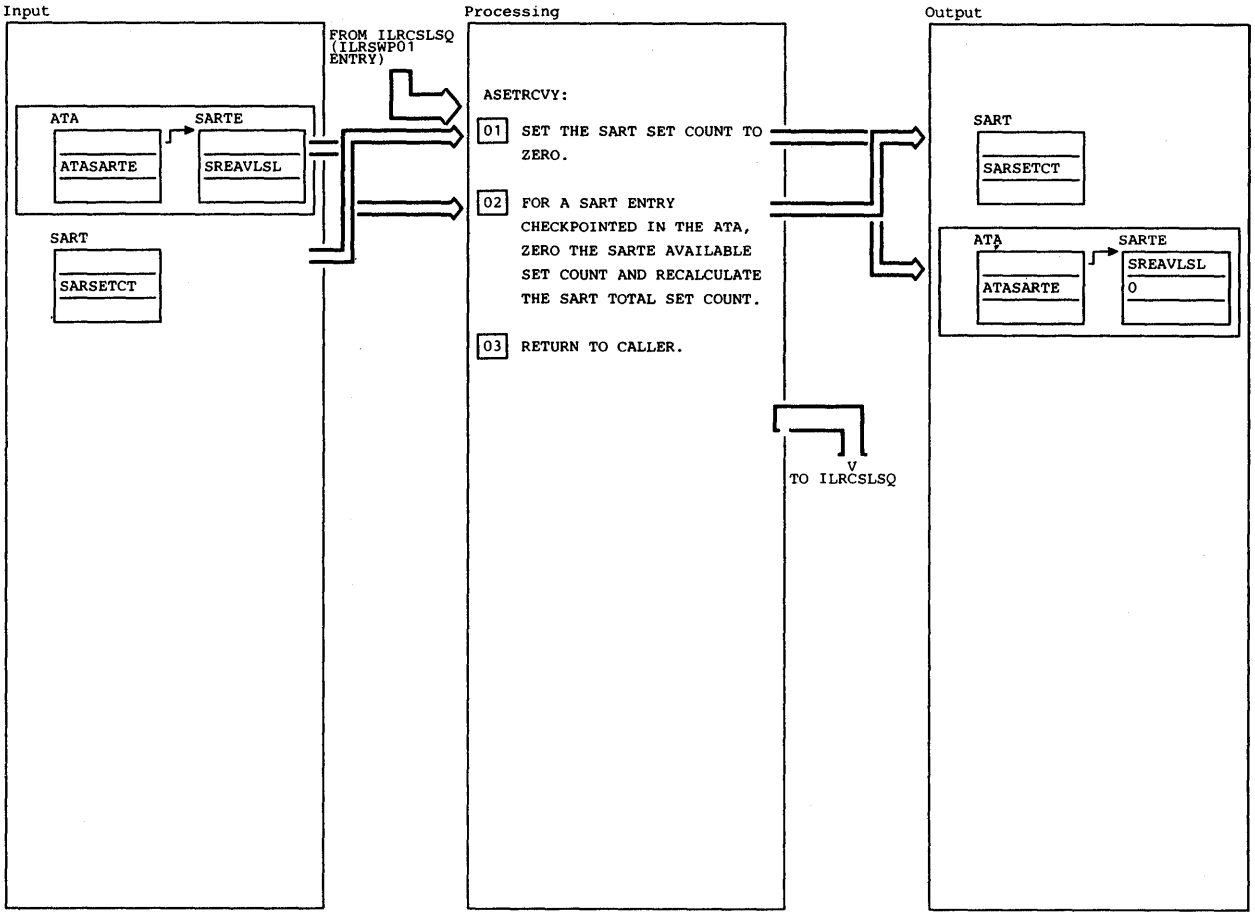
Notes	Routine	Label	Ref
05 THE SART ENTRY IS FLAGGED UNUSABLE (SRENUSE). ASM'S MESSAGE MODULE IS CALLED TO ISSUE AN UNUSABLE SWAP DATA SET MESSAGE,ILR009I. THE SART ENTRY ADDRESS IN THE ATA IS ZEROED TO PREVENT ILRSWP01 FROM UNLOCKING THIS SARTE.	ILRMSG00	ILRMSG00	

Diagram 25.21.3 RBDIORB (Part 1 of 1)



Notes	Routine	Label	Ref
01 WHEN THE LOGICAL SLOT ID IN AN AIA IS OUTSIDE THE RANGE OF VALID LSIDS, SCCWPROC (SUBROUTINE OF ILRSLSQA) ISSUES A RECORD ONLY ABEND. SCCWRCVY PROCESSING CONSISTS OF COPYING THE ERROR AIA (ATAAIA) INTO THE SDWA.			
02 IF AN AIA IS NOT CHECKPOINTED, SCCWPROC HAS COMPLETED PROCESSING FOR BOTH THE AIA AND SCCW.	ILRFRR01	ILRVAIA	25.27.9
03 AN AIA WHICH CONTAINS AN I/O ERROR FLAG (AIAPRIER OR AIABADID) IS IGNORED SINCE IT MAY ALREADY HAVE BEEN ADDED TO THE PART ERROR QUEUE (PARTAIAE) BY SCCWPROC. IF THE AIA IS VALID, IT IS ADDED TO THE SART WAIT QUEUE, SERIALIZED BY THE SALLOC LOCK.			
04 THE CHECKPOINTED SCCW (ATASCCW) IS VALIDITY CHECKED.	ILRFRR01	ILRVSCCW	25.27.12
05 A VALID SCCW IS ADDED TO THE QUEUE OF AVAILABLE SCCWS ANCHORED IN THE SART (SARSCCWQ) VIA COMPARE AND SWAP.			

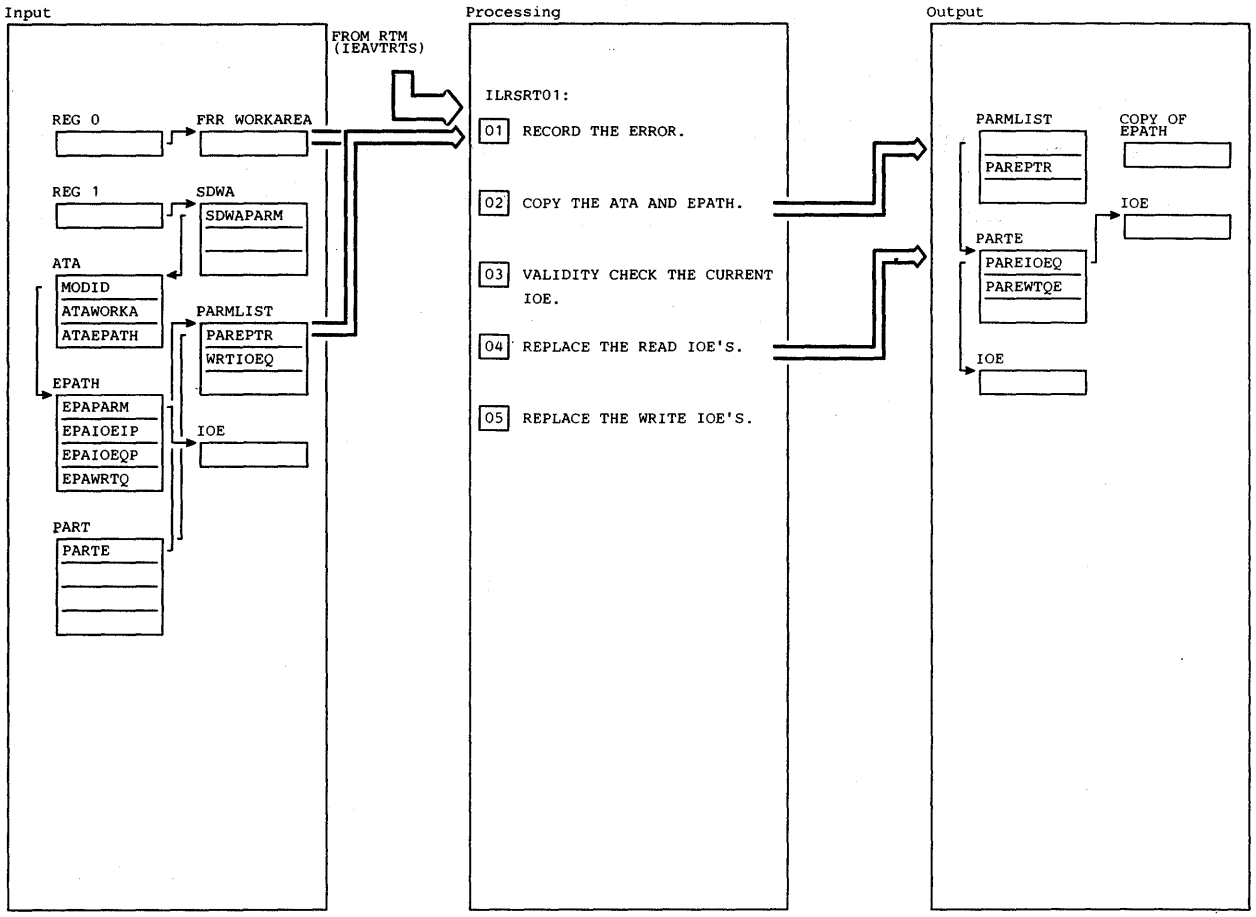
Diagram 25.21.4 SCCWRCVY (Part 1 of 1)



Notes	Routine	Label	Ref
01 THIS ROUTINE IS ENTERED FOR EITHER OF 2 COD ABENDS: (1) INCORRECT SART TOTAL SWAP SET COUNT (SARSETCT) OR (2) INCORRECT SART ENTRY AVAILABLE SET COUNT (SREAVLSL). IF THE SART ENTRY IS NOT CHECKPOINTED IN THE ATA (THE FIRST ABEND), THE SART SET COUNT IS SET TO ZERO.			
02 ASIGNSET (SUBROUTINE OF ILRSLSQA) CHECKPOINTS THE SART ENTRY IN THE ATA ONLY BEFORE ISSUING THE COD ABEND FOR AN INCORRECT SART ENTRY SET COUNT (THE SECOND ABEND). THIS SARTE'S SET COUNT IS ZEROED AND THE SART TOTAL SET COUNT IS RESET TO THE CURRENT TOTAL OF ALL AVAILABLE SWAP SET COUNTS IN EACH USABLE SART ENTRY.			

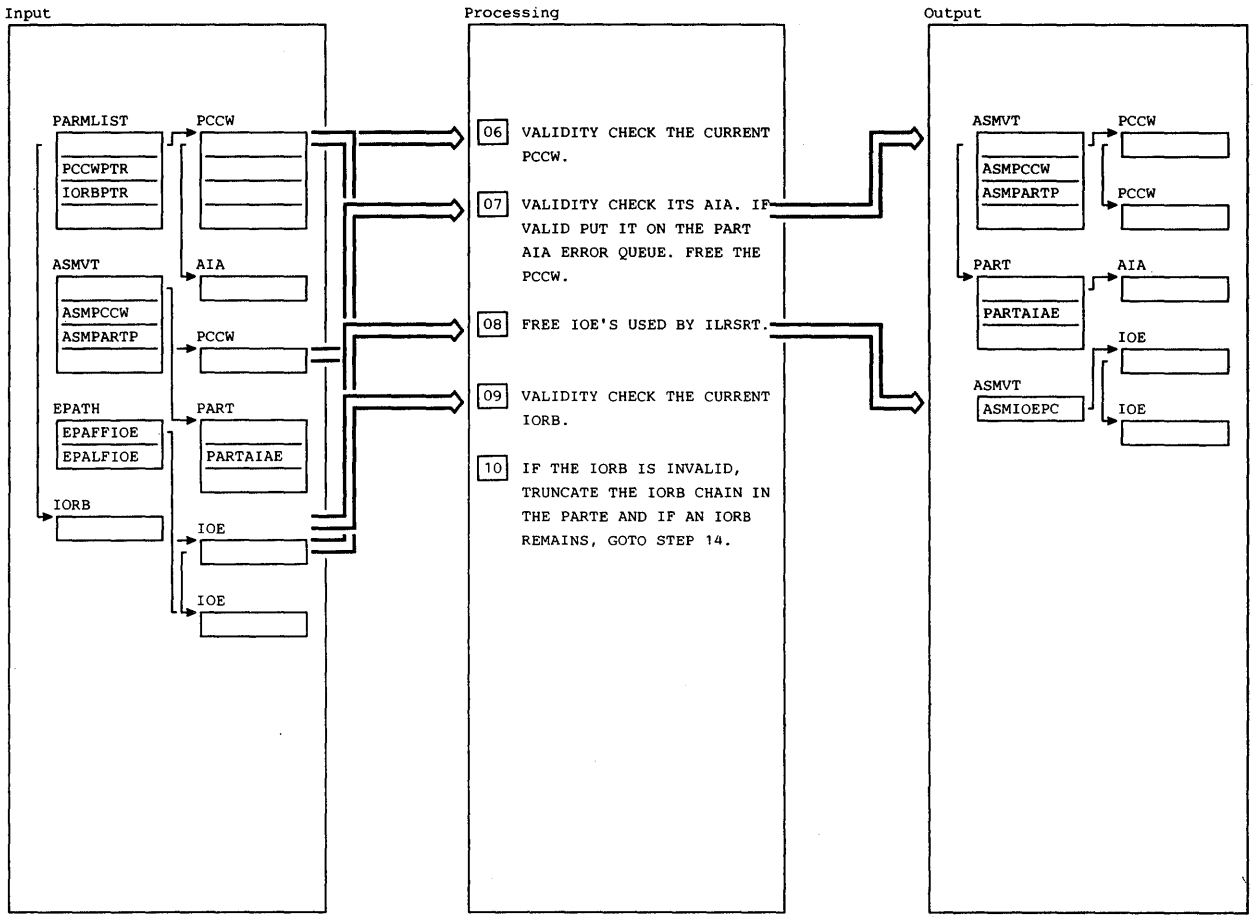
Notes	Routine	Label	Ref

Diagram 25.21.5 ASETRCVY (Part 1 of 1)



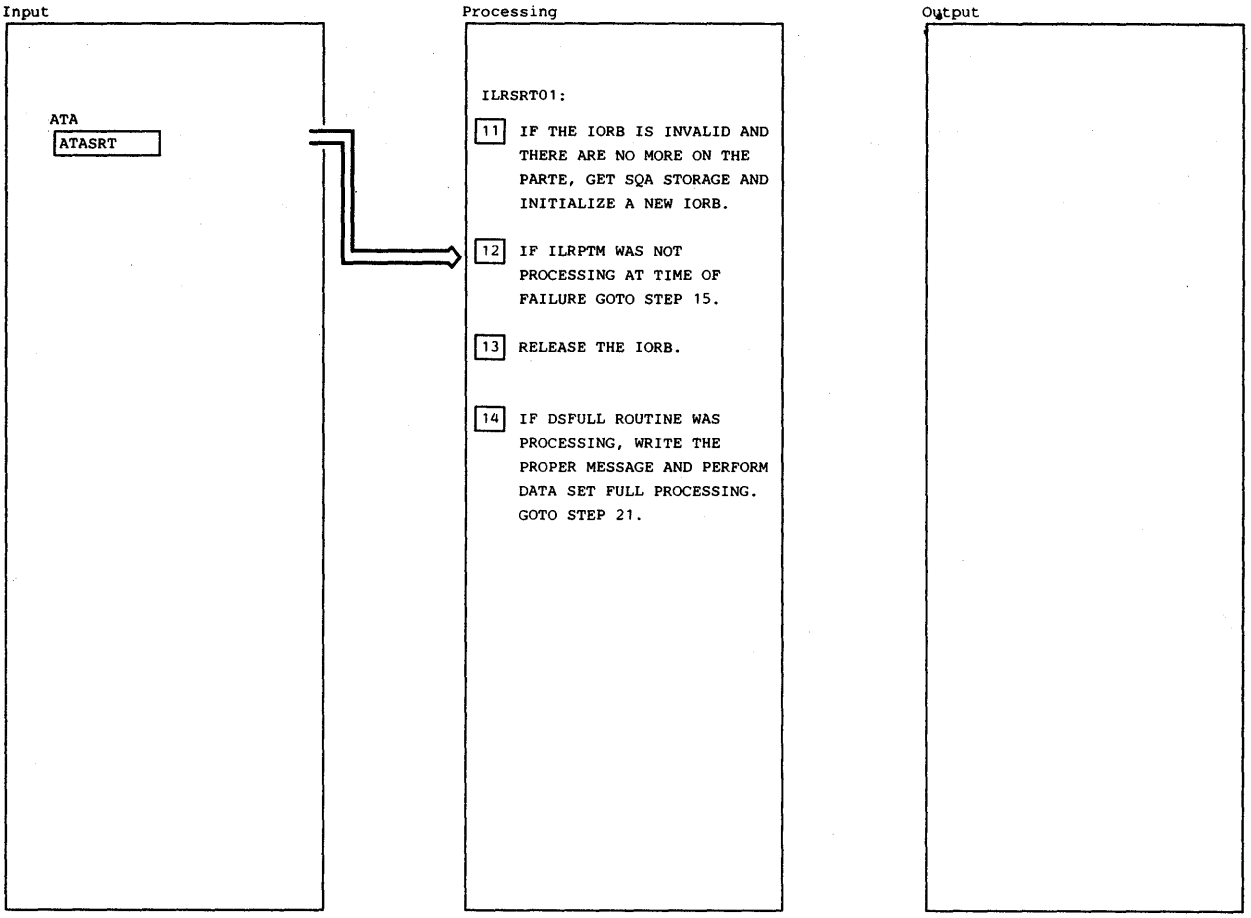
Notes	Routine	Label	Ref
01 MOVE THE MAINLINE MODULE-IN-ERROR (ILRPTM OR ILRSRT) NAME AND ILRSRT01 (RECOVERY NAME) TO THE SDWA. ISSUE SETRP TO REQUEST RECORDING AND RELEASING OF THE SALLOC AND THE CLASS LOCKS ON RETURN TO RTM.			
02 THE ATA AND EPATH (IF CHECKPOINTED) ARE COPIED TO THE VARIABLE RECORDING AREA IN THE SDWA. IF THE EPATH ADDRESS IS ZERO GOTO STEP 22, SINCE NO RECOVERY CAN BE DONE WITHOUT THE INFORMATION IN THE EPATH.			
03 THE CURRENT IOE (EPAIOEIP) IS ADDRESS VERIFIED. IF VALID IT IS PLACED ON THE APPROPRIATE IOE QUEUE, WRTIOEQ OR EPAIOEQP.	ILRFRR01	ILRVIOE	25.27.18
04 THE QUEUE OF READ IOE'S ON THE WORK QUEUE (EPAIOEQP) IS VALIDITY CHECKED. IF VALID IOE'S, THEY ARE REPLACED ON THE PARTE (PAREIOEQ). EPAIOEQP SHOULD BE NON-ZERO ONLY IF ILRPTM WAS PROCESSING.	ILRFRR01	ILRVIOEQ	25.27.7
05 THE QUEUE OF WRITE IOE'S IS VALIDITY CHECKED. IF ANY VALIDS THE CLASS LOCK IS OBTAINED AND THE IOE'S ARE REPLACED ON THE PART WRITE QUEUE (EPAWRTQ).	ILRFRR01	ILRVIOEQ	25.27.7

Diagram 25.22 ILRSRTO1 (Part 1 of 5)



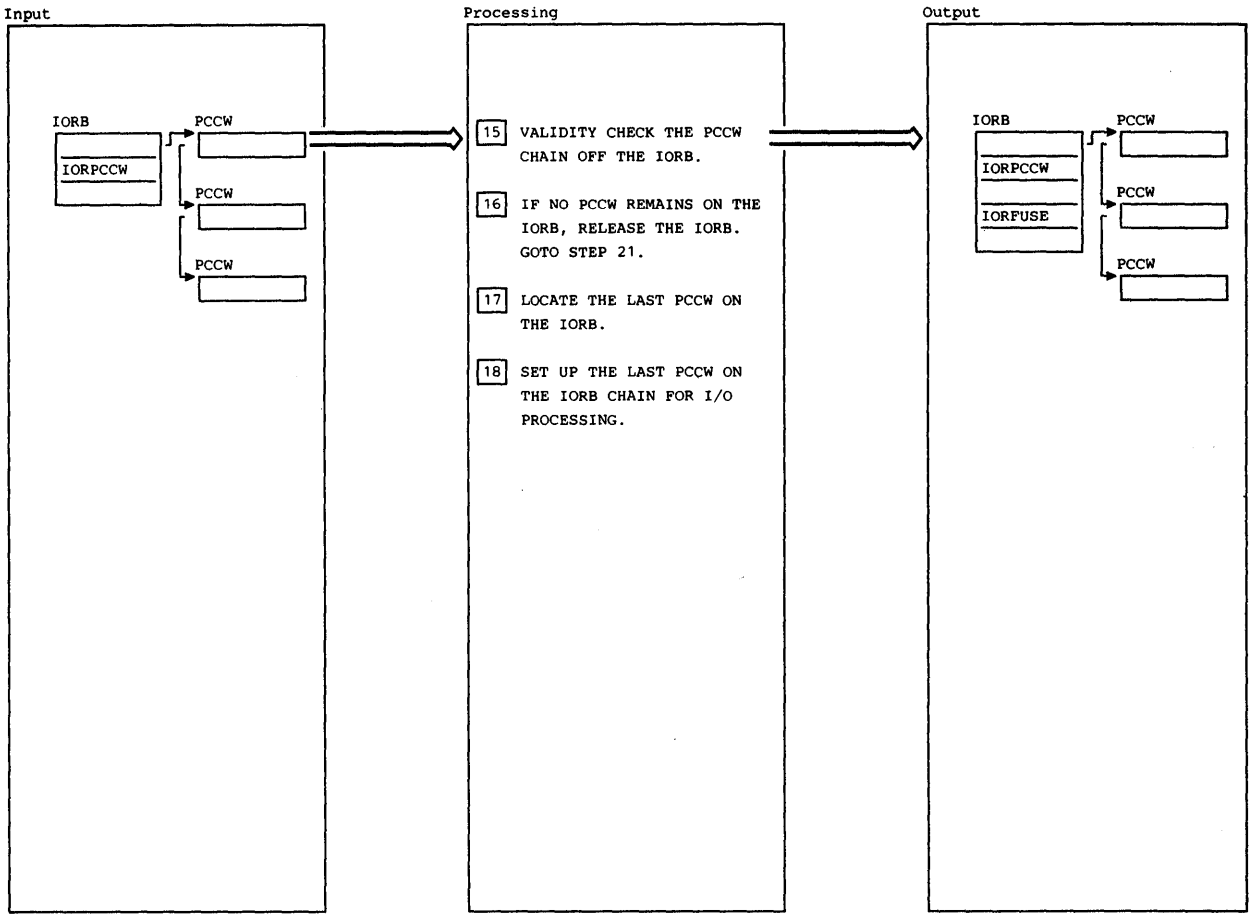
Notes	Routine	Label	Ref
06 CALL ILRVPCCW TO VALIDITY CHECK THE CURRENT PCCW (PCCWPTR) IN THE PARMLIST.	ILRFRR01	ILRVPCCW	25.27.14
07 THE AIA FROM THE PCCW IS VALIDITY CHECKED BY ILRVAIA. IF VALID IT IS PUT ON THE AIA ERROR QUEUE IN THE PART (PARTAIAE). IF ILRSRT HAD A CONVERT ERROR (083 ABEND) COPY THE AIA AND EDB TO THE VARIABLE RECORDING AREA IN THE SDWA. THE AIA FIELD IN THE PCCW IS SET TO ZERO AND THE PCCW IS RETURNED TO ITS POOL (ASMPCCWQ).	ILRFRR01	ILRVAIA	25.27.9
08 THE IOE'S ON THE ILRSRT FREE QUEUE (EPAFFIOE AND EPALFIOE) ARE VALIDITY CHECKED BY ILRVIOEQ. ANY VALID IOE'S ARE RETURNED TO THEIR POOL (ASMIOEPC).	ILRFRR01	ILRVIOEQ	25.27.7
09 CALL ILRVIORB TO VALIDITY CHECK THE CURRENT IORB (IORBPTR IN PARMLIST).	ILRFRR01	ILRVIORB	25.27.17
10 IF THE IORB IS INVALID, THE CHAIN (PAREIORB) IS TRUNCATED. IF PAREIORB IS NOW NON-ZERO, GO TO STEP 14.			

Diagram 25.22 ILRSRTO1 (Part 2 of 5)



Notes	Routine	Label	Ref
11 IF THE IORB IS INVALID AND PAREIORB IS ZERO, ISSUE A GETMAIN FOR SQA TO BUILD A NEW IORB, IOSB AND SRB. INITIALIZE THE REQUIRED FIELDS THEN CALL ILRVIORB TO FINISH THE CONSTRUCTION. STORE IN THE IORB THE ADDRESS IN PAREIORB AND GO TO STEP 14. IF THE GETMAIN FAILS SET PARENUSE=1 SINCE THIS PAGE DATA SET CANNOT BE USED. IF THE PARTE IS FOR A LOCAL PAGE DATA SET, DECREMENT TOTAL SLOTS AVAILABLE COUNT (ASMSLOTS). WRITE MESSAGE ILR009I. GO TO STEP 22.	ILRFRR01	ILRVIORB	25.27.17
12 IF ILRPTM WAS PROCESSING THERE IS NO I/O TO BE DONE AND POSSIBLY THE DSFULL ROUTINE IN ILRPTM FAILED.			
13 MAKE THE IORB AVAILABLE SINCE PROCESSING OF IT IS COMPLETE.			
14 IF THE DSFULL (DATA SET FULL) ROUTINE WAS PROCESSING, INSURE THAT THE PROPER MESSAGE IS WRITTEN AND THE PARTE IS PROPERLY ADJUSTED (AS IF DSFULL HAD COMPLETED PROCESSING). GO TO STEP 21 SINCE THERE IS NO I/O TO PERFORM.			

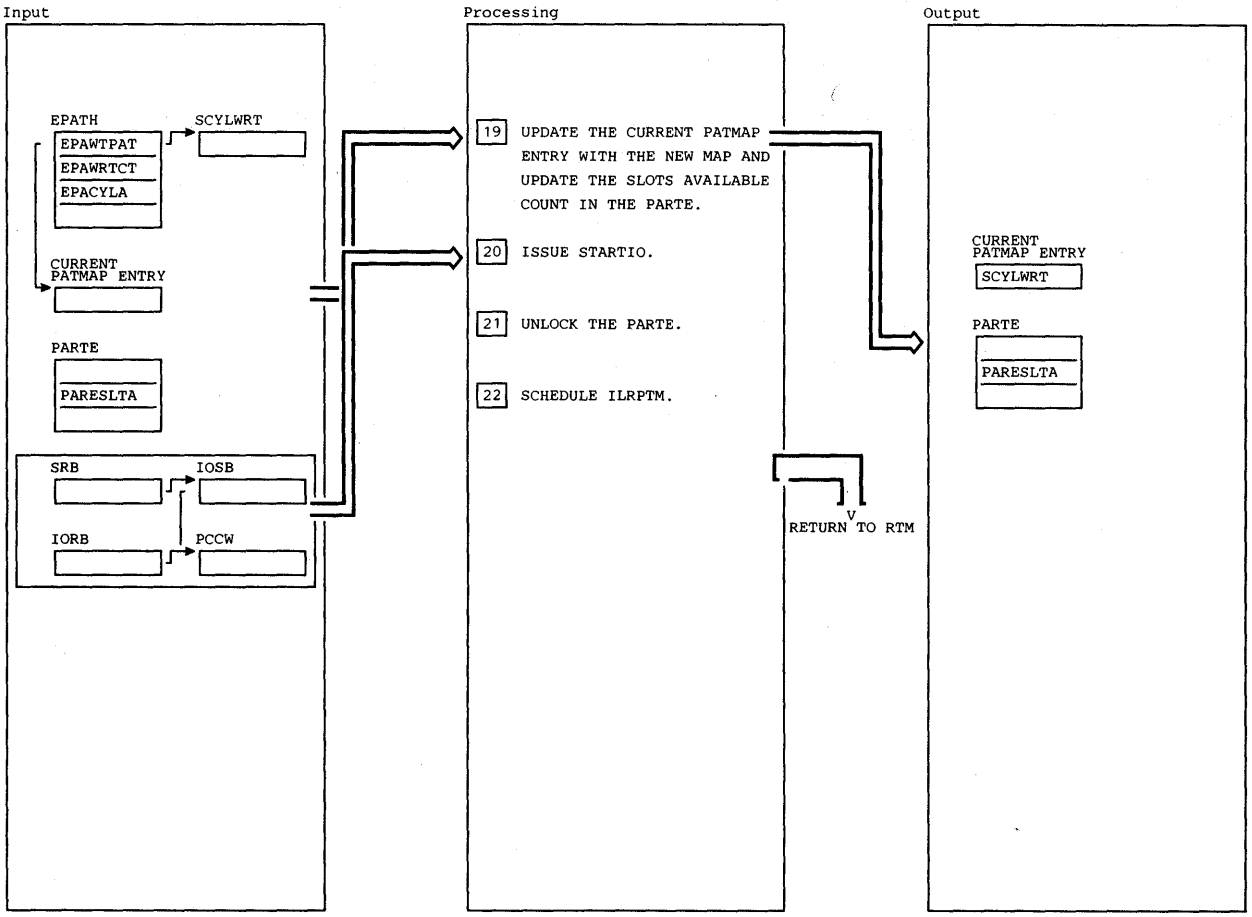
Diagram 25.22 ILRSTRO1 (Part 3 of 5)



Notes	Routine	Label	Ref
15 THE PCCW CHAIN (IORPCCW) IS VALIDITY CHECKED BY ILRVPCWQ AND A COUNT IS MAINTAINED FOR THE PCCW'S THAT ARE KEPT ON THE CHAIN. IF THE CURRENT PCCW (PCCWPTR) IS ON THE CHAIN, IT IS REMOVED.	ILRFRR01	ILRVPCWQ	25.27.13
16 IF NO PCCW IS LEFT TO SEND TO IOS, RELEASE THE IORB AND CONTINUE AT STEP 21.			
17 FOLLOW THE PCCW CHAIN FROM IORPCCW AND FIND THE LAST ONE SO IT CAN BE UPDATED.			
18 THE LAST PCCW REMAINING ON THE IORB CHAIN IS SET UP AS FOLLOWS: -PCCWPCCW=0, -THE LAST CCW IS CHANGED TO A NOP AND CHAINING BITS ARE SET=0.			

Notes	Routine	Label	Ref

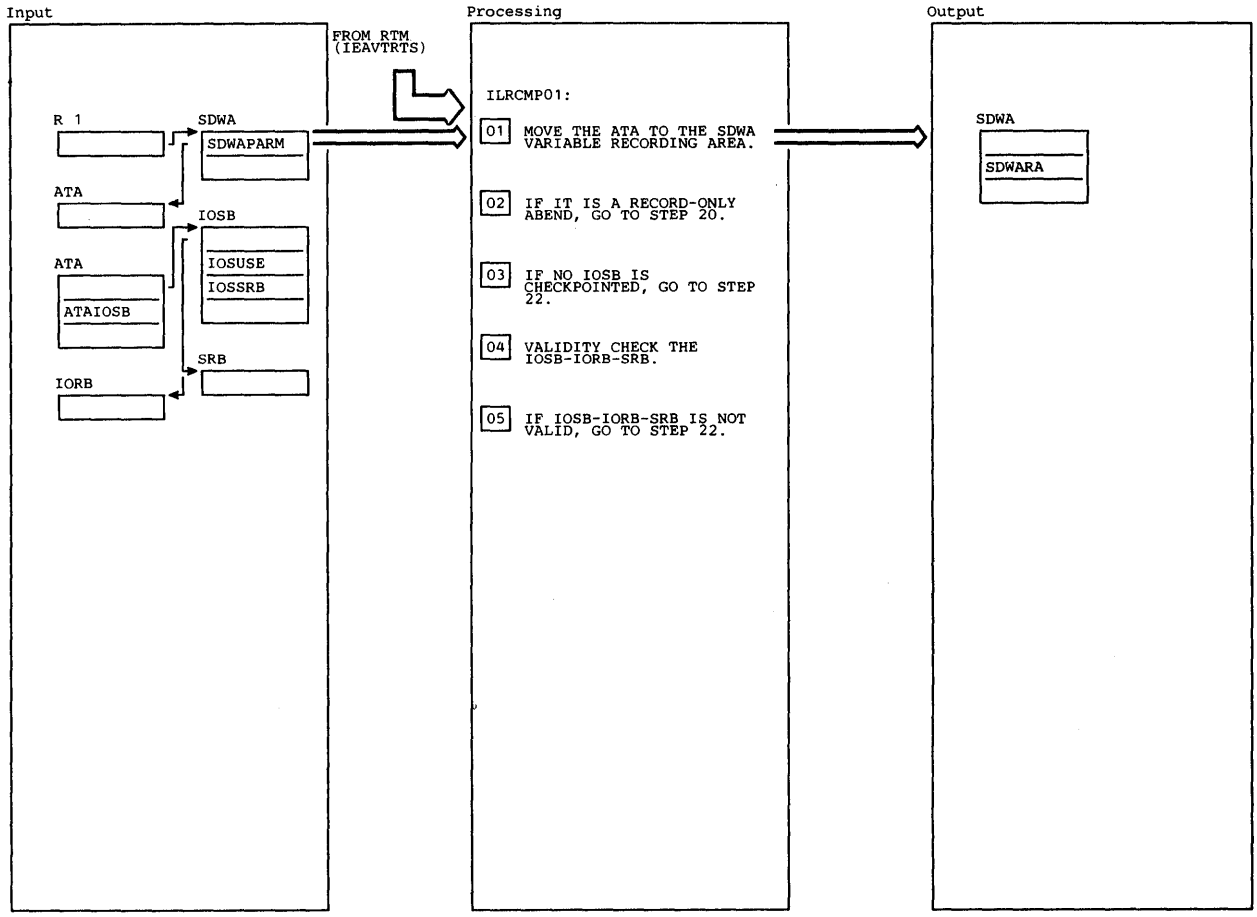
Diagram 25.22 ILRSRTO1 (Part 4 of 5)



Notes	Routine	Label	Ref
19 THE CURRENT PATMAP ENTRY AND AVAILABLE SLOT COUNT IN THE PARTE (PARESLTA) ARE UPDATED, USING EPAWTPAT,EPACYLA AND EPAWRTCT.			
20 FIELDS IN THE IORB AND IOSB ARE SET UP FOR IOS. THE SRB FOR IOS IS OBTAINED AND THE COUNT OF SRB (ASMIOCNT) FOR ILRIOC00 TO PROCESS IS UPDATED. THEN STARTIO IS ISSUED TO PROCESS PCCW'S ON THE IORB.			
21 PAREFSIP IS SET TO 0 TO UNLOCK THE PARTE IF LOCKED BY CURRENT PART MONITOR (EPACPUID).			
22 SCHEDULE ILRPTM SO THAT ANY IOE'S OR AIA'S PUT BACK ON THE QUEUES WILL BE PROCESSED.			

Notes	Routine	Label	Ref

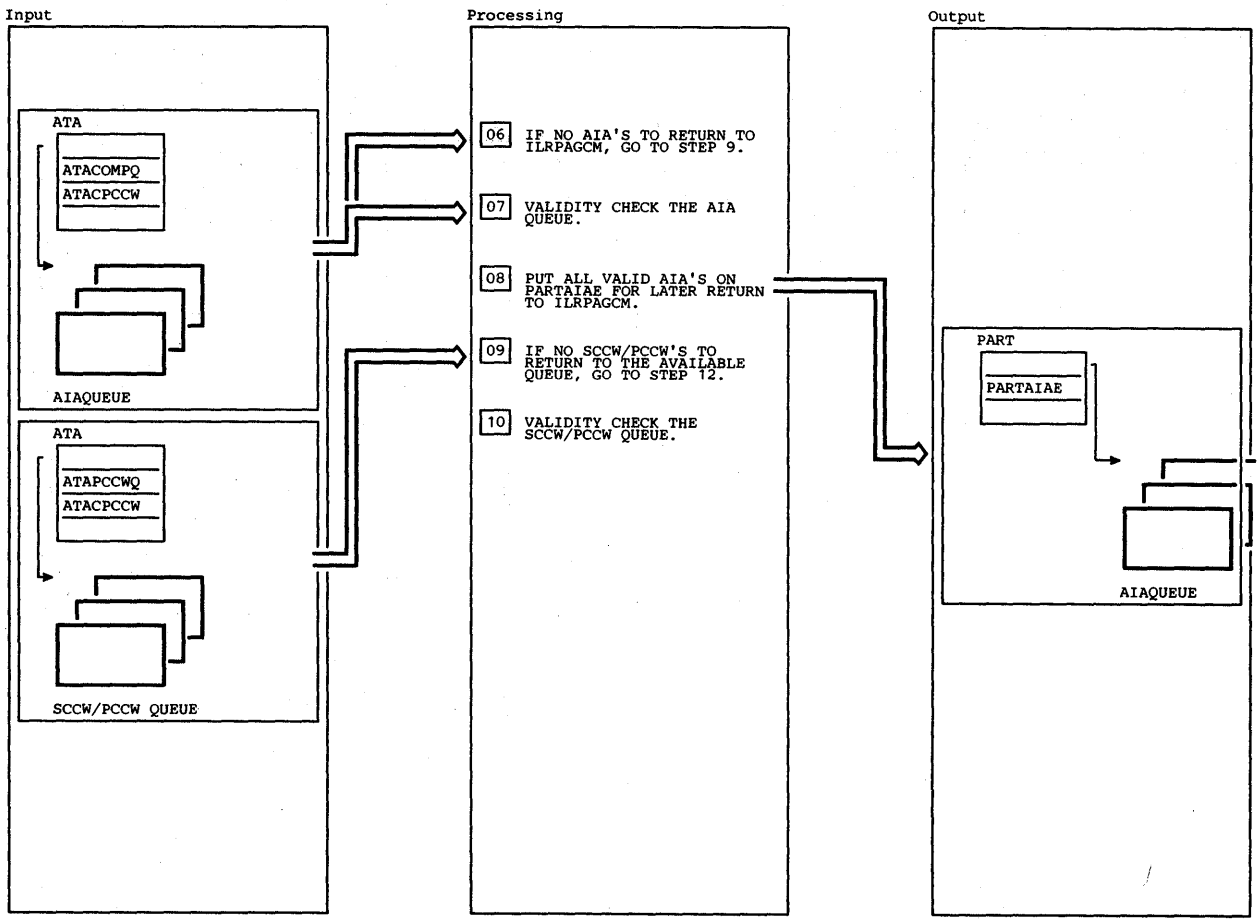
Diagram 25.22. ILRSRTO1 (Part 5 of 5)



Notes	Routine	Label	Ref
01 ILRCMP01 IS THE RECOVERY ROUTINE FOR ALL FOUR ENTRIES OF ILRCMP. THE ATA WILL ALWAYS BE RECORDED IN THE SDWA VARIABLE RECORDING AREA. THE MODID WILL BE SET IN THE SDWA IF NOT PERCOLATED TO.			
02 FOR A RECORD-ONLY ABEND (X'084', REASON CODE 4), ILRCMP WILL BE RESCHEDULED WITH A X'45' IN IOSCOD.			
03 IF THE IOSB HAS NOT BEEN CHECKPOINTED, EITHER THE IOSB HAS BEEN FREED OR THE ABEND OCCURRED BEFORE PROCESSING BEGAN. NO RECOVERY CAN BE DONE, SO GO TO STEP 22.			
04 THE IOSB-IORB-SRB WILL BE VALIDITY CHECKED FOR CERTAIN BASIC FIELDS AND THEN THE REMAINING FIELDS REFRESHED.	ILRFRR01	ILRVIORB	25.27.17
05 IF THE IOSB IS NOT VALID, NO RECOVERY IS DONE. GO TO STEP 22.			

Notes	Routine	Label	Ref

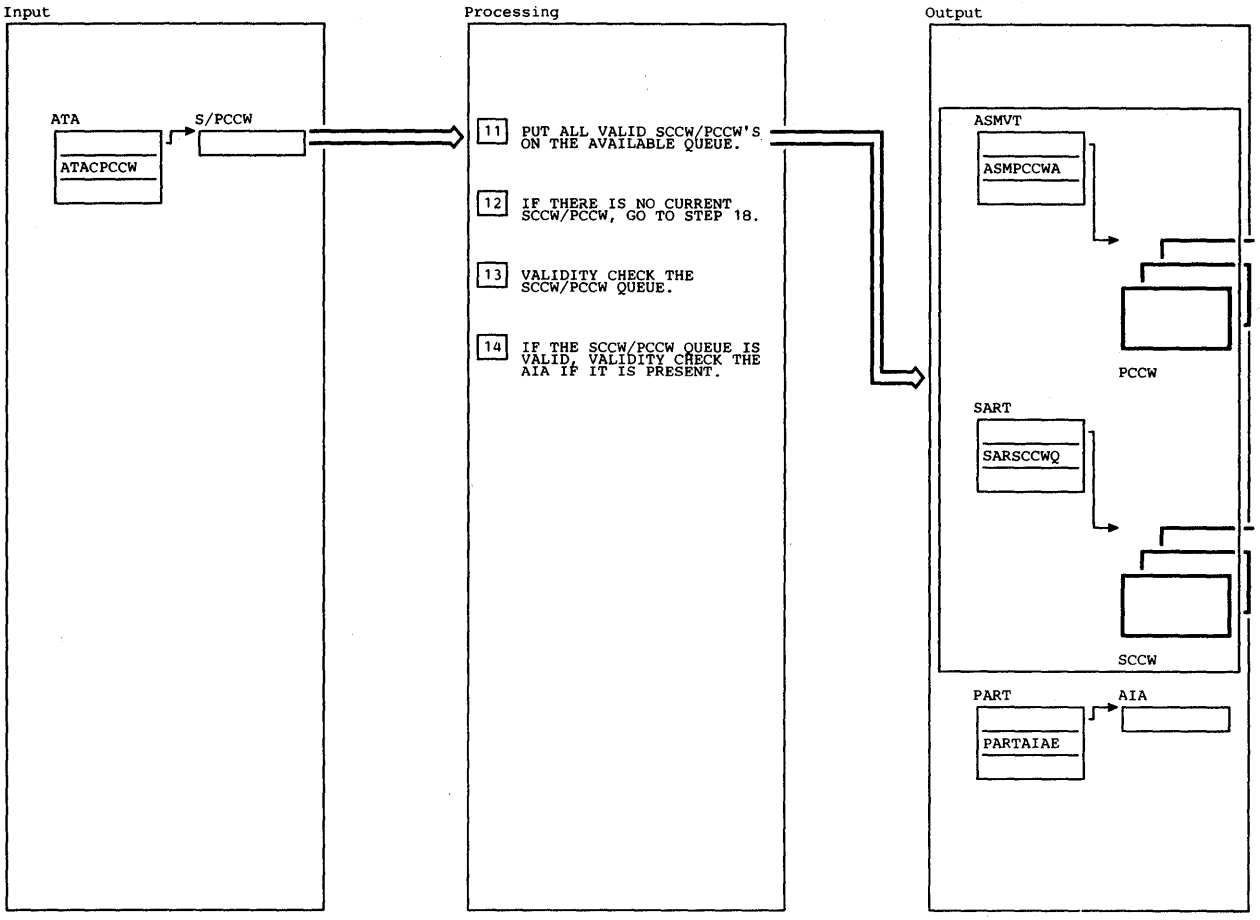
Diagram 25.23 ILRCMPO1 (Part 1 of 5)



Notes	Routine	Label	Ref
06 AIA'S TO BE RETURNED TO ILRPAGCM ARE CHECKPOINTED IN ATACOMPQ.			
07 BEFORE AIA'S ARE SENT TO THE VALIDITY CHECK ROUTINE, THE AIA POINTED TO BY THE CURRENT S/PCCW (ATACPCCW) WILL BE COMPARED WITH THE FIRST AIA ON THE ATACOMPQ QUEUE. IF A MATCH IS FOUND, THE AIA POINTER IN ATACPCCW IS SET TO ZERO SINCE THIS AIA ALREADY HAD BEEN PROCESSED BY ILRCMP BEFORE THE ERROR OCCURRED. THEN THE AIA'S ARE VALIDITY CHECKED.	ILRFRR01	ILRVAIAQ	25.27.4
08 IF A NON-ZERO QUEUE IS RETURNED, AIA'S ARE PUT ON PARTAIAE (PART AIA ERROR QUEUE). ILRPTM MUST BE SCHEDULED LATER.			
09 S/PCCW'S TO BE RETURNED TO THE AVAILABLE QUEUE ARE CHECKPOINTED IN ATAPCCWQ.			
10 BEFORE THE SCCW/PCCW'S ARE SENT TO THE VALIDITY CHECK ROUTINE, ATACPCCW WILL BE COMPARED TO THE FIRST SCCW/PCCW ON THE QUEUE. IF A MATCH IS FOUND, THE ATACPCCW IS SET TO ZERO SINCE THIS PCCW ALREADY HAD BEEN PROCESSED BY ILRCMP BEFORE THE ERROR OCCURRED. THEN THE VALIDITY CHECK ROUTINE IS CALLED.	ILRFRR01 ILRFRR01	ILRVSCWQ ILRVPCWQ	25.27.5 25.27.13

Notes	Routine	Label	Ref

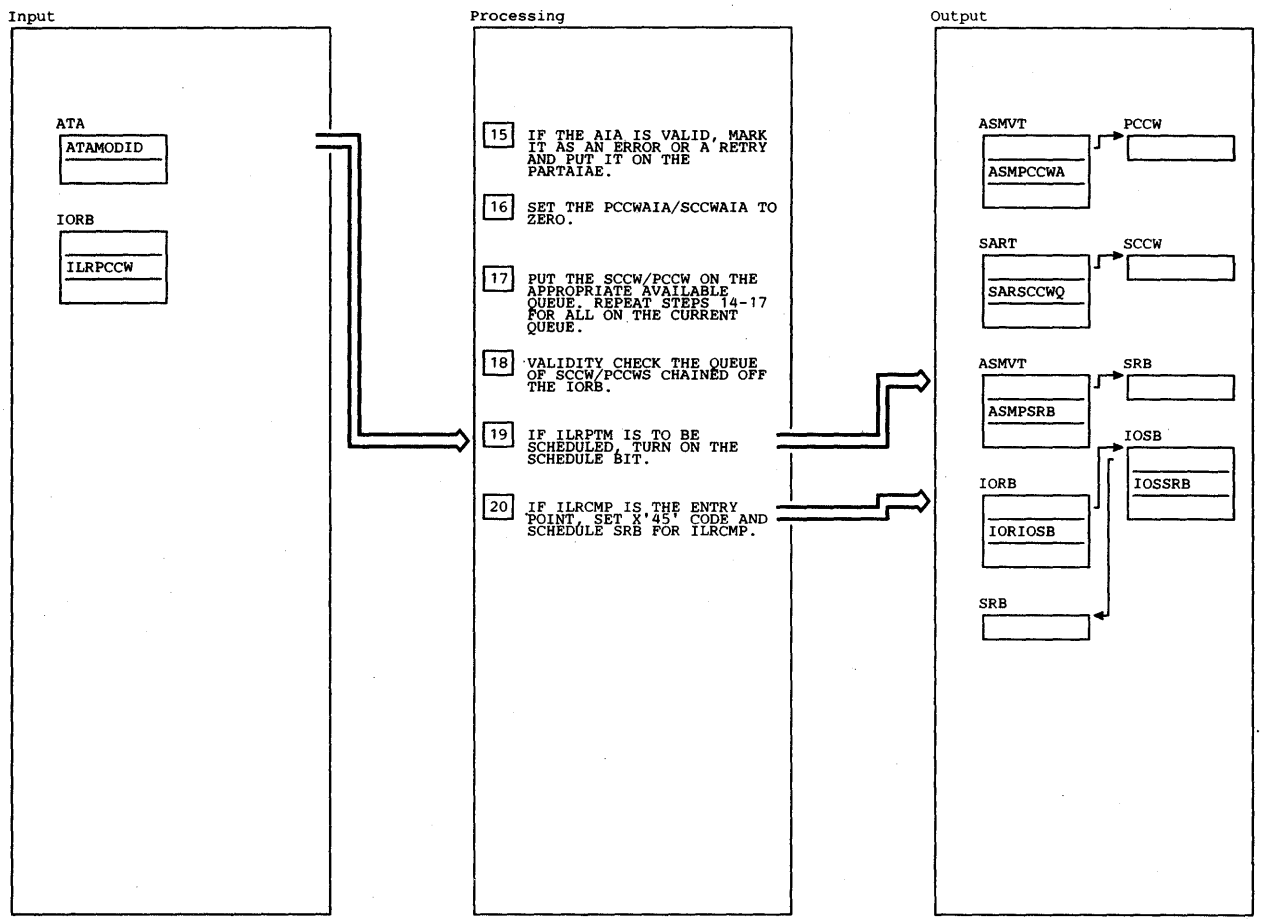
Diagram 25.23 ILRCMPO1 (Part 2 of 5)



Notes	Routine	Label	Ref
11 IF THE SCCW/PCCW'S ARE VALID, THEY ARE RETURNED TO THE APPROPRIATE AVAILABLE QUEUE.			
12 THE CURRENT SCCW/PCCW IS CHECKPOINTED IN THE ATACPCCW. THE AIA POINTER MAY BE ZERO. THE SCCW/PCCW MAY BE ON THE IORB CHAIN, OR THE ATAPCCWQ. SPECIAL CARE MUST BE TAKEN TO INSURE THAT NEITHER AN AIA NOR A SCCW/PCCW IS PROCESSED TWICE BY ILRSRT.			
13 THE SCCW/PCCW IS CHECKED AGAINST THE IORPCCW/IORSCCW FIELD. IF A MATCH IS FOUND, THE ATACPCCW WILL BE ZEROED AND CONTROL SENT TO STEP 18. IF NOT, THE SCCW/PCCW IS VALIDITY CHECKED.	ILRFRR01 ILRFRR01	ILRVSCWQ ILRVPCWQ	25.27.5 25.27.13
14 IF PCCWAIA/SCCWAIA IS NONZERO, THE AIA IS VALIDITY CHECKED.	ILRFRR01	ILRVAIA	25.27.9

Notes	Routine	Label	Ref

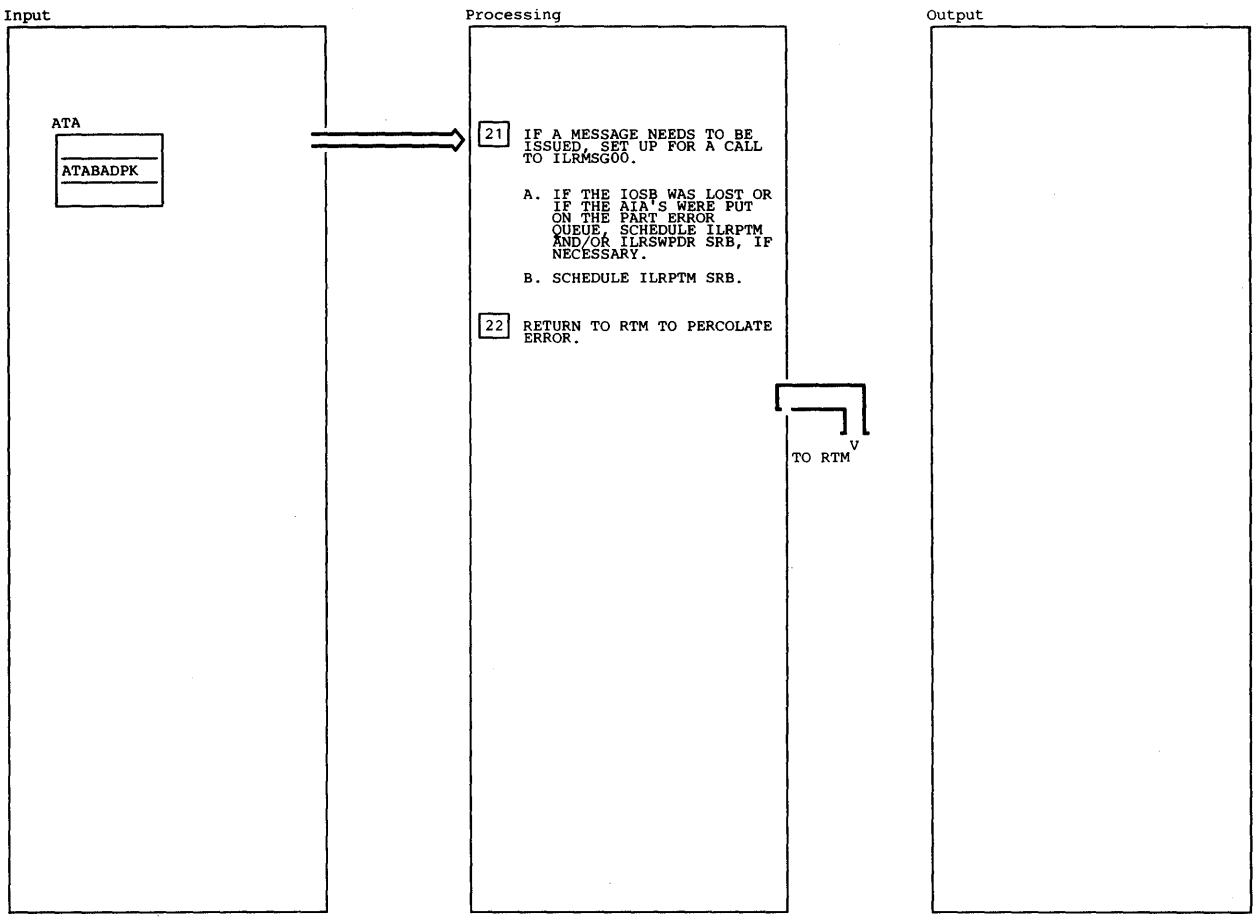
Diagram 25.23 ILRCMPO1 (Part 3 of 5)



Notes	Routine	Label	Ref
15 SINCE THE STATUS OF THIS AIA IS UNSURE, MARK IT AS A RETRY IF THERE IS ONLY ONE AIA. IF THERE IS A CHAIN OF AIAS, MARK IT AS AN ERROR SINCE IT MUST BE FOR BADPACK PROCESSING.			
16 THE AIA POINTER IS ALWAYS ZERO FOR A SCCW/PCCW ON THE AVAILABLE QUEUE.			
17 THE SCCW/PCCW IS MADE AVAILABLE FOR REUSE.			
18 WHATEVER REMAINS ON THE IORB IS PROCESSED BY THE MAINLINE TERMINATION ROUTINE - ILRCMP.	ILRFRR01 ILRFRR01	ILRVSCWQ ILRVPCWQ	25.27.5 25.27.13
19 IF ANY AIA'S WERE PUT ON THE PARTAIAE QUEUE, ILRPTM SHOULD BE SCHEDULED. THE SCHEDULE COUNT IS CHECKED TO DETERMINE IF ILRPTM IS ALREADY SCHEDULED.			
20 ILRCMP SHOULD BE RESCHEDULED USING THE SRB POINTED TO BY THE IOSB. A X'45' WILL BE PUT IN THE IOSB.			

Notes	Routine	Label	Ref

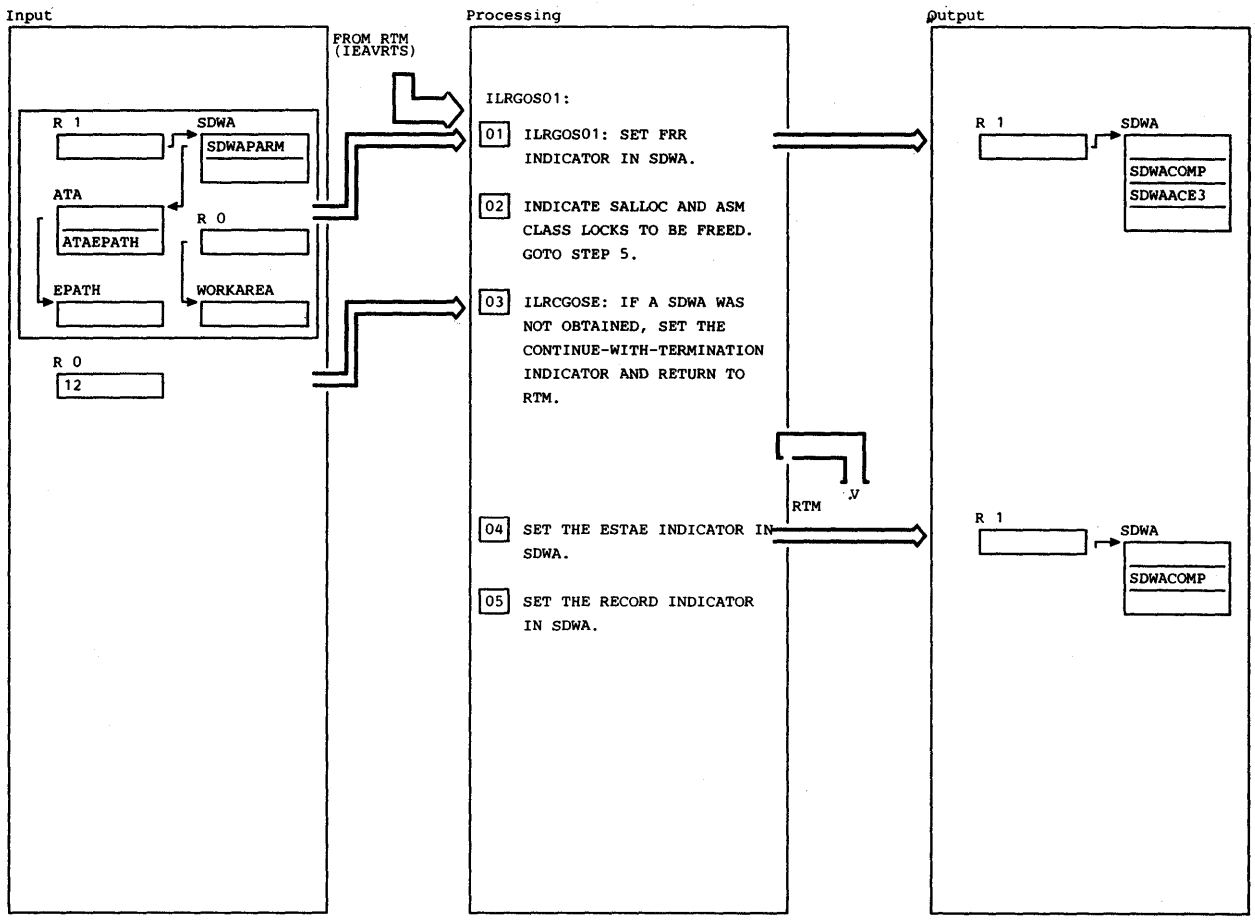
Diagram 25.23 ILRCMPO1 (Part 4 of 5)



Notes	Routine	Label	Ref
21 IF THE ERROR WAS IN THE BADPACK SUBROUTINE, COMPLETE SETTING UP OF PARAMETER LIST AND CALL ILRMSG00. UNCONDITIONALLY SET THE BADPACK FLAG IN THE PARTE OR THE SARTE. A. IF THE IOSB WAS LOST (ATAIOSB=0) OR THE AIAS WERE PUT ON THE PARTAIAE QUEUE, ILRPTM MUST BE SCHEDULED IF IT IS NOT ALREADY SCHEDULED. IF THE IOSB WAS LOST FOR A SWAP DATA SET THEN SWAP DRIVER IS ALSO SCHEDULED. B. IF THE ILRPTM SCHEDULE BIT IS ON, THEN SCHEDULE THE ILRPTM SRB IF IT IS NOT ALREADY SCHEDULED.	ILRMSG00	ILRMSG00	
22 FOR ILRCMPAE OR ILRCMPDI, IOS FRR WILL GET CONTROL AND SET THE IOSCOD TO X'45'. FOR IORIOC00 THERE IS NO FRR BELOW ILRCMP01 BUT ILRCMP HAS BEEN RESCHEDULED.			

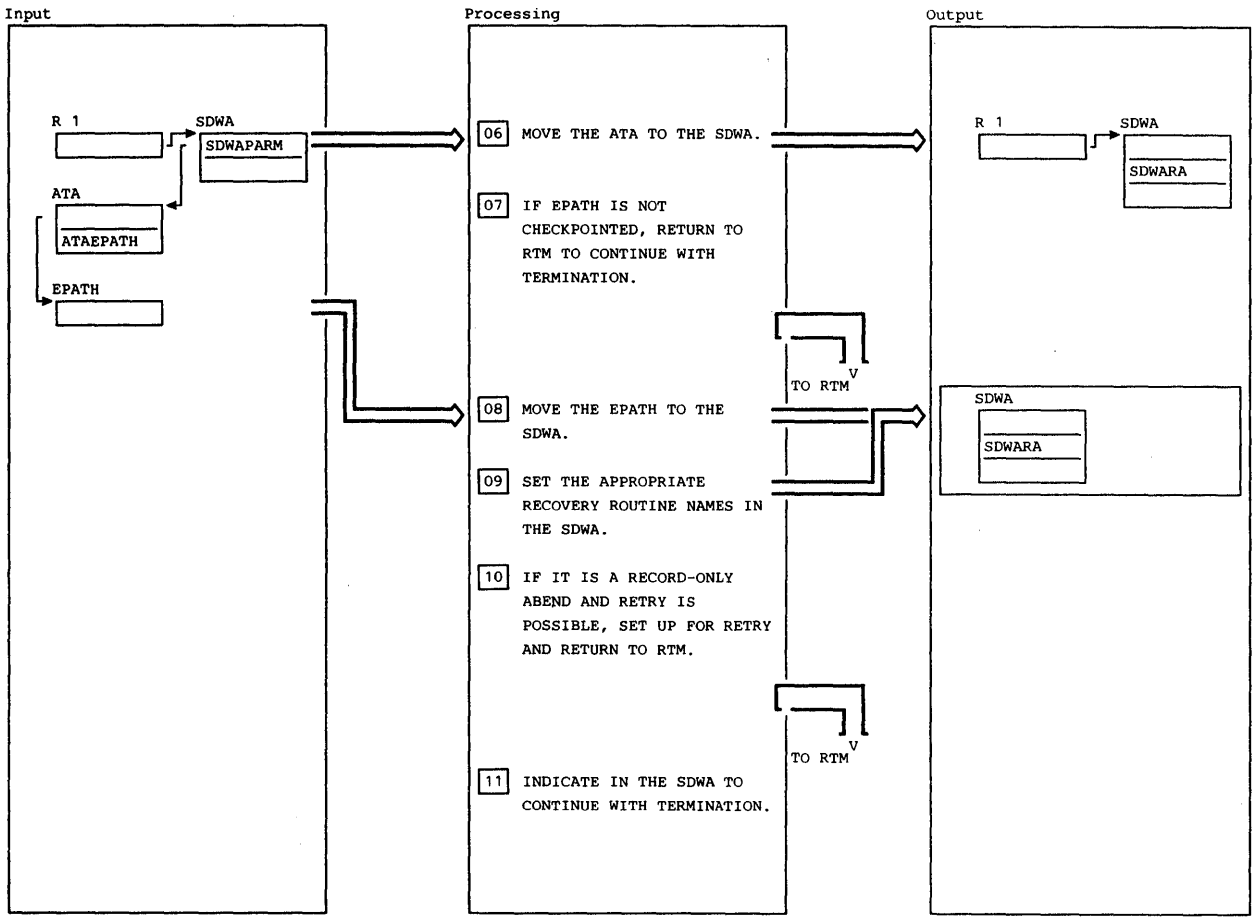
Notes	Routine	Label	Ref

Diagram 25.23 ILRCMPO1 (Part 5 of 5)



Notes	Routine	Label	Ref
01 ILRGOS01 IS THE RECOVERY ROUTINE FOR ILRGOS, ILRRLG, ILRACT, ILRSAV, AND ANY OF THEIR PATHS THROUGH ILRVSAMI. IT IS AN FRR FOR ILRGOS AND ILRRLG, AN ESTAE FOR THE OTHERS. FOR FRR ENTRY POINT, COMMUNICATION FIELD IN SDWA(SDWAPARM) WILL BE USED TO INDICATE WHETHER THIS IS THE FRR OR ESTAE PROCESSING. THE FRR WILL SET THE FIELD TO ZERO.			
02 WHEN ILRGOS RECEIVED CONTROL AND ESTABLISHED THE FRR, THE LOCAL LOCK WAS THE ONLY LOCK HELD. ALL OTHER LOCKS OBTAINED DURING MAINLINE OR RECOVERY PROCESSING SHOULD BE FREED BEFORE PERCOLATING TO VBP'S RECOVERY. GO TO STEP 5.			
03 THIS IS THE ESTAE ENTRY POINT GIVEN CONTROL BY RTM ROUTINE IEAVTAS1. IF NO SDWA WAS OBTAINED BY RTM, RECOVERY IS NOT ATTEMPTED.			
04 THE COMMUNICATION FIELD IN THE SDWA IS SET TO NON-ZERO FOR ESTAE PROCESSING. WHEN THE 200 BYTE WORKAREA IS OBTAINED ITS ADDRESS WILL BE PUT IN THAT FIELD.			
05 COMMON PROCESSING FOR ESTAE AND FRR - SDWA HAS BEEN OBTAINED. THE SDWA IS MARKED TO BE RECORDED IN SYS1.LOGREC.			

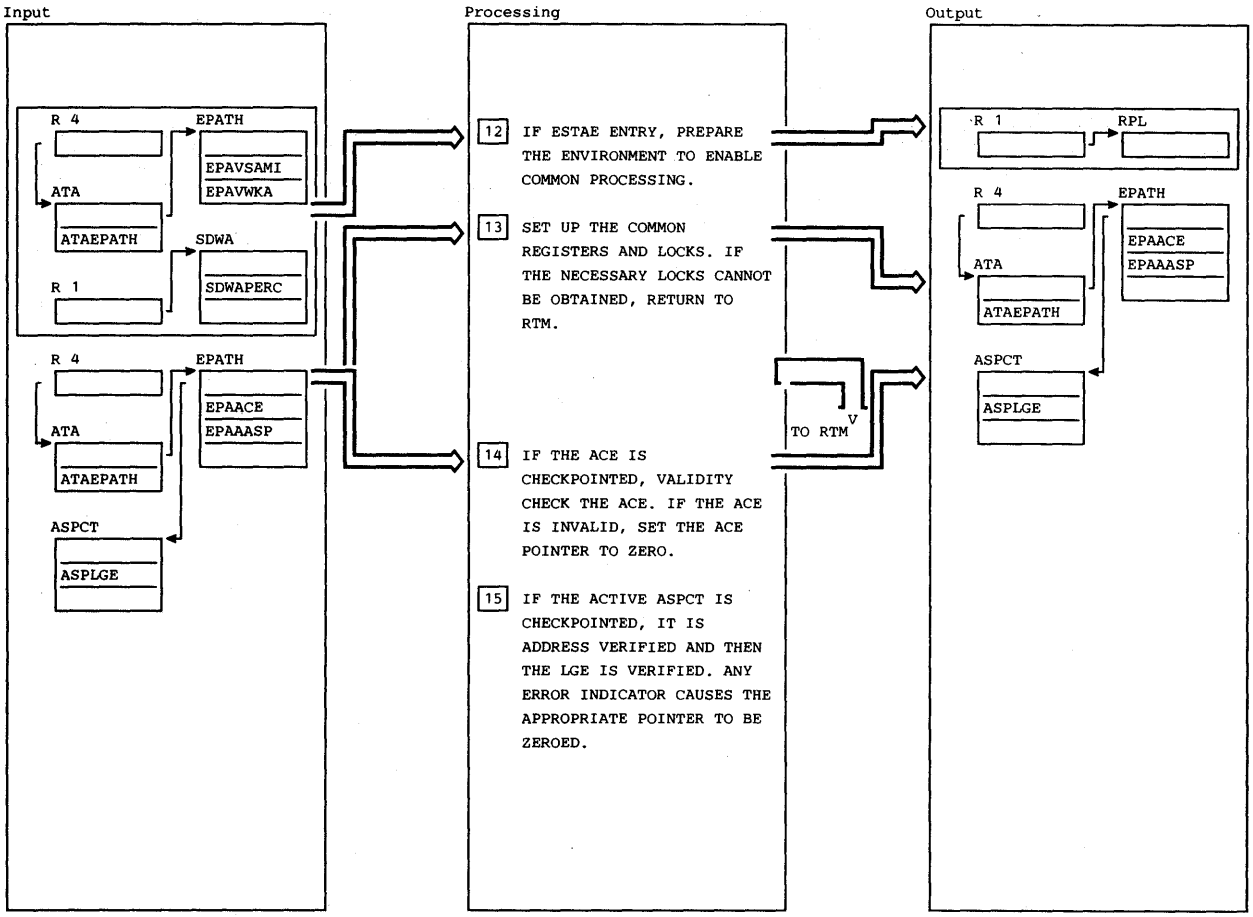
Diagram 25.24 ILRGOSO1 (Part 1 of 8)



Notes	Routine	Label	Ref
06 THE ATA IS RECORDED IN THE VARIABLE RECORDING AREA (SDWARA).			
07 IF THE EPATH HAS NOT BEEN CHECKPOINTED, NO RECOVERY IS ATTEMPTED.			
08 THE EPATH IS RECORDED IN THE VARIABLE RECORDING AREA.			
09 THE ROUTINE IN CONTROL AT THE TIME OF ERROR IS DETERMINED FROM THE ATA AND THE PROPER MODULE, CSECT, AND THE RECOVERY NAME IS PUT IN THE SDWA.			
10 IF IT IS A RECORD-ONLY ABEND (X'C0D', X'085', X'086', OR X'087'), SET UP THE RETRY REGISTERS FROM THE EPATH POINTER, INDICATE RETRY AT THE NEXT SEQUENTIAL INSTRUCTION, AND RETURN TO RTM.			
11 IF IT IS NOT A RECORD-ONLY ABEND OR RETRY IS IMPOSSIBLE, THEN			

Notes	Routine	Label	Ref
INDICATE IN THE SDWA TO CONTINUE WITH TERMINATION.			

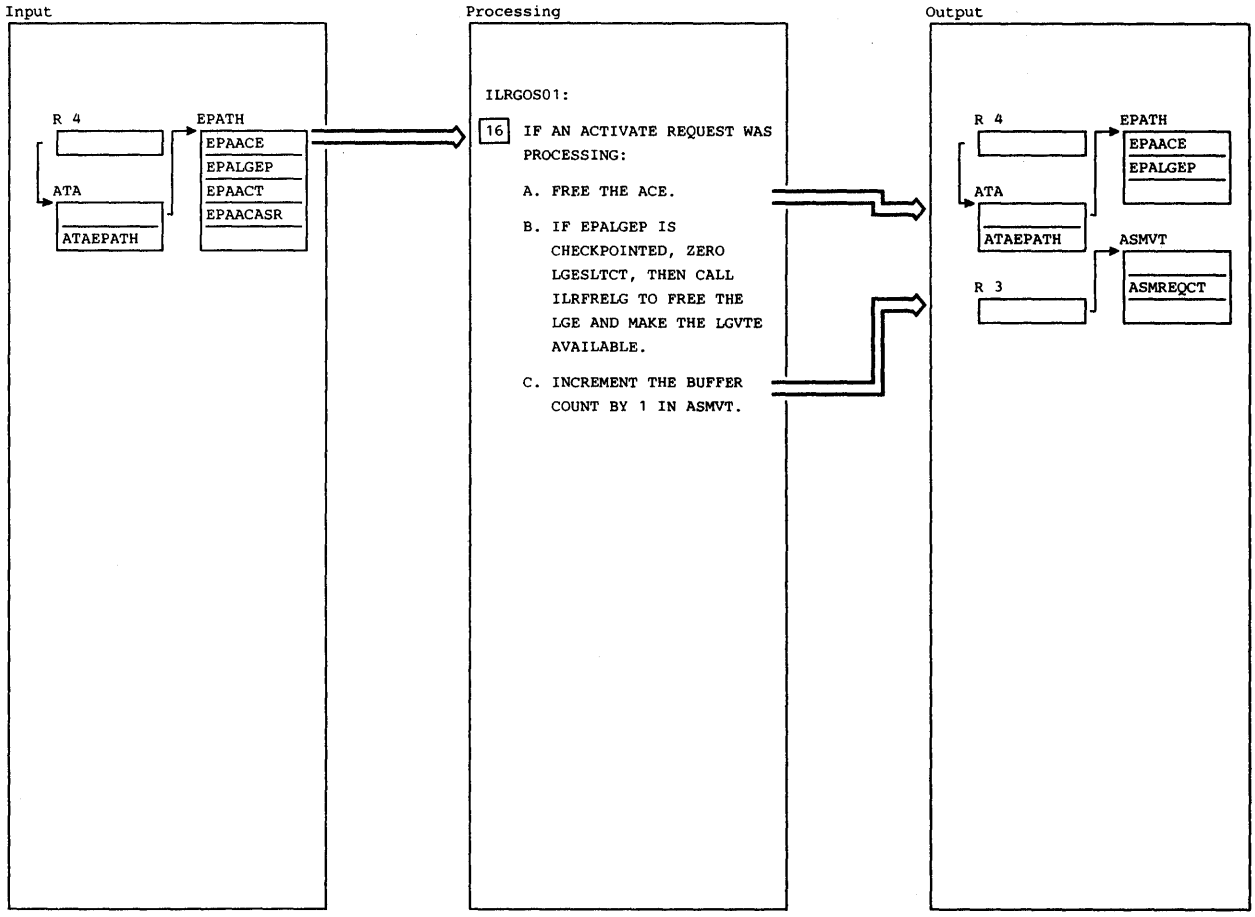
Diagram 25.24 ILRGOSO1 (Part 2 of 8)



Notes	Routine	Label	Ref
12 IF ESTAE ENTRY:			
A. OBTAIN A 200 BYTE WORKAREA. EACH FRR IS PASSED ONE.			
B. IF ILRVSAMI HAD CALLED VSAM (POSSIBLE ONLY IF ESTAE ENTRY), VSAM MUST BE ALLOWED TO CLEAN UP ITS RESOURCES. THE ENDREQ MACRO IS ISSUED.	ENDREQ		
C. IN PREPARATION OF OBTAINING THE SALLOC LOCK, PAGE FIX THE SDWA. SDWA IS FIXED IF FRR ENTRY.			
13 FOR COMMON RECOVERY PROCESSING, MAKE BOTH ENTRY POINTS HOLD THE SAME LOCKS AND SET UP COMMON REGISTERS. FOR THE ESTAE ENTRY POINT, THE LOCAL, SALLOC, AND ASM LOCKS ARE OBTAINED. FOR THE FRR ENTRY, THE LOCAL LOCK WAS ALREADY HELD AND THE OTHER LOCKS MAY BE HELD. THE SALLOC AND ASM LOCKS ARE OBTAINED. IF THE ASM WAS HELD AND THE SALLOC CANNOT BE OBTAINED CONDITIONALLY, NO RECOVERY IS DONE AND CONTROL IS	SETLOCK		

Notes	Routine	Label	Ref
RETURNED TO RTM.			
14 IF THE ACE IS CHECKPOINTED IT IS VALIDITY CHECKED. IF THE POINTER IS INVALID, THE ACE POINTER IN THE EPATH IS SET TO ZERO.	ILRFRR01	ILRVACE	25.27.10
15 IF THE ACTIVE ASPCT IS CHECKPOINTED, ADDRESS VERIFY THE ASPCT. IF THE ADDRESS IS INVALID, SET THE POINTER TO THE ACTIVE ASPCT TO ZERO. IF ASPCT ADDRESS IS VALID, VERIFY THE LGE ADDRESS IN THE ASPCT. IF LGE ADDRESS IS INVALID, ZERO POINTER TO LGE IN ASPCT.	ILRFRR01 IEAVEADV	ILRVLGE IEAVEADV	25.27.11

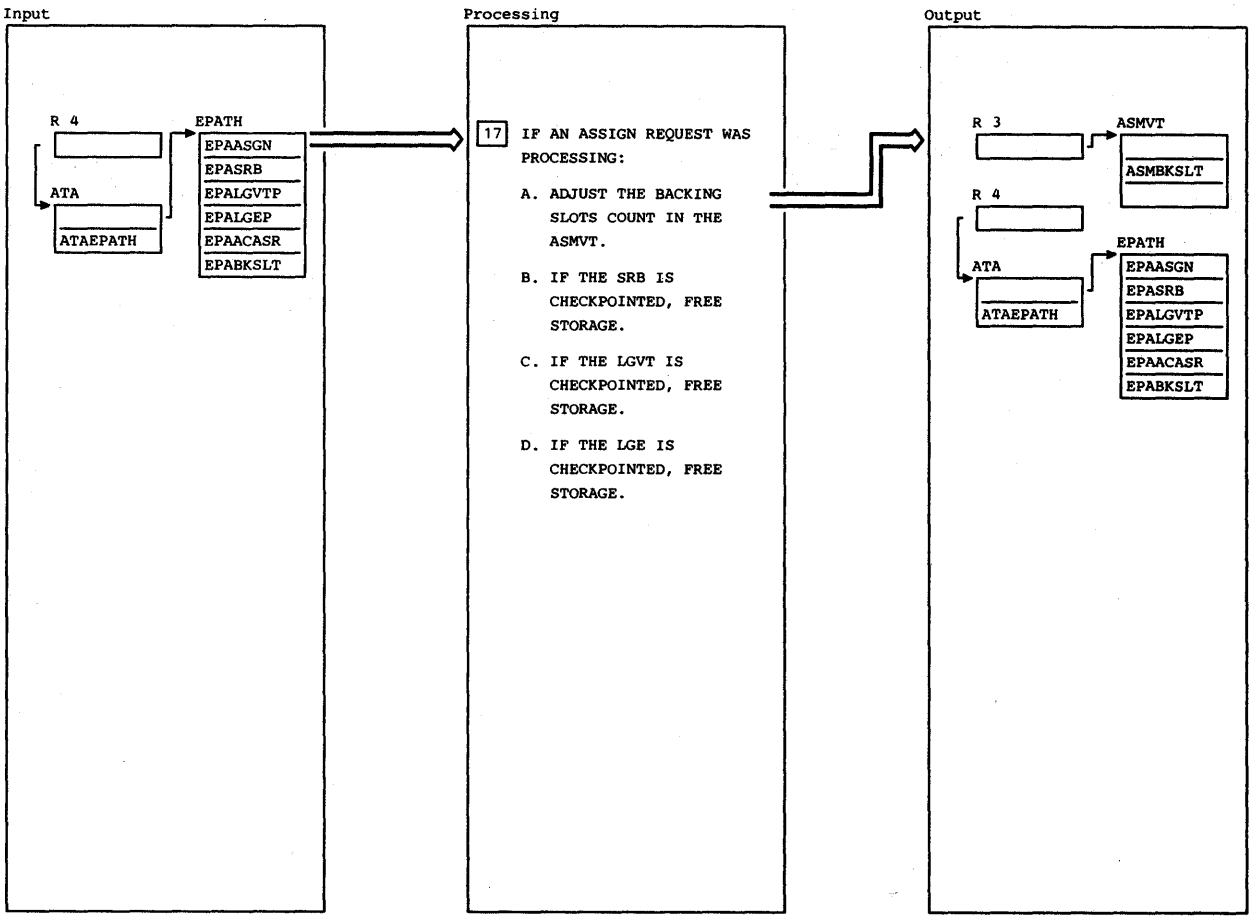
Diagram 25.24 ILRGOSO1 (Part 3 of 8)



Notes	Routine	Label	Ref
16 IF AN ACTIVATE REQUEST WAS PROCESSING, CLEAN UP ANY RESOURCES OBTAINED ON BEHALF OF THIS REQUEST.			
A. IF THE ACE IS STILL CHECKPOINTED, DEQUEUE THE ACE FROM THE PROCESS QUEUE AND RETURN IT TO THE ACE POOL.	ILRGMA		
B. IF AN LGE WAS OBTAINED, ILRFRELG IS CALLED TO FREE STORAGE AND MAKE THE LGVTE AVAILABLE.	ILRGOS	ILRFRELG	
C. THE BUFFER COUNT IN THE ASMVT MUST BE INCREMENTED SO THAT THE GROUP OPERATORS CAN CONTINUE TO DO I/O.			

Notes	Routine	Label	Ref

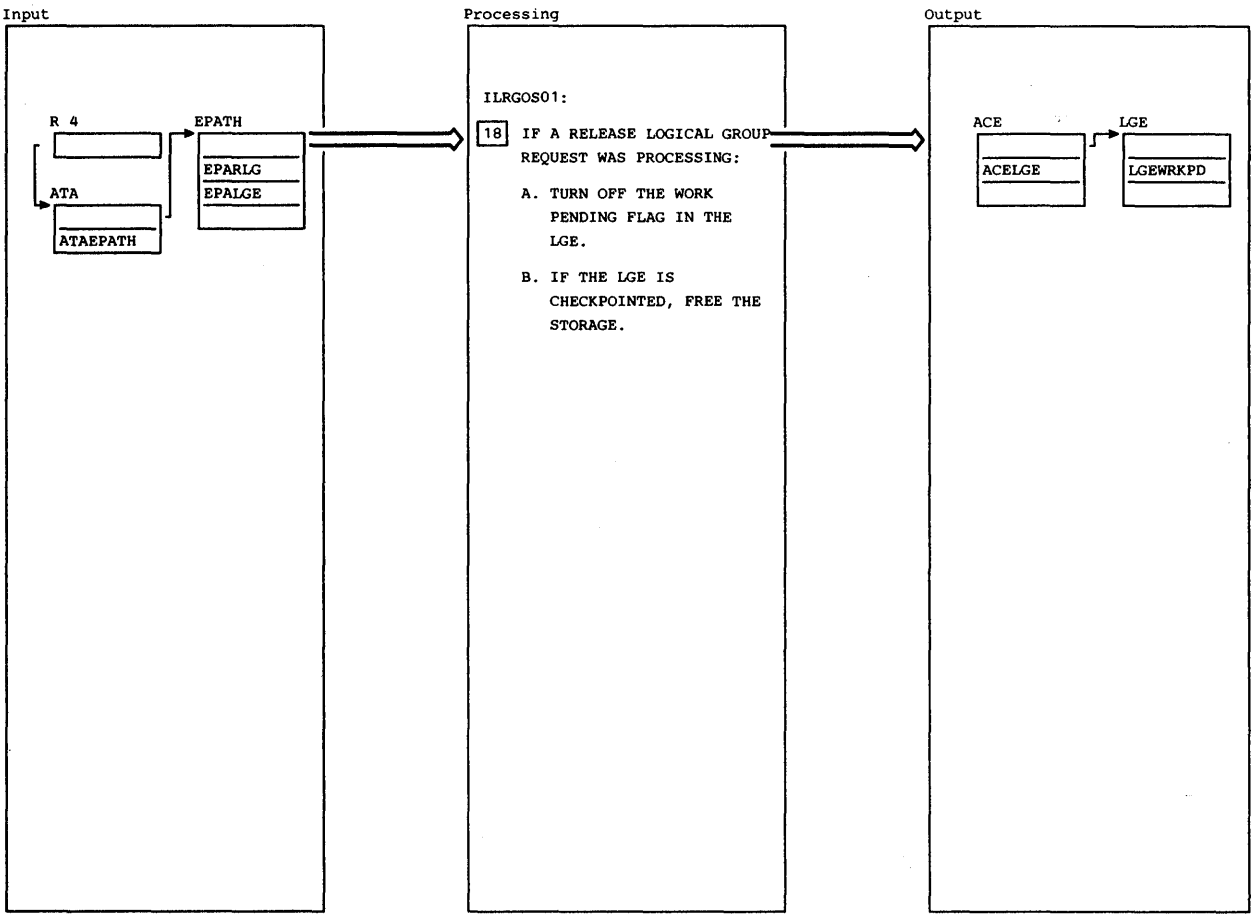
Diagram 25.24 ILRGOSO1 (Part 4 of 8)



Notes	Routine	Label	Ref
17 IF AN ASSIGN REQUEST IS BEING PROCESSED, CLEAN UP ANY RESOURCES OBTAINED ON BEHALF OF THIS REQUEST.			
A. IF EPAASGN IS ON, THE BACKING SLOTS COUNT OBTAINED FOR THIS LOGICAL GROUP MUST BE RETURNED. THE NUMBER OF SLOTS RETURNED IS ADDED TO THE ASMBKSLT COUNT IN ASMVT.			
B. IF THE SRB IS CHECKPOINTED, FREE THE SRB STORAGE.	FREEMAIN		
C. IF THE LGVT IS CHECKPOINTED, FREE THE LGVT STORAGE.			
D. IF THE LGE IS CHECKPOINTED, CALL ILRFRELG TO FREE LG RELATED STORAGE. IF UNSUCCESSFUL, FREE THE LGE STORAGE.	ILRGOS FREEMAIN	ILRFRELG	

Notes	Routine	Label	Ref

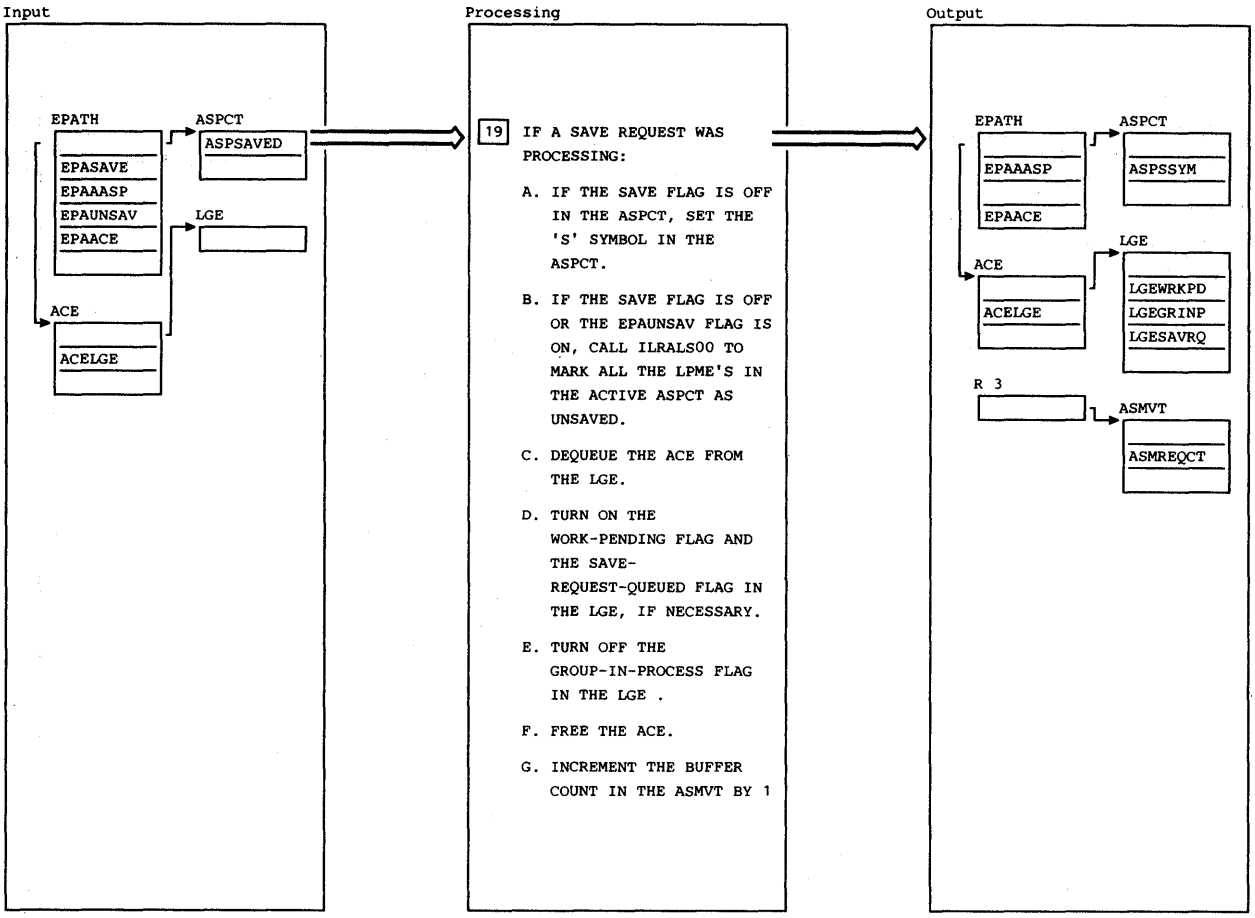
Diagram 25.24 ILRGOSO1 (Part 5 of 8)



Notes	Routine	Label	Ref
18 IF A RELEASE LOGICAL GROUP REQUEST WAS PROCESSING (ILRRLG), ALLOW THE REQUEST TO REMAIN ON THE PROCESS QUEUE UNTIL MEMORY TERMINATION.			
A. THE WORK PENDING FLAG IN THE LGE BEING OFF PREVENTS THE SRB CONTROLLER FROM PROCESSING THIS LGE.			
B. IF THE LGE IS STILL CHECKPOINTED, IT HAS NOT BEEN QUEUED, SO THIS SQA IS FREED.	FREEMAIN		

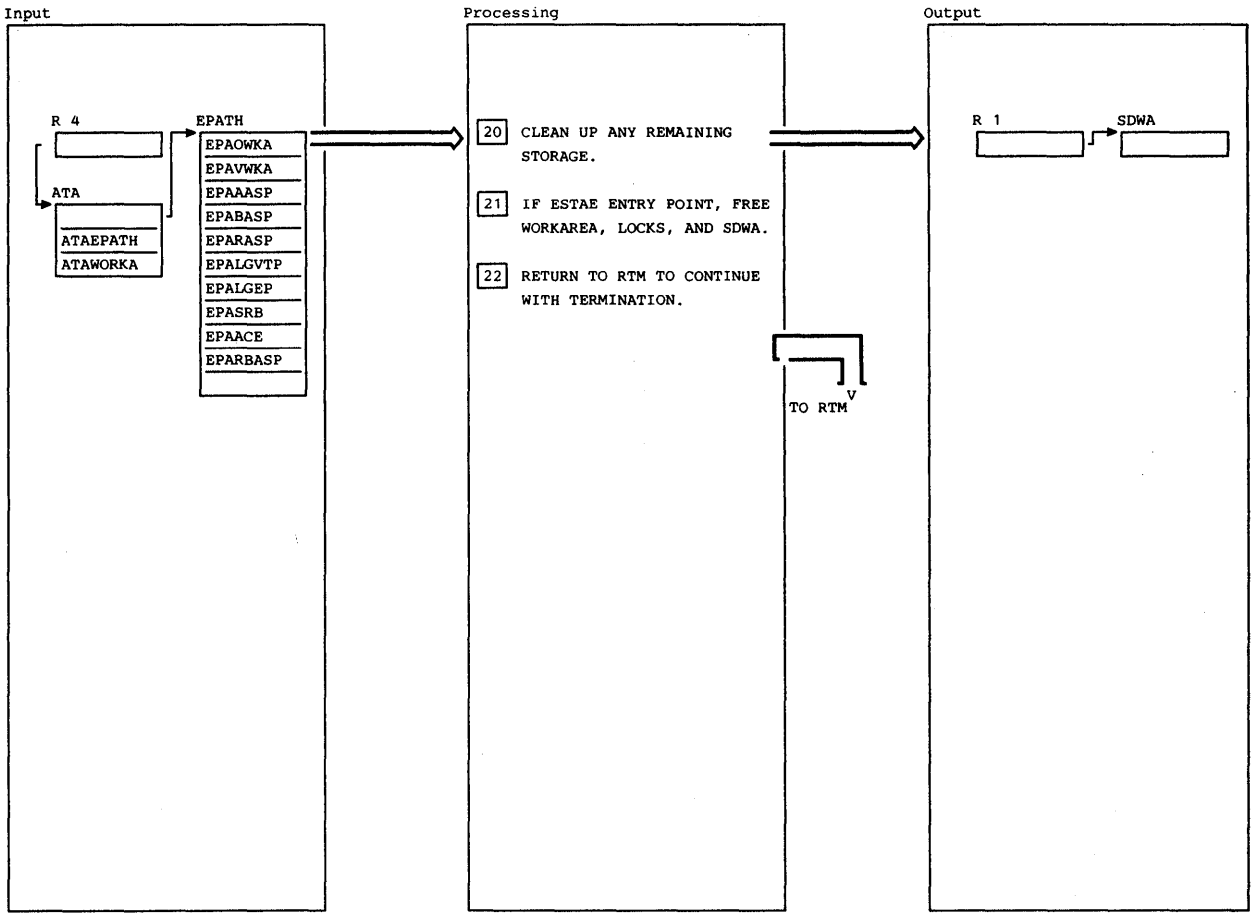
Notes	Routine	Label	Ref

Diagram 25.24 ILRGOS01 (Part 6 of 8)



Notes	Routine	Label	Ref
19 IF ILRSAV WAS PROCESSING, CLEAN UP THE RESOURCES OBTAINED FOR THIS REQUEST.			
A. IF THE ASPCT HAS NOT BEEN MARKED SAVE, ZERO THE 'S' SYMBOL SO THAT FUTURE RELEASE REQUEST WILL BE HONORED.			
B. IF THE ASPCT HAS NOT BEEN MARKED SAVED OR, IF THE EPAUNSAV FLAG IS ON, MARK ALL LPME'S AS UNSAVED. THIS WILL ALLOW SLOTS TO BE FREED LATER.	ILRALS00		
C. THE ACE SHOULD BE THE FIRST ACE ON THE LGE PROCESS QUEUE (LGEPROCQ).			
D. THE WORK-PENDING AND THE SAVE-REQUEST FLAGS IN THE LGE ARE TURNED OFF. IF MORE ACE'S ARE QUEUED, THE WORK-PENDING FLAG IS TURNED ON. IF MORE SAVE REQUESTS EXIST ON QUEUE, THE SAVE- REQUEST-QUEUED FLAG IS TURNED ON.			
E. THE GROUP-OP FLAG WAS ON DURING THE SAVE TO SERIALIZE WORK BEING DONE FOR THIS LGE.			
F. THE ACE IS FREED AND RETURNED TO THE POOL VIA ILRGMA.	ILRGMA		
G. THE BUFFER COUNT IN THE ASMVT SHOULD BE INCREMENTED BY 1 TO ALLOW ADDITIONAL I/O PROCESSING BY THE GROUP OPERATORS.			

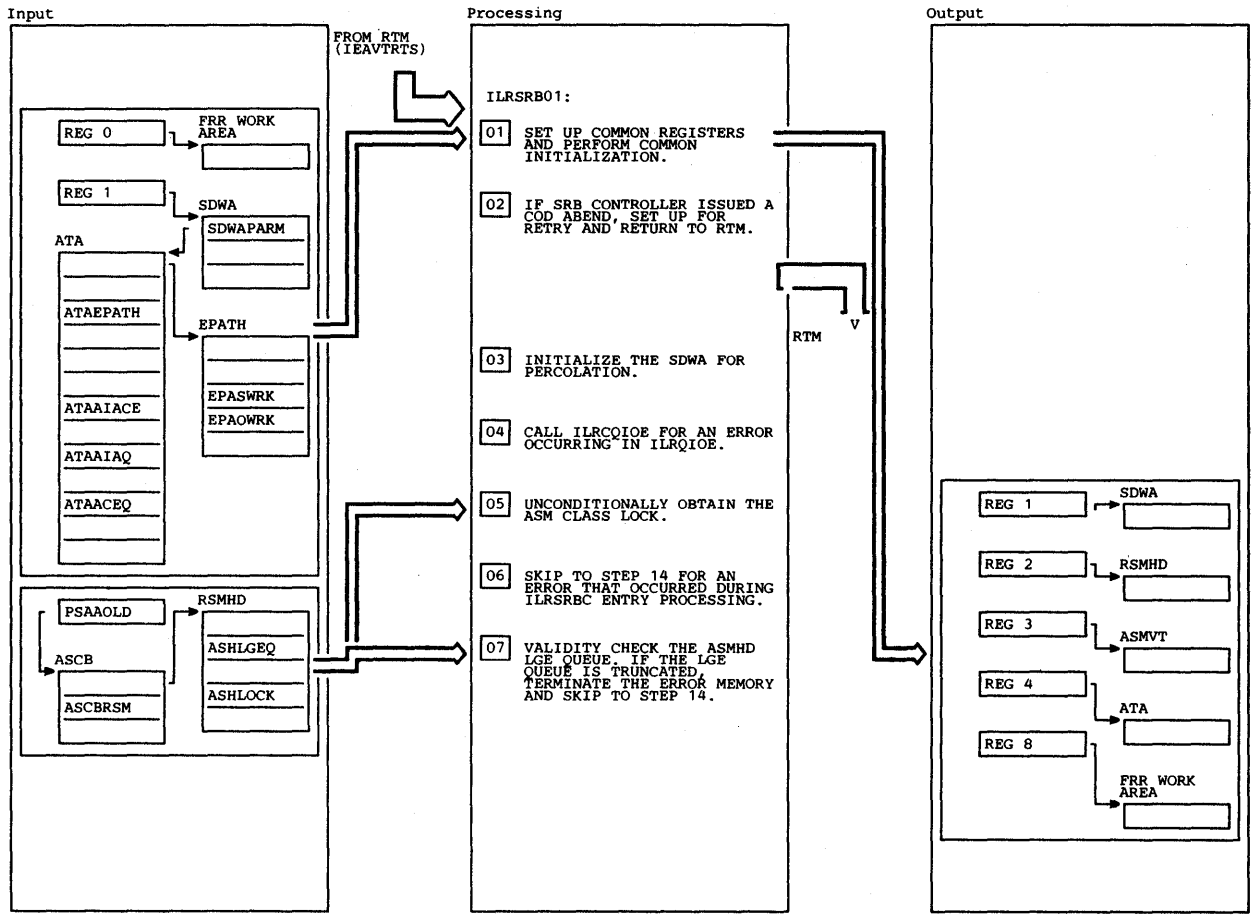
Diagram 25.24 ILRGOSO1 (Part 7 of 8)



Notes	Routine	Label	Ref
20 ANY WORKAREAS OR CONTROL BLOCKS STILL CHECKPOINTED AT THIS POINT ARE FREED. IN PREVIOUS STEPS WHERE AREAS HAVE BEEN FREED, THE EPATH POINTERS HAVE BEEN CLEARED.	FREEMAIN		
21 IF ESTAE ENTRY POINT, THE 200 BYTE WORKAREA MUST BE FREED, ALL LOCKS OBTAINED MUST BE FREED, AND THE SDWA MUST BE PAGE FREED.	FREEMAIN SETLOCK PGFREE		
22 THE SDWA HAS ALREADY BEEN SET UP TO CONTINUE WITH TERMINATION.			

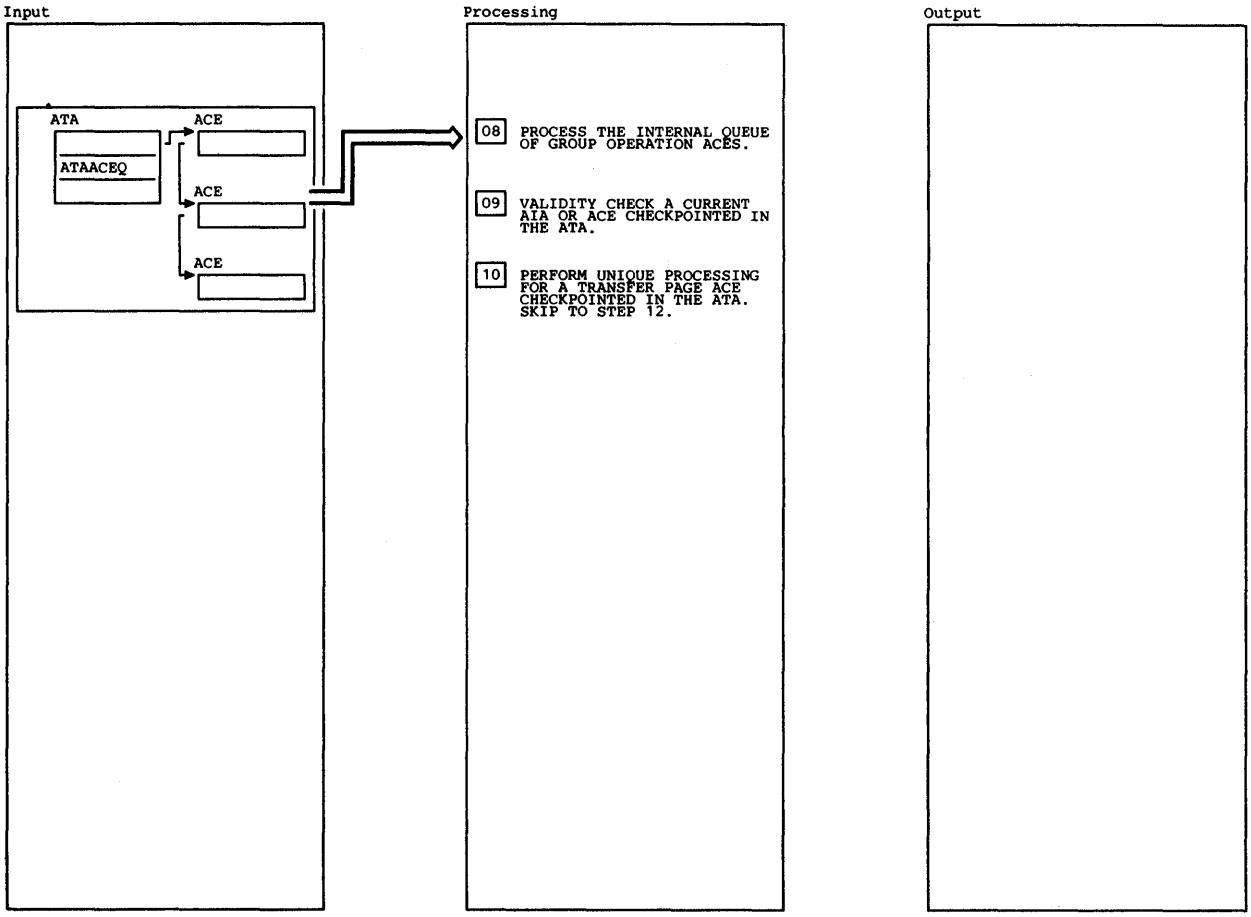
Notes	Routine	Label	Ref

Diagram 25.24 ILRGOSO1 (Part 8 of 8)



Notes	Routine	Label	Ref
01 PLACE THE NECESSARY POINTERS IN REGISTERS TO STANDARDIZE THE INTERFACE TO RECOVERY SUBROUTINES. COMMON INITIALIZATION INCLUDES: SETTING THE FRR WORKAREA TO ZERO AND COPYING THE ATA INTO THE SDWA.			
02 THE SRB CONTROLLER ISSUES A COD ABEND FOR AN AIA THAT DOES NOT CONTAIN A LOGICAL TO PHYSICAL MAPPING ENTRY (LPME). THE SDWA IS SET UP FOR RETRY AT THE NEXT SEQUENTIAL INSTRUCTION AFTER THE ABEND. MODULE, CSECT, AND RECOVERY ROUTINE IDS ARE COPIED INTO THE SDWA. A RETURN IS ISSUED TO RT/M.	SETRP		
03 THIS ROUTINE PERCOLATES FOR UNEXPECTED ABENDS. THE MODULE, CSECT, AND RECOVERY ROUTINE IDS ARE COPIED INTO THE SDWA. SDWA FLAGS, WHICH REQUEST THAT RT/M FREE THE ASM CLASS, SALLOC, AND LOCAL LOCKS, ARE TURNED ON. RT/M'S DEFAULT RECORDING PROCEDURE IS USED.	SETRP		
04 A FLAG IN THE ATA (ATAQIOE) INDICATES THAT ILRQIOE WAS IN CONTROL AT THE TIME OF ERROR. ILRSRBC'S AIA PROCESSING IS COMPLETED BEFORE THE CALL TO ILRQIOE. THE SDWA CSECT ID (SDWACSCT) IS RESET TO ILRQIOE. THE CALL TO ILRCQIOE MUST BE DONE PRIOR TO OBTAINING THE ASM CLASS LOCK.	ILRIOFRR	ILRCQIOE	25.20.1
05 THE LOCK MAY HAVE BEEN HELD ON ENTRY. THE LOCK IS USED TO SERIALIZE ASM PROCESSING FOR THIS MEMORY. IT WILL BE FREED BY RT/M ON PERCOLATION.	SETLOCK		
06 IF SRB CONTROLLER'S MODID IS NOT INITIALIZED IN THE ATA, IT IS NOT NECESSARY TO PERFORM QUEUE VERIFICATION. THE SRB SCHEDULED FLAG IN THE ASM HEADER (ASMSCHED) IS TURNED OFF TO INSURE THAT THE SRB CONTROLLER'S SRB IS RESCHEDULED.			
07 CALL ILRVLGEQ TO VERIFY EACH LGE ON THE ASM HEADER LGE QUEUE (ASHLGEQ) AND THE PROCESS QUEUE ANCHORED IN THE LGE (LGEPROCQ). ASM CANNOT BE ALLOWED TO PROCESS AYN FUTURE REQUESTS FOR THIS MEMORY ON A LOGICAL GROUP ENTRY (LGE) THAT MAY NOT EXIST. TERMINATE THE MEMORY USING THE ERROR SYSTEM COMPLETION CODE (SDWACMPC). ILRTERMR WILL RECOVER ASM RESOURCES.	ILRFRR01 ILRFRR01 CALLRTM	ILRVLGEQ ILRVLPRQ	25.27.11 25.27.2

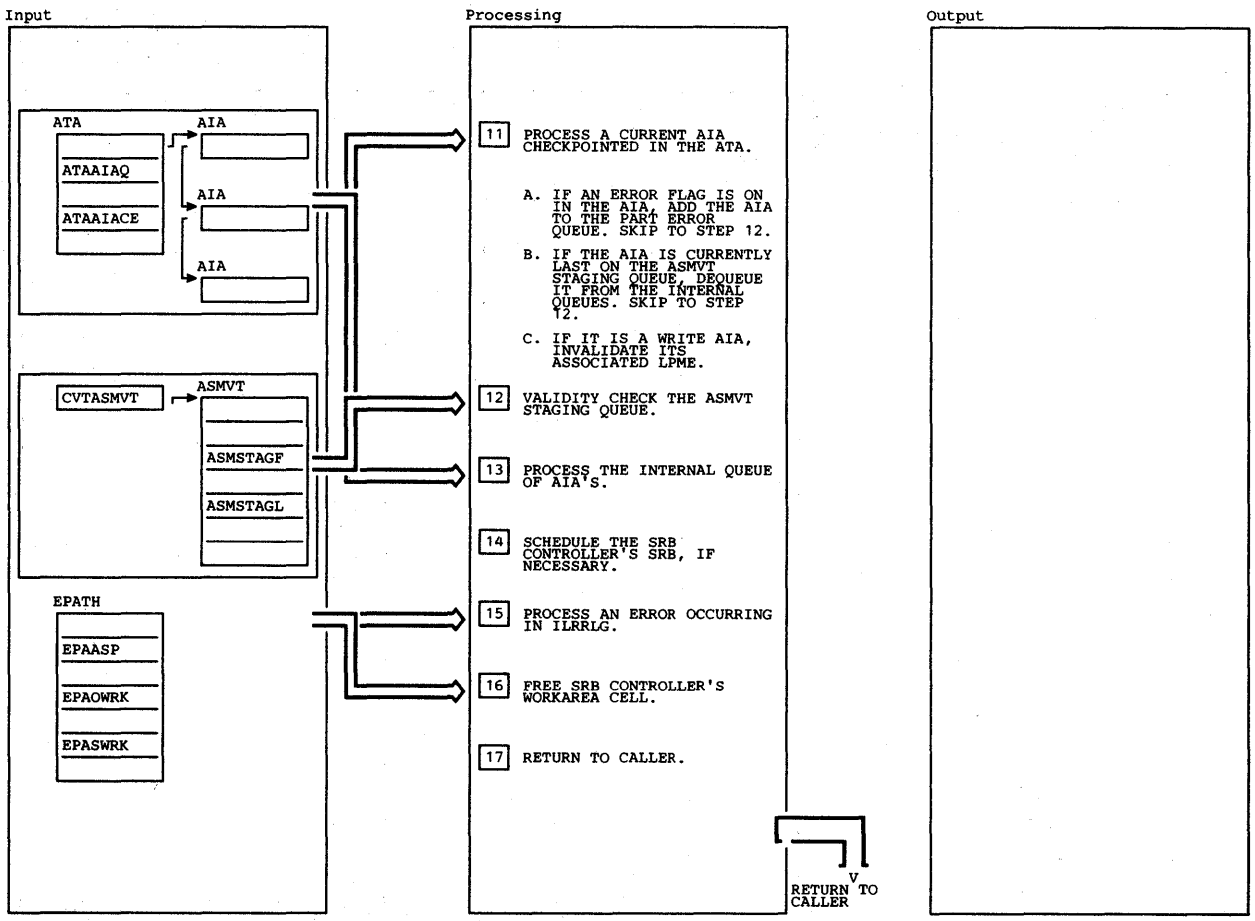
Diagram 25.25 ILRSRBO1 (Part 1 of 3)



Notes	Routine	Label	Ref
08 SRB CONTROLLER MAINTAINS IN THE ATA A LIFO QUEUE OF GROUP OPERATION ACES TO BE PROCESSED. AFTER VALIDITY CHECKING THE QUEUE, SET THE GROUP OP IN PROCESS FLAG (LGEGRINP) AND WORK PENDING FLAG (LGEWRKPD) IN THE ASSOCIATED LGE (ACELGE) SO THAT SRB CONTROLLER WILL REPROCESS THE ACES ON THIS INTERNAL QUEUE.	ILRFRR01	ILRVACEQ	25.27.16
09 VALIDITY CHECK THE CURRENT ACE (ILRTRANS PROCESSING) OR AIA (ILRESTRT PROCESSING). THE FIELD ATAAIA MAY BE EITHER AN ACE OR AN AIA. AN ACE IS IDENTIFIED BY THE TRANSFER PAGE ACE OPERATION CODE (ACEOP=X'04').	ILRFRR01	ILRVAIAC	25.27.8
10 IF THE ACE TARGET LPME (ACETLPME) IS MARKED VALID AND NOT SAVED, MARK THE LPME INVALID TO AVOID FREEING A SLOT TWICE.(IN NORMAL PROCESSING, IF THE TARGET LPME IS MARKED VALID BUT NOT SAVED, THE SLOT IS FREED. SINCE IT MAY HAVE BEEN FREED BEFORE THE ERROR OCCURRED, FREEING IT AGAIN WILL ALLOW ILRSRT TO ALLOCATE IT TWICE. IF IT HAD NOT BEEN FREED BEFORE THE ERROR, THE SLOT IS NEVER FREED.) TURN ON THE WORK PENDING FLAG FOR THE LGE (ACELGE) SO THAT THIS ACE IS REPROCESSED BY THE SRB CONTROLLER.			

Notes	Routine	Label	Ref

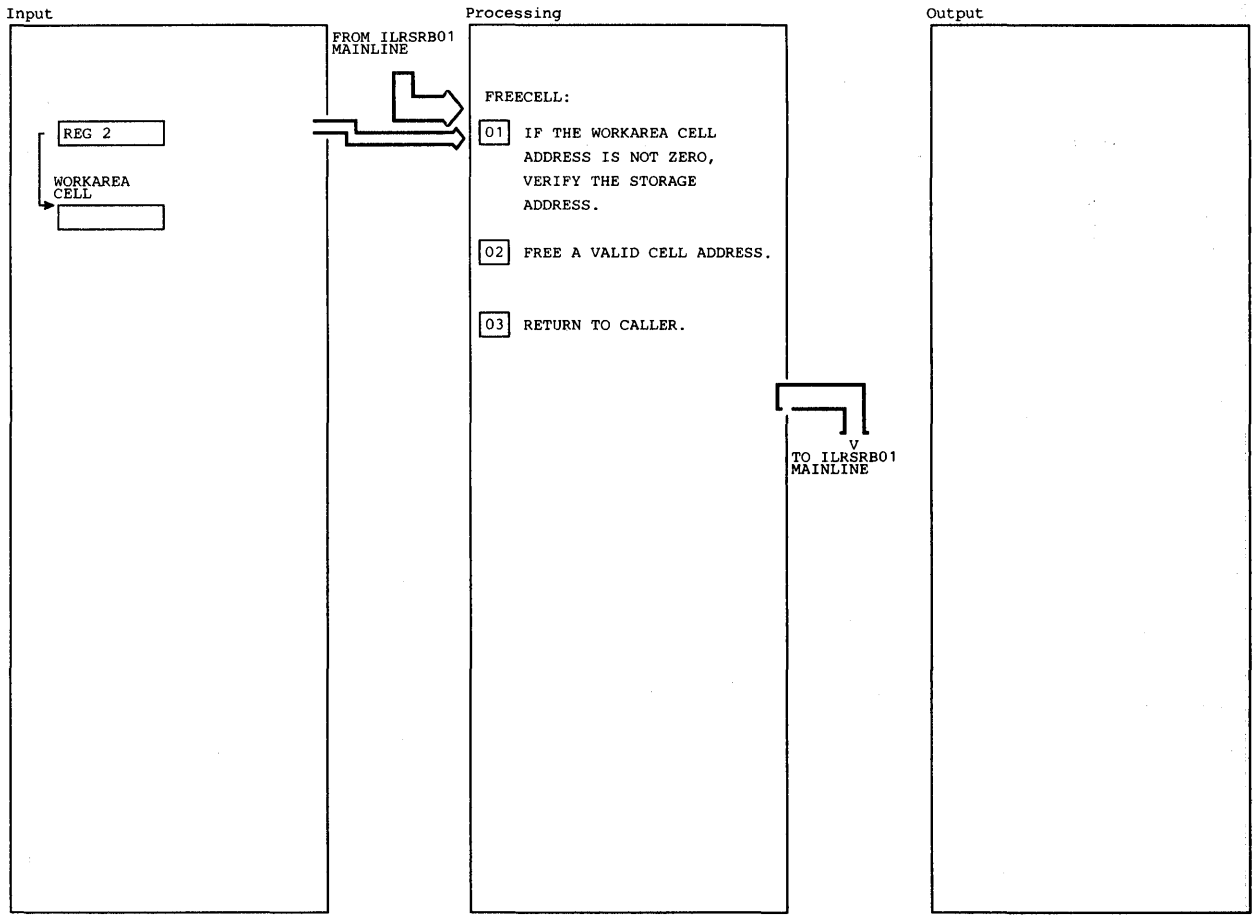
Diagram 25.25 ILRSRBO1 (Part 2 of 3)



Notes	Routine	Label	Ref
11 IF AN AIA WAS CURRENTLY BEING PROCESSED BY ILRESTRT, IT IS NECESSARY TO DETERMINE THE STAGE OF PROCESSING FOR THE AIA. A. IF AN ERROR FLAG IS ON IN THE CURRENT AIA, IT IS DEQUEUED FROM THE INTERNAL AIA QUEUE POINTED TO BY THE ATA AND ADDED TO THE PART ERROR QUEUE (PARTAIAE). B. DEQUEUEING THE AIA FROM THE INTERNAL AIA QUEUE INSURES THAT THIS AIA WILL NOT BE SETUP FOR REPROCESSING BY THIS RECOVERY. C. THE SLOT ASSOCIATED WITH A WRITE AIA MAY HAVE ALREADY BEEN FREED. INVALIDATING THE LPME INSURES THAT IT WILL NOT BE FREED AGAIN.			
12 THE VALIDITY CHECKING ROUTINE REMOVES A PARTIALLY QUEUED AIA.	ILRFRR01	ILRVASGQ	25.27.1
13 SRB CONTROLLER MAINTAINS A QUEUE OF STARTABLE AIAS (ATAAIAQ). THE CURRENT AIA (ATAAIACE) REPRESENTS THAT PART OF THE AIA QUEUE NOT YET PROCESSED BY THE RESTART SUBROUTINE. IF THE CURRENT AIA IS NOT ZERO, RESET THE INTERNAL QUEUE TO THE CURRENT AIA. VALIDITY CHECK THE INTERNAL QUEUE OF AIAS. TURN OFF THE AIA IN PROCESS FLAG AND TURN ON THE LGE WORK PENDING FLAG FOR EACH AIA ON THE INTERNAL QUEUE IN ORDER TO ALLOW THESE AIAS TO BE REPROCESSED BY ILRSRBC.	ILRFRR01	ILRVAIAQ	25.27.4
14 THE SRB CONTROLLER'S SRB IS RESCHEDULED IF NOT ALREADY SCHEDULED. THE ASM HEADER SCHEDULE FLAG, ASHSCHED, IS TURNED ON TO INDICATE THE SRB CONTROLLER IS SCHEDULED FOR THIS MEMORY.	SCHEDULE		

Notes	Routine	Label	Ref
15 IF RELEASE LOGICAL GROUP WAS IN CONTROL AT THE TIME OF ERROR, AN ACTIVE ASPCT (EPAASP) AND A WORKAREA CELL (EPAOWRK) MAY BE CHECKPOINTED IN THE EPATH. THE ACTIVE ASPCT ADDRESS IN THE ASSOCIATED LGE (LGEASP) IS CLEARED TO PREVENT FREEING THE ASPCT'S SLOTS TWICE IN THE EVENT OF ANOTHER RELEASE LOGICAL GROUP REQUEST FOR THE LGE. THE WORKAREA CELL IS FREED.		FREECELL	25.25.1
16 SRB CONTROLLER'S WORKAREA CELL IS CHECKPOINTED IN THE EPATH(EPASWRK).		FREECELL	25.25.1
17 THIS RECOVERY PATH ALWAYS PERCOLATES.			

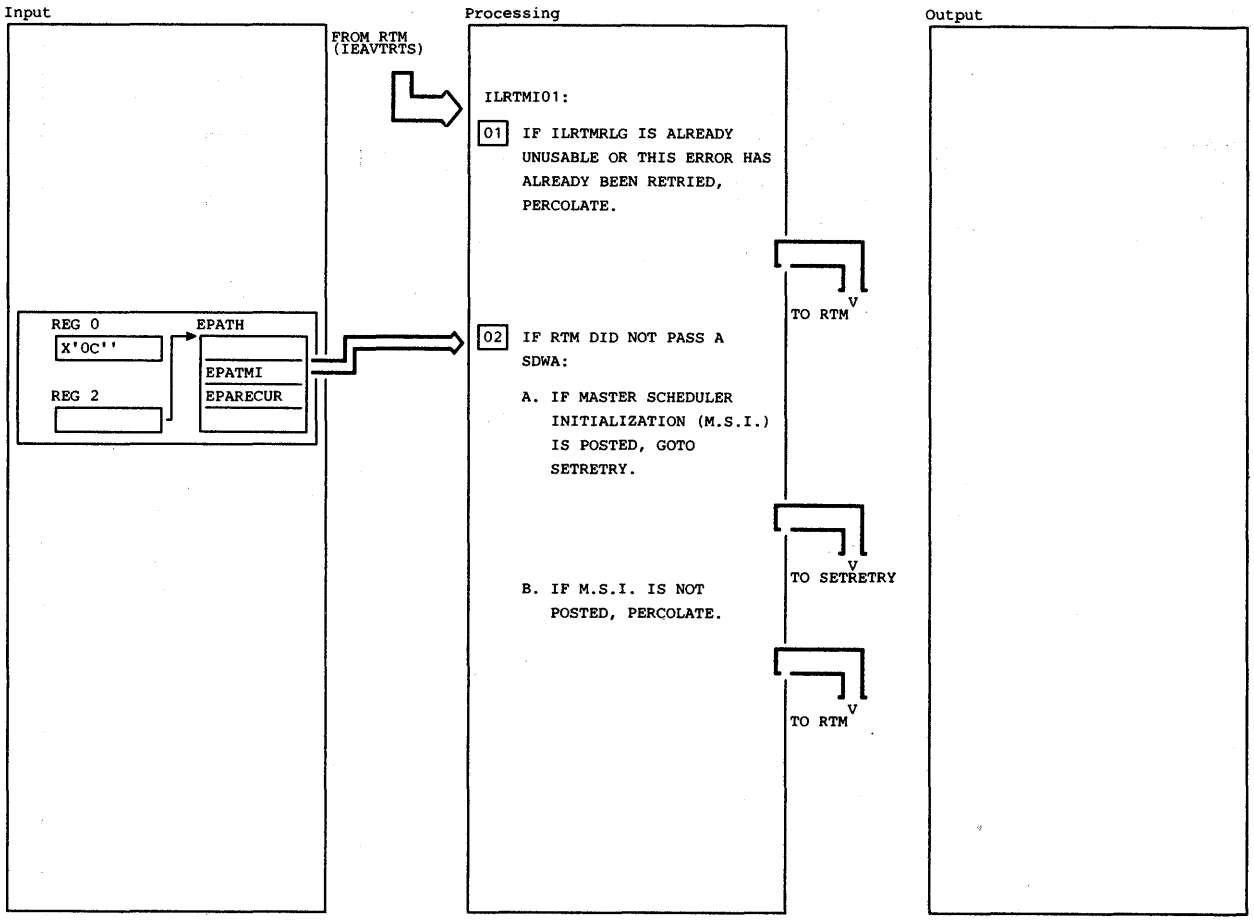
Diagram 25.25 ILRSRBO1 (Part 3 of 3)



Notes	Routine	Label	Ref
01 THE STORAGE POINTED TO BY THE WORKAREA CELL ADDRESS IS VERIFIED TO BE ADDRESSABLE AND FREE OF STORAGE CHECKS.	IEAVEADV	IEAVEADV	
02 IF THE STORAGE IS VALID THE WORKAREA CELL IS RETURNED TO THE PROPER ASM CELL POOL.	ILRGMA		

Notes	Routine	Label	Ref

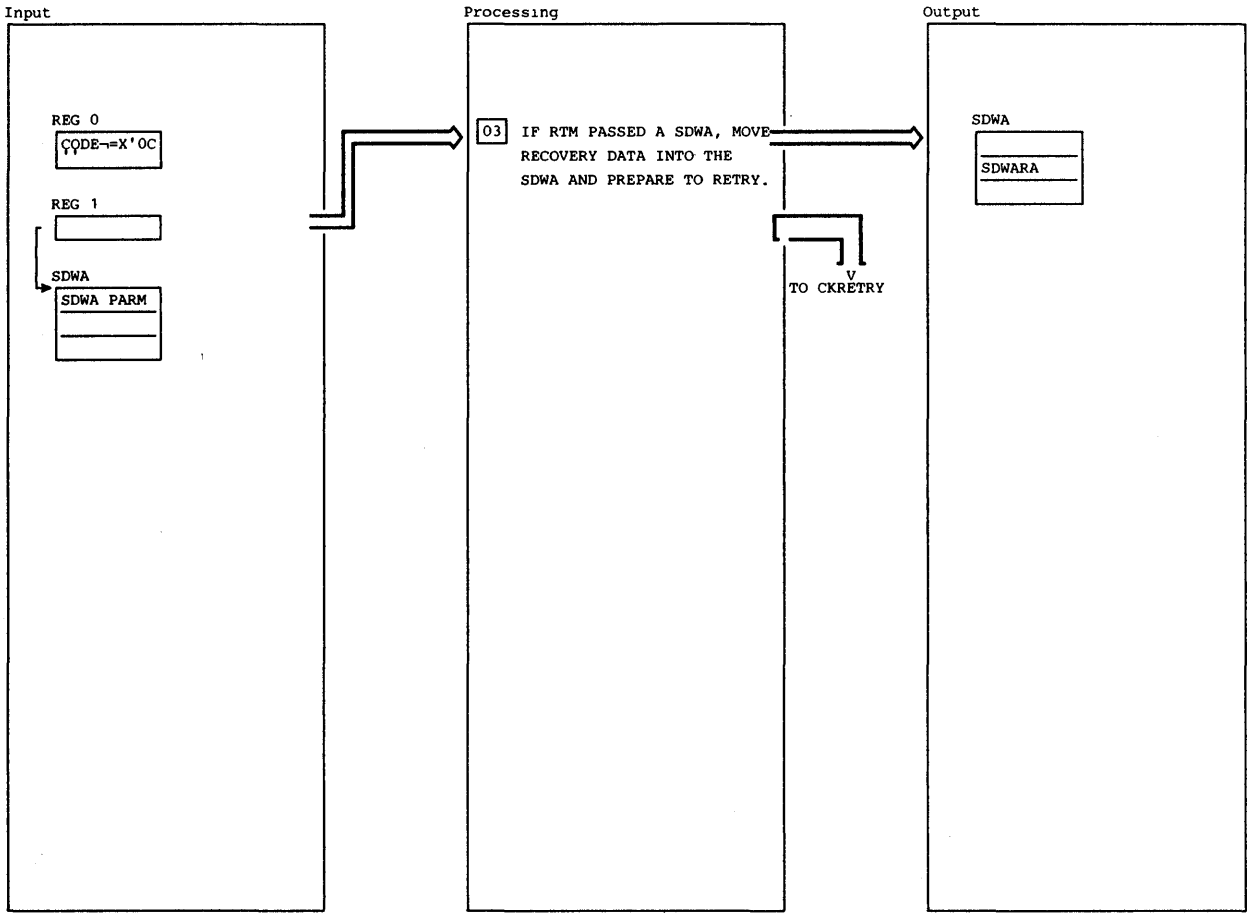
Diagram 25.25.1 FREECELL (Part 1 of 1)



Notes	Routine	Label	Ref
01 ILRTMI01 IS THE RECOVERY ROUTINE FOR ILRTMRLG AND ILRTMI00. IF ASMNOTMR=1 OR EPARECUR=1, A DOUBLE ERROR HAS OCCURRED SO SET ASMNOTMR TO ONE, IF NOT ALREADY, AND PERCOLATE.			
02 RTM COULD NOT OBTAIN A SDWA.			
A. IF RTM DID NOT PASS A SDWA, THERE IS NO WAY TO TELL RTM TO RETRY WITH UPDATED REGISTERS. IF MASTER SCHEDULER INITIALIZATION HAS BEEN POSTED (EPAMAST=1), ILRTMRLG MAIN LINE CODE WAS PROCESSING, THUS WE HAVE ENOUGH INFORMATION TO DO A SPECIAL RETRY.		SETRETRY	25.26.1
B. IF MASTER SCHEDULER INITIALIZATION IS NOT POSTED AND RTM COULD NOT GET STORAGE FOR A SDWA, PERCOLATE AND ALLOW M.S.I. TO TERMINATE THE IPL.			

Notes	Routine	Label	Ref

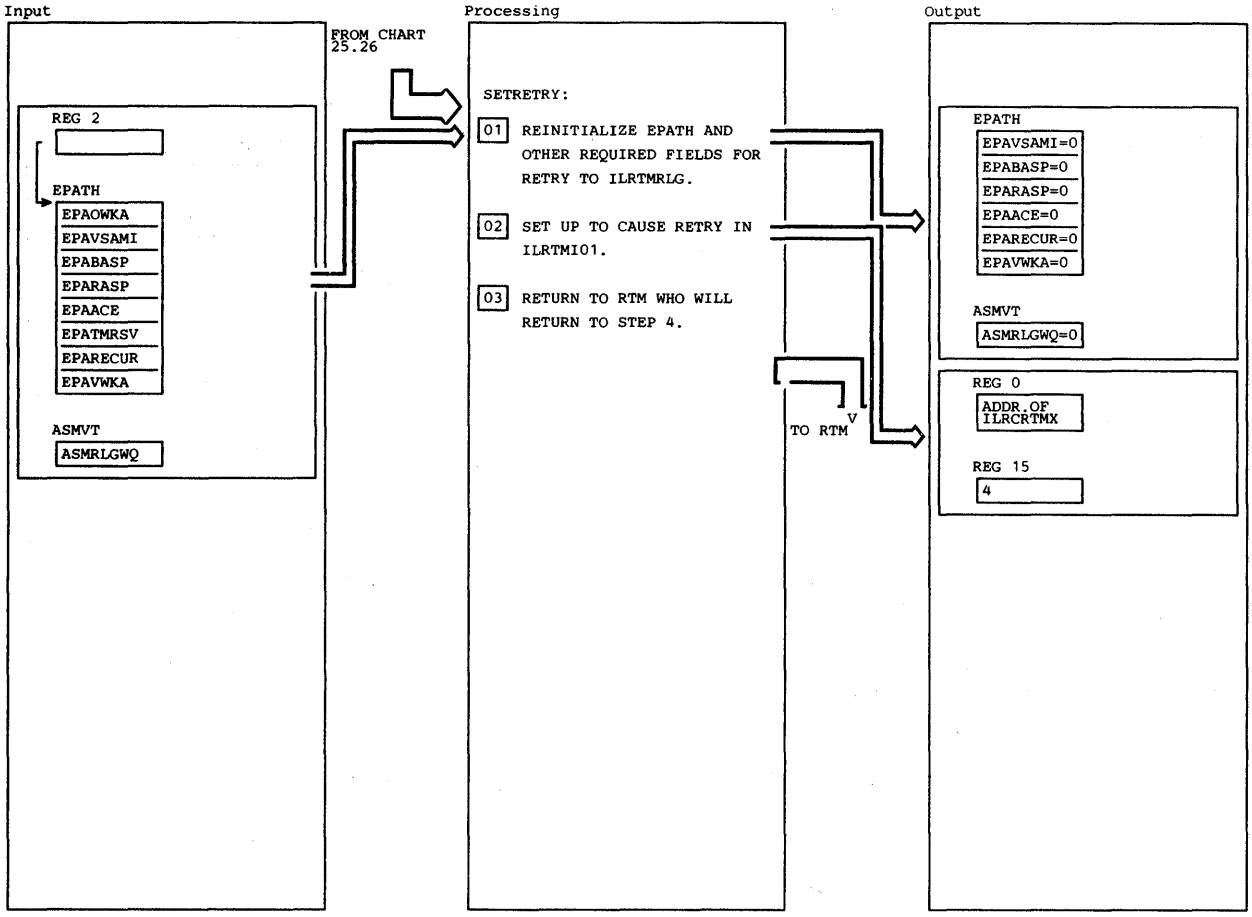
Diagram 25.26 ILRTMIO1 (Part 1 of 2)



Notes	Routine	Label	Ref
03 PLACE ERROR RECORDING INFORMATION INTO THE SDWA, INCLUDING A COPY OF THE EPATH TO THE VARIABLE RECORDING AREA, THEN GO PREPARE TO RETRY INTO ILRTMI00 OR ILRTMRLG.		CKRETRY	25.26.2

Notes	Routine	Label	Ref

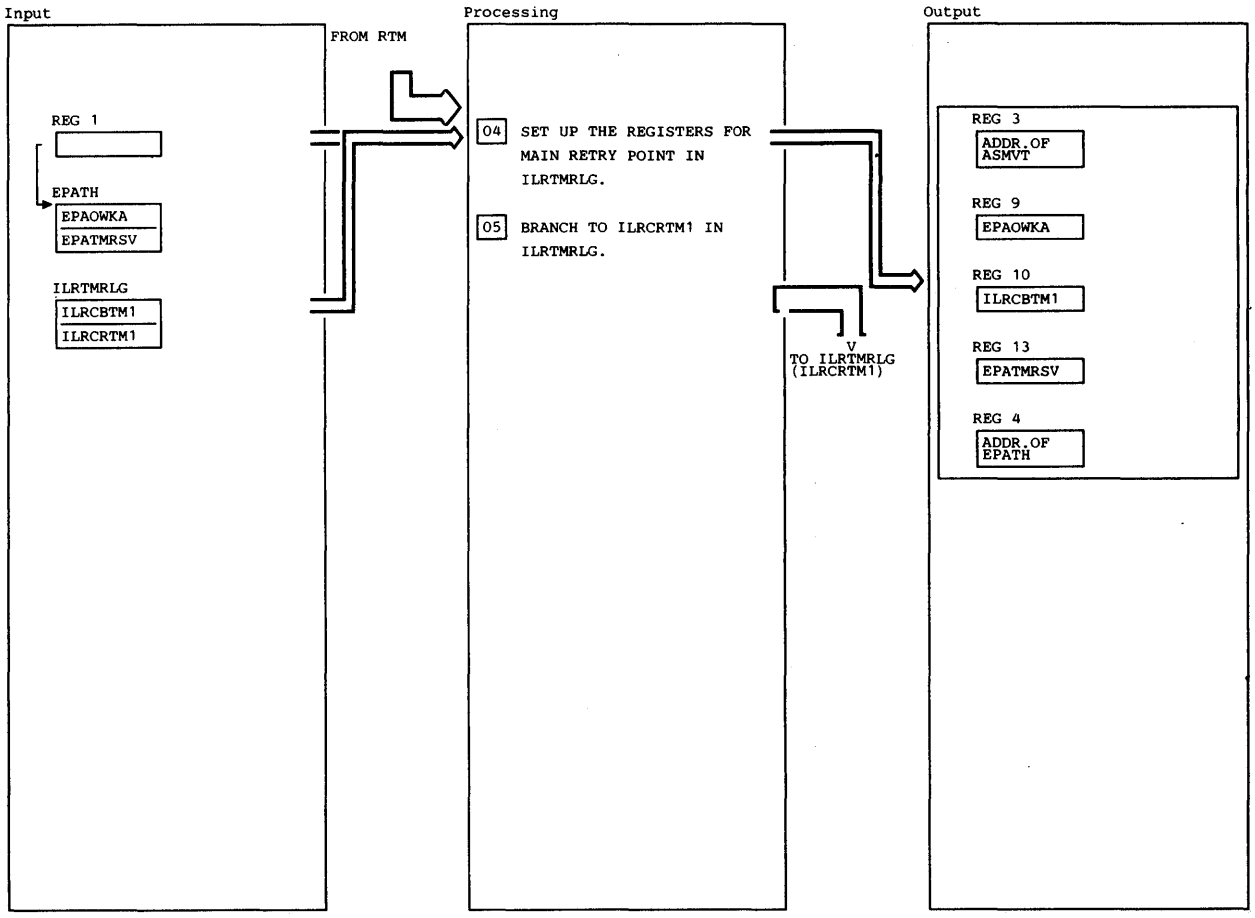
Diagram 25.26 ILRTMIO1 (Part 2 of 2)



Notes	Routine	Label	Ref
01 WITHOUT THE SDWA NO ERROR RECORDING OR VALIDITY CHECKING IS POSSIBLE. SET EPARECUR=1 SO THAT RECURSION CAN BE DETECTED. THE FOLLOWING EPATH FIELDS ARE SET TO ZERO UNCONDITIONALLY: EPAVSAMI, EPABASP, EPARASP, EPAACE, ASMRLGWQ, EPAVWKA.			
02 THE RETRY ADDRESS ILRCRTMX IN ILRTMI01 IS PUT IN REGISTER ZERO AND A 4 IS PUT IN REGISTER 15 TO CAUSE RTM TO RETRY.			
03 CONTROL IS RETURNED TO RTM WHO WILL CONTINUE AT STEP 4.			

Notes	Routine	Label	Ref

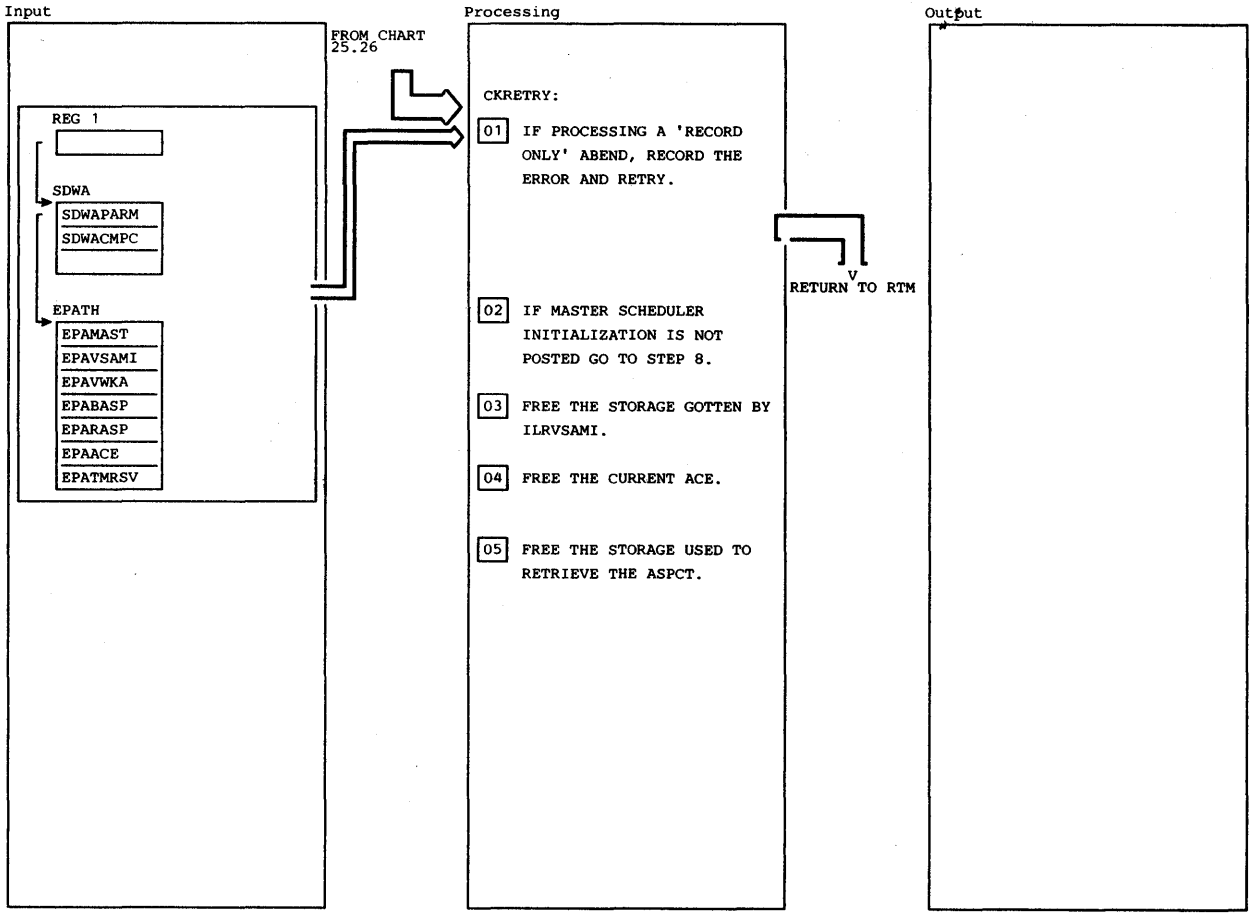
Diagram 25.26.1 SETRETRY (Part 1 of 2)



Notes	Routine	Label	Ref
04 AT THIS POINT, IT APPEARS (TO THE SYSTEM) THAT ILRTMI01 IS NO LONGER RUNNING AS AN ESTAE BUT AS A MAINLINE RETRY ROUTINE. THE FOLLOWING REGISTERS ARE LOADED WITH VALUES REQUIRED TO BRANCH TO THE MAIN RETRY POINT IN ILRTMRLG: REGISTER 3, REGISTER 4, REGISTER 9, REGISTER 10, REGISTER 13.			
05 BRANCH TO ILRTMRLG TO CONTINUE PROCESSING ACES OR WAIT IF NO MORE ACES ARE ON THE QUEUES (ASMRLGWQ AND ASMRLGRQ).	ILRTMRLG	ILRCRTM1	

Notes	Routine	Label	Ref

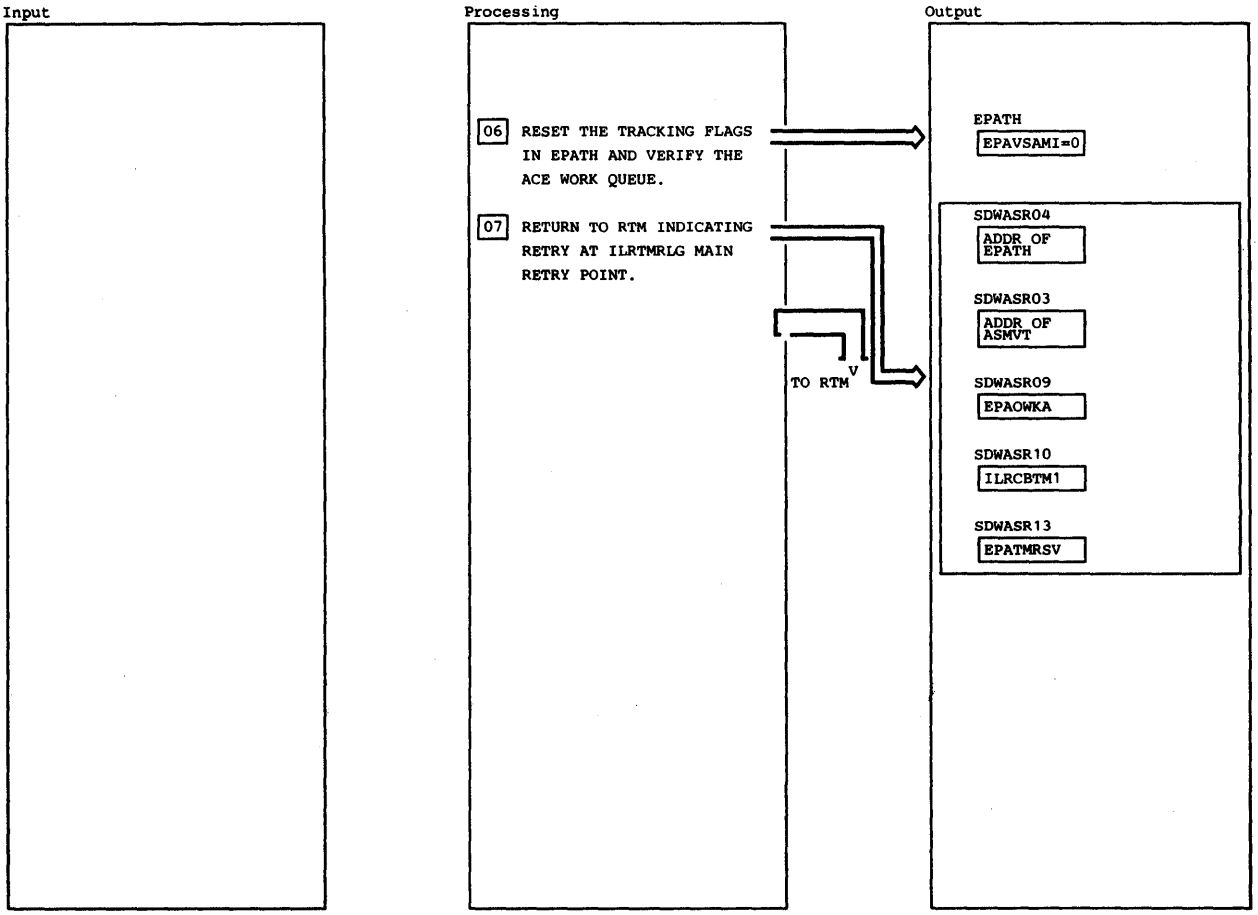
Diagram 25.26.1 SETRETRY (Part 2 of 2)



Notes	Routine	Label	Ref
01 SET UP FOR RETRY IF ONE OF THE FOLLOWING CODES IS IN SDWACMPC: X'086000', X'087000'. RETURN TO RTM.			
02 IF MASTER SCHEDULER INITIALIZATION IS NOT POSTED, ILRTMRLG MAIN LINE ACE PROCESSOR WAS NOT IN CONTROL, SO GO TO STEP 8.			
03 IF THE ERROR OCCURRED DURING ILRTMRLG'S CALL TO VSAM (SDWAPERC=1), ISSUE ENDREQ FOR VSAM CLEAN UP. IF EPAVWKA IS NON-ZERO, ISSUE ILRGMA TO FREE THE VSAMI WORKAREA. ISSUE FREEMAIN FOR THE ASPCT BUFFER.	ENDREG ILRGMA FREEMAIN		
04 VALIDITY CHECK THE ACE ADDRESSED BY EPAACE AND ISSUE ILRGMA TO FREE IT.	ILRFRR01 ILRGMA	ILRVACE	25.27.10
05 CALL ILRAFS00 TO FREE ASPCT STORAGE IF EPARASP IS NON-ZERO.			

Notes	Routine	Label	Ref

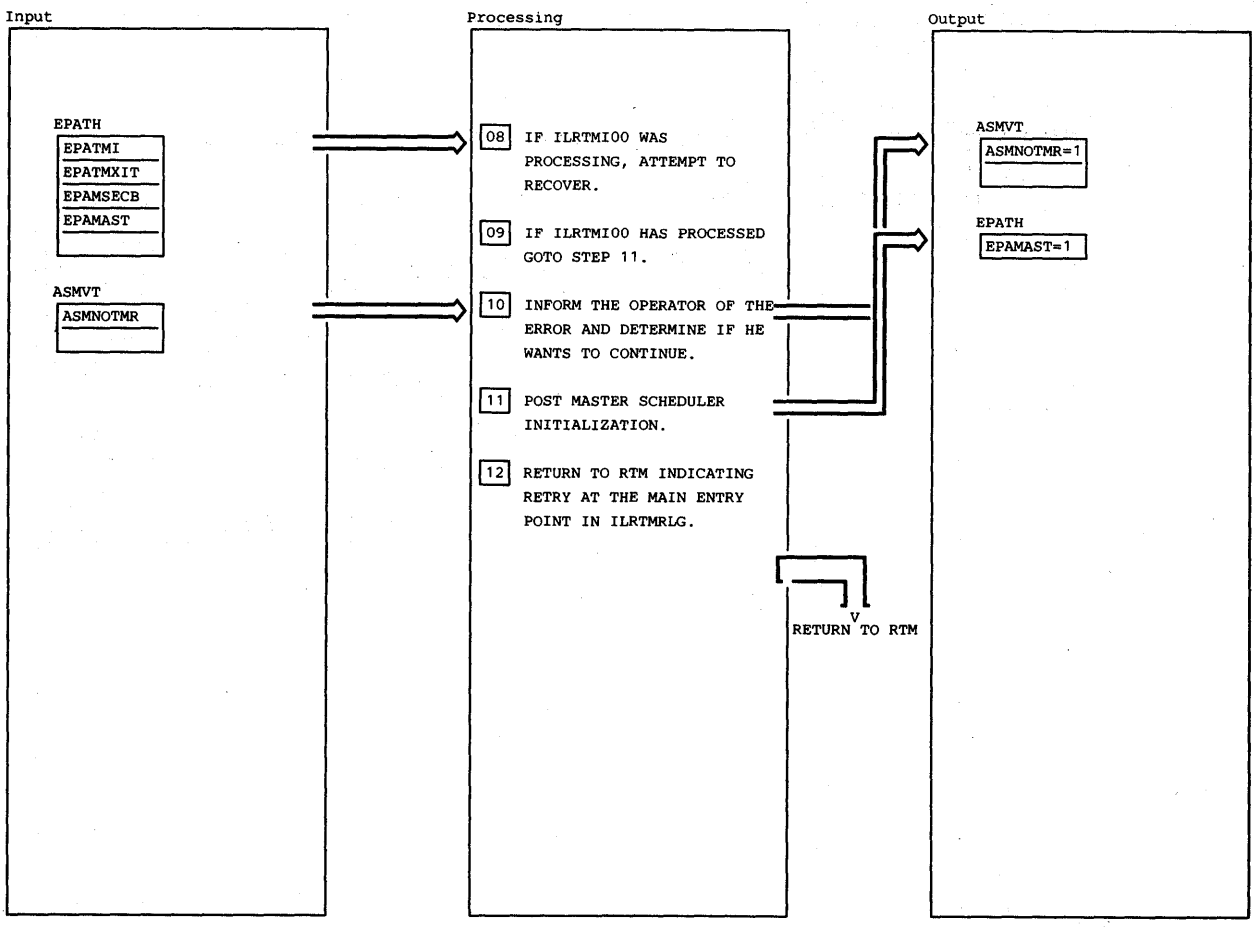
Diagram 25.26.2 CKRETRY (Part 1 of 3)



Notes	Routine	Label	Ref
06 VERIFY THE ACE WORK QUEUE (ASMRLGWQ): SET EPAVSAMI=0 AND SET EPARECUR=1 TO STOP RECURSION DUE TO ERRORS IN ILRTMRLG.	ILRFRR01	ILRVACEQ	25.27.16
07 UPDATE THE FOLLOWING REGISTER VALUES IN THE SDWA TO CONTAIN THE VALUES REQUIRED AT THE MAIN RETRY POINT (ILRCRTM1) IN ILRTMRLG: REG 3, REG 4, REG 9, REG 10, REG 13.			

Notes	Routine	Label	Ref

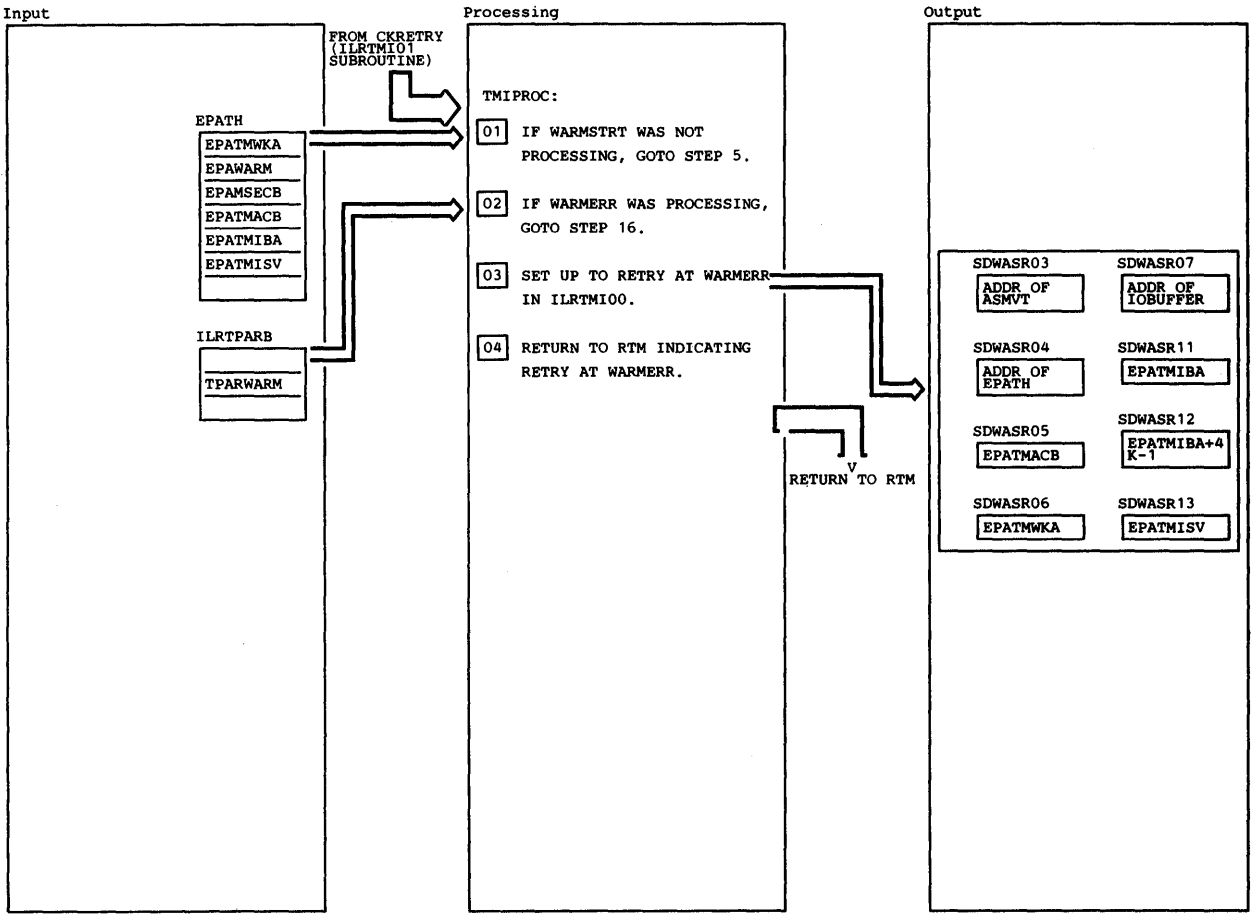
Diagram 25.26.2 CKRETRY (Part 2 of 3)



Notes	Routine	Label	Ref
08 IF EPATMI IS 1, GOTO TMIPROC.		TMIPROC	25.26.3
09 IF EPATMXIT FLAG HAS BEEN TURNED ON BY TMIPROC, GO TO STEP 11.			
10 THE FAILURE HAPPENED SOMETIME BEFORE ILRTMRLG CALLED ILRTMI00 SO ISSUE MESSAGES ILR021I AND ILR022A TO INFORM THE OPERATOR OF AN ERROR AND TO DETERMINE IF HE WANTS TO CONTINUE WITHOUT VIO JOURNALING. IF HE DOES, ASMNOTMR IS SET TO 1 TO INDICATE ILRTMRLG IS NOT AVAILABLE. SET TO ZERO SARDSNL,PARTDSNL, AND PARTTPAR.			
11 EPAMSECB IS USED TO POST MASTER SCHEDULER INITIALIZATION - SETPAMAST=1.			
12 RETRY AT THE MAIN RETRY POINT IN ILRTMRLG (ILRCRTM1), WHICH WILL PUT THE ILRTMRLG TASK IN A WAIT.			

Notes	Routine	Label	Ref

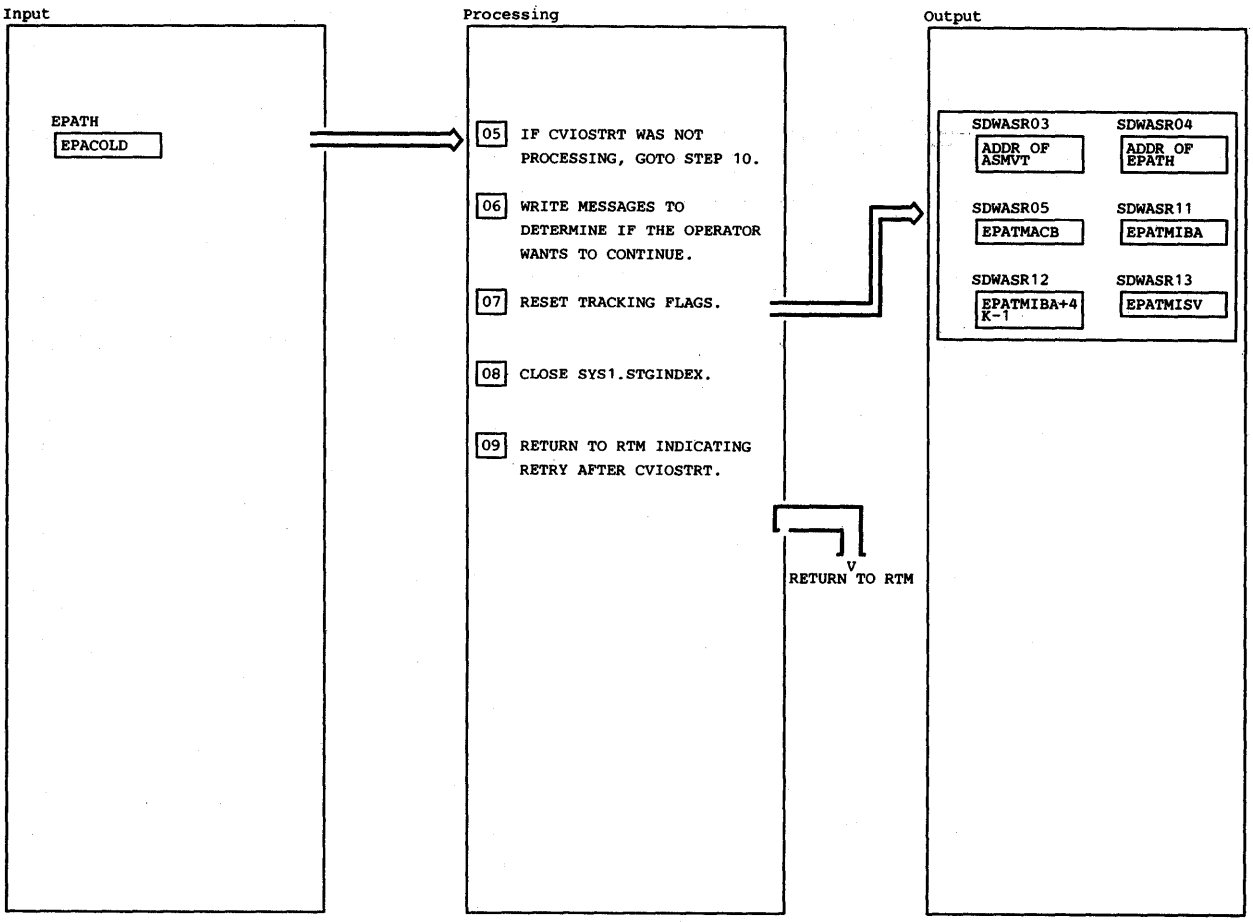
Diagram 25.26.2 CKRETRY (Part 3 of 3)



Notes	Routine	Label	Ref
01 IF EPAWARM=0, THE WARMSTRT (WARM START) SECTION OF ILRTMI00 WAS NOT EXECUTING. GO TO STEP 5 TO DETERMINE WHERE THE ERROR OCCURRED.			
02 A WARM START WAS PROCESSING. IF TPARWARM=0, THEN WARMERR (WARM START RETRY CODE) WAS PROCESSING. SO THERE IS A DOUBLE OR RECURSIVE ERROR. GO TO STEP 16 TO ISSUE MESSAGES.			
03 SINCE THIS IS A SINGLE WARM START ERROR, PREPARE TO RETRY. THE FOLLOWING REGISTERS ARE REQUIRED BY WARMERR: 3,4,5,6,7,11,12 AND 13.			
04 RETURN TO RTM TO RETRY AT WARMERR (ILRCRTM2) IN ILRTMI00.			

Notes	Routine	Label	Ref

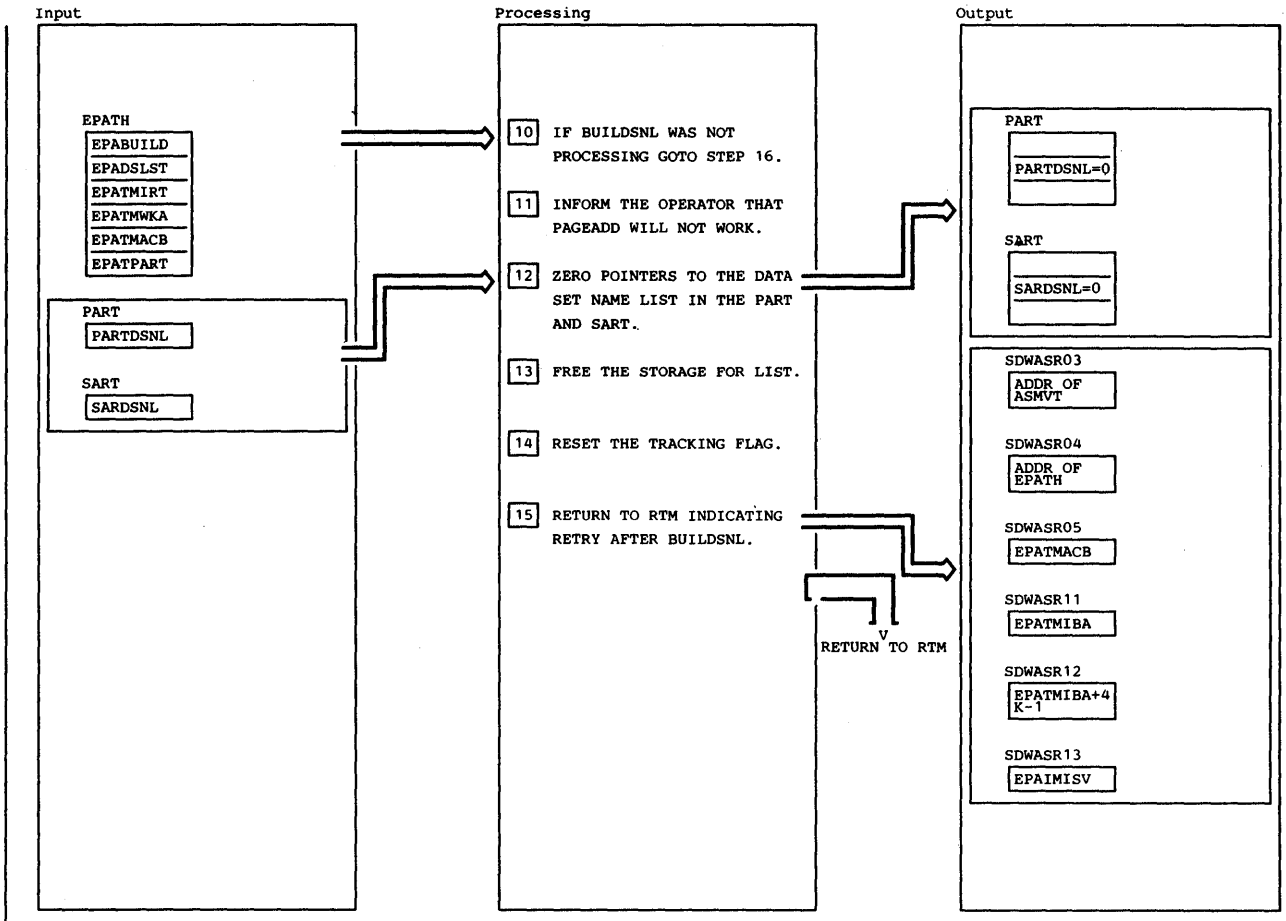
Diagram 25.26.3 TMIPROC (Part 1 of 4)



Notes	Routine	Label	Ref
05 IF EPACOLD=0 GOTO STEP 10.			
06 WRITE MESSAGES ILR001I AND ILR022A TO INDICATE AN ERROR OCCURRED AND SEE IF THE OPERATOR WANTS TO CONTINUE.			
07 CONTINUING, SET EPACOLD=0 (CVIOSTRT NO LONGER PROCESSING) AND ASMNOTMR=1 (ILRTMRLG WILL NOT BE USED TO RELEASE SAVED LG THIS IPL).			
08 IF ASMSTGXA IS NOT ZERO ISSUE CLOSE FOR SYS1.STGINDEX AND SET ASMSTGXA=0.			
09 SET UP SDWA WITH VALUES FOR REGISTERS REQUIRED AFTER CVIOSTRT (ILRCRTM3) AND RETURN TO RTM TO RETRY.			

Notes	Routine	Label	Ref

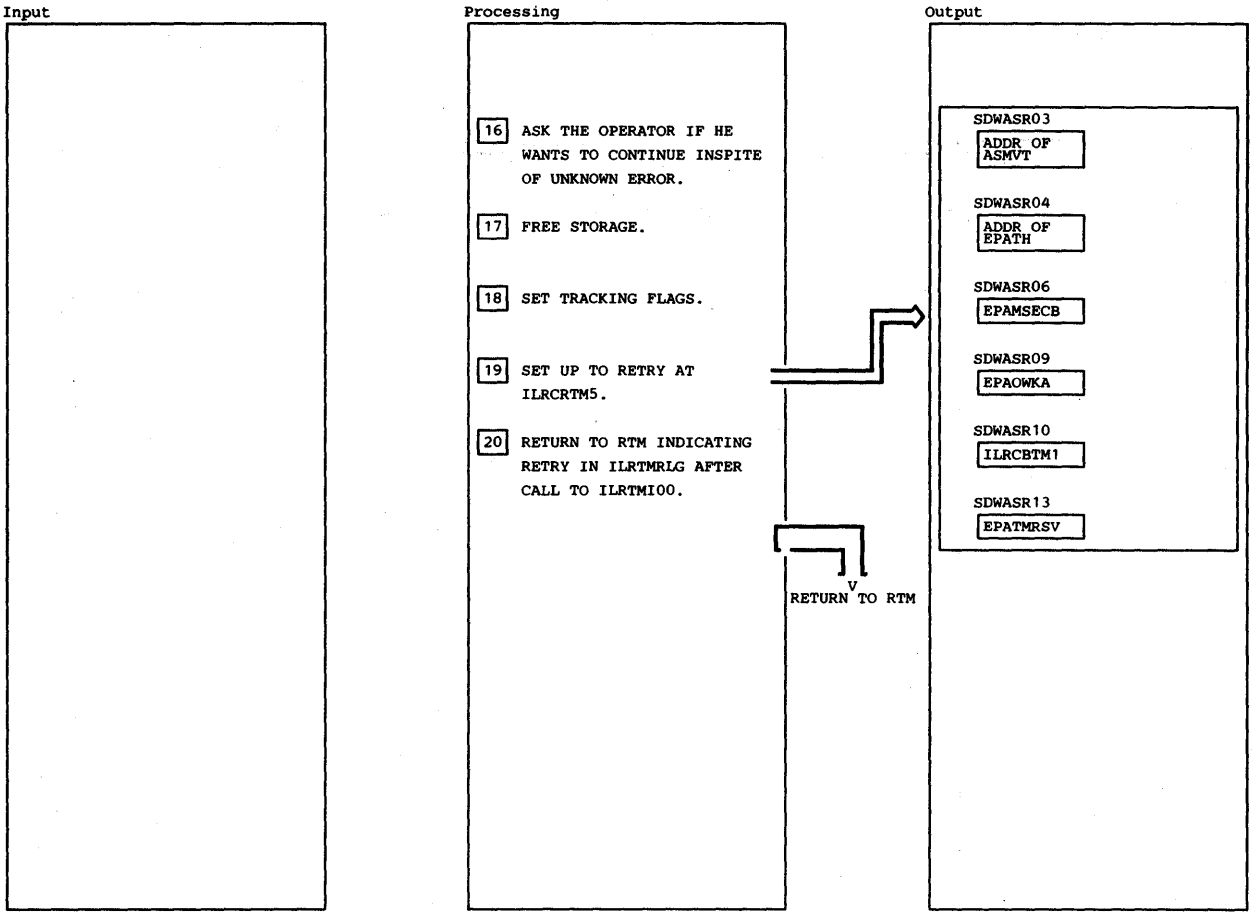
Diagram 25.26.3 TMIPROC (Part 2 of 4)



Notes	Routine	Label	Ref
10 IF EPABUILD=0, GOTO STEP 16.			
11 WRITE MESSAGE ILR003I TO INFORM THE OPERATOR THAT PAGEADD WILL NOT WORK.			
12 SET THE DATA SET NAME LISTS (PARTDSNL AND SARDSNL) TO ZERO.			
13 ISSUE FREEMAIN FOR LIST ADDRESSED BY EPADSLST.	FREEMAIN		
14 SET EPABUILD=0.			
15 SET UP SDWA WITH REQUIRED REGISTER VALUES AND RETURN TO RTM TO RETRY AFTER BUILDSNL (ILRCRTM4 IN ILRTMI00).			

Notes	Routine	Label	Ref

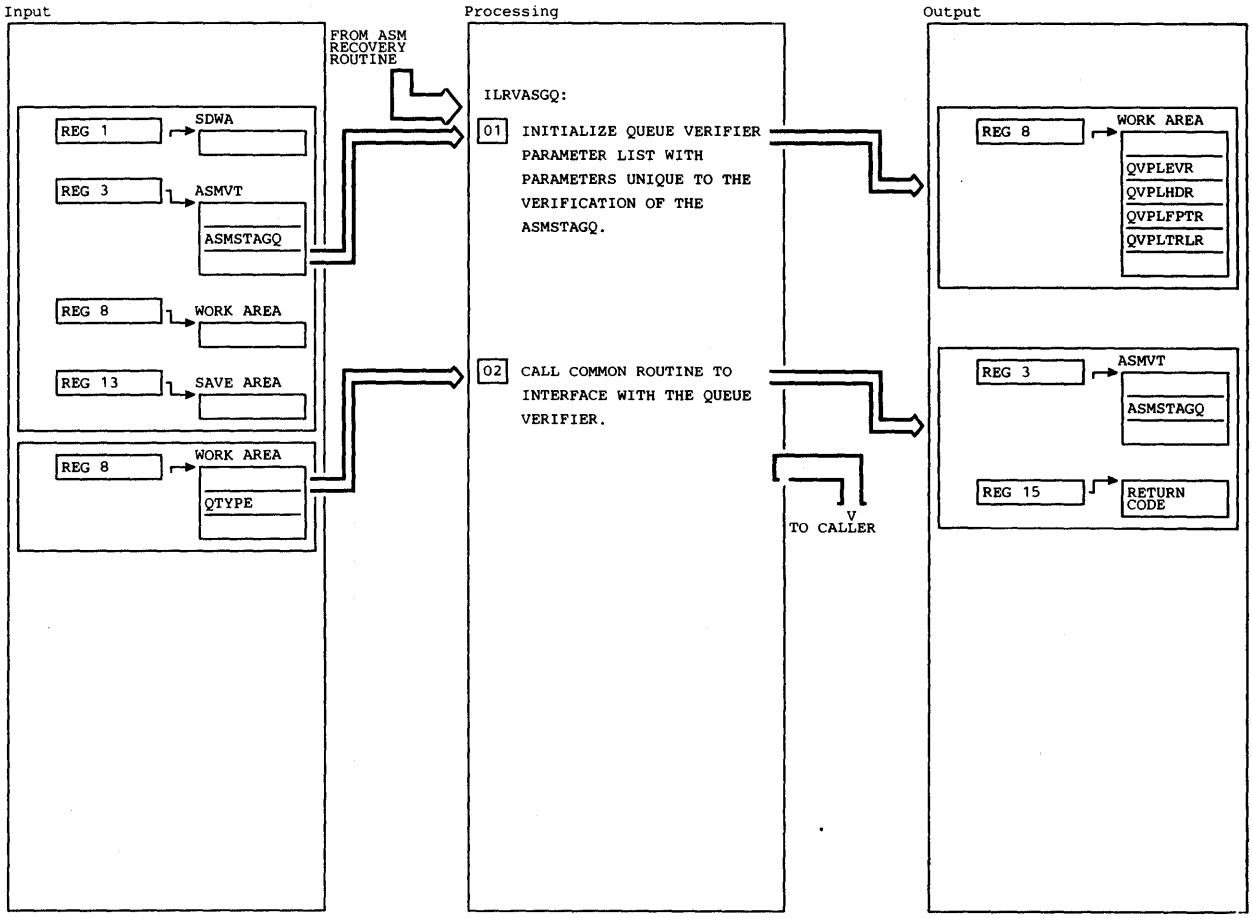
Diagram 25.26.3 TMIPROC (Part 3 of 4)



Notes	Routine	Label	Ref
16 AT THIS POINT THE PLACE OF FAILURE IS UNKNOWN UNLESS WARMERR FAILED. ISSUE MESSAGES ILR021I AND ILR022A TO SEE IF THE OPERATOR WANTS TO CONTINUE.			
17 CONTINUING, SET ASMNOTMR=1 (INDICATING ILRTMRLG WILL NOT BE USED TO RELEASE LG ON SAVED LOGICAL GROUPS). IF ASMSTGXA IS ZERO, FREE THE STORAGE USED FOR THE ACB(EPATMACB). FREE THE WORK AREA FOR ILRTMI00 (EPATMWKA). FREE THE STORAGE FOR TPARTBLE (EPATPART). ZERO PARTTPAR.	FREEMAIN		
18 SET EPATMXIT=1.			
19 ILRTMRLG AT ILRCRTM5 REQUIRES REGISTERS 3, 4, 6, 9, 10 AND 13.			
20 RETURN TO RTM TO RETRY AT ILRCRTM5.			

Notes	Routine	Label	Ref

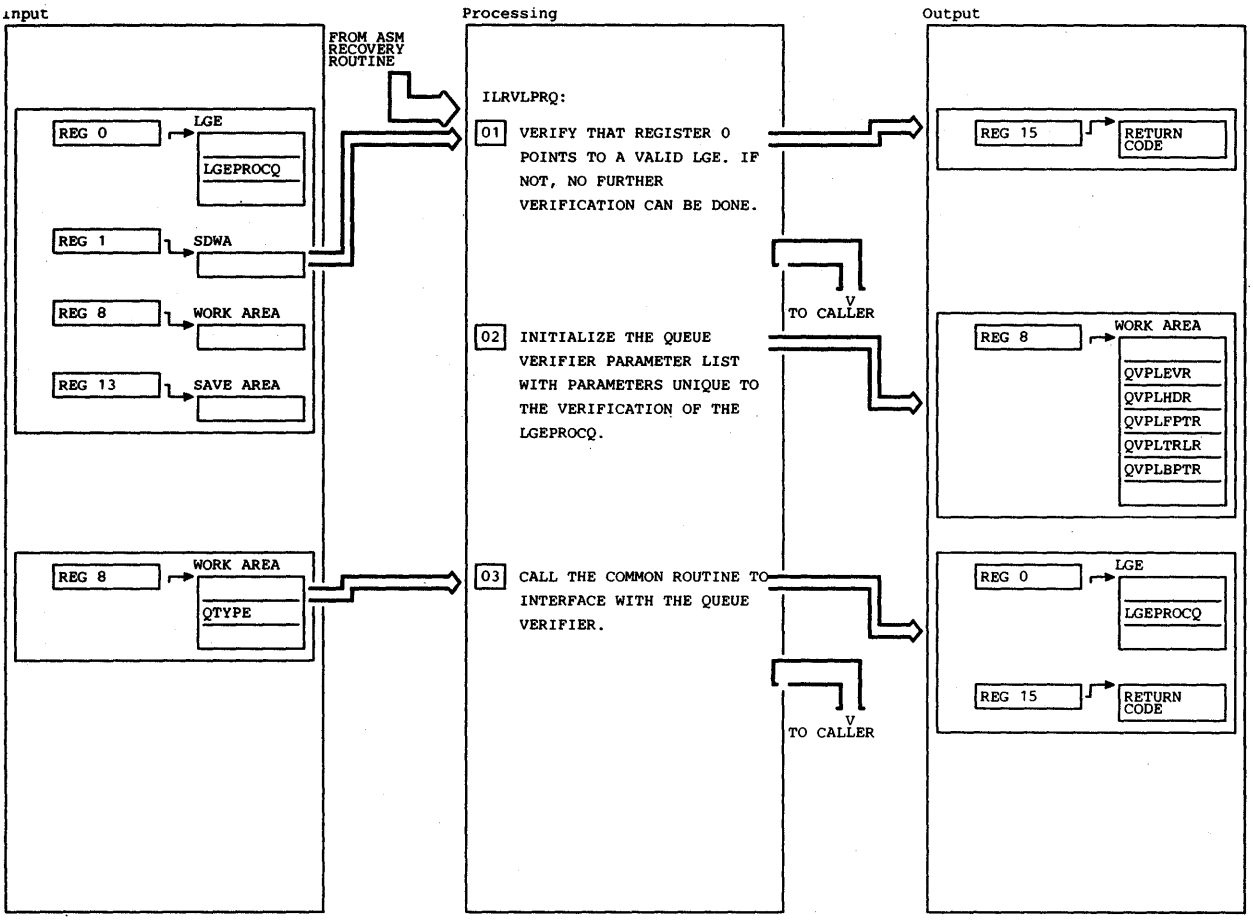
Diagram 25.26.3 TMIPROC (Part 4 of 4)



Notes	Routine	Label	Ref
01 INITIALIZE THE PARAMETERS OF THE QUEUE VERIFIER PARAMETER LIST THAT ARE UNIQUE FOR THE VERIFICATION OF THE ASMSTAGQ. THESE PARAMETERS ARE THE ADDRESS OF THE QUEUE HEADER (ASMSTAGF), THE ADDRESS OF THE QUEUE TRAILER (ASMSTAGL), THE ADDRESS OF THE ELEMENT VERIFICATION ROUTINE (ILRVAIA), AND THE OFFSET OF THE FORWARD CHAIN POINTER (AIANXAIA).			
02 CALL COMMON ROUTINE TO SET UP THE INTERFACE FOR THE QUEUE VERIFIER. AN INTERNAL VARIABLE, QTYPE, IS SET TO INDICATE THE QUEUE IS A SINGLE-THREADED, DOUBLE-HEADED QUEUE (QTYPE=2).	ILRFRR01	COMQRTN	25.27.20

Notes	Routine	Label	Ref

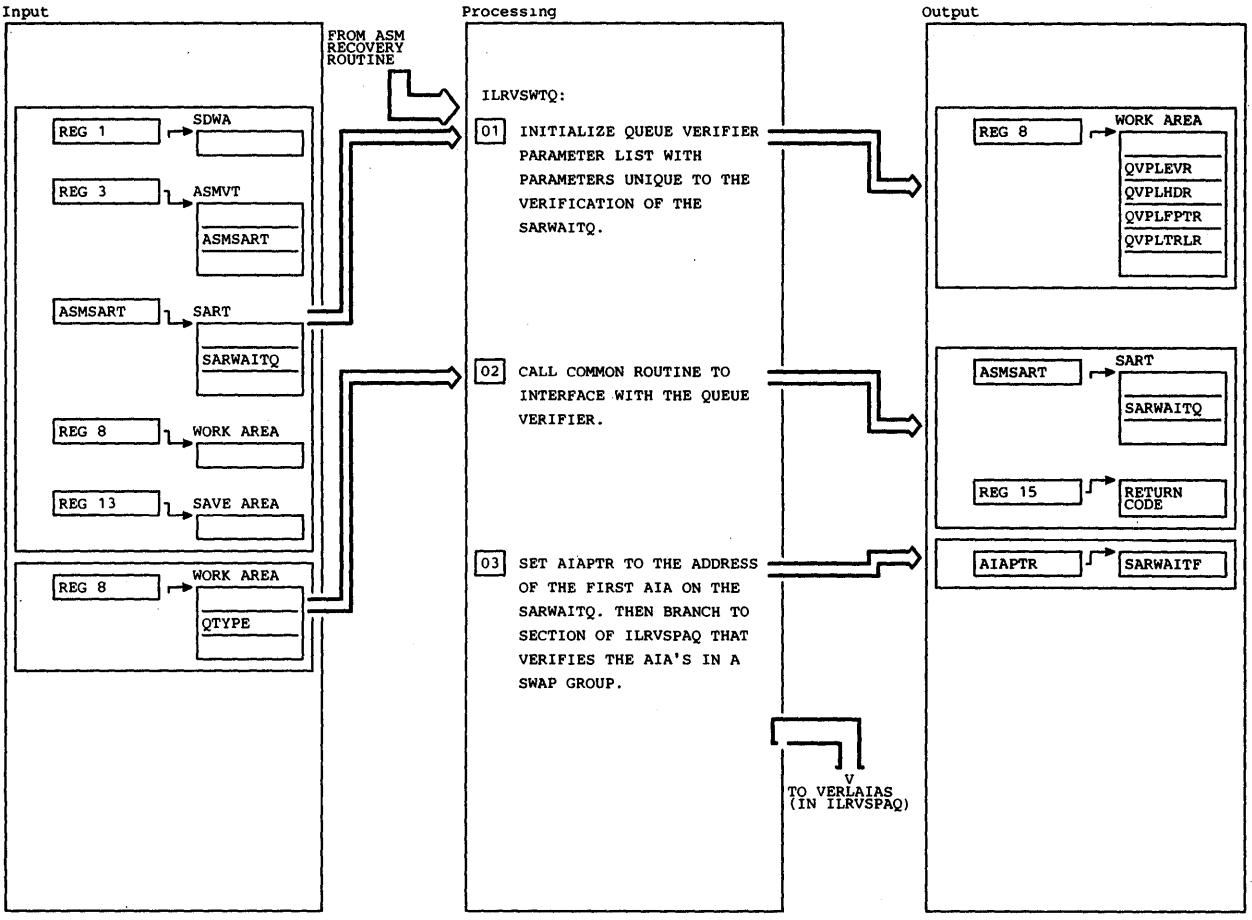
Diagram 25.27.1 ILRVASGQ (Part 1 of 1)



Notes	Routine	Label	Ref
01 VERIFY THAT REGISTER 0 POINTS TO A VALID LGE. IF IT DOES NOT, RETURN TO THE CALLER SINCE NO FURTHER VERIFICATION CAN BE DONE.	ILRFRR01	ILRVLGE	25.27.11
02 INITIALIZE THE PARAMETERS OF THE QUEUE VERIFIER PARAMETER LIST THAT ARE UNIQUE FOR THE VERIFICATION OF THE LGEPROCQ. THESE PARAMETERS ARE THE ADDRESS OF THE QUEUE HEADER (LGEPROCF), THE ADDRESS OF THE QUEUE TRAILER (LGEPROCL), THE ADDRESS OF THE ELEMENT VERIFICATION ROUTINE (ILRVAIAC), THE OFFSET OF THE FORWARD CHAIN POINTER (AIAFQPA), AND THE OFFSET OF THE BACKWARD CHAIN POINTER (AIABQPA).			
03 CALL THE COMMON ROUTINE TO SET UP THE INTERFACE FOR THE QUEUE VERIFIER. SET QTYPE=3 TO INDICATE THE QUEUE IS A DOUBLE-HEADED,DOUBLE-THREADED QUEUE.	ILRFRR01	COMQRTN	25.27.20

Notes	Routine	Label	Ref

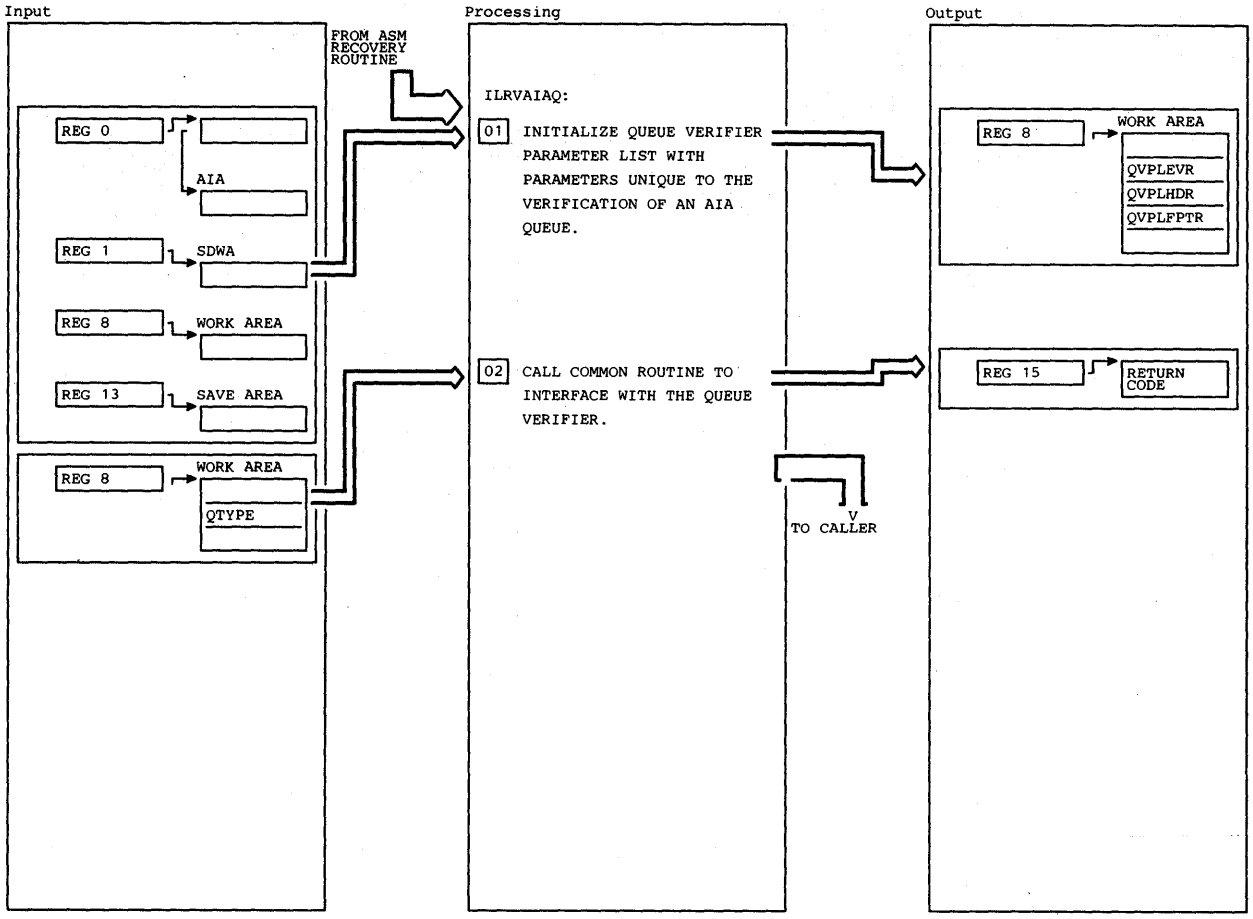
Diagram 25.27.2 ILRVLPRQ (Part 1 of 1)



Notes	Routine	Label	Ref
01 INITIALIZE THE PARAMETERS OF THE QUEUE VERIFIER PARAMETER LIST THAT ARE UNIQUE FOR VERIFICATION OF THE SARWAITQ. THESE PARAMETERS ARE THE ADDRESS OF THE QUEUE HEADER (SARWAITF), THE ADDRESS OF THE QUEUE TRAILER (SARWAITL), THE ADDRESS OF THE ELEMENT VERIFICATION ROUTINE (ILRVAIA), AND THE OFFSET OF THE FORWARD CHAIN POINTER (AIANXAIA).			
02 CALL COMMON ROUTINE TO SET UP THE INTERFACE FOR THE QUEUE VERIFIER. SET QTYPE=2 TO INDICATE THE QUEUE IS A SINGLE-THREADED, DOUBLE-HEADED QUEUE.	ILRFRR01	COMQRTN	25.27.20
03 THE VARIABLE AIAPTR IS INITIALIZED TO THE ADDRESS OF THE FIRST AIA ON THE SARWAITQ. THIS IS DONE TO SET UP FOR THE VERIFICATION OF THE LATERAL AIA'S OF EACH AIA ON THE SARWAITQ. THIS VERIFICATION IS ACTUALLY DONE IN ILRVSPAQ.	ILRFRR01	ILRVSPAQ	25.27.6

Notes	Routine	Label	Ref

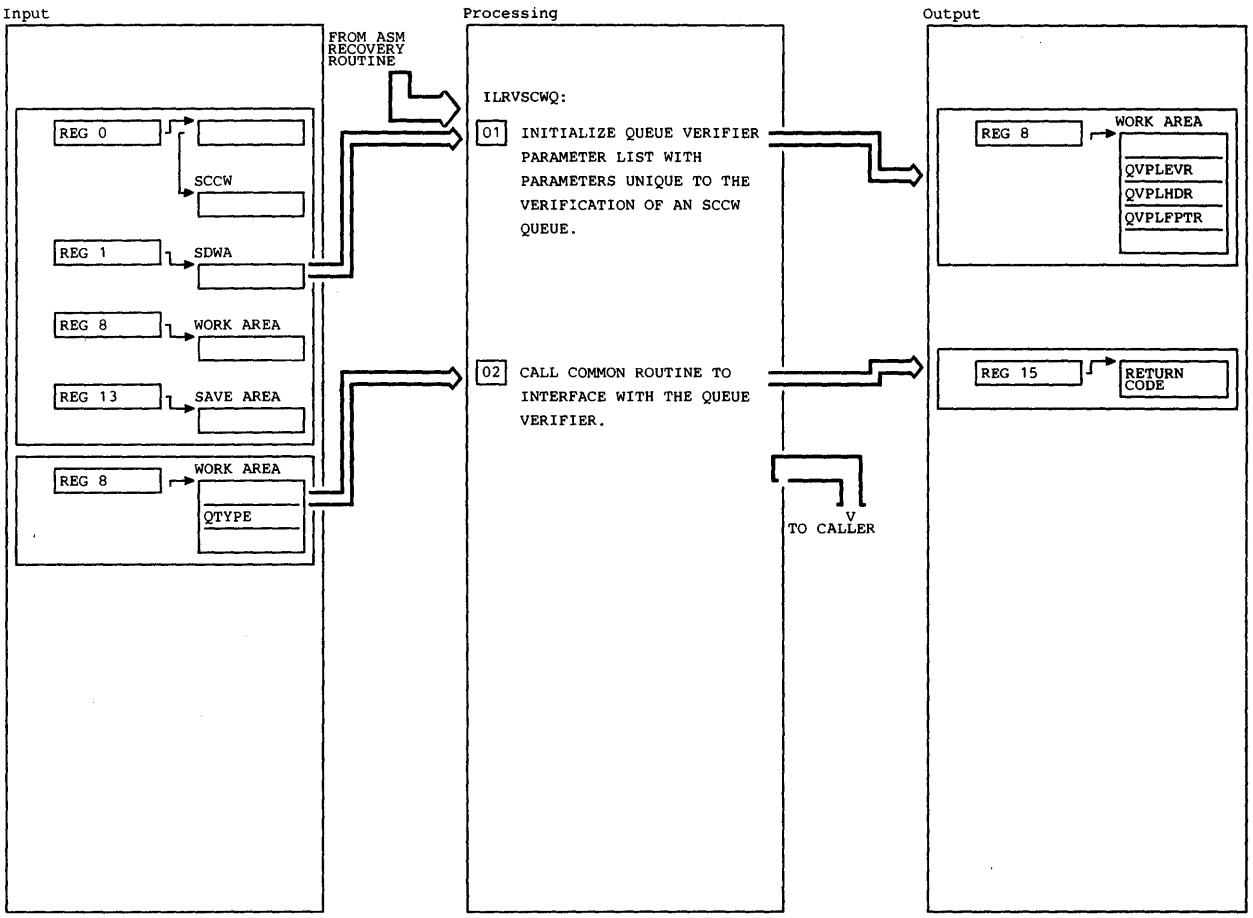
Diagram 25.27.3 ILRVSWTQ (Part 1 of 1)



Notes	Routine	Label	Ref
01 INITIALIZE THE PARAMETERS OF THE QUEUE VERIFIER PARAMETER LIST THAT ARE UNIQUE FOR VERIFICATION OF AN AIA QUEUE. THESE PARAMETERS ARE THE ADDRESS OF THE QUEUE HEADER (VALUE OF REGISTER 0), THE ADDRESS OF THE ELEMENT VERIFICATION ROUTINE (ILRVAIA), AND THE OFFSET OF THE FORWARD CHAIN POINTER (AIANXAIA).			
02 CALL COMMON ROUTINE TO SET UP THE INTERFACE FOR THE QUEUE VERIFIER. SET QTYPE=1 TO INDICATE THE QUEUE IS A SINGLE-THREADED, SINGLE HEADED QUEUE.	ILRFRR01	COMQRTN	25.27.20

Notes	Routine	Label	Ref

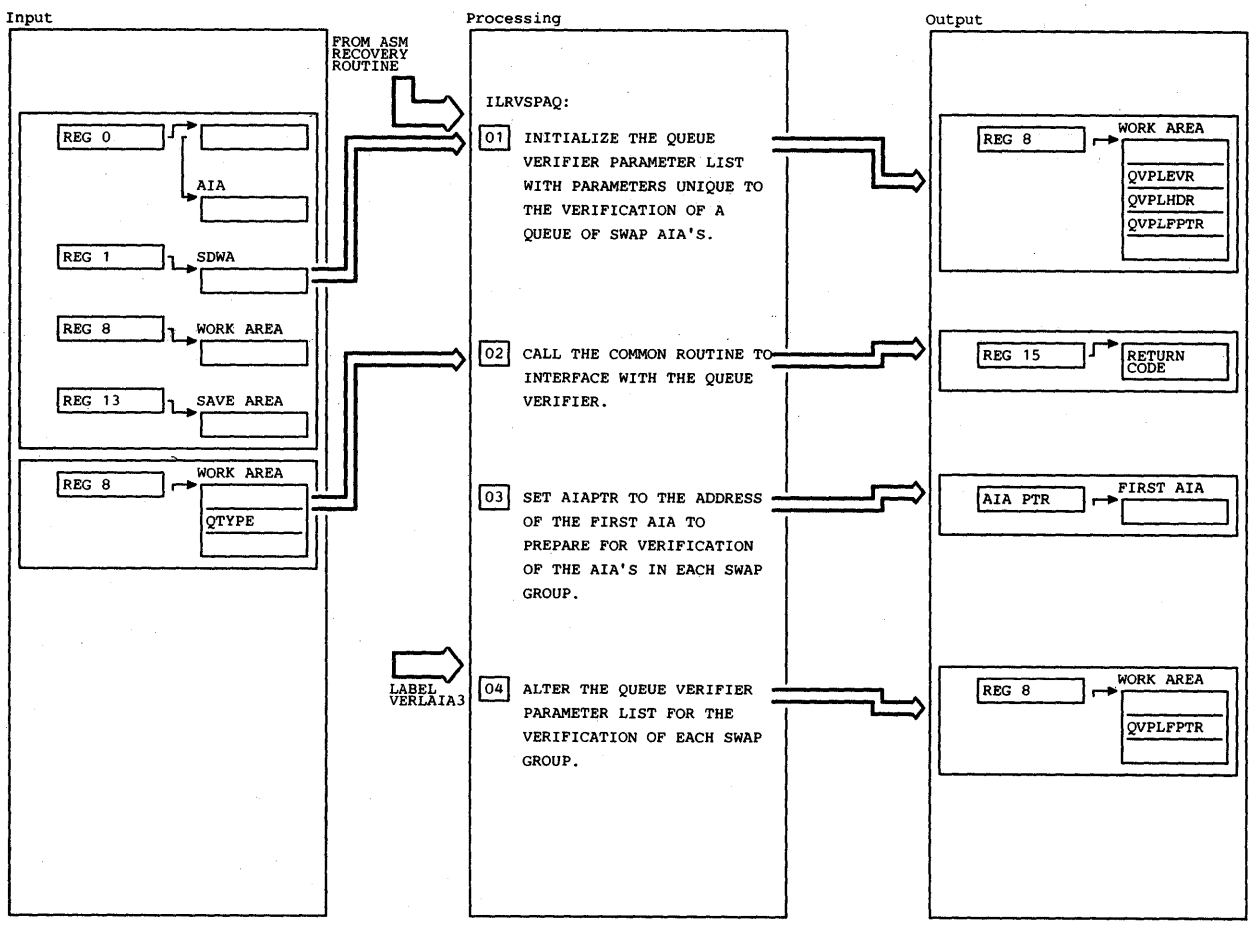
Diagram 25.27.4 ILRVAIAQ (Part 1 of 1)



Notes	Routine	Label	Ref
01 INITIALIZE THE PARAMETERS OF THE QUEUE VERIFIER PARAMETER LIST THAT ARE UNIQUE FOR VERIFICATION OF AN SCCW QUEUE. THESE PARAMETERS ARE THE ADDRESS OF THE QUEUE HEADER (VALUE OF REGISTER 0), THE ADDRESS OF THE ELEMENT VERIFICATION ROUTINE (ILRVSCCW), AND THE OFFSET OF THE FORWARD CHAIN POINTER (SCCWSCCW).			
02 CALL COMMON ROUTINE TO SET UP THE INTERFACE FOR THE QUEUE VERIFIER. SET QTYPE=1 TO INDICATE THE QUEUE IS A SINGLE-THREADED, SINGLE-HEADED QUEUE.	ILRFRR01	COMQRTN	25.27.20

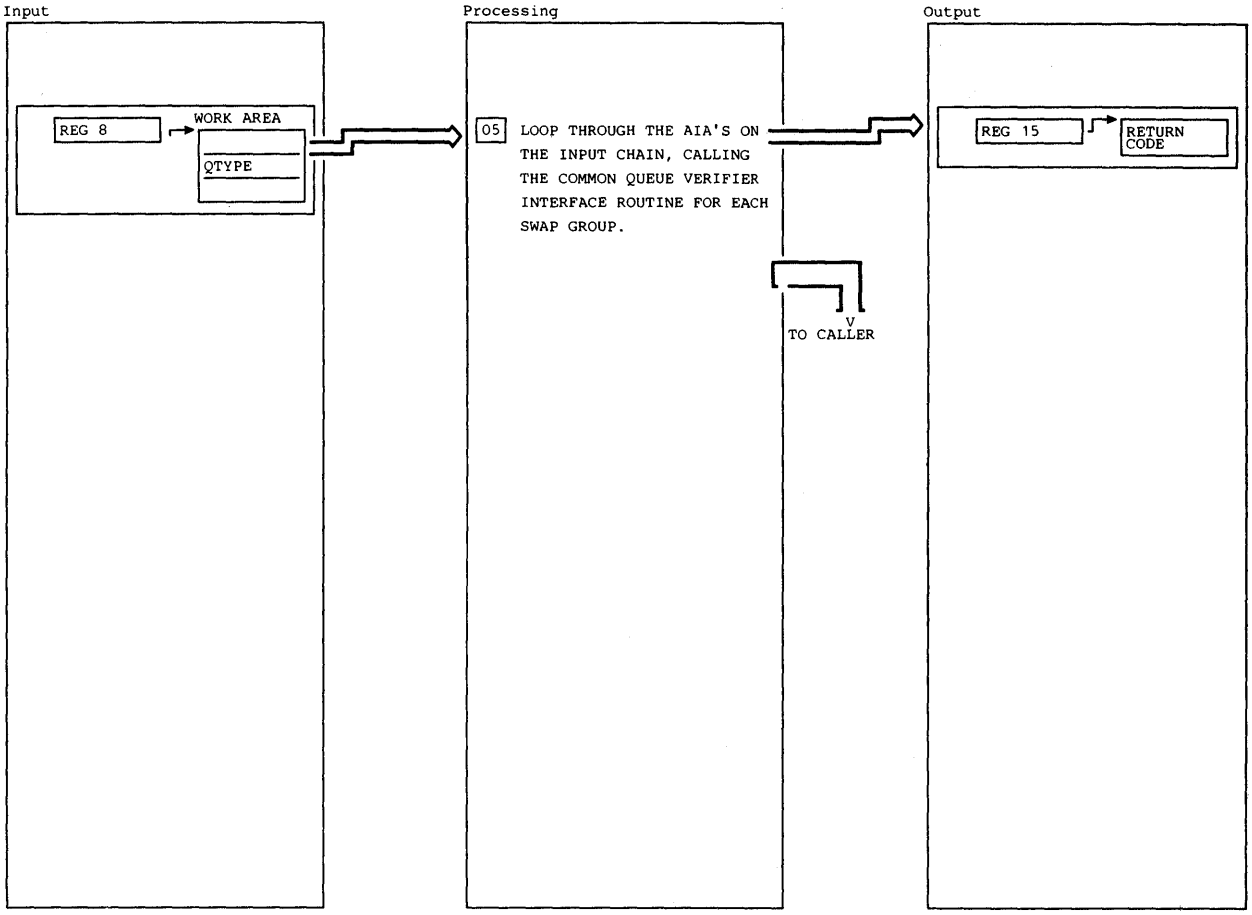
Notes	Routine	Label	Ref

Diagram 25.27.5 ILRVSCWQ (Part 1 of 1)



Notes	Routine	Label	Ref
01 INITIALIZE THE PARAMETER OF THE QUEUE VERIFIER PARAMETER LIST THAT ARE UNIQUE FOR THE VERIFICATION OF A QUEUE OF SWAP AIA'S. THESE PARAMETERS ARE THE ADDRESS OF THE QUEUE HEADER (VALUE OF REGISTER 0), THE ADDRESS OF THE ELEMENT VERIFICATION ROUTINE (ILRVAIA), AND THE OFFSET OF THE FORWARD CHAIN POINTER (AIANXAIA).			
02 CALL THE COMMON ROUTINE TO SET UP THE INTERFACE FOR THE QUEUE VERIFIER. SET QTYPE=1 TO INDICATE THE QUEUE IS A SINGLE-THREADED,SINGLE-HEADED QUEUE.	ILRFRR01	COMQRTN	25.27.20
03 THE VARIABLE AIAPTR IS INITIALIZED TO THE ADDRESS OF THE FIRST AIA ON THE INPUT CHAIN. THIS IS DONE TO SET UP FOR THE VERIFICATION OF THE LATERAL AIA'S OF EACH AIA ON THE INPUT CHAIN.			
04 CHANGE THE OFFSET OF THE FORWARD CHAIN POINTER (TO AIAFQPA) TO VERIFY THE LATERAL AIA'S.			

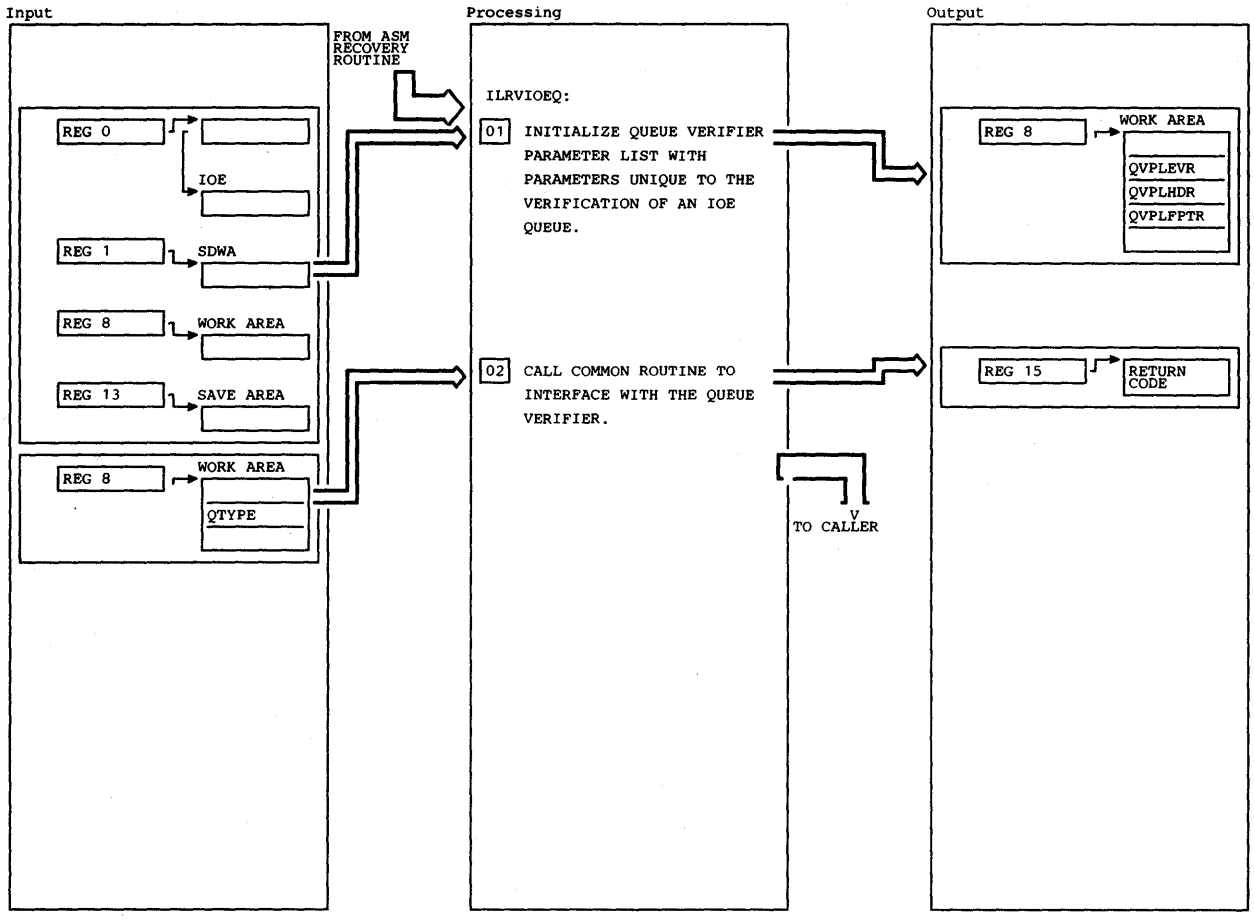
Diagram 25.27.6 ILRVSPAQ (Part 1 of 2)



Notes	Routine	Label	Ref
05 FOR EACH AIA ON THE INPUT CHAIN, CALL THE COMMON ROUTINE TO INTERFACE WITH THE QUEUE VERIFIER. QTYPE=1 TO INDICATE THE QUEUE IS A SINGLE-THREADED,SINGLE-HEADED QUEUE. AFTER ALL THE SWAP GROUPS HAVE BEEN VALIDITY CHECKED, RETURN TO THE CALLER. THE RETURN CODE IS SET TO THE LARGEST RETURN CODE PASSED BACK BY THE QUEUE VERIFIER.	ILRFRR01	COMQRTN	25.27.20

Notes	Routine	Label	Ref

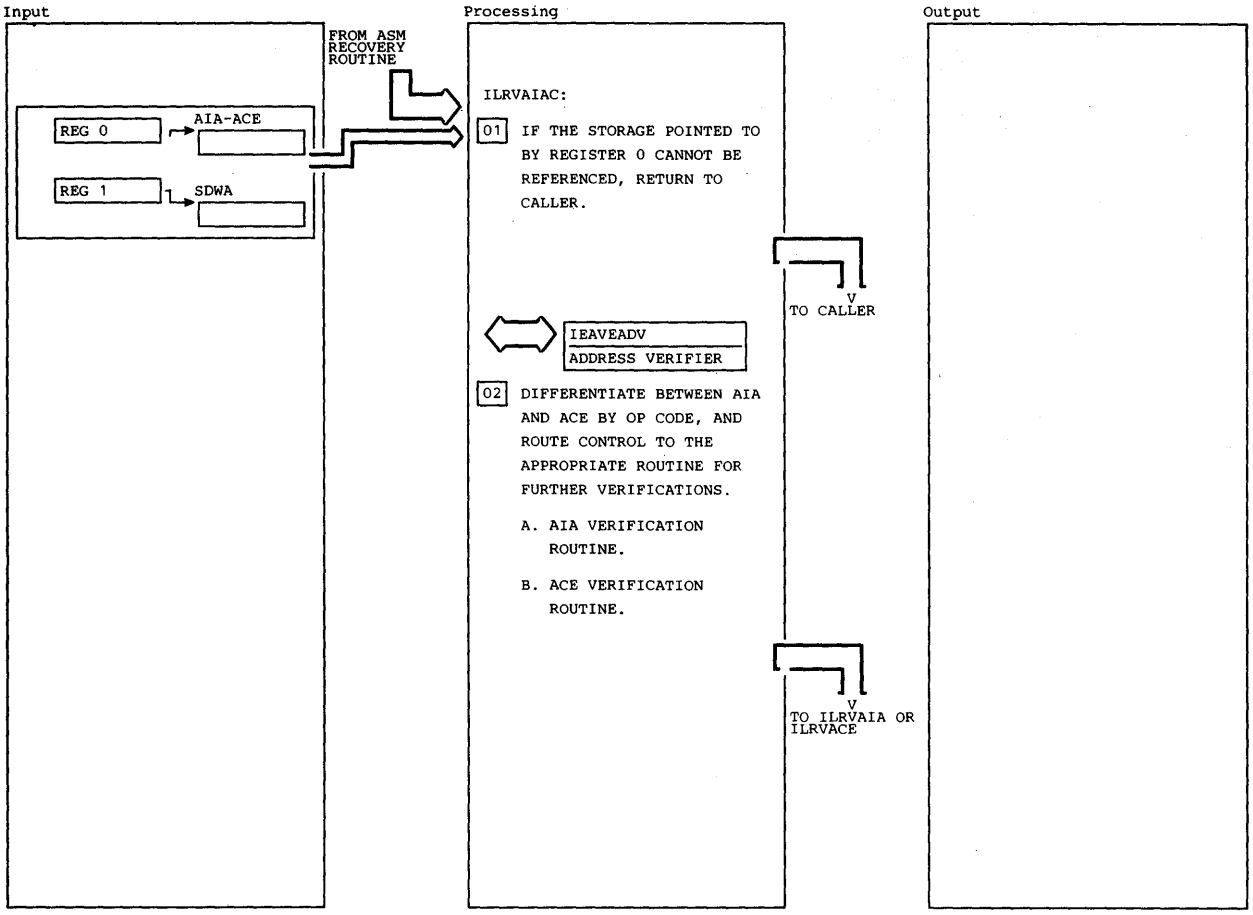
Diagram 25.27.6 ILRVSPAQ (Part 2 of 2)



Notes	Routine	Label	Ref
01 INITIALIZE THE PARAMETERS OF THE QUEUE VERIFIER PARAMETER LIST THAT ARE UNIQUE FOR VERIFICATION OF AN IOE QUEUE. THESE PARAMETERS ARE THE ADDRESS OF THE QUEUE HEADER (VALUE OF REGISTER 0), THE ADDRESS OF THE ELEMENT VERIFICATION ROUTINE (ILRVIOE), AND THE OFFSET OF THE FORWARD CHAIN POINTER (IOENEXT).			
02 CALL COMMON ROUTINE TO SET UP THE INTERFACE FOR THE QUEUE VERIFIER. SET QTYPE=1 TO INDICATE THE QUEUE IS A SINGLE-THREADED, SINGLE-HEADED QUEUE.	ILRFRR01	COMQRTN	25.27.20

Notes	Routine	Label	Ref

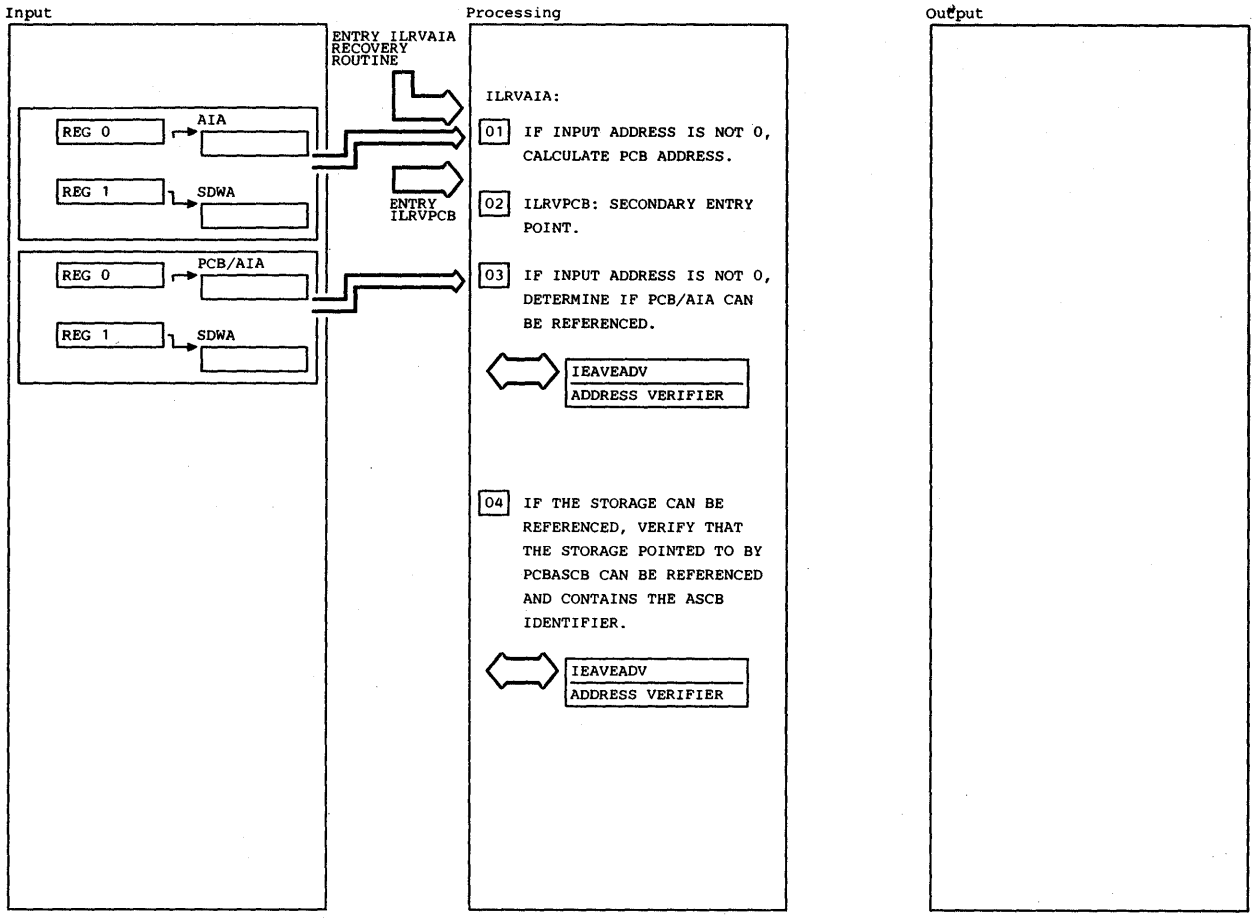
Diagram 25.27.7 ILRVIOEQ (Part 1 of 1)



Notes	Routine	Label	Ref
01 IF THE STORAGE POINTED TO BY REGISTER ZERO CANNOT BE REFERENCED, RETURN IS MADE TO THE CALLER WITH A RETURN CODE OF 8, MEANING THAT THE ELEMENT IS NEITHER AN AIA NOR ACE.	IEAVEADV	IEAVEADV	
02 IF THE STORAGE CAN BE REFERENCED, AN AIA IS DISTINGUISHED FROM AN ACE BY THE OPERATION CODE. SEPARATE ROUTINES PERFORM FURTHER VERIFICATIONS FOR AN AIA AND AN ACE.			
A. AIA VERIFICATION ROUTINE.	ILRFRR01	ILRVAIA	25.27.9
B. ACE VERIFICATION ROUTINE.	ILRFRR01	ILRVACE	25.27.10

Notes	Routine	Label	Ref

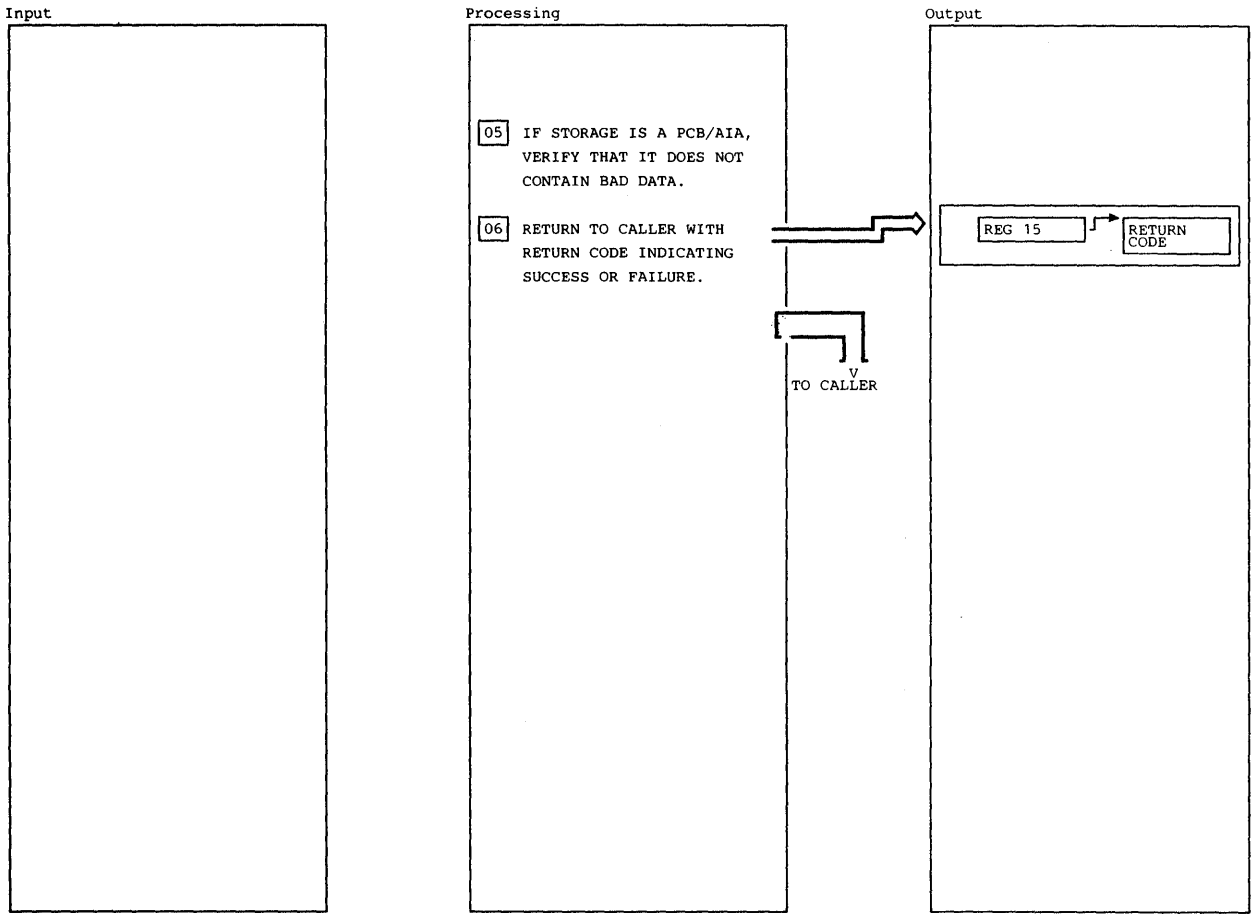
Diagram 25.27.8 ILRVAIAC (Part 1 of 1)



Notes	Routine	Label	Ref
01 FOR ENTRY ILRVAIA, IF INPUT ADDRESS IS 0, A RETURN CODE OF 8 IS SET. OTHERWISE, THE OFFSET TO THE PCB IS CALCULATED.			
02 THIS IS THE ENTRY POINT FOR ILRVPCB.			
03 IF INPUT ADDRESS IS 0, A RETURN CODE OF 8 IS SET. OTHERWISE, VERIFY THAT THE STORAGE POINTED TO BY THE PCB ADDRESS CAN BE REFERENCED. IF IT CANNOT, A RETURN CODE OF 8 IS SET.	IEAVEADV	IEAVEADV	
04 IF THE PCB/AIA CAN BE REFERENCED, VERIFY THAT THE STORAGE POINTED TO BY PCBASCB CAN ALSO BE REFERENCED. IF IT CANNOT, A RETURN CODE OF 8 IS SET. IF IT CAN BE REFERENCED, THE ASCBASCB FIELD IS CHECKED FOR THE ACRONYM 'ASCB'. IF THE ACRONYM IS NOT THERE, A RETURN CODE OF 8 IS SET.	IEAVEADV	IEAVEADV	

Notes	Routine	Label	Ref

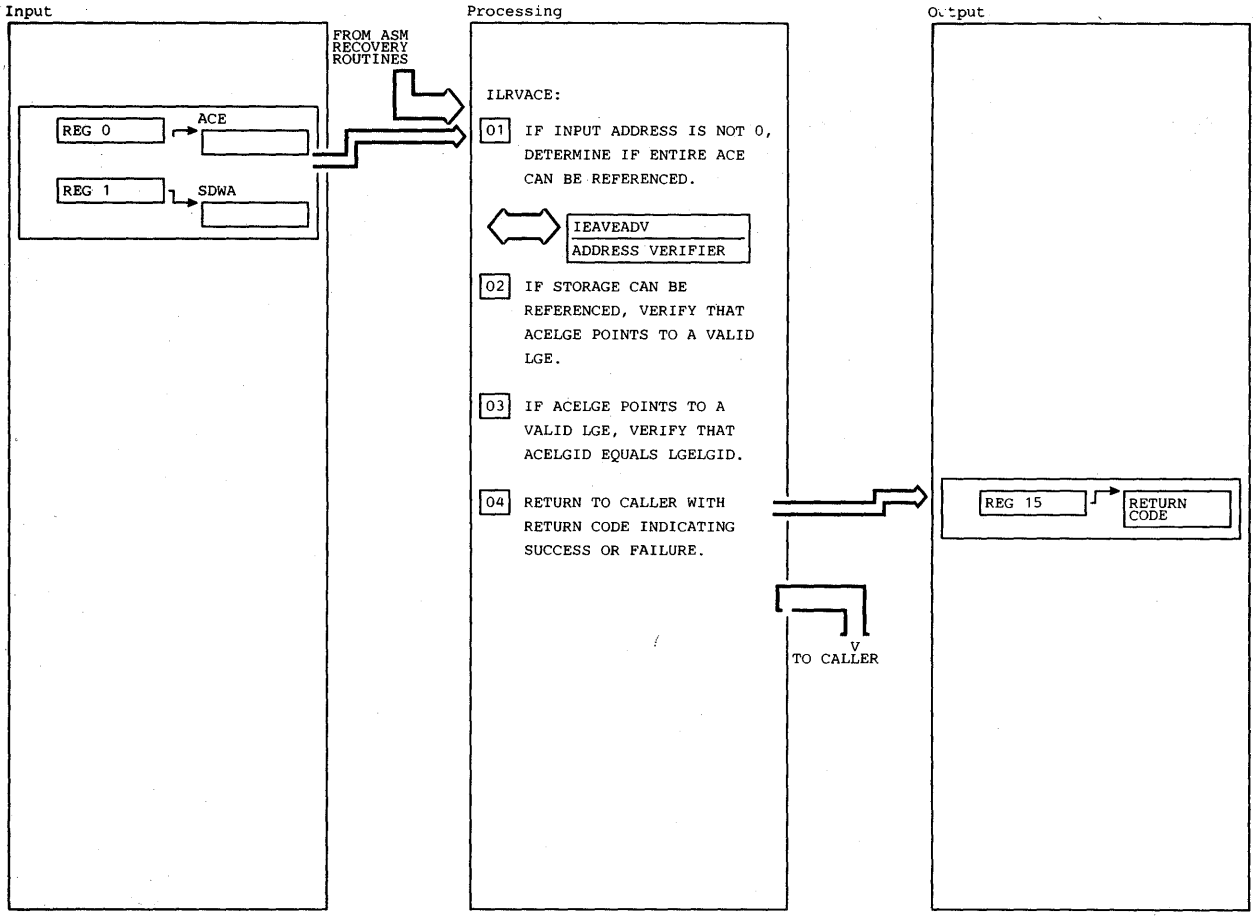
Diagram 25.27.9 ILRVAIA/ILRVPCB (Part 1 of 2)



Notes	Routine	Label	Ref
05 IF THE PREVIOUS VERIFICATIONS ARE SUCCESSFUL, PERFORM VERIFICATIONS TO CHECK FOR BAD DATA (AIAOP=X'00'). IF THIS VERIFICATION FAILS, A RETURN CODE OF 4 IS SET.			
06 RETURN IS MADE TO THE CALLER WHEN AN ERROR IS FOUND OR VALIDITY CHECKING IS COMPLETE. THE POSSIBLE RETURN CODES ARE:			
A. 0 - ELEMENT PASSED ALL TESTS.			
B. 4 - ELEMENT IS A PCB/AIA, BUT IT CONTAINS BAD DATA.			
C. 8 - ELEMENT IS NOT A PCB/AIA.			

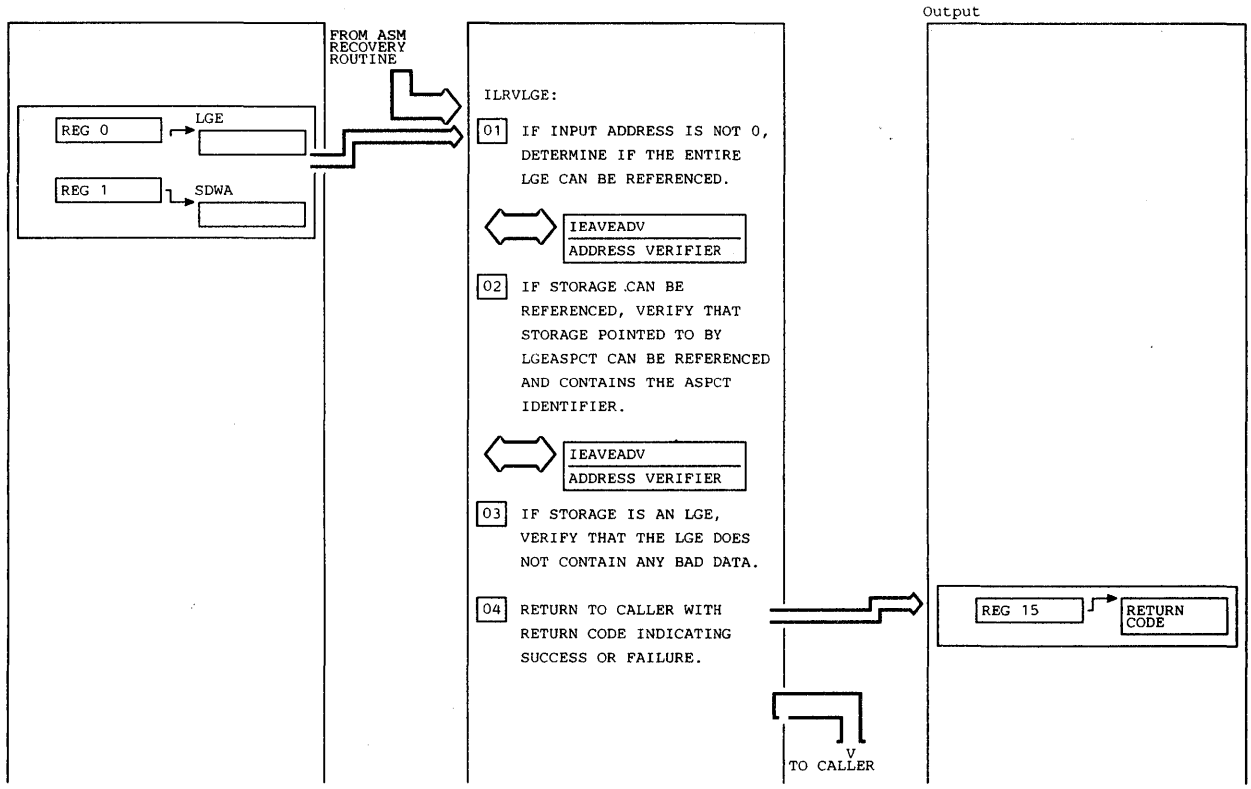
Notes	Routine	Label	Ref

Diagram 25.27.9 ILRVAIA/ILRVPCB (Part 2 of 2)



Notes	Routine	Label	Ref
01 IF INPUT ADDRESS IS 0, A RETURN CODE OF 8 (NOT AN ACE) IS SET. OTHERWISE, VERIFY THAT THE STORAGE POINTED TO BY THE ACE ADDRESS CAN BE REFERENCED. IF IT CANNOT, A RETURN CODE OF 8 IS SET.	IEAVEADV	IEAVEADV	
02 IF THE STORAGE CAN BE REFERENCED, VERIFY THAT ACELGE POINTS TO A VALID LGE. IF IT DOES NOT POINT TO A VALID LGE, A RETURN CODE OF 8 IS SET.	ILRFRR01	ILRVLGE	25.27.11
03 IF THE PREVIOUS VERIFICATIONS ARE SUCCESSFUL, CHECK FOR BAD DATA BY VERIFYING THAT ACELGID EQUALS LGELGID. IF IT DOES NOT,, A RETURN CODE OF 4 IS SET.			
04 RETURN IS MADE TO THE CALLER WHEN AN ERROR IS FOUND OR VALIDITY CHECKING IS COMPLETE. THE POSSIBLE RETURN CODES ARE:			
A. 0 - ELEMENT PASSED ALL TESTS.			
B. 4 - ELEMENT IS AN ACE BUT CONTAINS BAD DATA.			
C. 8 - ELEMENT IS NOT AN ACE.			

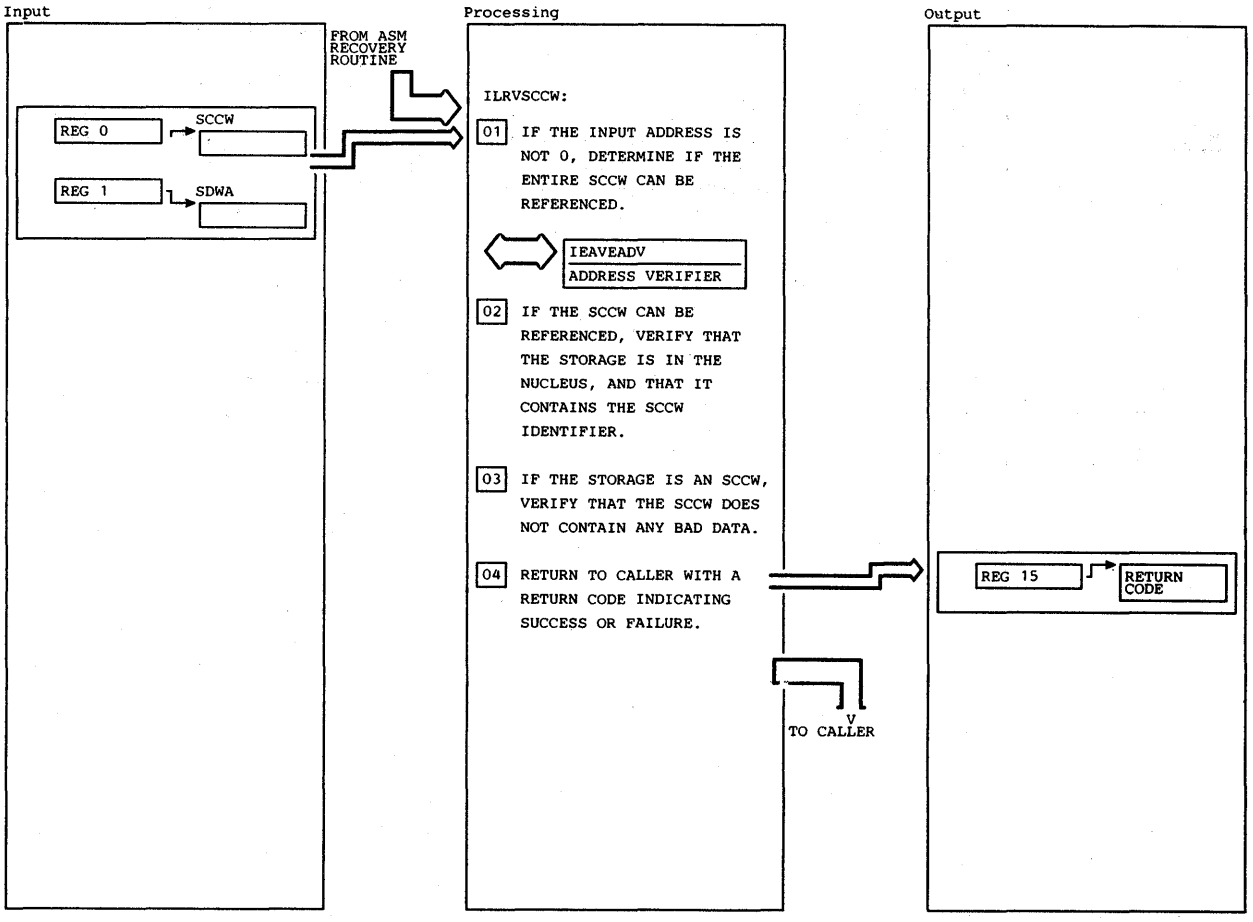
Diagram 25.27.10 ILRVACE (Part 1 of 1)



Notes	Routine	Label	Ref
01 IF INPUT ADDRESS IS 0, A RETURN CODE OF 8 (NOT AN LGE) IS SET. OTHERWISE, VERIFY THAT THE STORAGE POINTED TO BY THE LGE ADDRESS CAN BE REFERENCED. IF IT CANNOT, A RETURN CODE OF 8 IS SET.	IEAVEADV	IEAVEADV	
02 IF THE LGE CAN BE REFERENCED, VERIFY THAT THE STORAGE POINTED TO BY LGEASPCT CAN ALSO BE REFERENCED. IF THIS STORAGE CANNOT BE REFERENCED, A RETURN CODE OF 8 IS SET. IF IT CAN, THE ASPIDENT FIELD IS CHECKED FOR THE ACRONYM 'ASPC'. IF THE ACRONYM IS NOT THERE, A RETURN CODE OF 8 IS SET.	IEAVEADV	IEAVEADV	
03 IF THE PREVIOUS VERIFICATIONS ARE SUCCESSFUL, CHECK FOR AN LGE CONTAINING BAD DATA BY VERIFYING THAT LGELGID EQUALS ASPLGID. IF IT DOES NOT, A RETURN CODE OF 4 (BAD DATA) IS SET.			
04 RETURN IS MADE TO THE CALLER WHEN AN ERROR IS FOUND, OR			

Notes	Routine	Label	Ref
VALIDITY CHECKING IS COMPLETE. THE POSSIBLE RETURN CODES ARE: A. 0 - ELEMENT PASSED ALL TESTS. B. 4 - ELEMENT IS AN LGE BUT CONTAINS BAD DATA. C. 8 - ELEMENT IS NOT AN LGE.			

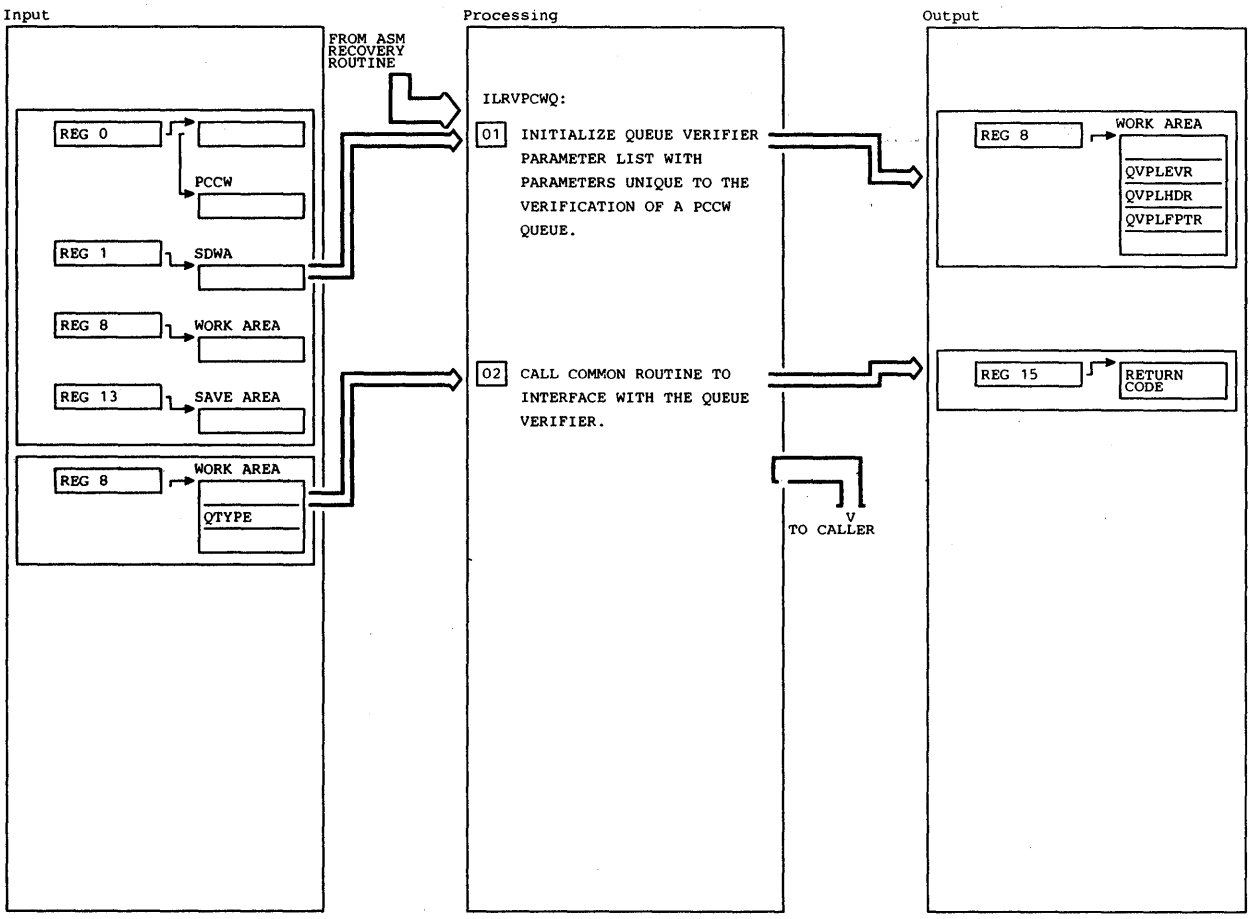
Diagram 25.27.11 ILRVLGE (Part 1 of 1)



Notes	Routine	Label	Ref
01 IF THE INPUT ADDRESS IS 0, A RETURN CODE OF 8 (NOT AN SCCW) IS SET. OTHERWISE, VERIFY THAT THE STORAGE POINTED TO BY THE SCCW ADDRESS CAN BE REFERENCED. IF IT CANNOT, A RETURN CODE OF 8 IS SET.	IEAVEADV	IEAVEADV	
02 IF THE STORAGE CAN BE REFERENCED, VERIFY THAT THE STORAGE IS IN THE NUCLEUS (< CVTNUCB). IF IT IS NOT, A RETURN CODE OF 8 IS SET. ALSO VERIFY THAT THE SCCWID FIELD CONTAINS THE SCCW IDENTIFIER, X'87'. IF IT IS NOT THERE, A RETURN CODE OF 8 IS SET.			
03 IF THE PREVIOUS VERIFICATIONS ARE SUCCESSFUL, MAKE THE FOLLOWING TESTS TO CHECK FOR BAD DATA. IF ANY OF THESE TESTS FAIL, A RETURN CODE OF 4 IS SET. A. CHECK THAT THE FIRST BYTE OF SCCWSEEK IS THE SEEK COMMAND CODE (X'0B'). B. CHECK THAT THE FIRST BYTE OF			

Notes	Routine	Label	Ref
SCCWSSEC IS THE SET SECTOR COMMAND CODE (X'23'). C. CHECK THAT THE DATA ADDRESS OF SCCWSSEC IS THE ADDRESS OF SCCWSECT. 04 RETURN IS MADE TO THE CALLER WHEN AN ERROR IS FOUND OR VALIDITY CHECKING IS COMPLETE. THE POSSIBLE RETURN CODES ARE: A. 0 - ELEMENT PASSED ALL TESTS. B. 4 - ELEMENT IS AN SCCW BUT CONTAINS BAD DATA. C. 8 - ELEMENT IS NOT AN SCCW.			

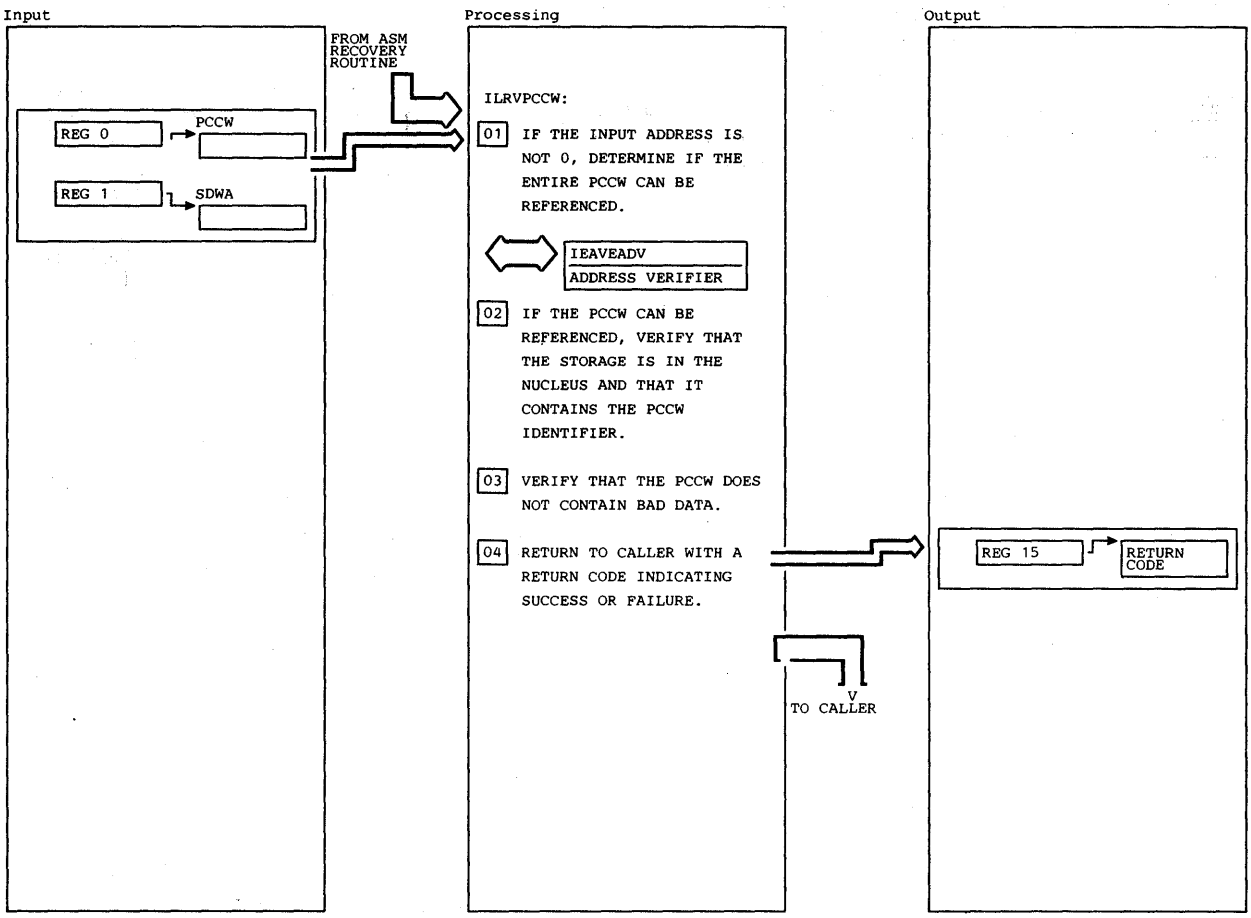
Diagram 25.27.12 ILRVSCCW (Part 1 of 1)



Notes	Routine	Label	Ref
01 INITIALIZE THE PARAMETERS OF THE QUEUE VERIFIER PARAMETER LIST THAT ARE UNIQUE TO THE VERIFICATION OF A PCCW QUEUE. THESE PARAMETERS ARE THE ADDRESS OF THE QUEUE HEADER (VALUE OF REGISTER 0), THE ADDRESS OF THE ELEMENT VERIFICATION ROUTINE (ILRVPCCW), AND THE OFFSET OF THE FORWARD CHAIN POINTER (PCCWPCCW).			
02 CALL COMMON ROUTINE TO SET UP THE INTERFACE FOR THE QUEUE VERIFIER. SET QTYPE=1 TO INDICATE THE QUEUE IS A SINGLE-HEADED,SINGLE-THREADED QUEUE.	ILRFRR01	COMQRTN	25.27.20

Notes	Routine	Label	Ref

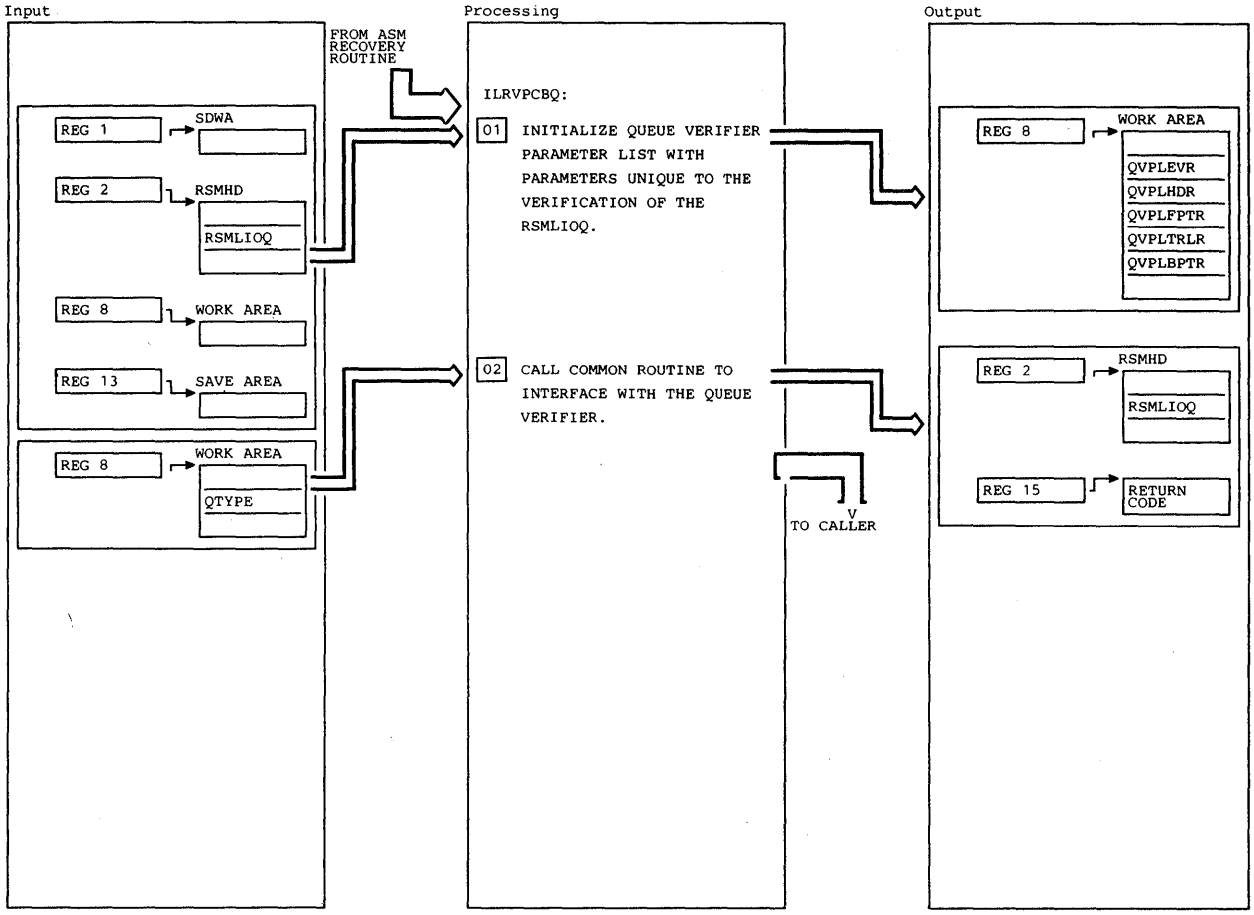
Diagram 25.27.13 ILRVPCWQ (Part 1 of 1)



Notes	Routine	Label	Ref
01 IF THE INPUT ADDRESS IS 0, A RETURN CODE OF 8 (NOT A PCCW) IS SET. OTHERWISE, VERIFY THAT THE STORAGE POINTED TO BY THE PCCW ADDRESS CAN BE REFERENCED. IF IT CANNOT, A RETURN CODE OF 8 IS SET.	IEAVEADV	IEAVEADV	
02 IF THE STORAGE CAN BE REFERENCED, VERIFY THAT THE STORAGE IS IN THE NUCLEUS (< CVTNUCB). IF IT IS NOT, A RETURN CODE OF 8 IS SET. ALSO CHECK THAT THE PCCWID FIELD CONTAINS THE IDENTIFIER X'86'. IF IT DOES NOT, A RETURN CODE OF 8 IS SET.			
03 MAKE THE FOLLOWING TESTS TO CHECK FOR BAD DATA. IF ANY OF THESE TESTS FAILS, A RETURN CODE OF 4 IS SET. A. CHECK THAT THE FIRST BYTE OF PCCWSRCH IS THE SEARCH COMMAND CODE (X'31'). B. CHECK THAT THE FIRST BYTE OF PCCWTIC IS THE TIC COMMAND CODE (X'08').			

Notes	Routine	Label	Ref
04 RETURN IS MADE TO THE CALLER WHEN AN ERROR IS FOUND OR VALIDITY CHECKING IS COMPLETE. THE POSSIBLE RETURN CODES ARE: A. 0 - ELEMENT PASSED ALL TESTS. B. 4 - ELEMENT IS A PCCW BUT CONTAINS BAD DATA. C. 8 - ELEMENT IS NOT A PCCW.			

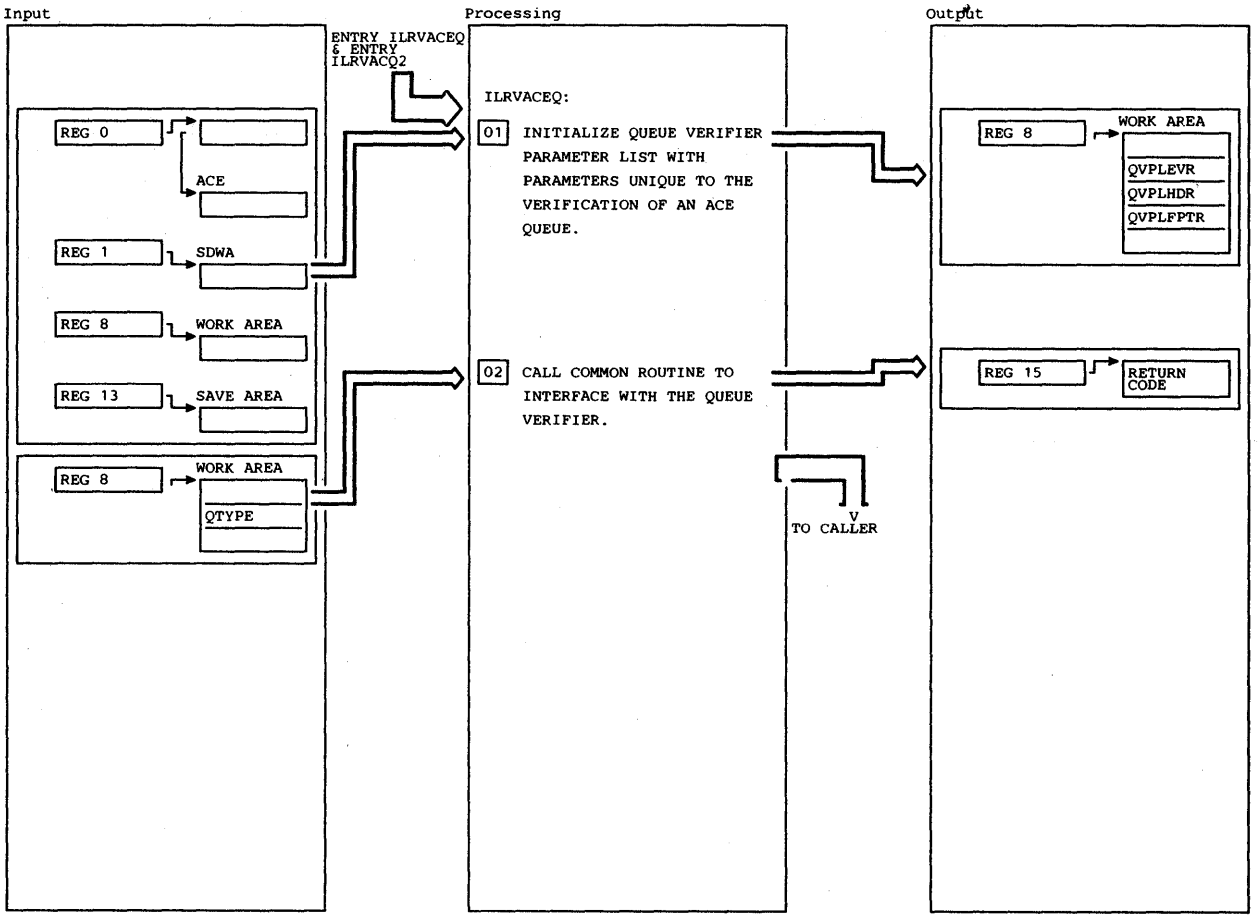
Diagram 25.27.14 ILRVPCCW (Part 1 of 1



Notes	Routine	Label	Ref
01 INITIALIZE THE PARAMETERS OF THE QUEUE VERIFIER PARAMETER LIST THAT ARE UNIQUE TO THE VERIFICATION OF THE RSMLIOQ. THESE PARAMETERS ARE THE ADDRESS OF THE QUEUE HEADER (RSMLIOQF), THE ADDRESS OF THE QUEUE TRAILER (RSMLIOQL), THE ADDRESS OF THE ELEMENT VERIFICATION ROUTINE (ILRVPCB), THE OFFSET OF THE FORWARD CHAIN POINTER (PCBFQP, 3 BYTE POINTER), AND THE OFFSET OF THE BACKWARD CHAIN POINTER (PCBBQP, 4 BYTE POINTER).			
02 CALL COMMON ROUTINE TO SET UP THE INTERFACE FOR THE QUEUE VERIFIER. SET QTYPE=3 TO INDICATE THE QUEUE IS A DOUBLE-HEADED, DOUBLE-THREADED QUEUE.	ILRFRR01	COMQRTN	25.27.20

Notes	Routine	Label	Ref

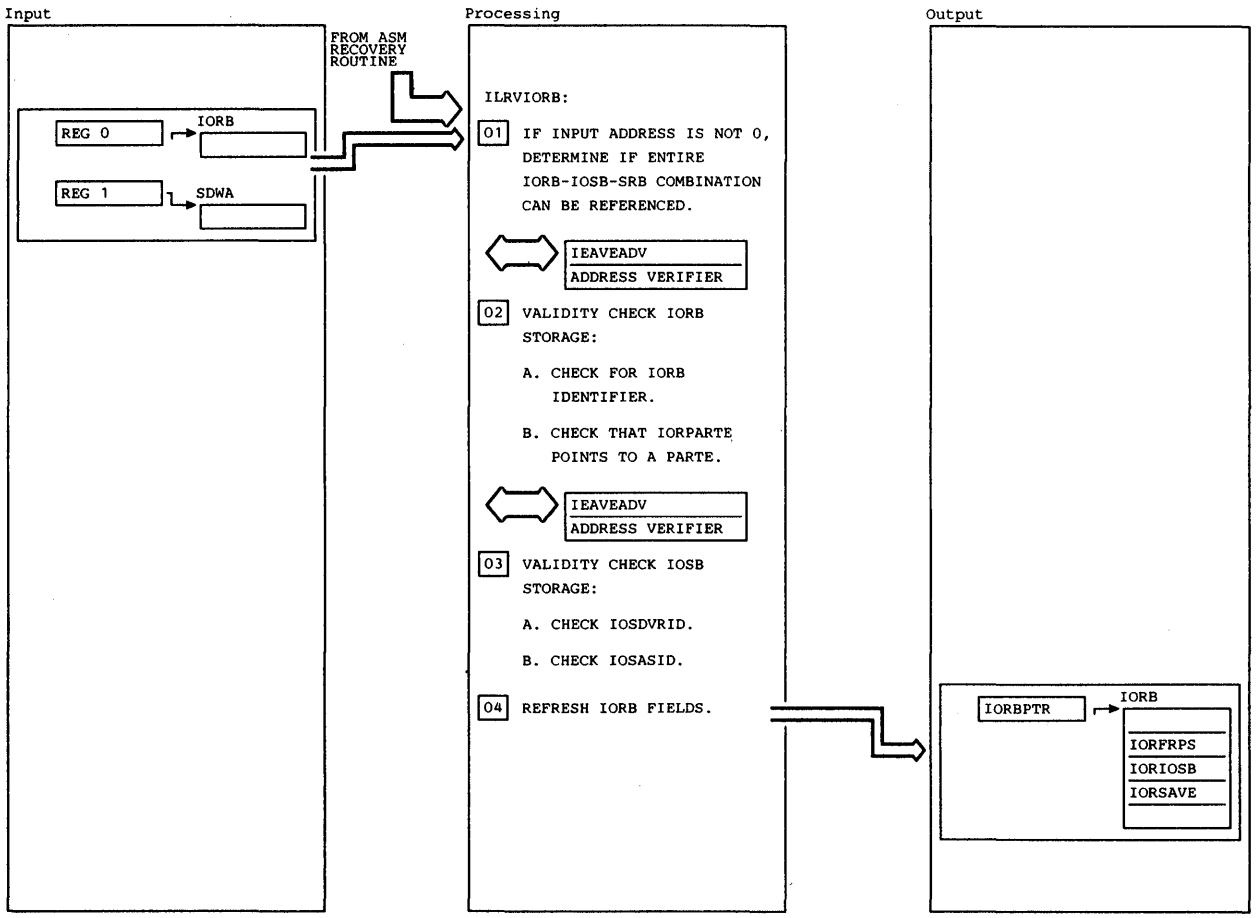
Diagram 25.27.15 ILRVPCBQ (Part 1 of 1)



Notes	Routine	Label	Ref
01 ENTRY IS FROM ASM RECOVERY ROUTINES. INITIALIZE THE PARAMETERS OF THE QUEUE VERIFIER PARAMETER LIST THAT ARE UNIQUE TO THE VERIFICATION OF AN ACE QUEUE. THESE PARAMETERS ARE THE ADDRESS OF THE QUEUE HEADER (VALUE OF REGISTER 0), THE ADDRESS OF THE ELEMENT VERIFICATION ROUTINE (ILRVACE), AND THE OFFSET OF THE FORWARD CHAIN POINTER (ACESRBWK FOR ENTRY ILRVACEQ, ACEFQPA FOR ENTRY ILRVACQ2).			
02 CALL COMMON ROUTINE TO SET UP THE INTERFACE FOR THE QUEUE VERIFIER. SET QTYPE=1 TO INDICATE THE QUEUE IS A SINGLE-HEADED,SINGLE-THREADED QUEUE.	ILRFRR01	COMQRTN	25.27.20

Notes	Routine	Label	Ref

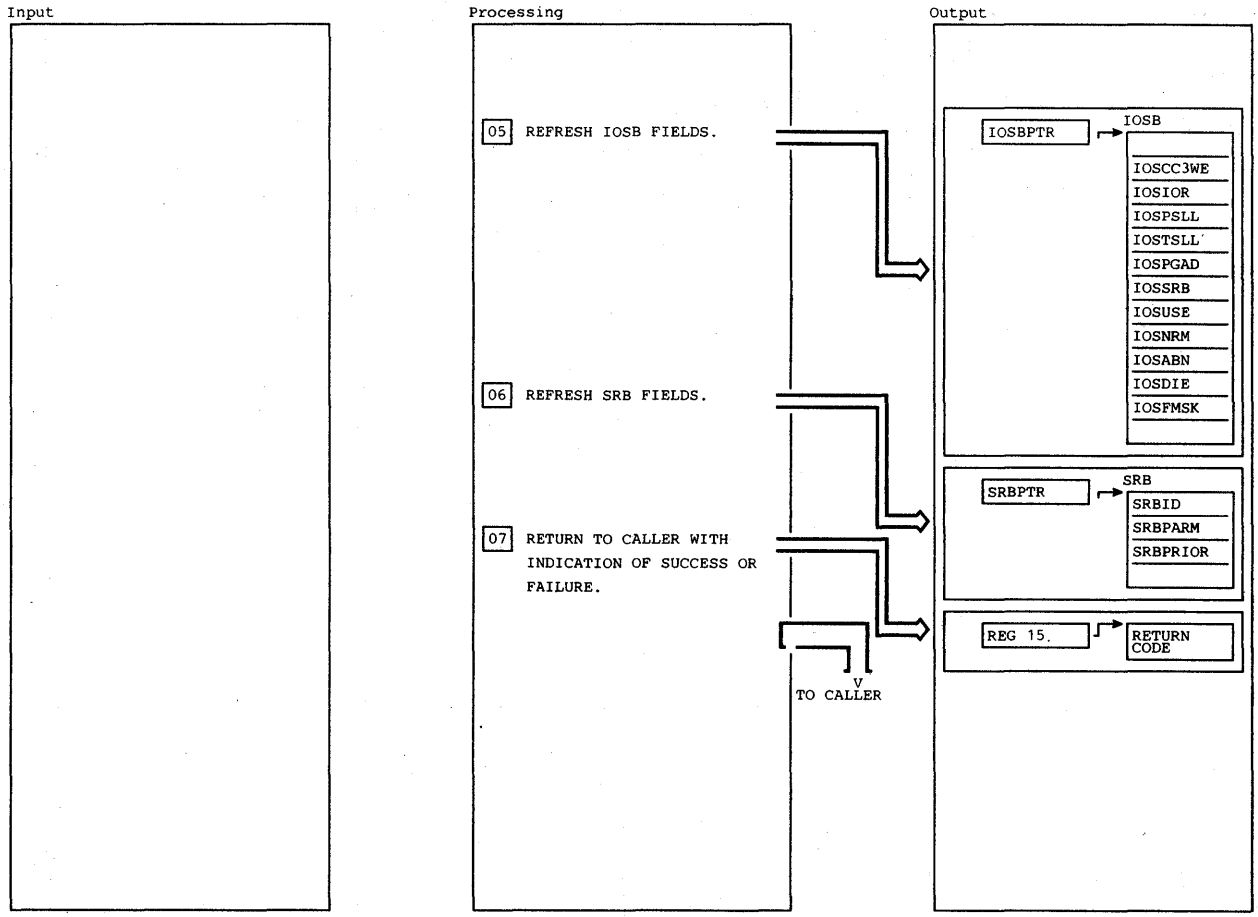
Diagram 25.27.16 ILRVACEQ (Part 1 of 1)



Notes	Routine	Label	Ref
01 IF INPUT ADDRESS IS 0, A RETURN CODE OF 8 (NOT AN IORB-IOSB-SRB) IS SET. OTHERWISE, VERIFY THAT THE STORAGE POINTED TO BY THE IORB ADDRESS CAN BE REFERENCED. IF IT CANNOT, A RETURN CODE OF 8 IS SET.	IEAVEADV	IEAVEADV	
02 IF THE STORAGE CAN BE REFERENCED, VALIDITY CHECK THE IORB STORAGE. IF ANY OF THE TESTS FAIL, A RETURN CODE OF 8 IS SET.			
A. CHECK THAT IORID CONTAINS THE IORB IDENTIFIER, X'88'.			
B. VERIFY THAT THE STORAGE POINTED TO BY IORPARTE CAN BE REFERENCED. IF IT CAN, CHECK THAT THE PARTE INDICATED BY PARENN (OR SARTE INDICATED BY SRENN, IF IORSWAP IS ON) IS THE SAME AS IORPARTE.	IEAVEADV	IEAVEADV	
03 IF THE IORB VALIDITY CHECKS, VALIDITY CHECK THE IOSB STORAGE. IF ANY TEST FAILS, A RETURN CODE OF 8 IS SET.			

Notes	Routine	Label	Ref
A. CHECK THAT IOSDVRID CONTAINS IOSMISID (THE MISCELLANEOUS ID).			
B. CHECK THAT IOSASID CONTAINS 1 (MASTER SCHEDULER'S ADDRESS SPACE ID).			
04 IF ALL PREVIOUS VERIFICATIONS ARE SUCCESSFUL, REFRESH IORB FIELDS ORIGINALLY SET BY ILROPS00: THE IORFRPS FLAG, IORSAVE, AND IORIOSB.			

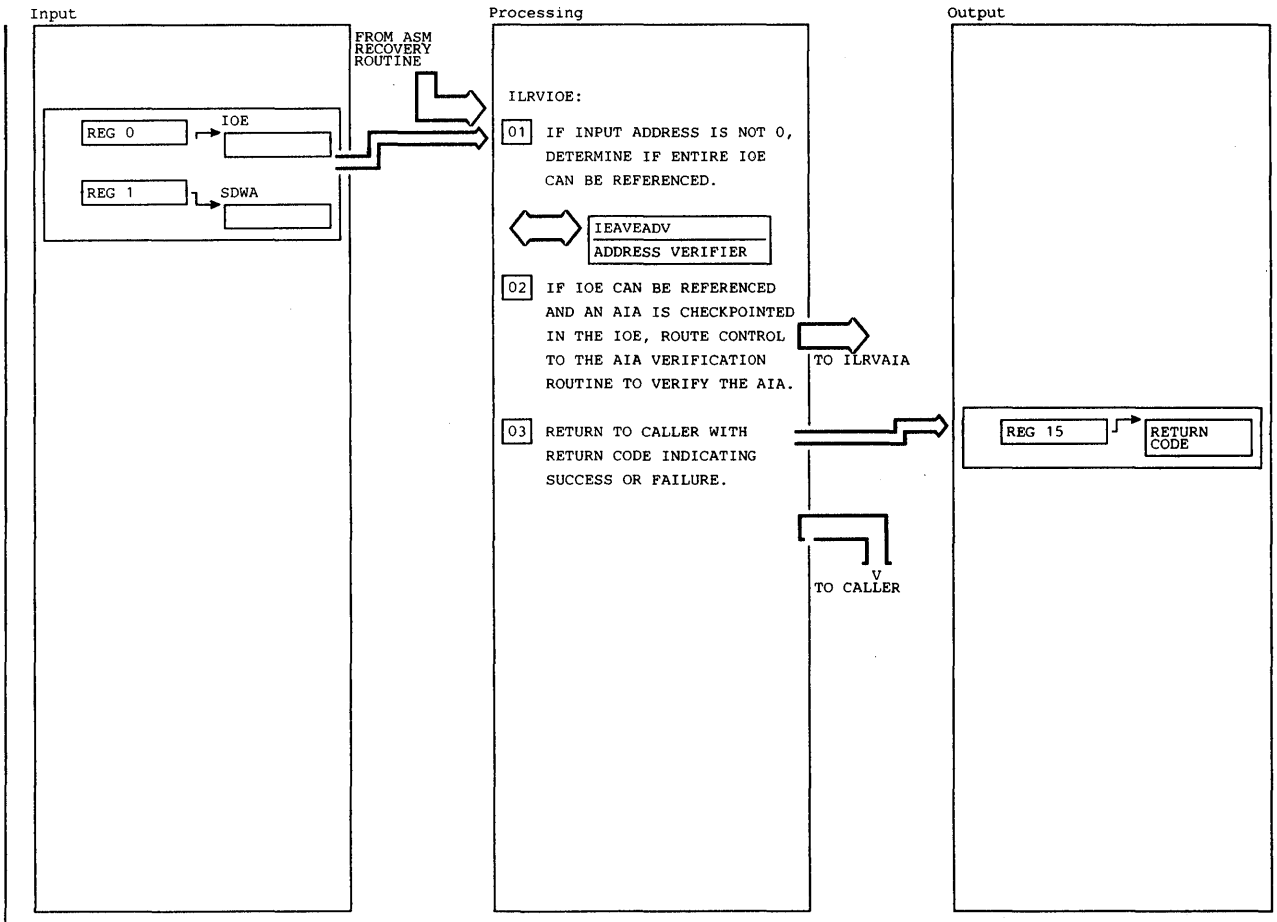
Diagram 25.27.17 ILRVIORB (Part 1 of 2)



Notes	Routine	Label	Ref
05 REFRESH IOSB FIELDS ORIGINALLY SET BY ILROPS00: FLAGS IOSCC3WE, IOSIDR, IOSPESLL, IOSTSLL, AND FIELDS IOSPGAD, IOSSRB, IOSUSE, IOSNRM, IOSABN, IOSDIE (WITH HIGH ORDER BIT ON), AND IOSFMSK.			
06 REFRESH SRB FIELDS: SRBID, SRBPARM, AND SRBPRIOR.			
07 RETURN IS MADE TO THE CALLER WHEN AN ERROR IS FOUND OR THE REFRESH IS COMPLETE. THE POSSIBLE RETURN CODES ARE: A. 0 - STORAGE PASSED ALL TESTS AND WAS REFRESHED. B. 8 - STORAGE IS NOT AN IORB-IOSB-SRB.			

Notes	Routine	Label	Ref

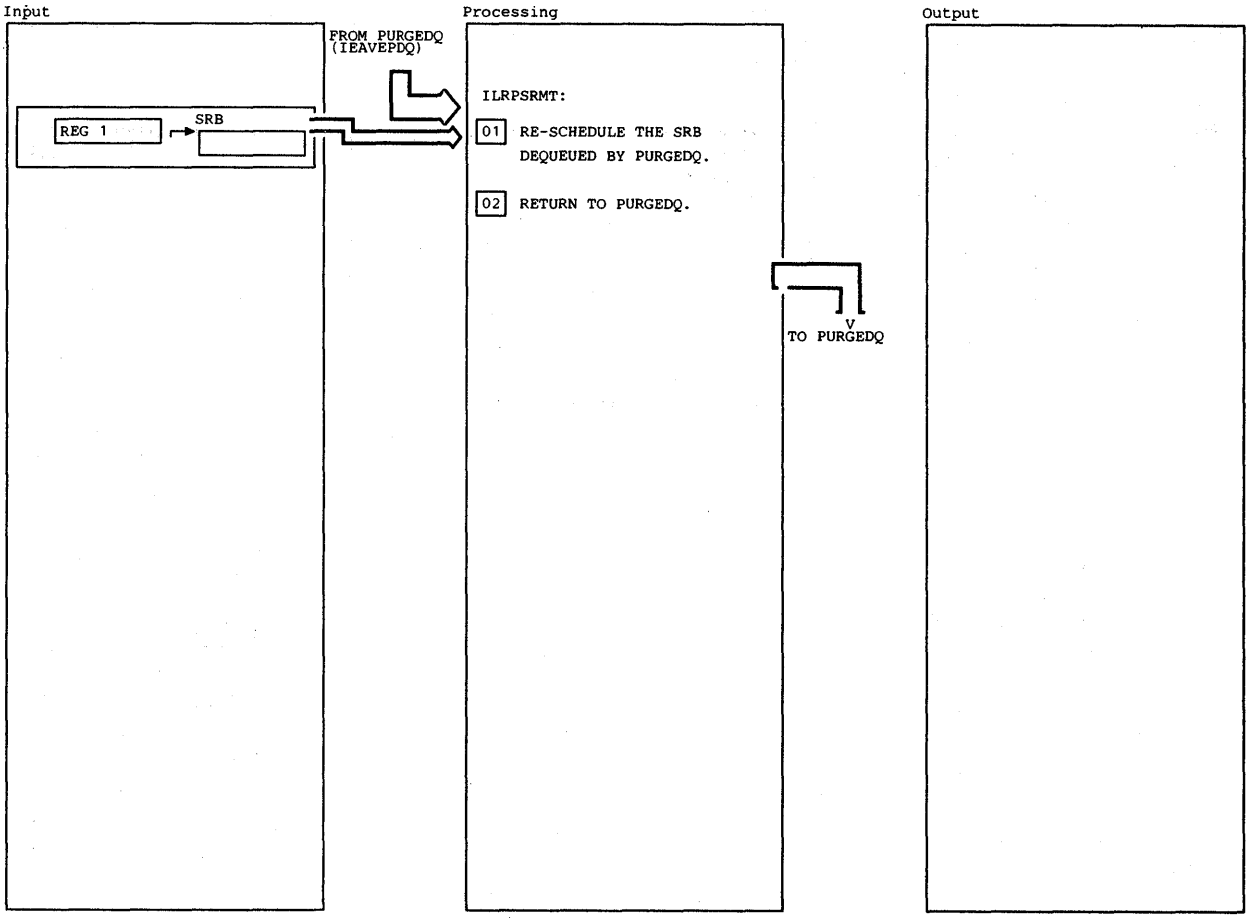
Diagram 25.27.17 ILRVIORB (Part 2 of 2)



Notes	Routine	Label	Ref
01 IF INPUT ADDRESS IS 0, A RETURN CODE OF 8 (NOT AN IOE) IS SET. OTHERWISE, VERIFY THAT THE STORAGE POINTED TO BY THE IOE ADDRESS CAN BE REFERENCED. IF IT CAN NOT, A RETURN CODE OF 8 IS SET.	IEAVEADV	IEAVEADV	
02 IF THE STORAGE CAN BE REFERENCED AND IOEAIA IS NOT 0, THE AIA VERIFICATION ROUTINE IS GIVEN CONTROL TO VERIFY THE AIA. THIS ROUTINE WILL RETURN DIRECTLY TO THE CALLER OF ILRVIOE.	ILRFRR01	ILRVAIA	25.27.9
03 RETURN IS MADE TO THE CALLER WHEN AN ERROR IS FOUND OR IF ILRVAIA HAS NOT BEEN GIVEN CONTROL. THE POSSIBLE RETURN CODES ARE: A. 0 - ELEMENT PASSED ALL TESTS. B. 8 - ELEMENT IS NOT AN IOE.			

Notes	Routine	Label	Ref

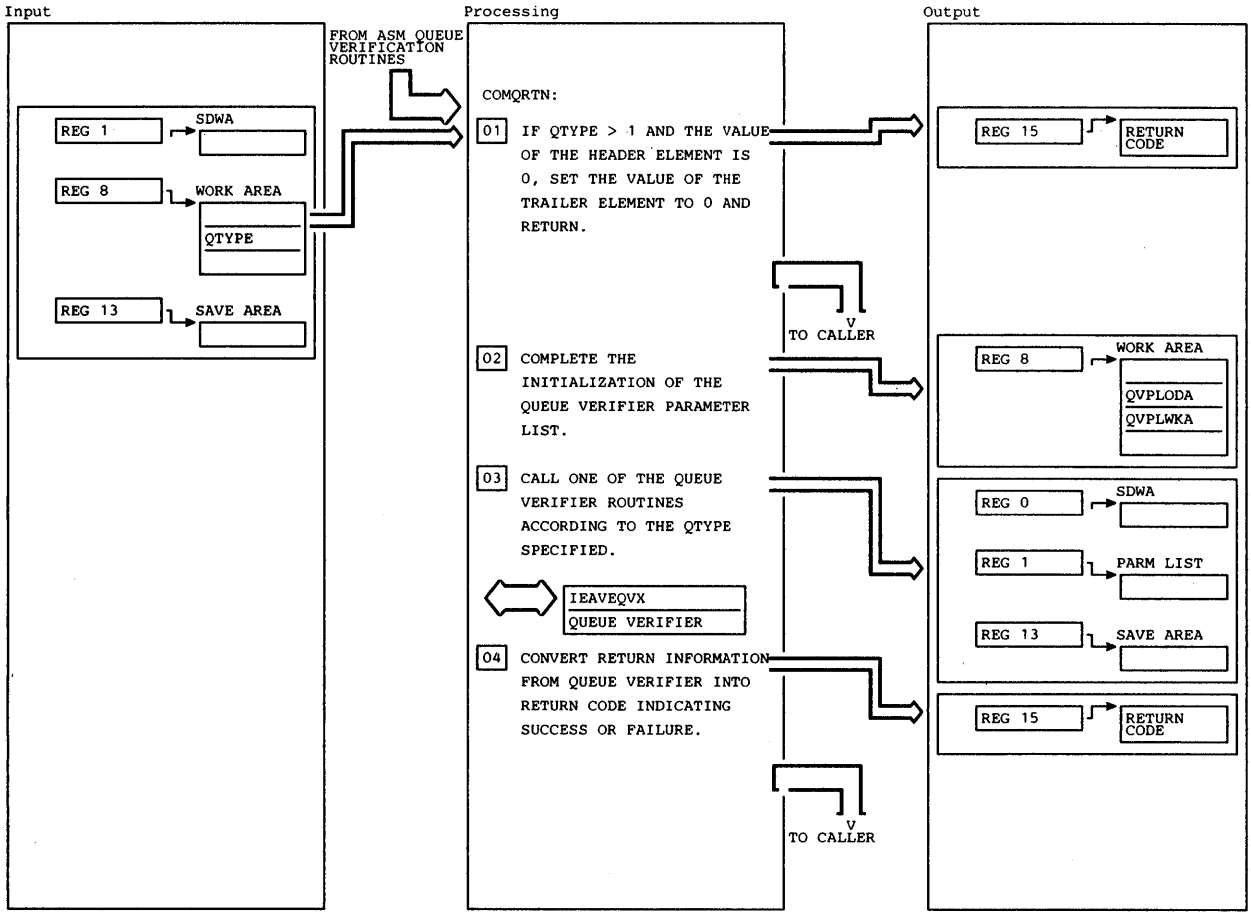
Diagram 25.27.18 ILRVIOE (Part 1 of 1)



Notes	Routine	Label	Ref
01 THE SRB'S FOR PART MONITOR AND SWAP DRIVER SHOULD NOT BE PURGED. IF PURGEDQ IS EVER CALLED TO PURGE ALL SRB'S IN THE MASTER SCHEDULER'S ADDRESS SPACE, THIS ROUTINE RESCHEDULES THE SRB THAT WAS PURGED.			

Notes	Routine	Label	Ref

Diagram 25.27.19 ILRPSRMT (Part 1 of 1)



Notes	Routine	Label	Ref
01 FOR DOUBLE-HEADED QUEUES (QTYPE > 1), CHECK THE VALUE OF THE HEADER ELEMENT FOR 0. IF IT IS 0, NO FURTHER VERIFICATION NEEDS TO BE DONE. INSURE THAT THE VALUE OF THE TRAILER ELEMENT IS ALSO 0 AND RETURN TO THE CALLER. IF THE HEADER ELEMENT IS NOT ZERO, CONTINUE.			
02 INITIALIZE THE COMMON PARAMETERS FOR THE QUEUE VERIFIER, SUCH AS THE ADDRESS OF THE VARIABLE RECORDING AREA IN THE SDWA AND THE ADDRESS OF THE WORKAREA FOR THE QUEUE VERIFIER.			
03 CALL THE APPROPRIATE QUEUE VERIFIER ROUTINE ACCORDING TO THE QTYPE SPECIFIED BY THE CALLER.			
A. QTYPE=1, VERIFY SINGLE-THREADED, SINGLE-HEADED QUEUE.	IEAVEQV0	IEAVEQV1	
B. QTYPE=2, VERIFY SINGLE-THREADED, DOUBLE-HEADED QUEUE.	IEAVEQV0	IEAVEQV2	
C. QTYPE=3, VERIFY DOUBLE-THREADED, DOUBLE-HEADED QUEUE.	IEAVEQV0	IEAVEQV3	
04 THE QUEUE VERIFIER RETURNS INFORMATION ABOUT HOW THE QUEUE WAS CORRECTED IN ADDITION TO A RETURN CODE INDICATING WHETHER ANY ERRORS WERE FOUND. PRESERVE ONLY THE RETURN CODE.			

Diagram 25.27.20 COMQRTN (Part 1 of 1)

Service Routines

This section describes three service routines in ASM:

- ILRPEX — Pool Extender,
- ILRTERMR — Address Space Termination Resource Manager,
- ILRFMT00 — Control Block Formatter.

Pool Extender

The Pool Extender routine (ILRPEX) expands a pool of control blocks or work areas if the pool becomes temporarily empty. The ILRGMA macro (issued in mainline ASM routines) passes control to ILRPEX. Input to ILRPEX is the address of the appropriate pool controller. The pool controller contains the size of each cell and the number of cells to build for the expansion. ILRPEX obtains sufficient space from SQA and formats it by chaining together cell-sized portions of the storage. ILRPEX returns the first cell to the caller and places the remaining cells on the available queue of the pool controller.

Address Space Termination Resource Manager

The ASM Address Space Termination routine (ILRTERMR) provides clean-up of ASM resources during normal or abnormal address space termination and attempts to recover auxiliary storage resources from an address space that is terminating abnormally. All ASM resources for the address space including storage, control blocks, and auxiliary storage slots, are freed or marked to be freed when in-process operations complete.

RTM (Recovery Termination Manager) gives control to ILRTERMR during termination of any address space. ILRTERMR attempts to free auxiliary storage slots assigned to private area address space pages and VIO data set logical groups. Abnormal address space terminations are not scheduled while the address space is swapped out. If the address is swapped out, ASM tables for the address space are unavailable; only swap sets assigned to the address space can be freed.

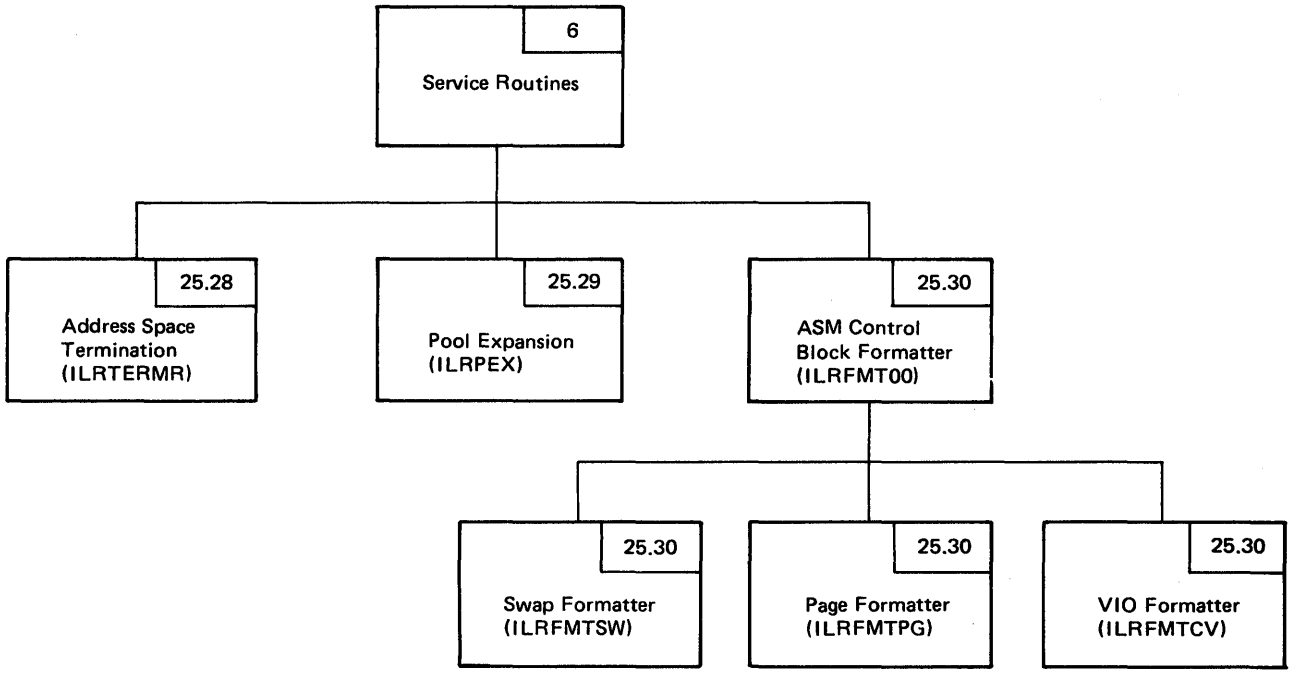
The ASM termination routine also receives control during normal address space termination. This is a safety-valve type operation to assure that all auxiliary storage resources assigned to an address space have in fact been freed. If resources have not been freed, an error is assumed to have occurred and the error is recorded before attempting to free the resources.

Another entry point of ILRTERMR (ILRSLTRV) receives control during memory creation (from IEAVITAS) to determine if there are enough slots to create a new memory.

Control Block Formatter

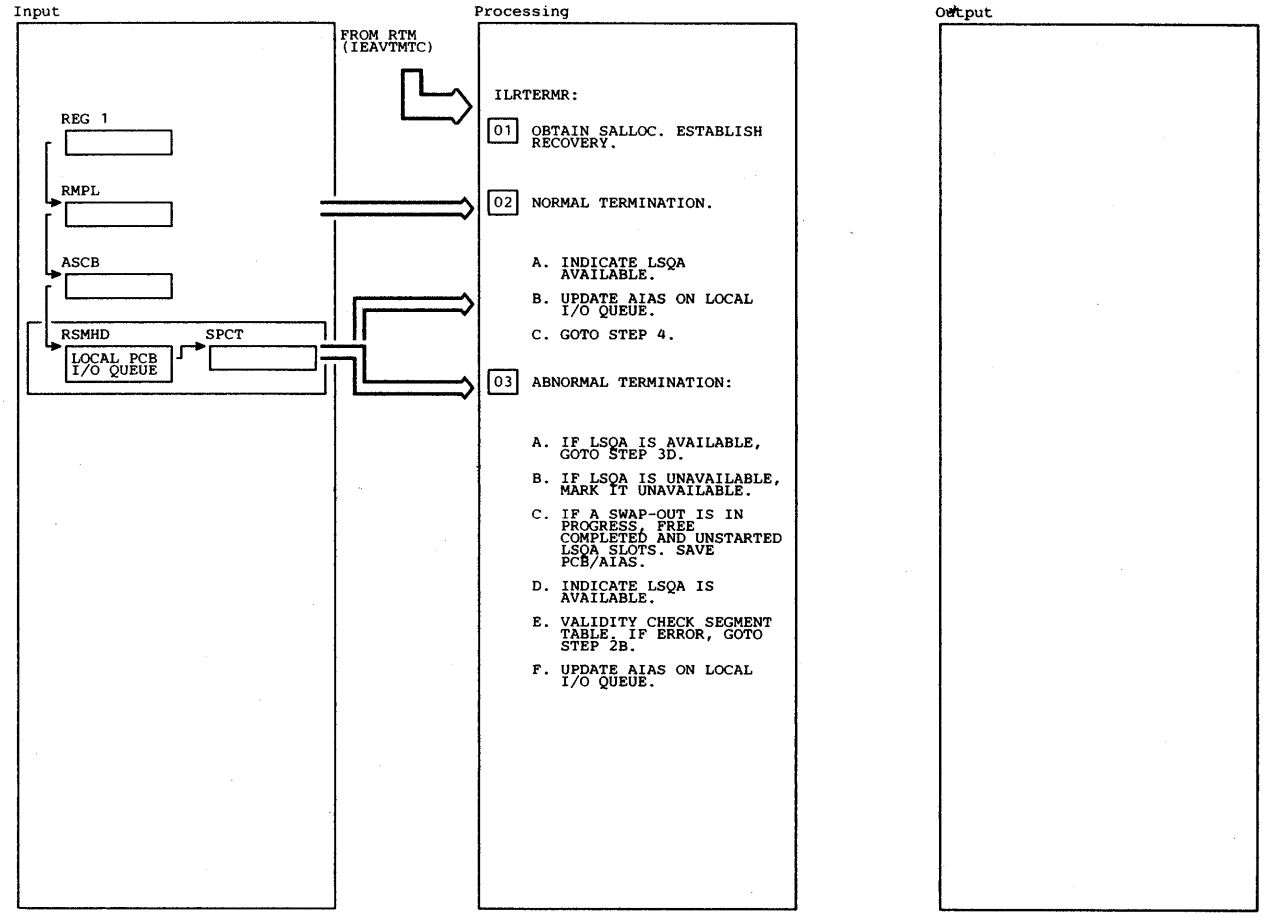
The system dump-printing routine (AMDPRDMP) invokes the Control Block Formatter (ILRFMT00). ILRFMT00 calls ILRFMTPG, ILRFMTSW, and ILRFMTCV to format ASM and shared RSM control blocks. The control blocks are contained in storage areas passed by AMDPRDMP. Formatting is done as follows:

1. Beginning from the CVT address passed in the input parameter list, the routine attempts to access the ASMVT. If successful, it formats the ASMVT (including the bad slot error record, message buffer, ACEs and AIAs).
2. Calls module ILRFMTPG to format the PART and its associated blocks (AIAs, IOEs, PARTEs, PCTs, PATs, IORBs, IOSBs and PCCWs).
3. Calls ILRFMTSW to format the SART and its associated blocks (AIAs, SARTEs, SATs, SDCTs, IORBs, IOSBs and SCCWs).
4. Calls module ILRFMTCV at entry point ILRFMTC to format the common service area page tables (PGTs) and external page tables (XPTs).
5. Calls module ILRFMTCV at entry point ILRFMTH for each address space to format RSMHD, SPCT, ASMHD, AIAs and private area PGT/XPTs.
6. Calls module ILRFMTCV at entry point ILRFMTV to format LGVT and its associated blocks (LGEs, ASPCTs, LPMEs, ASSTs, AIA/ACEs).



25.x. – Module
25.x.y. – Entry point in module 25.x.

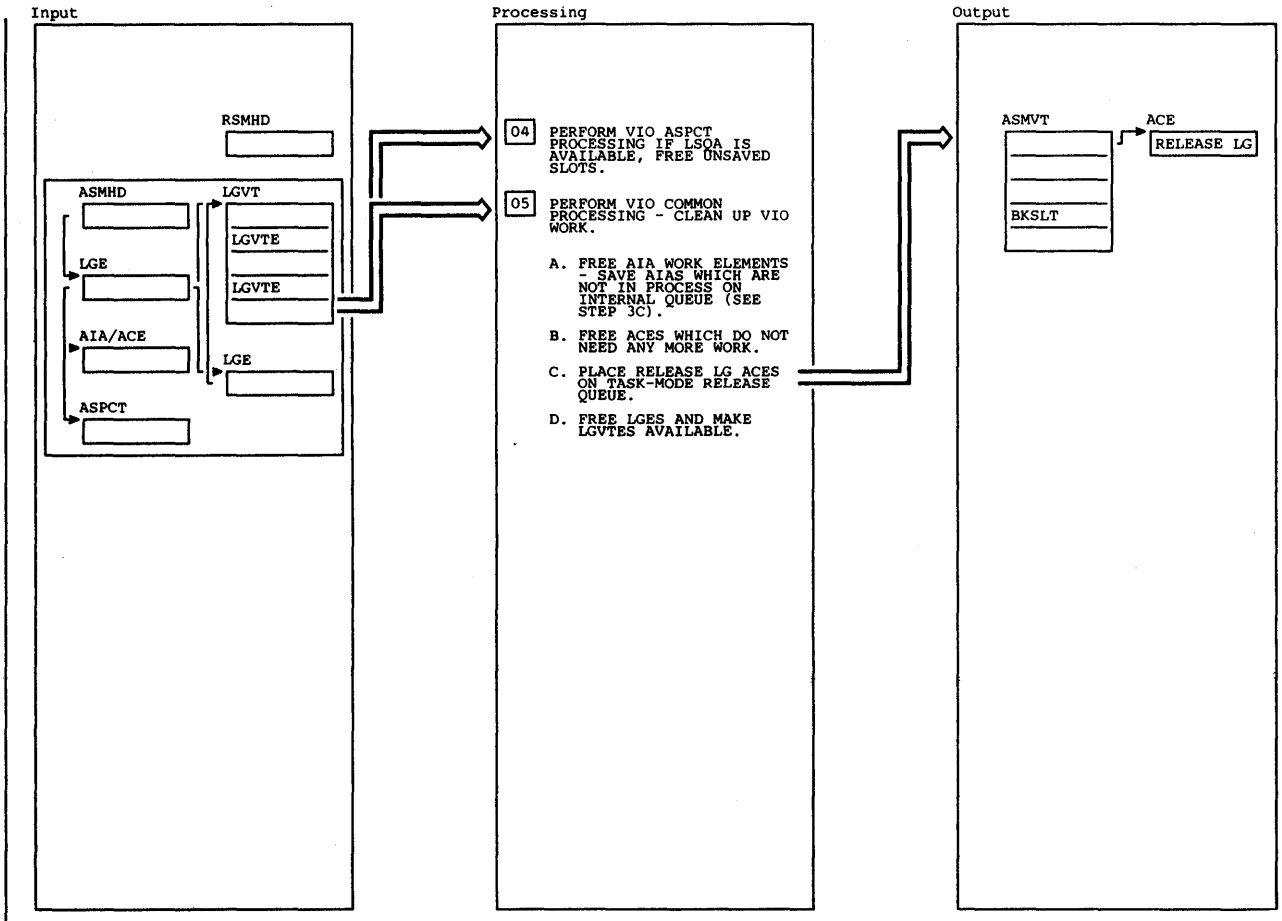
Figure 2-62. Service Routines Overview



Notes	Routine	Label	Ref
01 ILRTERMR RECEIVES CONTROL FROM R/TM DURING TERMINATION OF ANY ADDRESS SPACE. ALL ASM RESOURCES FOR THE ADDRESS SPACE, INCLUDING STORAGE, CONTROL BLOCKS AND AUXILIARY STORAGE SLOTS, ARE FREED OR MARKED TO BE FREED WHEN IN-PROCESS OPERATIONS COMPLETE. THE SALLOC IS OBTAINED IN ORDER TO SERIALIZE ASM/RSM PROCESSING. AN FRR IS ESTABLISHED FOR RECOVERY. TERMRFRR, ANOTHER ENTRY IN ILRTERMR, HANDLES ERRORS OCCURRING IN ILRTERMR.			
02 THE RMPL IS CHECKED TO DETERMINE IF THE ADDRESS SPACE IS TERMINATING NORMALLY OR ABNORMALLY. THE FOLLOWING IS DONE FOR NORMAL TERMINATION:			
A. A LOCAL FLAG IS SET INDICATING THAT LSQA IS AVAILABLE.			
B. THE LOCAL I/O QUEUE IS SCANNED. ALL AIAS ARE MARKED TO FREE THE SLOTS WHEN THE OPERATION COMPLETES AND TO INDICATE THAT TERMINATION HAS PROCESSED THEM.		TERMAIA1	
C. CONTINUE COMMON PROCESSING AT STEP4.			
03 FOR ABNORMAL TERMINATION, THE LSQA IS NEEDED TO PERFORM SOME OF THE CLEAN-UP.			
A. IF LSQA IS AVAILABLE, AS INDICATED BY RSMFAIL=0, RSMLSQAF=0, AND NO SWAP OPERATION IN PROCESS, CONTINUE AT STEP 3D.			
B. IF THE ADDRESS SPACE IS SWAPPED OUT (RSMLSQAF=0), OR IF A SWAP-IN IS IN PROGRESS BUT HAS NOT FAILED, AN ERROR EXISTS. A COD ABEND IS ISSUED		TERMRFRR	

Notes	Routine	Label	Ref
IN ORDER TO RECORD THE ERROR, LSQA IS MARKED UNAVAILABLE, AND PROCESSING CONTINUES AT STEP 2B. IF A SWAP-IN FAILURE HAS OCCURRED (RSMFAIL=1), MARK LSQA UNAVAILABLE. CONTINUE AT STEP 2B.			
C. IF A SWAP-OUT IS IN PROGRESS, THE SLOTS ALLOCATED TO COMPLETED LSQA FRAMES ARE FREED BY SCANNING THE CAPTURE QUEUE. ALL AIAS ARE REMOVED FROM THE CAPTURE QUEUE AND THE SWAP QUEUE, AND SAVED ON AN INTERNAL QUEUE.	ILRFRSLT ILRFRSLT	ILRFRSL1 ILRFRSW1 TERMSOUT	
D. SET LOCAL FLAG INDICATING THAT LSQA IS AVAILABLE.			
E. THE SEGMENT TABLE IS VALIDITY CHECKED TO PREVENT ANY OBVIOUS ERRORS. IF IT IS INVALID, LSQA IS MARKED UNAVAILABLE AND PROCESSING CONTINUES AT STEP 2B.			
F. THE LOCAL I/O QUEUE IS SCANNED. PROCESSING IS IDENTICAL WITH STEP 2B. AN ADDITIONAL SCAN OF THE QUEUE IS MADE IF LSQA IS AVAILABLE. XPTVALID IS TURNED OFF FOR INCOMPLETE, NON-VIO, NON-LSQA WORK NOT MARKED TO BE FREED BECAUSE THE SLOT WILL BE FREED WHEN THE OPERATION COMPLETES. THIS PREVENTS ILRTERMR FROM FREEING THE SLOT IN LATER PROCESSING.		TERMAIA1 TERMAIA2	

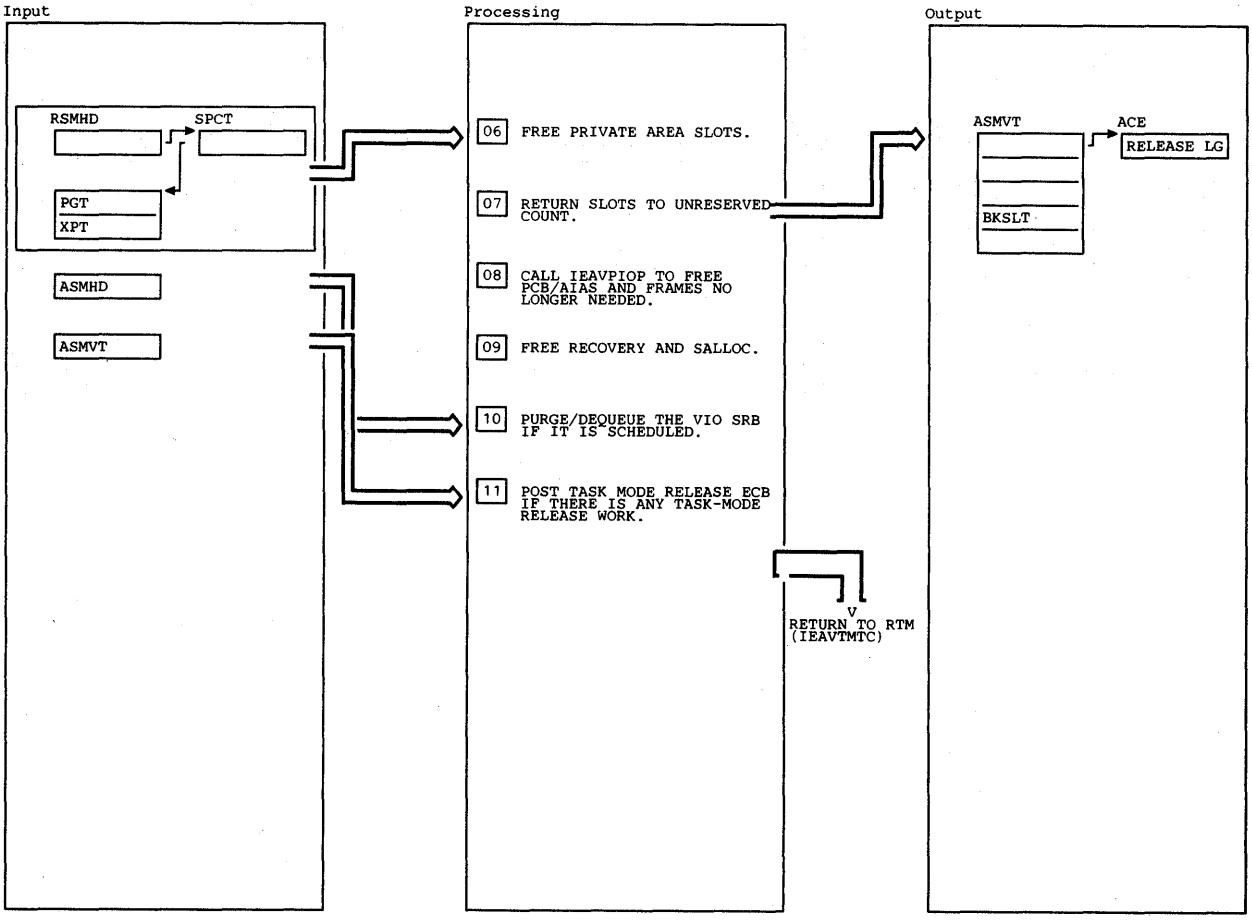
Diagram 25.28 ILRTERMR (Part1 of 3)



Notes	Routine	Label	Ref
04 IF LSQA IS AVAILABLE, ASPCT PROCESSING IS DONE. ALL LGE'S QUEUED FROM THE ASMHD ARE PROCESSED. VIO SLOTS ARE FREED. THESE ARE UNSAVED SLOTS, OR SLOTS RELEASED AFTER SAVE.	ILRFRSLT ILRALS00	ILRFRSL1	
05 IN PROCESS VIO WORK IS CLEANED UP AND GLOBAL RESOURCES ARE FREED. ALL LGE'S QUEUED FROM THE ASMHD ARE PROCESSED.		TERMVIO	
A. ALL WORK ELEMENTS ARE DEQUEUED FROM THE LGE PROCESS QUEUES. AIAS WHICH ARE FOR UNSTARTED WORK (AIA NOT IN PROCESS) ARE SAVED ON AN INTERNAL QUEUE SINCE THEY WILL NEVER BE FREED ANY OTHER WAY.			
B. ALL ACES ARE DEQUEUED AND RETURNED TO THE POOL EXCEPT FOR RELEASE LG ACES FOR WHICH TASK MODE RELEASE PROCESSING IS NEEDED.	ILRGMA		
C. RELEASE LG ACES NEEDING TASK MODE RELEASE PROCESSING ARE PLACED ON THE TASK MODE RELEASE QUEUE ANCHORED IN THE ASMVT. TASK MODE RELEASE PROCESSING IS NOT POSTED AT THIS TIME. THE RELEASE PROCESSING WILL OCCUR WHEN TASK MODE RELEASE PROCESSING IS POSTED LATER.			
D. THE LGES AND SRB USED FOR VIO ARE FREED. LGVTES ARE MADE AVAILABLE BY QUEUEING THEM TO THE AVAILABLE QUEUE IN THE LGVT.			

Notes	Routine	Label	Ref

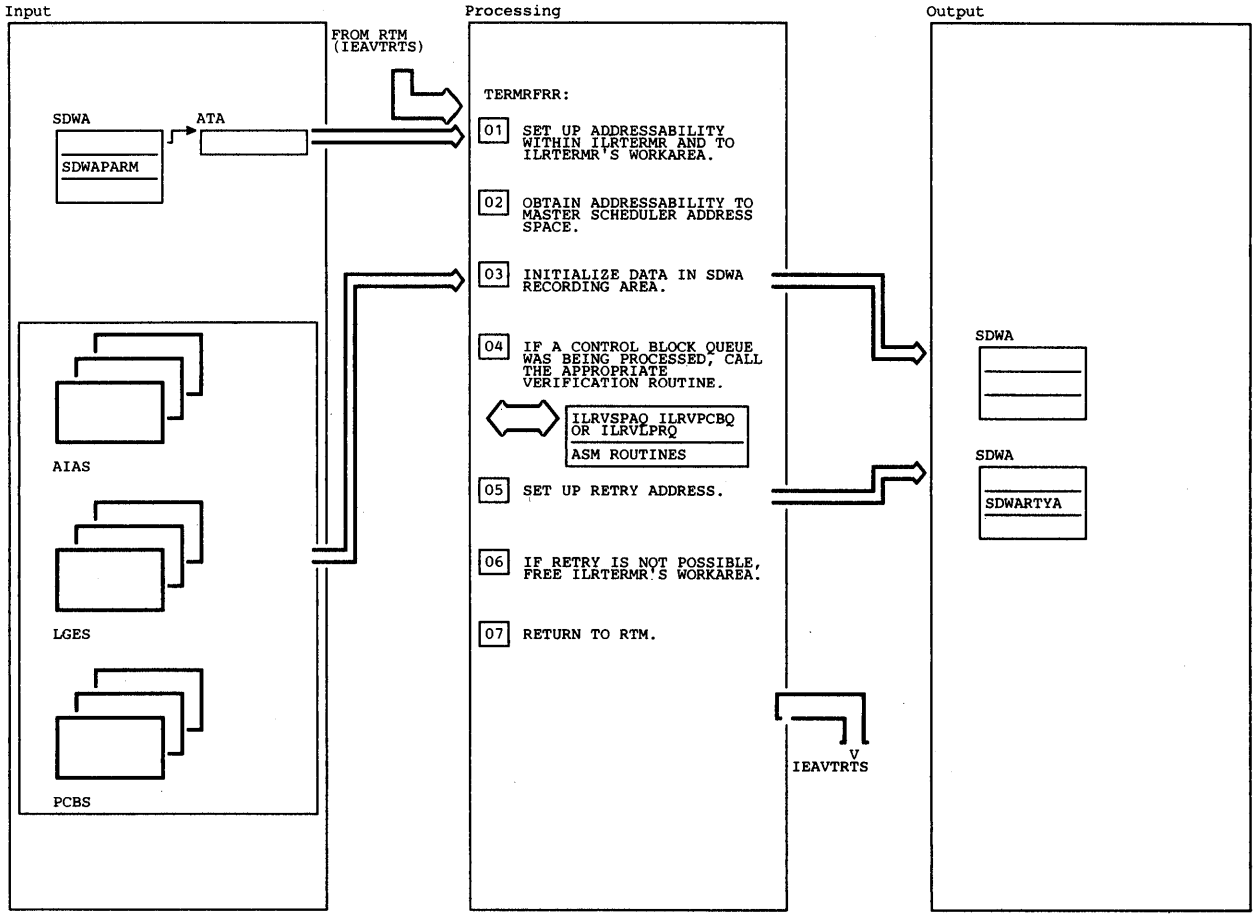
Diagram 25.28 ILRTERMR (Part 2 of 3)



Notes	Routine	Label	Ref
06 FOR ABNORMAL TERMINATION WITH LSQA AVAILABLE, PRIVATE AREA SLOTS WHICH ARE STILL ALLOCATED ARE FREED. THIS IS DONE BY USING THE SPCT SEGMENT ENTRIES TO FIND ALL THE VALID PAGE TABLES AND EXTERNAL PAGE TABLES, AND FREEING ALL SLOTS WHICH ARE STILL MARKED VALID.	ILRFRSLT	ILRFRSL1 TERMPA	
07 THE UNRESERVED SLOT COUNT IN THE ASMVT IS UPDATED.			
08 ALL PCB/AIAS (AND THEIR ASSOCIATED FRAMES) WHICH ARE NO LONGER NEEDED ARE FREED. THESE INCLUDE THE LSQA SWAP-OUT PCB/AIAS WHICH HAVE COMPLETED, AND VIO WORK WHICH HAS NOT BEEN STARTED. ALL IN-PROCESS WORK WILL BE FREED AS THEY COMPLETE BECAUSE OF THE FLAGGING OF THE AIAS.	IEAVPIOP		
09 THE FRR IS DELETED AND THE SALLOC IS RELEASED.			
10 THE VIO SRB IS PURGE/DEQUEUED IF IT HAS BEEN SCHEDULED. THIS IS NECESSARY SINCE THE STORAGE CANNOT BE FREED IF THE SRB IS ON A DISPATCHER QUEUE. THE SRB STORAGE WILL BE FREED BY IEAVDLAS.			
11 IF THERE IS ANY WORK ON THE TASK MODE RELEASE REQUEST QUEUE IN THE ASMVT, THE TASK MODE RELEASE PROCESSOR ECB IN THE ASMVT IS POSTED.			

Notes	Routine	Label	Ref

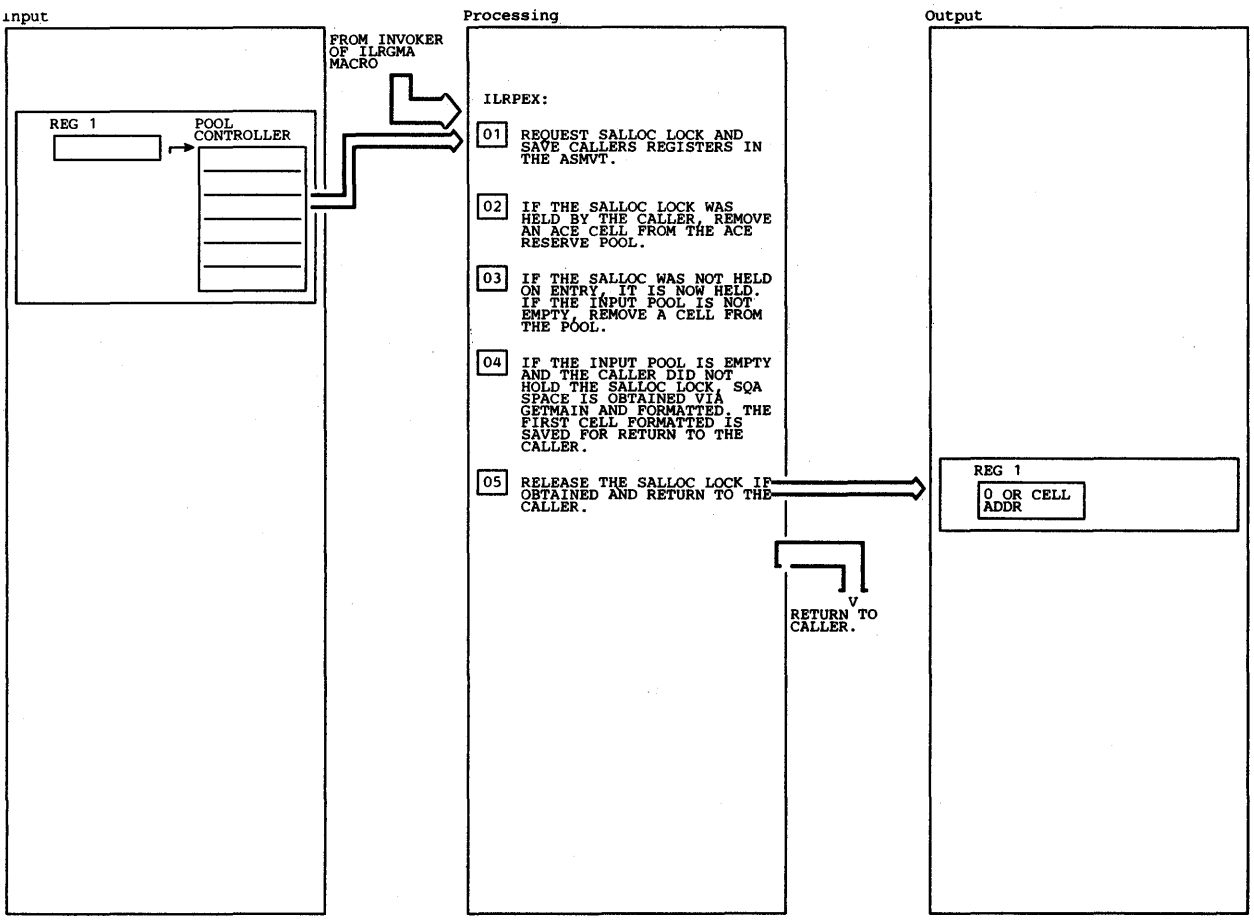
Diagram 25.28 ILRTERMR (Part 3 of 3)



Notes	Routine	Label	Ref
01 REGISTER 1 CONTAINS THE ADDRESS OF THE SDWA WHICH CONTAINS THE ADDRESS OF THE ATA. THE ATA CONTAINS OTHER NEEDED ADDRESSES (RMPL, WORKAREA).			
02 TRAS IS NECESSARY TO GET ADDRESSABILITY TO THE CORRECT ADDRESS SPACE TO RECOVER.			
03 THE FAILING CSECT NAME IS PUT IN THE SDWA.			
04 THE ROUTINES THAT COULD BE CALLED ARE: ILRVSPAQ, ILRVPCBQ, AND ILRVLPRQ.	ILRFRR01	ILRVSPAQ	
	ILRFRR01	ILRVPCBQ	
	ILRFRR01	ILRVLPRQ	
05 THE ADDRESS AT WHICH TO RETRY IS IN ILRTERMR'S WORKAREA.			
06 RETRY IS NOT POSSIBLE IF SO INDICATED IN THE SDWA (SDWARCDE).			
07 IF RETRY IS POSSIBLE, ILRTERMR WILL GET CONTROL AT ITS NEXT RETRY POINT.			

Notes	Routine	Label	Ref

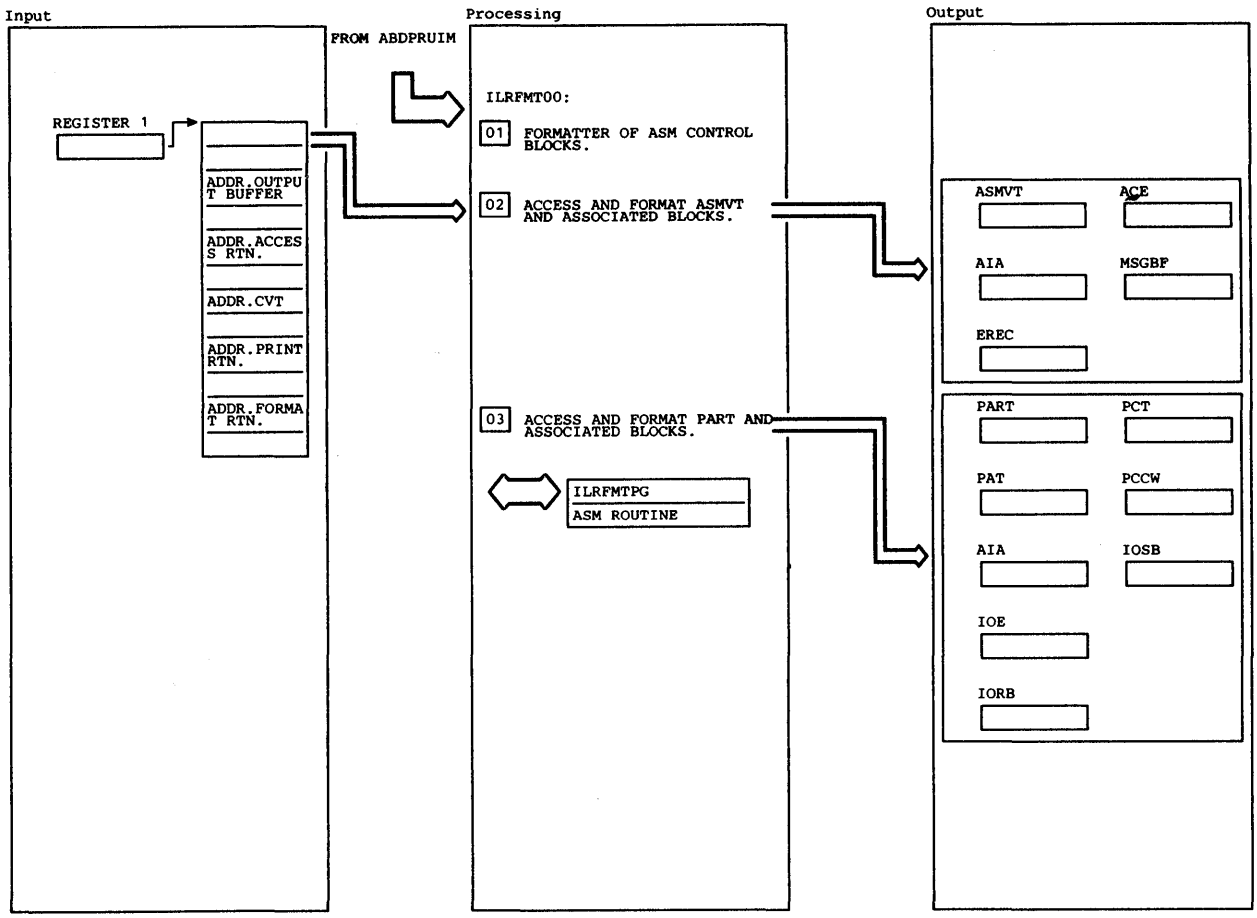
Diagram 25.28.1 TERMRFRR (Part 1 of 1)



Notes	Routine	Label	Ref
01 THE ASM POOL EXTENDER, ILRPEX, RECEIVES CONTROL FROM ANY ASM ROUTINE INVOKING THE ILRGMA MACRO WHEN THE POOL BEING PROCESSED IS EMPTY. ILRPEX WILL BE CALLED ONLY FOR EXPANDABLE POOLS. INPUT REGISTERS 13 AND 14 MUST BE SAVED ACROSS THE SETLOCK FOR THE SALLOC LOCK. ONCE THE SALLOC LOCK IS HELD, THE CALLER'S REGISTERS ARE SAVED IN THE ASMVT WORK SAVE AREA RESERVED FOR ILRPEX.			
02 THE RETURN CODE FROM THE SETLOCK REQUEST IS NON-ZERO IF THE CALLER HELD SALLOC ON ENTRY. THE ONLY POOL FROM WHICH CELLS CAN BE REQUESTED HOLDING SALLOC IS THE ACE POOL. IF THE ACE POOL CONTROLLER ADDRESS WAS PASSED AS INPUT, A CELL IS REMOVED TO BE RETURNED TO THE CALLER. IF THE POOL WAS EMPTY OR THE ACE POOL WAS NOT PASSED AS INPUT, A ZERO CELL ADDRESS IS RETURNED.			
03 THE SALLOC LOCK WAS OBTAINED BY ILRPEX IF THE SETLOCK RETURN CODE WAS ZERO. IN THIS CASE, THE POOL MUST BE CHECKED TO SEE IF IT HAD ALREADY BEEN EXPANDED BY ILRPEX RUNNING ON ANOTHER CPU. IF SO, THE POOL IS NOT EMPTY AND A CELL IS REMOVED FROM THE POOL TO BE RETURNED TO THE CALLER.			
04 THE AMOUNT OF SQA SPACE TO OBTAIN IS DETERMINED BY THE SIZE AND NUMBER OF CELLS INDICATED IN THE POOL CONTROLLER. IF THE GETMAIN FAILS, A ZERO ADDRESS IS RETURNED TO THE CALLER.	IEAVGM00	GLBRANCH	
05 REGISTERS ARE RESTORED AND THE CELL ADDRESS IS PASSED BACK IN REG 1. THE SALLOC LOCK IS RELEASED ONLY IF THE CALLER DID NOT HOLD IT AT ENTRY TO ILRPEX.			

Notes	Routine	Label	Ref

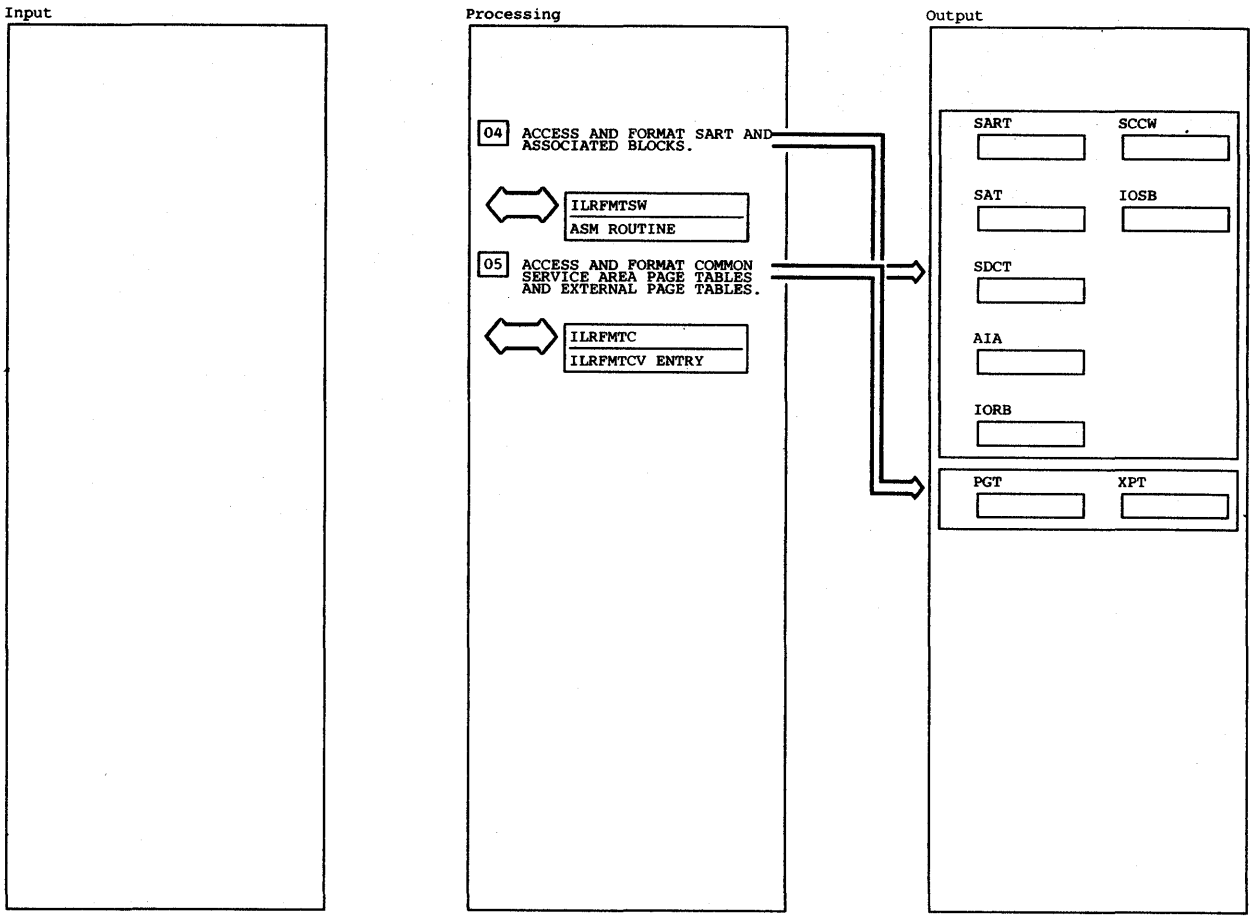
Diagram 25.29 ILRPEX (Part 1 of 1)



Notes	Routine	Label	Ref
01 WHEN PRINT DUMP (AMDPRDMP) PROGRAM IS EXECUTED WITH THE ASMDATA CONTROL STATEMENT INCLUDED, ILRFMT00 IS ENTERED. THE ASM CONTROL BLOCKS AND SOME RSM CONTROL BLOCKS ALSO USED BY ASM ARE FORMATTED. THE CONTROL BLOCK INFORMATION CAN THEN BE EASILY FOUND AND READ.			
02 THE ASMVT IS ACCESSED VIA THE CVTASMVT FIELD. IT IS FORMATTED FOLLOWED BY: A. AIA'S CHAINED FROM ASMSTAGQ. B. THE MESSAGE BUFFER, ASMMSGBF. C. THE BAD SLOT ERROR RECORD, ASMEREC. D. ACE'S CHAINED FROM ASMRLGRQ AND ASMRLGWQ. THEN THE OTHER MODULES ARE CALLED IN TURN, ILRFMTPG, ILRFMTSW, ILRFMTCV(ILRFMTC), ILRFMTCV (ILRFMTH), ILRFMTCV (ILRFMTV).			
03 THE PART IS ACCESSED FROM ASMPART AND FORMATTED, ALONG WITH: (A) PCTS CHAINED FROM PARTPCTQ, (B) ERROR AIA'S FROM PARTAIAE AND (C) IOES FROM PARTCOMQ, PARTSPLQ, PARTDUPQ, PARTLOCQ. NEXT, EACH PART ENTRY IS FORMATTED ALONG WITH: (A) PAT FROM PAREPATP, (B) IOES FROM PAREIOE, AND (C) IORBS, IOSBS AND PCCWS AND AIAS FOUND VIA PAREIORB.	ILRFMTPG	ILRFMTPG	

Notes	Routine	Label	Ref

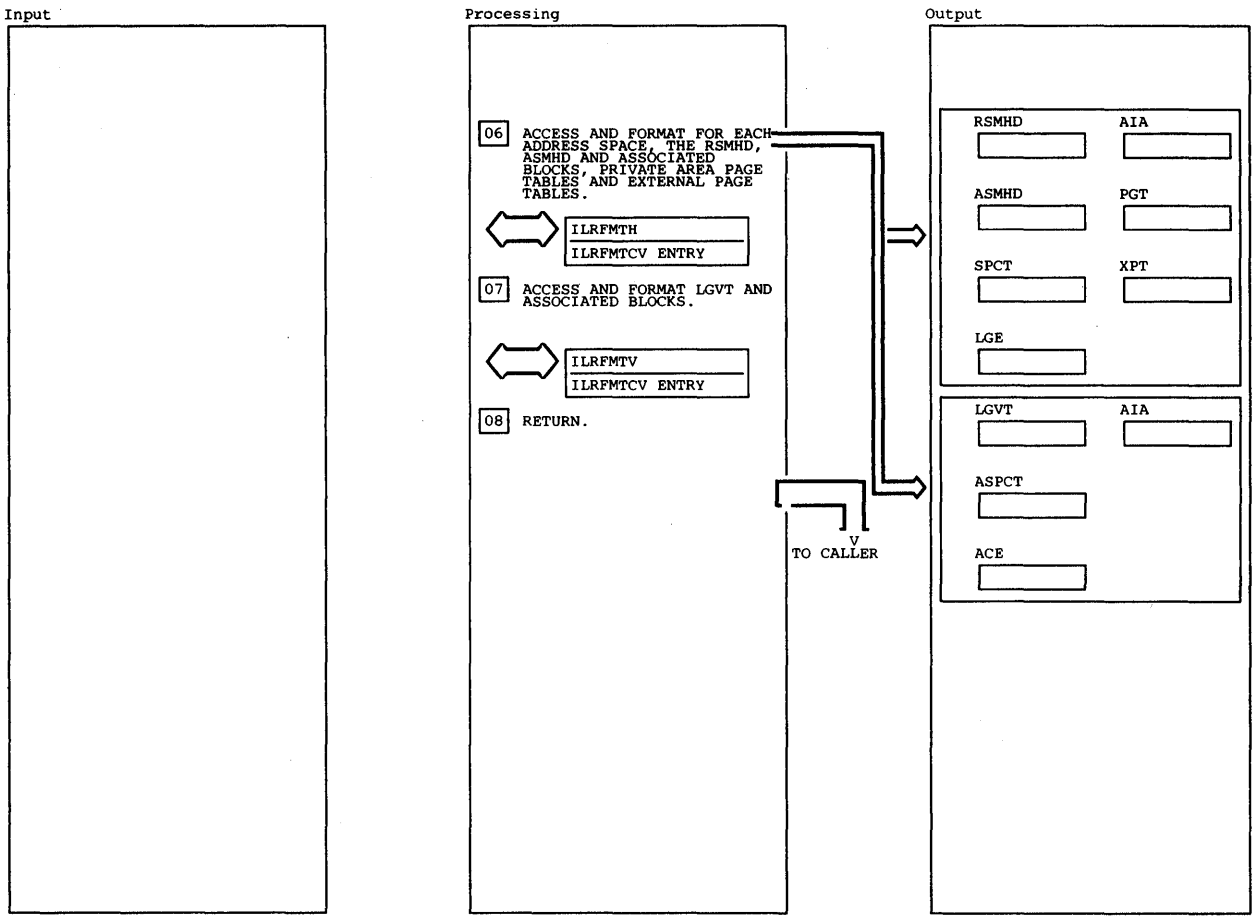
Diagram 25.30 ILRFMT00 (Part 1 of 3)



Notes	Routine	Label	Ref
04 THE SART IS ACCESSED FROM ASMSART AND FORMATTED ALONG WITH AIAS FROM SARWAITQ, AND THE SDCT AND ITS ENTRIES. EACH SART ENTRY IS FORMATTED WITH ITS ASSOCIATED: A. SAT FROM SRESAT B. SCCW FROM SRESCCW AND. C. IORBS, IOSBS, SCCWS AND AIAS FROM SREIORB.	ILRFMTSW	ILRFMTSW	
05 THE COMMON SERVICE AREA PAGE AND EXTERNAL PAGE TABLES ARE ACCESSED VIA THE MASTER SCHEDULER SEGMENT TABLE (PSASTOR). EACH PAGE TABLE (PGT) AND ITS ASSOCIATED EXTERNAL PAGE TABLE (XPT) ARE FORMATTED.	ILRFMTCV	ILRFMTC	

Notes	Routine	Label	Ref

Diagram 25.30 ILRFMT00 (Part 2 of 3)



Notes	Routine	Label	Ref
06 THE RSMHD IS ACCESSED VIA CVTASVT AND ASCBRSM. IT AND THE SPCT ARE FORMATTED. THEN THE ASMHD IS FORMATTED WITH: A. AIAS FROM ASHSWAPQ AND ASHCAPQ. B. PGT/XPTS FOR THE PRIVATE AREA OF THAT ADDRESS SPACE VIA SPCTPGT.	ILRFMTCV	ILRFMTH	
07 THE LGVT IS ACCESSED VIA ASMLGVT. THE LGVT ENTRY FOR THAT ADDRESS SPACE IS LOCATED AND FORMATTED WITH THE ASSOCIATED: A. LGE FROM LGVELGEP. B. AIAS FROM LGEPROCQ AND C. ASPCT FROM LGEASPCT AND LPME'S AND/OR ASSTS, IF PRESENT. ALSO AIA/ACE'S CHAINED FROM AIACEPTR.	ILRFMTCV	ILRFMTV	
08 RETURN TO CALLER.			

Notes	Routine	Label	Ref

Diagram 25.30 ILRFMT00 (Part 3 of 3)

Page Expansion

The dynamic page expansion facility (ILRPGEXP) allows the system operator to add page or swap data sets to the system by entering the PAGEADD command. The number of page or swap data sets that can be added throughout one ILP is limited to the number specified by the PAGNUM system parameter at IPL time.

Control Blocks Used

The major control blocks ILRPGEXP uses are:

- ASMVT — Auxiliary Storage Management Vector Table
- PART — Page Activity Reference Table
- PAT — Page Allocation Table
- SART — Swap Activity Reference Table
- SAT — Swap Allocation Table
- Data Set Name List
- ILRTPARB— TPARTBLE
- PCT — Performance Characteristics Table

The ASMVT resides in the nucleus and is ASM's extension of the CVT. It contains a count of available slots and back slots that are changed when slots are allocated and page data sets are added. The ASMVT also contains an indicator that specifies whether TPARTBLE is valid or not (ASMNOTPT).

The PART resides in SQA and consists of one header and an entry (PARTE) for each page data set that is open and for the number of entries required to support page expansion. When a page data set is added, ILRPGEXP updates an empty PARTE and chains it to the others.

The PAT resides in SQA and contains a bit map of allocated and unallocated page slots.

The SART resides in SQA and consists of a header and an entry (SARTE) for each swap data set and additional entries for the number of swap sets that may be added.

The SAT resides in SQA and describes the allocated and available slots for a swap data set.

The Data Set name lists reside in CSA and contain lists of swap and page data set names that are currently in use; they also have additional entries for page and swap data sets to be added by page expansion.

ILRTPARB (TPARTBLE) resides on the PLPA page data set. It is built during IPL, contains the page data set information that is used when quick or warm starting. ILRPGEXP updates ILRTPARB when a new page data set is added.

Processing

The Master Scheduler attaches ILRPGEXP when the operator issues the PAGEADD command, passing the command text in a CSCB (Command Scheduling control block). ILRPGEXP loads the read/write routine (ILRPREAD) and the open routine (ILROPS00), establishes an ESTAE, syntax checks the command, and then processes the page or swap data set request(s). If there is more than one request the process is repeated as many times as necessary.

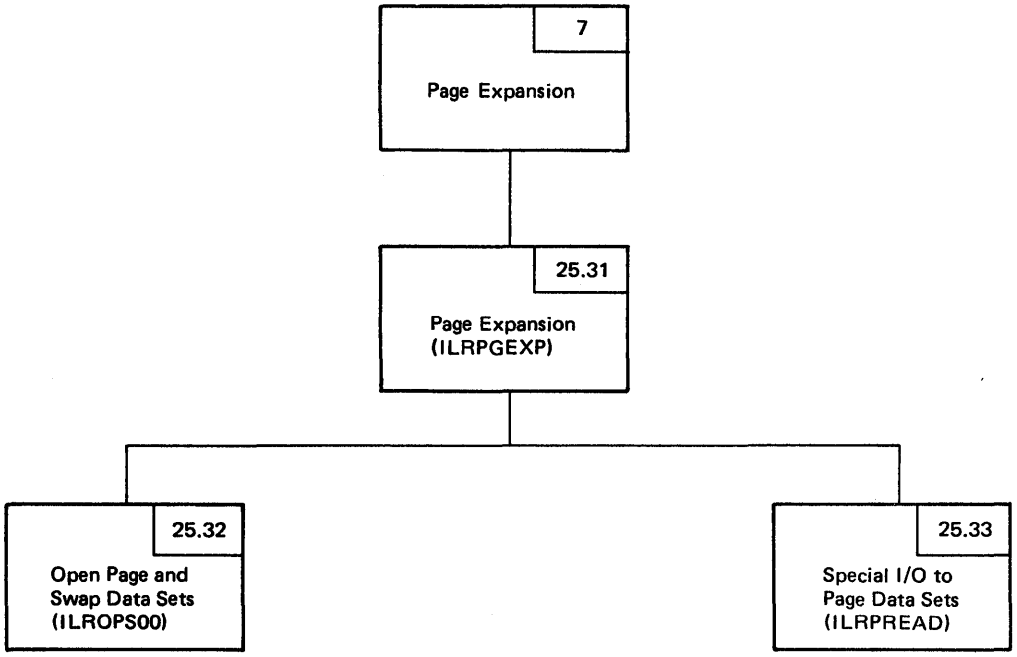
Page Data Sets

ILRPGEXP calls ILRPREAD to read the TPARTBLE and calls ILROPS00 to open the page data set.

ILRPGEXP then updates the following control blocks. If the page data set being added is of a different device type than the existing page data sets, ILRPGEXP builds a PCT. Then it finds an empty PARTE, fills it in, and chains it to the others. Next ILRPGEXP gets storage for and initializes a PAT to reflect the available slots on the new page data set. Finally, ILRPGEXP updates the TPARTBLE, the data set name list, increases the slot and back slot counts in ASMVT, and calls ILRPREAD to write the TPARTBLE back to the PLPA data set.

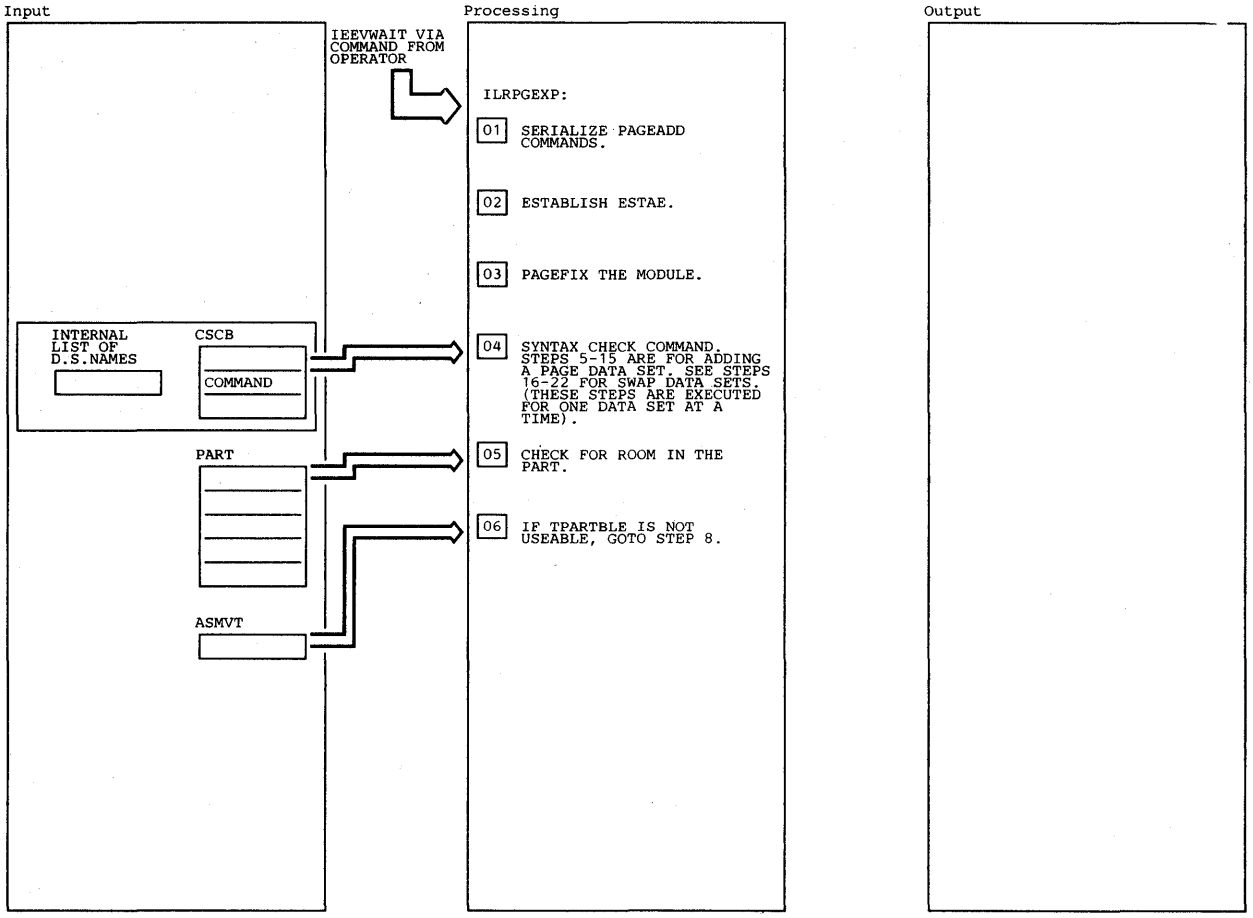
Swap Data Sets

ILRPGEXP calls ILROPS00 to open the data set, fills in and chains a SARTE, updates the SART header, builds a SAT to reflect available swap space, and updates the data set name list.



25.x. – Module
25.x.y. – Entry point in module 25.x.

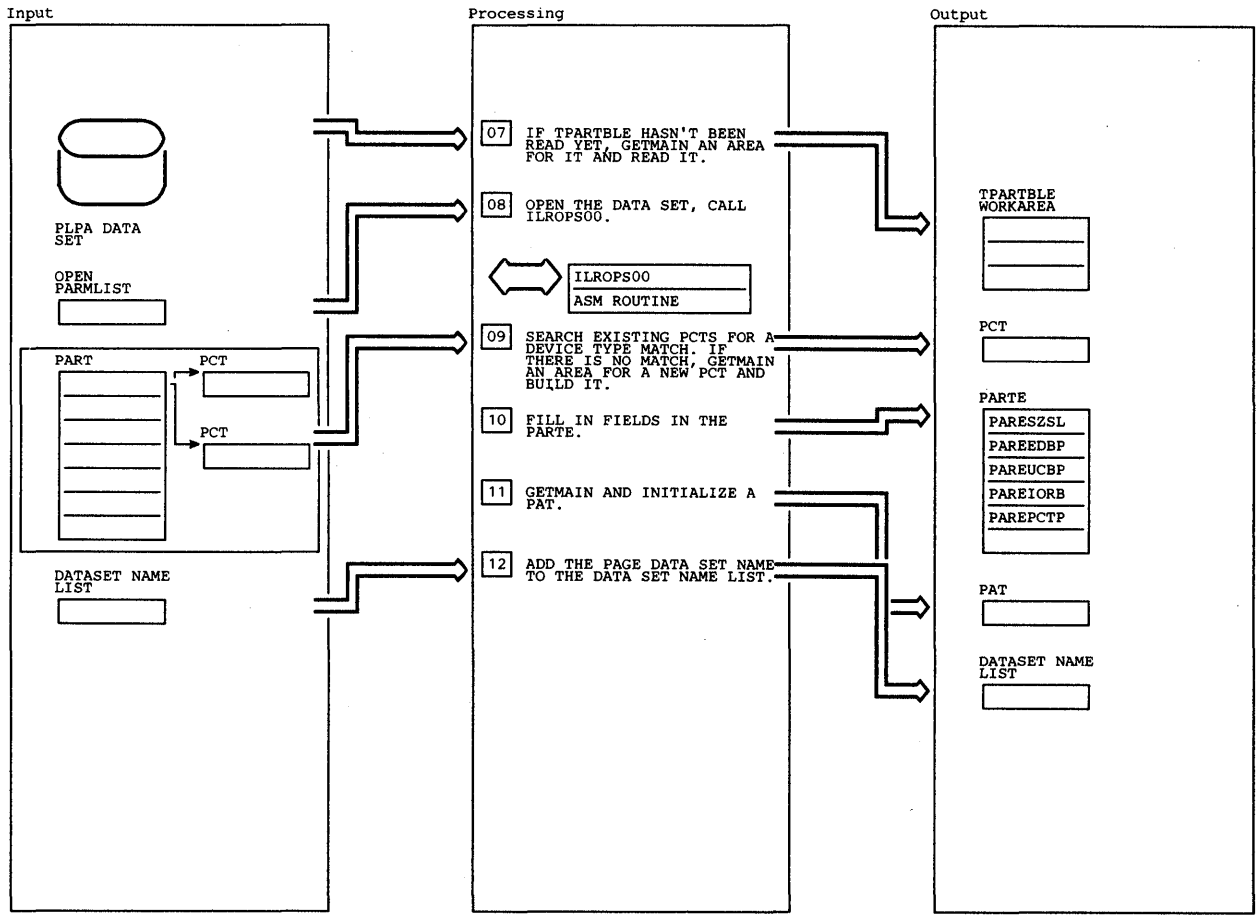
Figure 2-63. Page Expansion Overview



Notes	Routine	Label	Ref
01 ILRPGEXP (ALIAS IEEPGEXP) IS ENTERED WHENEVER THE PAGEADD COMMAND IS ISSUED BY THE OPERATOR. ITS PURPOSE IS TO ADD PAGE DATA SET(S) OR SWAP DATA SET(S) TO THE SYSTEM. THE ENQ MACRO IS USED TO KEEP SUBSEQUENT PAGEADD COMMANDS FROM EXECUTING BEFORE THIS ONE HAS COMPLETED.			
02 ILRPGEXP SETS UP THE INTERFACE TO IEECB860, THE MASTER SCHEDULER USER ESTAE ROUTINE AND THEN ESTABLISHES ITS OWN ESTAE, ESTAER (ANOTHER ENTRY IN ILRPGEXP).			
03 A PAGEFIX IS NEEDED IN ORDER TO OBTAIN THE SALLOC LOCK FOR SERIALIZATION OF CONTROL BLOCKS.			
04 THE DATA SET NAMES ARE CHECKED FOR THE CORRECT LENGTH AND FOR DUPLICATES. IF THE LENGTH IS INCORRECT OR IF THERE ARE DUPLICATES, THE OPERATOR IS NOTIFIED.			
05 THERE IS ONLY ROOM IN THE PART TO PROCESS THE NUMBER OF PAGE DATA SETS SPECIFIED ON THE PAGNUM SYSTEM PARAMETER.			
06 IF TPARTBLE IS UNUSEABLE, (ASMNOTPT=1), THEN IT IS NOT USED OR UPDATED.			

Notes	Routine	Label	Ref

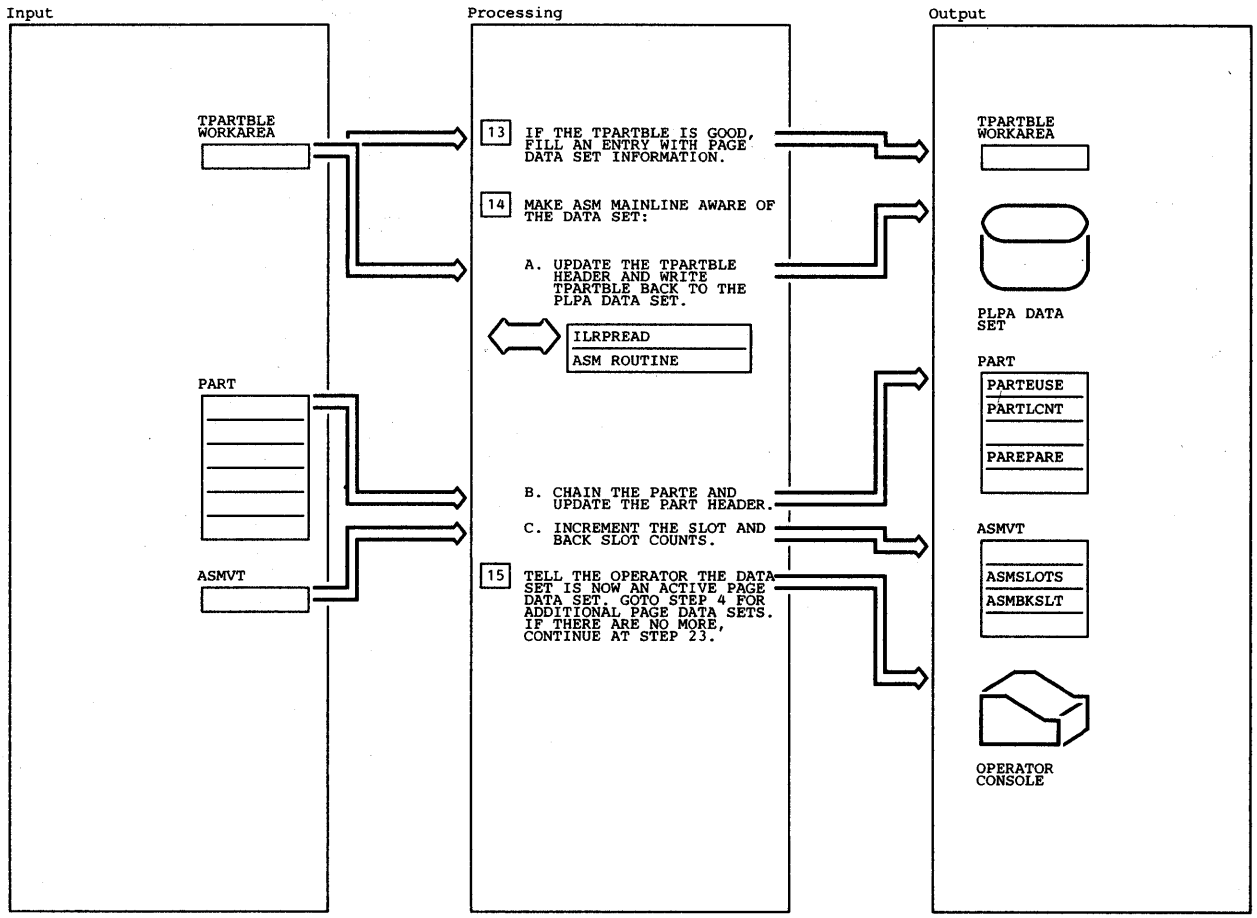
Diagram 25.31 ILRPGEXP (Part 1 of 4)



Notes	Routine	Label	Ref
07 IF THE GETMAIN OR READ FOR TPARTBLE FAILS, ASK THE OPERATOR IF HE WANTS TO CONTINUE.	ILRPREAD	ILRPREAD	
08 IF SUCCESSFUL, ILROPS00 WILL RETURN WITH THE NUMBER OF SLOTS IN THE DATA SET AND ADDRESSES OF THE IORB, EDB, AND UCB CONTROL BLOCKS. FOR MOUNT, GETMAIN AND LOCATE ERRORS ON OPEN PROCESSING, THE REQUEST FOR THIS DATA SET IS FAILED. THE OPERATOR IS NOTIFIED.	ILROPS00	ILROPS00	
09 A DIFFERENT PCT IS NEEDED FOR EACH OPEN PAGE DATA SET DEVICE TYPE. IF THE NEW PAGE DATASET IS ON A DIFFERENT DEVICE FROM THE EXISTING ONES A NEW PCT IS NEEDED. IF THE GETMAIN FOR THE PCT FAILS THE OPERATOR IS TOLD AND THIS PAGE DATA SET REQUEST IS FAILED.			
10 ADDRESSES OF CONTROL BLOCKS AND THE NUMBER OF SLOTS FOR THIS DATA SET RETURNED FROM ILROPS00 ARE PUT INTO THE PARTE. THE ADDRESS OF THE PCT IS ALSO INCLUDED.			
11 IF THE GETMAIN FOR THE PAT FAILS, NOTIFY THE OPERATOR AND FAIL THIS DATA SET REQUEST.			
12 THE DATA SET NAME LIST IS USED FOR CHECKING NEW DATA SET NAMES AGAINST EXISTING ONES.			

Notes	Routine	Label	Ref

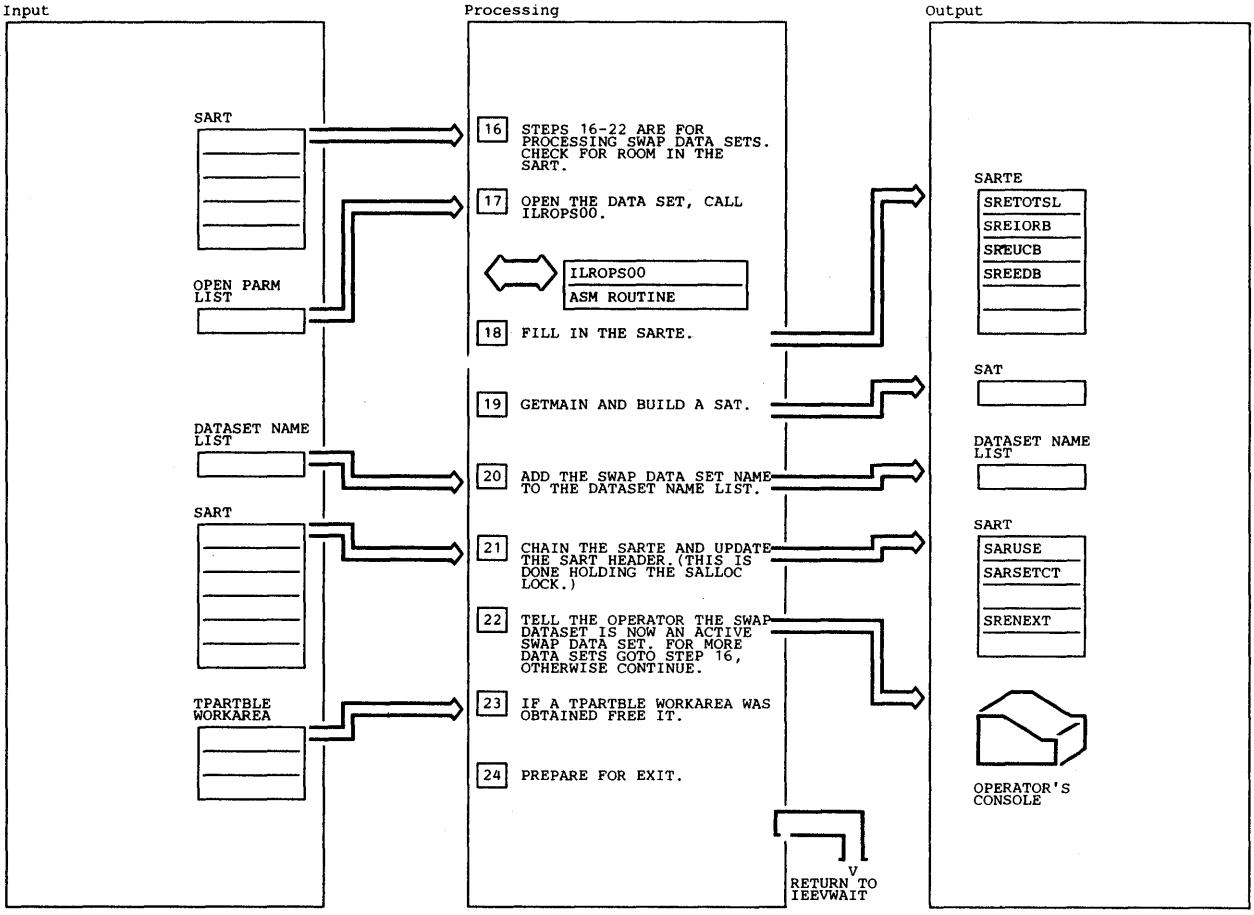
Diagram 25.31 ILRPGEXP (Part 2 of 4)



Notes	Routine	Label	Ref
13 IF TPARTBLE IS NOT GOOD (ASMNOTPT=1), IT IS NOT UPDATED.			
14 AFTER THE DATA SET IS OPENED AND COMPLETELY INITIALIZED IT CAN BE USED BY ASM.			
A. IF THE WRITE OF TPARTBLE FAILS, THE OPERATOR IS ASKED IF HE WANTS TO CONTINUE.	ILRPREAD	ILRPREAD	
B. THE SALLOC LOCK IS OBTAINED TO UPDATE THE COUNT OF LOCAL PAGE DATA SETS (PARTLCNT) AND THEN RELEASED. THE PAGE DATA SETS IN USE COUNT (PARTEUSE) IS INCREMENTED, AND THE PARTE IS CHAINED VIA PAREPARE.			
C. THE TOTAL PAGE SLOTS COUNT IN THE ASMVT IS INCREMENTED. ALSO THE AVAILABLE SLOTS FOR BACKING (ASMBKSLT) AN ADDRESS SPACE OR VIO DATA SET IS INCREMENTED.			

Notes	Routine	Label	Ref

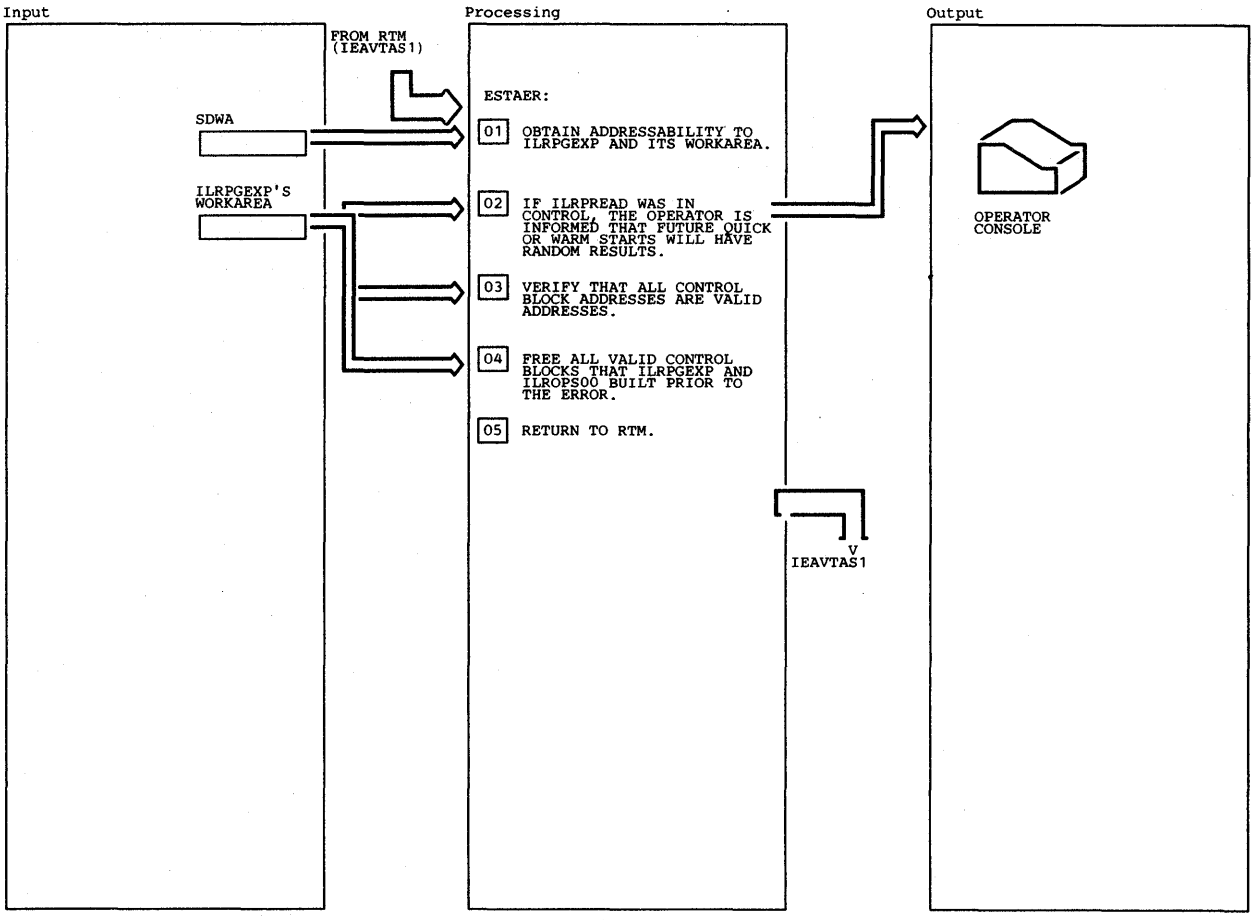
Diagram 25.31 ILRPGEXP (Part 3 of 4)



Notes	Routine	Label	Ref
16 THERE IS ONLY ROOM IN THE SART TO PROCESS THE NUMBER OF SWAP DATA SETS SPECIFIED ON THE PAGNUM SYSTEM PARAMETER.			
17 IF SUCCESSFUL, ILROPS00 RETURNS WITH THE NUMBER OF SLOTS ON THIS DATA SET AND ADDRESSES OF THE IORB, EDB, AND UCB CONTROL BLOCKS. UPON RETURN, THE NUMBER OF SLOTS IS CONVERTED TO THE NUMBER OF SWAP SETS. FOR MOUNT GETMAIN AND LOCATE ERRORS ON OPEN, THE OPERATOR IS TOLD AND PROCESSING FOR THIS DATASET IS FAILED.	ILROPS00	ILROPS00	
18 ADDRESSES OF CONTROL BLOCKS RETURNED BY ILROPS00 ARE PUT INTO THE SART ENTRY. ALSO PUT IN THE ENTRY IS THE NUMBER OF SWAP SETS (SRETOTSL).			
19 IF THE GETMAIN FOR THE SAT FAILS, THE OPERATOR IS TOLD AND PROCESSING FOR THIS DATASET IS FAILED.			
20 THE DATA SET NAME LIST IS USED FOR CHECKING NEW SWAP DATA SETS AGAINST ALREADY EXISTING ONES.			
21 THE SALLOC LOCK IS NEEDED TO KEEP ASM FROM USING THE SART WHILE IT IS BEING UPDATED.			
24 BEFORE RETURNING CONTROL, ILRPGEXP REMOVES ITS ESTAE FOR RECOVERY, DEQUEUES FROM THE PAGEADD RESOURCE AND ISSUES A PGFREE MACRO.			

Notes	Routine	Label	Ref

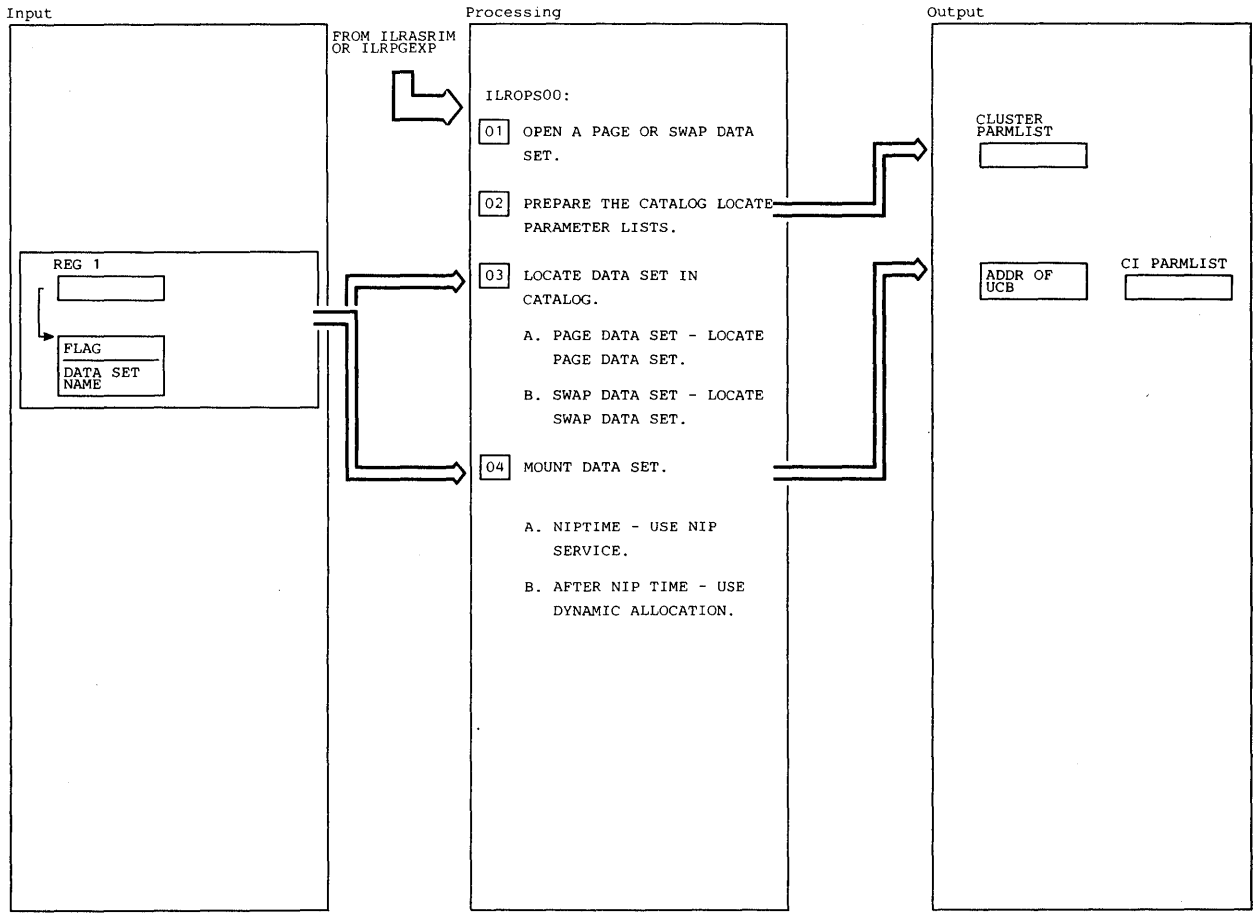
Diagram 25.31 ILRPGEXP (Part 4 of 4)



Notes	Routine	Label	Ref
01 ESTAER IS PHYSICALLY CONTAINED WITHIN ILRPGEXP'S FIRST BASE REGISTER.			
02 IF ILRPREAD WAS IN CONTROL, AN ATTEMPT WAS BEING MADE TO READ OR WRITE TPARTBLE. IF TPARTBLE IS BAD, IT MAY NO LONGER BE POSSIBLE TO QUICK OR WARM START. MESSAGE IEE789I IS ISSUED TO INFORM THE OPERATOR OF THE SITUATION.			
03 ALL ADDRESSES OF CONTROL BLOCKS OBTAINED BY ILRPGEXP AND ILROPS00 ARE VALIDITY CHECKED.			
04 THE CONTROL BLOCKS FREED INCLUDE THE PCT, PAT, SAT, EDB, AND IORB. ALSO, THE RELATED CONTROL BLOCK ENTRIES IN THE PART, SART, AND DATA SET NAME LIST ARE CLEARED.			

Notes	Routine	Label	Ref

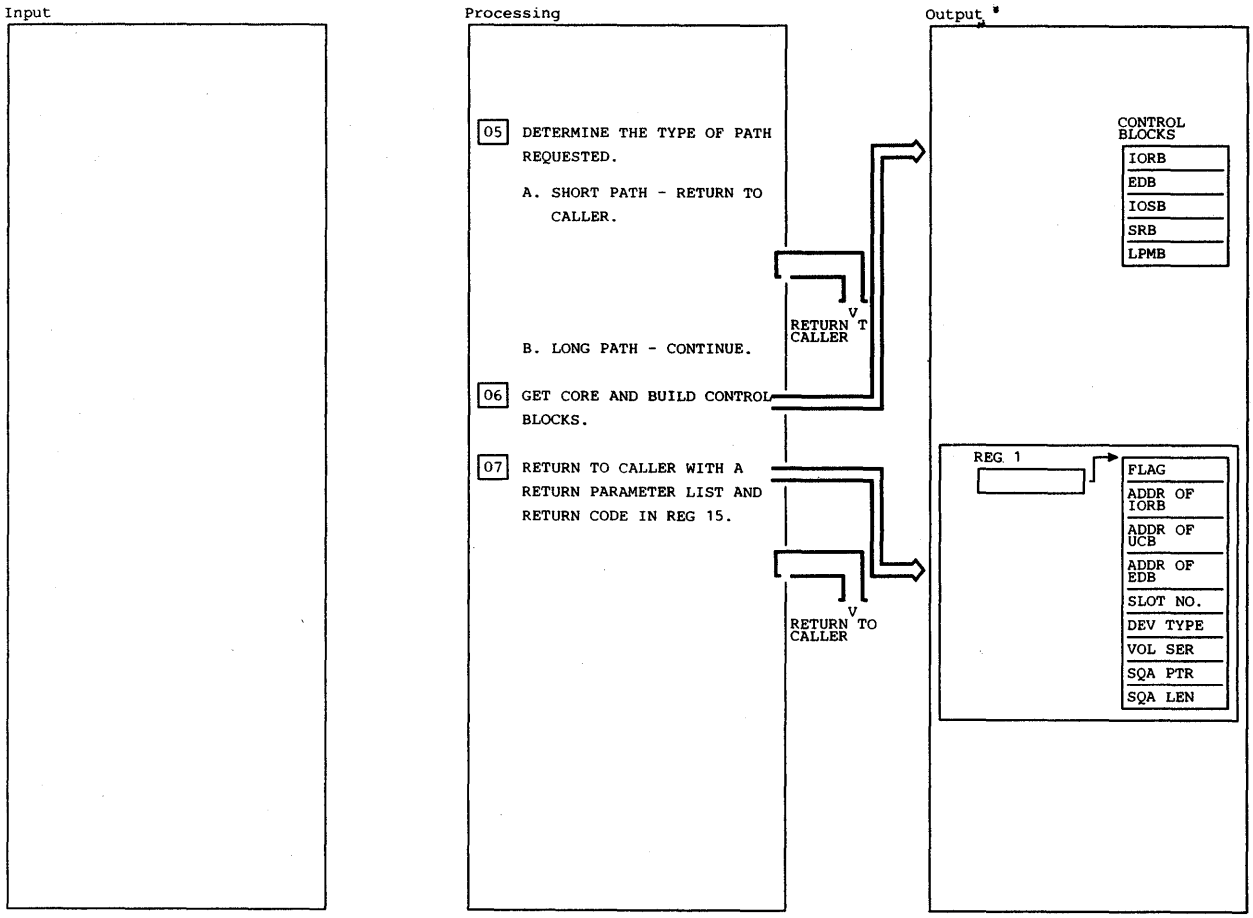
Diagram 25.31.1 ESTAER (Part 1 of 1)



Notes	Routine	Label	Ref
01 ILROPS00 OPENS A PAGE OR SWAP DATA SET FOR ASM'S RIM (ILRASRIM) DURING SYSTEM INITIALIZATION OR FOR PAGE EXPANSION (ILRPGEXP) AFTER IPL.			
02 INITIALIZE THE CATALOG LOCATE PARAMETER LISTS (CTGPL) AND CATALOG FIELD PARAMETER LISTS (CTGFLS) FOR CLUSTER LOCATE AND DATA CONTROL INTERVAL (CI) LOCATE, RESPECTIVELY. INFORMATION, SUCH AS DATA SET TYPE, VOLSER, ATTRIBUTES, DEVICE TYPE AND STATISTICAL DATA WILL BE REQUESTED FROM VSAM CATALOG.			
03 ISSUE THE TWO LOCATES TO LOCATE PAGE OR SWAP DATA SET . CHECK THE RETURN PARAMETER TO INSURE THE REQUESTED DATA SET IS LOCATED, AND CHECK THAT TRACK OVERFLOW IS NOT INDICATED FOR SWAP DATA SET. IF LOCATE FAILS, SET RETURN CODE TO '12' IN REGISTER 15 AND RETURN TO CALLER.		LOCPAGE LOCSWAP	25.32.1 25.32.2

Notes	Routine	Label	Ref
04 MOUNT METHOD IS BASED ON THE TIME REQUESTED. NIPMOUNT IS USED AT NIPTIME,DYNAMIC ALLOCATION IS USED AFTER NIP TIME. IF MOUNT FAILS, RETURN TO CALLER WITH '08' IN REGISTER 15.		VMTVER DYNALLO	25.31.3 25.31.4

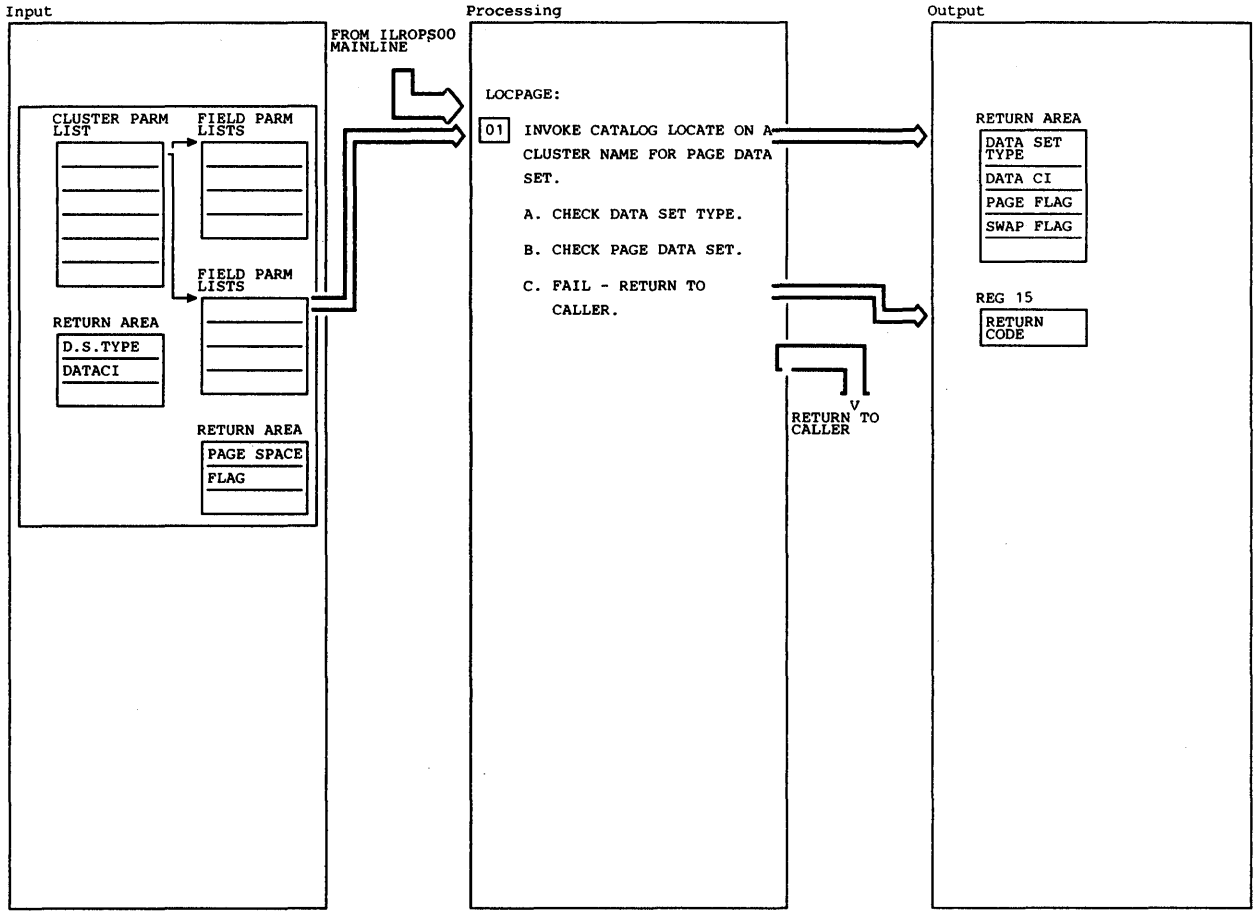
Diagram 25.32 ILROPS00 (Part 1 of 2)



Notes	Routine	Label	Ref
05 SHORT PATH IS ONLY (MEANING CONTROL BLOCKS SHOULD NOT BE BUILT) IS ONLY REQUESTED DURING SYSTEM INITIALIZATION. THE TYPE OF PROCESSING IS BASED ON AN INPUT FLAG. IF A SHORT PATH IS DESIRED, RETURN TO THE CALLER. OTHERWISE CONTINUE PROCESSING.			
06 BUILD LPME, EDB, IORB, SRB, IOSB AND SAVE AREA FOR PAGE OR SWAP DATA SET. IF THE REQUESTED DATA SET IS A PAGE DATA SET AND ON NIPTIME, PCCW'S WILL BE BUILT IN NUCLEUS BUFFER SPACE. IF SQA SPACE IS NOT AVAILABLE FOR THE CONTROL BLOCKS, RETURN CODE IS SET TO '16' ('20' FOR NUCLEUS BUFFER SPACE NOT AVAILABLE).		GETCORE	25.31.5
07 IF ALL ABOVE PROCESSING IS SUCCESSFUL, A RETURN PARAMETER LIST WILL BE SET AND PASSED BACK TO CALLER WITH A RETURN CODE OF ZERO.			

Notes	Routine	Label	Ref

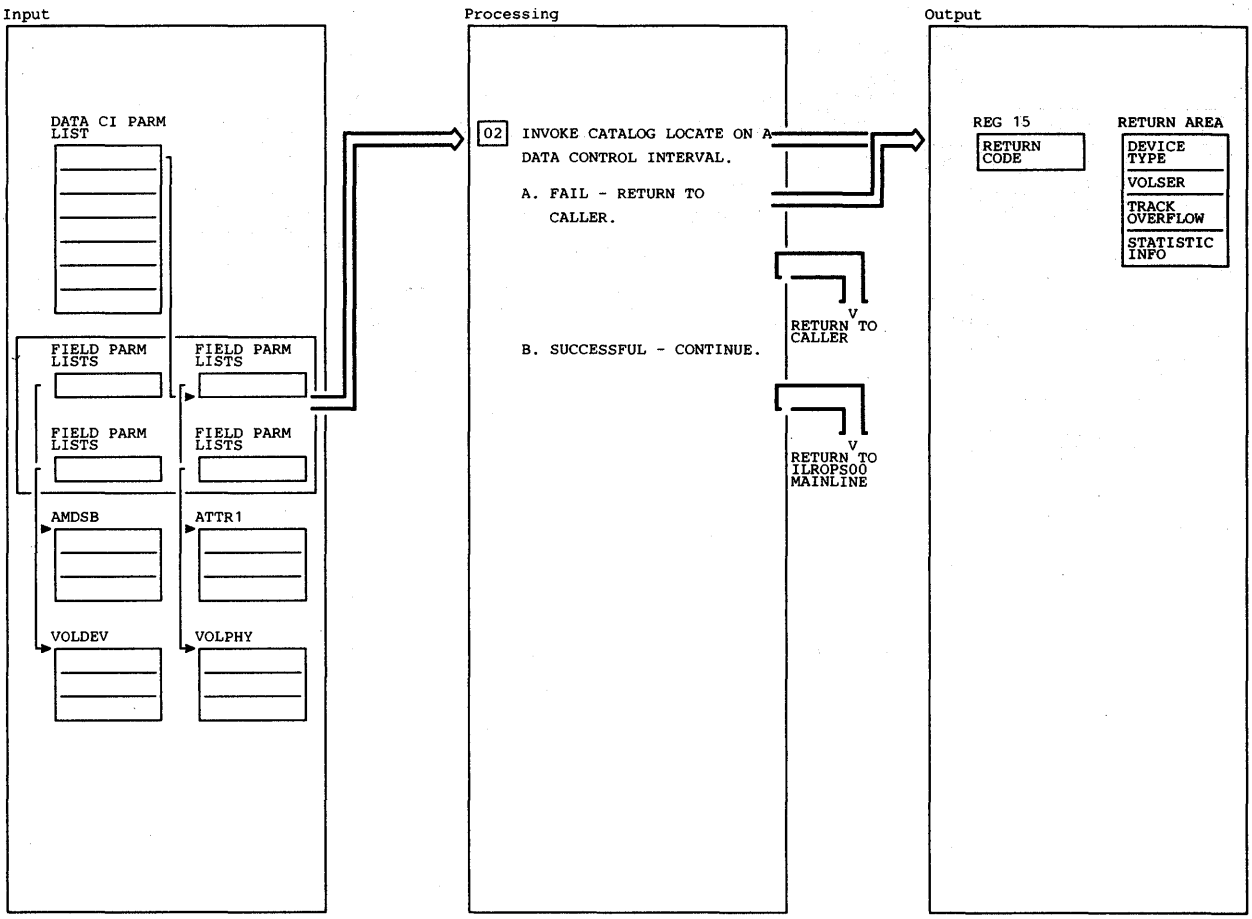
Diagram 25.32 ILROPS 00 (Part 2 of 2)



Notes	Routine	Label	Ref
01 ISSUE A LOCATE ON A CLUSTER NAME REQUESTING NAMEDS AND CATTR INFORMATION. NAMEDS INCLUDES THE DATA SET TYPE('INDEX' OR 'DATA') AND THE DATA CONTROL INTERVAL FOR ALL OTHER INFORMATION ON THIS DATA SET. CATTR CONTAINS A PAGE SPACE FLAG. A. CHECK THAT THE REQUESTED DATA SET TYPE IS ONLY 'DATA'. B. CHECK THAT THE PAGE DATA SET FLAG IS ON AND SWAP FLAG IS OFF. C. IF LOCATE FAILS OR THE DATA SET TYPE IS NOT 'DATA' OR THE DATA SET IS NOT A PAGE DATA SET, THEN SET REGISTER 15 TO '12' AND RETURN TO CALLER.	SVC26		

Notes	Routine	Label	Ref

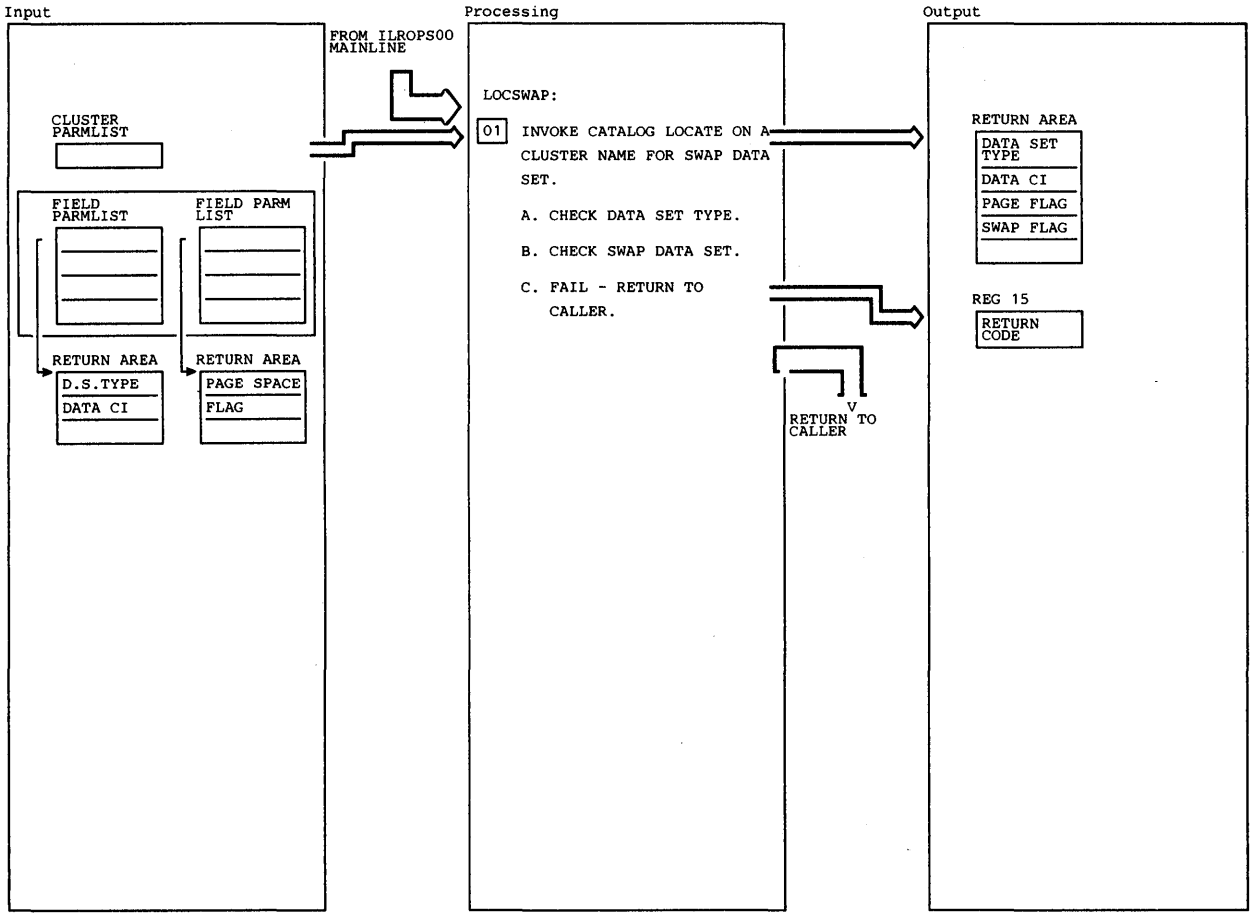
Diagram 25.32.1 LOCPAGE (Part 1 of 2)



Notes	Routine	Label	Ref
02 USING THE DATA CONTROL INTERVAL FROM THE FIRST LOCATE, ISSUE A LOCATE ON A DATA CONTROL INTERVAL, REQUESTING VOLPHV, VOLDEV, AMDSB AND ATTR1 INFORMATION. VOLPHY AND VOLDEV CONTAIN RESPECTIVELY PHYSICAL AND DEVICE RELATED DATA. THE AMDSB CONTAINS CONTROL INTERVAL DATA . ATTR1 CONTAINS A TRACK OVERFLOW FLAG.			
A. IF 2ND LOCATE FAILS, ILROPS00 SETS REGISTER 15 TO 12 AND RETURN TO CALLER.	SVC26		
B. IF SUCCESSFUL, THEN CONTINUE.			

Notes	Routine	Label	Ref

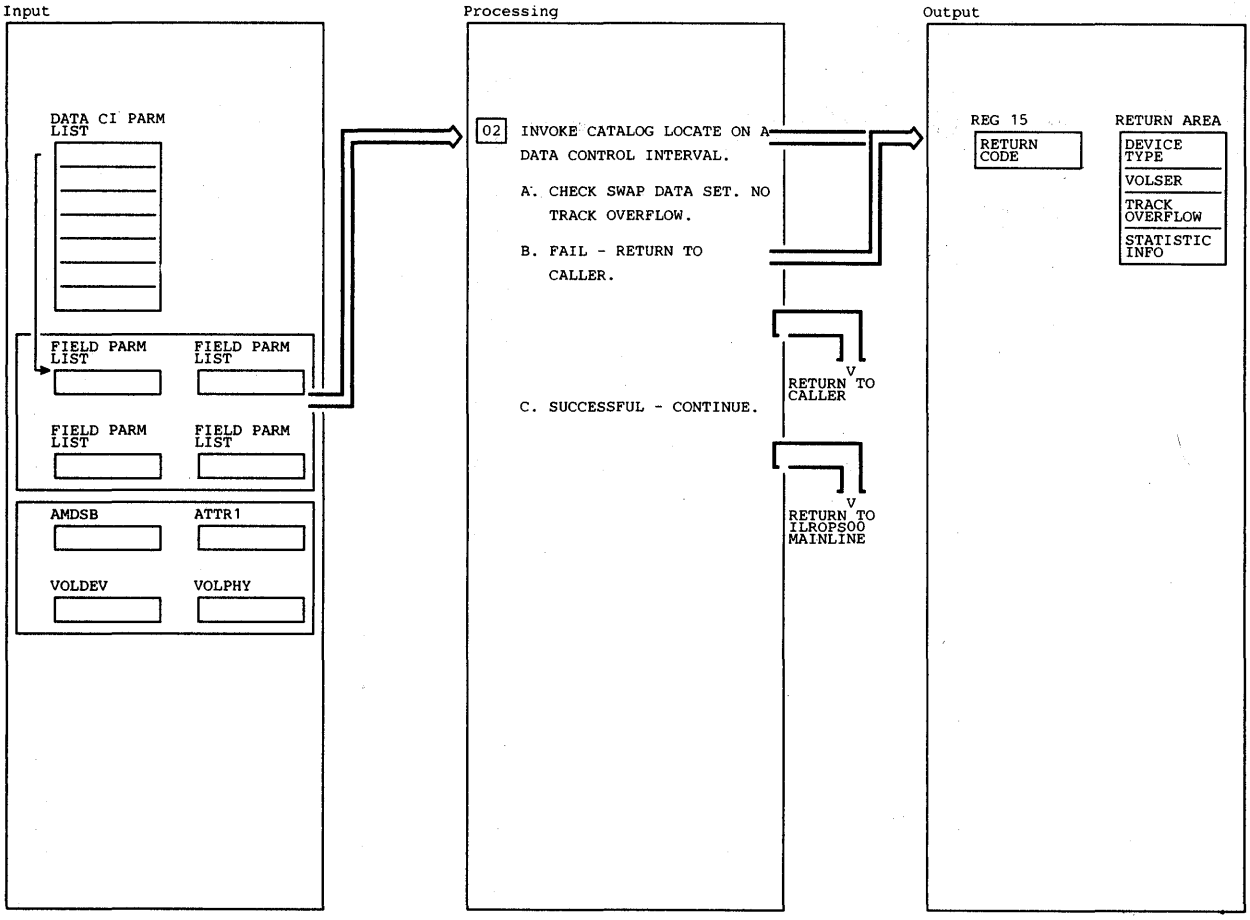
Diagram 25.32.1 LOCPAGE (Part 2 of 2)



Notes	Routine	Label	Ref
01 ISSUE A LOCATE ON A CLUSTER NAME REQUESTING NAMEDS AND CATTR INFORMATION. NAMEDS INCLUDES THE DATA SET TYPE ('INDEX' OR 'DATA') AND THE DATA CONTROL INTERVAL FOR ALL OTHER INFORMATION ON THIS DATA SET. CATTR CONTAINS A SWAP DATA SET FLAG AND PAGE DATA SET FLAG. A. CHECK THAT THE SWAP DATA SET TYPE IS 'DATA'. B. CHECK THAT THE SWAP AND PAGE DATA SET FLAGS ARE BOTH ON. C. IF LOCATE FAILS OR THE TYPE IS NOT 'DATA' OR THE DATA SET IS NOT A SWAP DATA SET, THEN SET RETURN CODE TO '12' AND RETURN TO THE CALLER.	SVC26		

Notes	Routine	Label	Ref

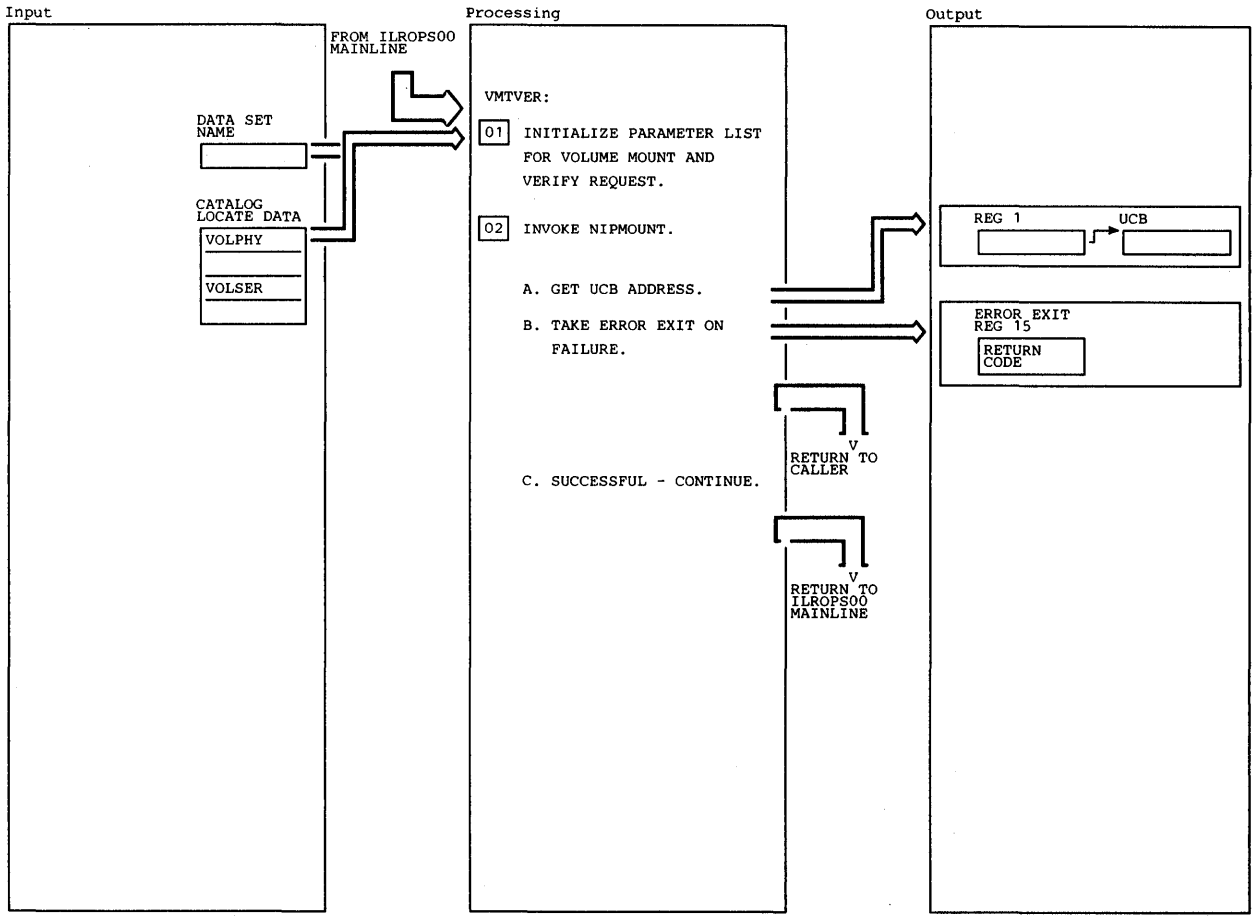
Diagram 25.32.2 LOCSWAP (Part 1 of 2)



Notes	Routine	Label	Ref
02 USING THE DATA CONTROL INTERVAL FROM THE FIRST LOCATE, ISSUE A LOCATE ON A DATA CONTROL INTERVAL, REQUESTING VOLPHV, VOLDEV, AMDSB, AND ATTRC INFORMATION. VOLPHY AND VOLDEV CONTAIN RESPECTIVELY PHYSICAL AND DEVICE RELATED DATA. THE AMDSB CONTAINS CONTROL INTERVAL DATA. ATTRC CONTAINS A TRACK OVERFLOW FLAG. A. TRACK OVERFLOW IS NOT ALLOWED FOR SWAP DATA SET. B. IF THE SECOND LOCATE FAILS OR THE TRACK OVERFLOW FLAG IS ON, SET THE RETURN CODE TO 12 AND RETURN TO CALLER. C. IF SUCCESSFUL, THEN CONTINUE.	SVC26		

Notes	Routine	Label	Ref

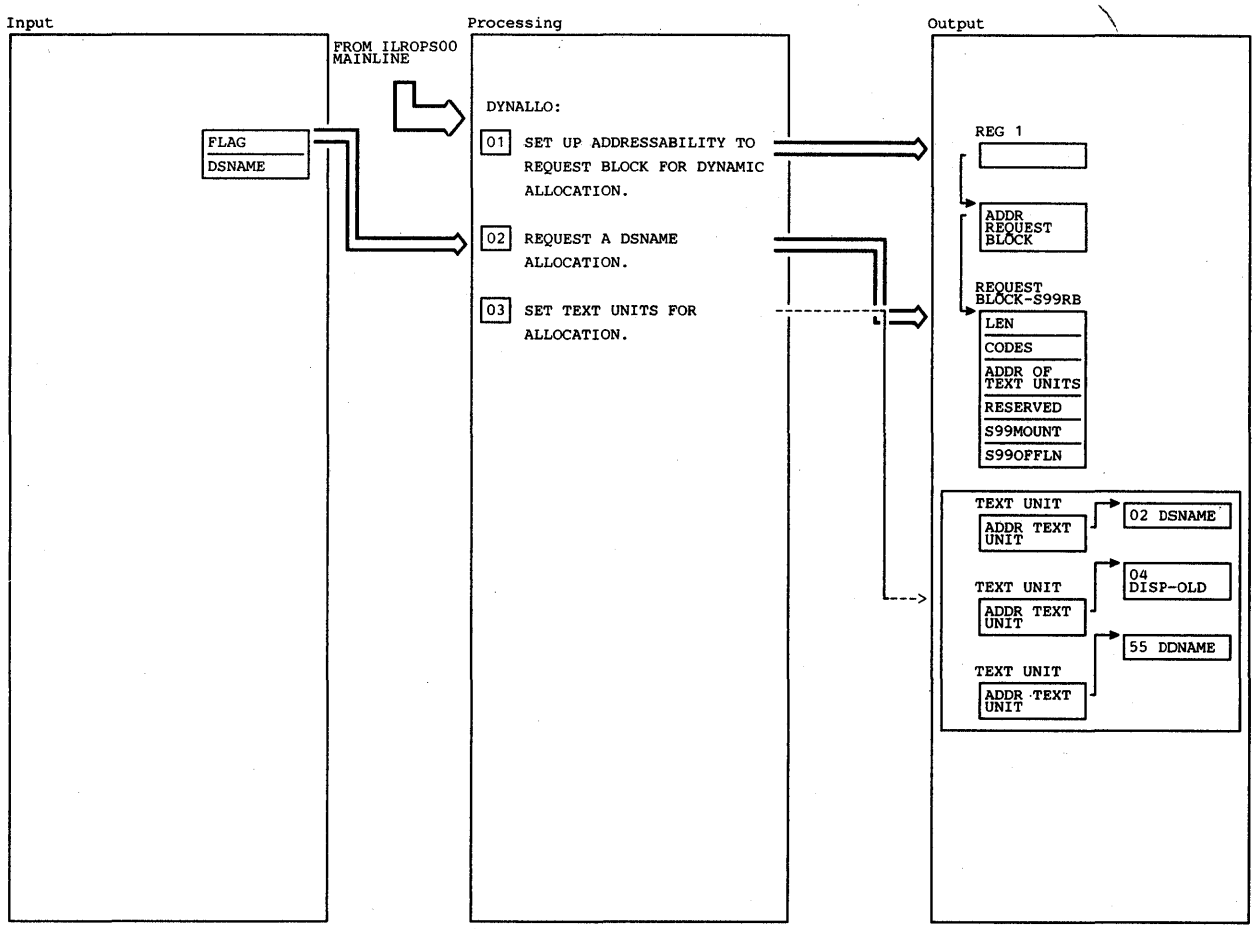
Diagram 25.32.2 LOCSWAP (Part 2 of 2)



Notes	Routine	Label	Ref
01 PUT THE DATA SET NAME AND VOLSER IN THE PARAMETER LIST FOR NIPMOUNT.			
02 INVOKE NIPMOUNT SERVICE.	IEAPMNIP		
A. IF SUCCESSFUL, REGISTER 1 WILL CONTAIN A UCB POINTER.			
B. IF NIPMOUNT FAILS, PUT AN 8 IN REGISTER 15, AND RETURN TO CALLER.			
C. IF SUCCESSFUL, THEN CONTINUE.			

Notes	Routine	Label	Ref

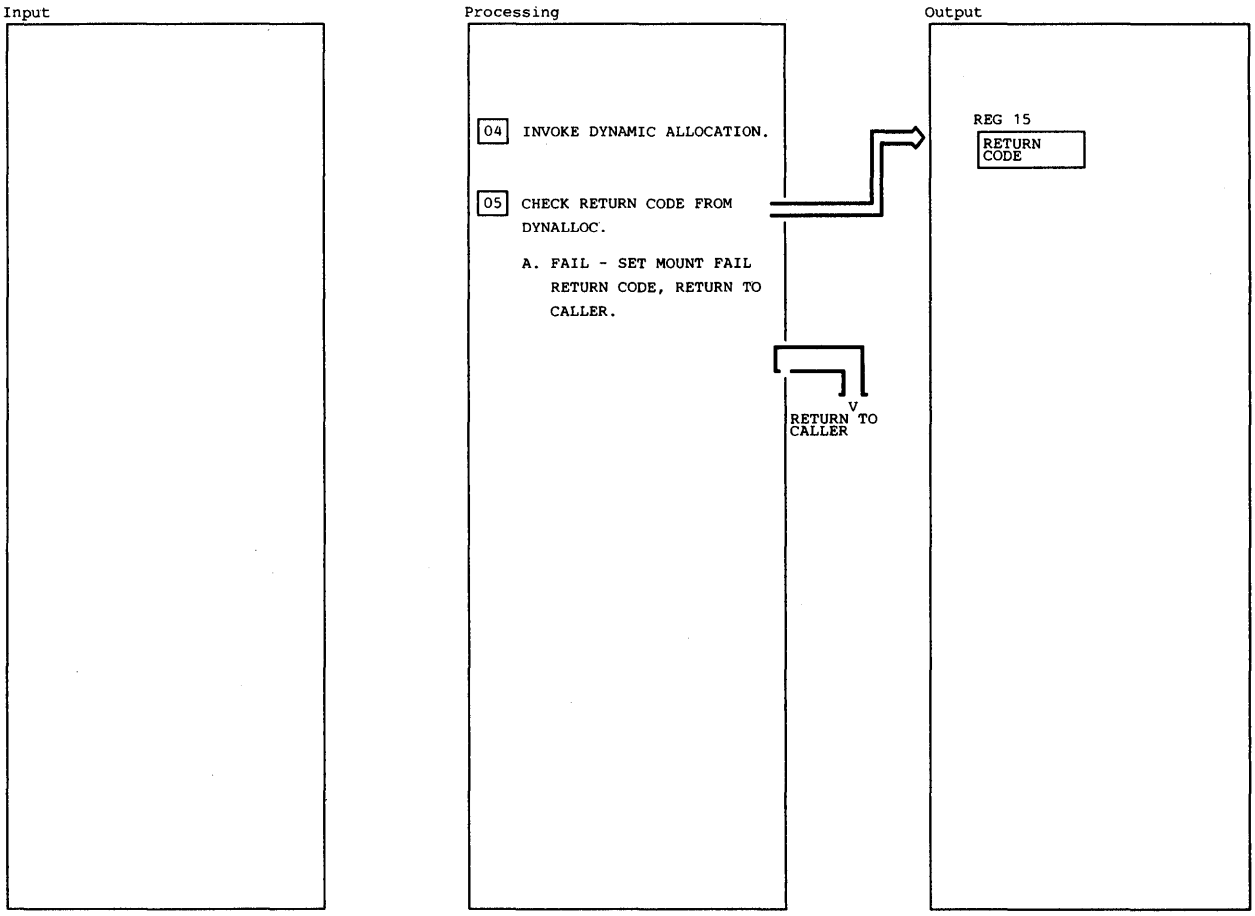
Diagram 25.32.3 VMTVER (Part 1 of 1)



Notes	Routine	Label	Ref
01 THE REQUEST BLOCK - S99RB IS MAPPED BY IEFZB4D0. IT IS THE INPUT PARAMETER TO DYNAMIC ALLOCATION.			
02 SET CODES IN REQUEST BLOCK TO INDICATE THAT DSNAME ALLOCATION IS DESIRED. TURN ON THE OFFLINE UNITS BIT (S99OFFLN) AND THE MOUNT VOLUME BIT (S99MOUNT), SO DYNAMIC ALLOCATION WILL NOTIFY THE OPERATOR WHEN THESE CONDITIONS OCCUR.			
03 TEXT UNITS, POINTED TO BY THE REQUEST BLOCK, ACTUALLY CONTAIN THE INPUT DATA (DSNAME AND DISPOSITION) AND THE EXPECTED OUTPUT DATA (DDNAME). THE OUTPUT DATA IS FILLED IN BY DYNAMIC ALLOCATION.			

Notes	Routine	Label	Ref

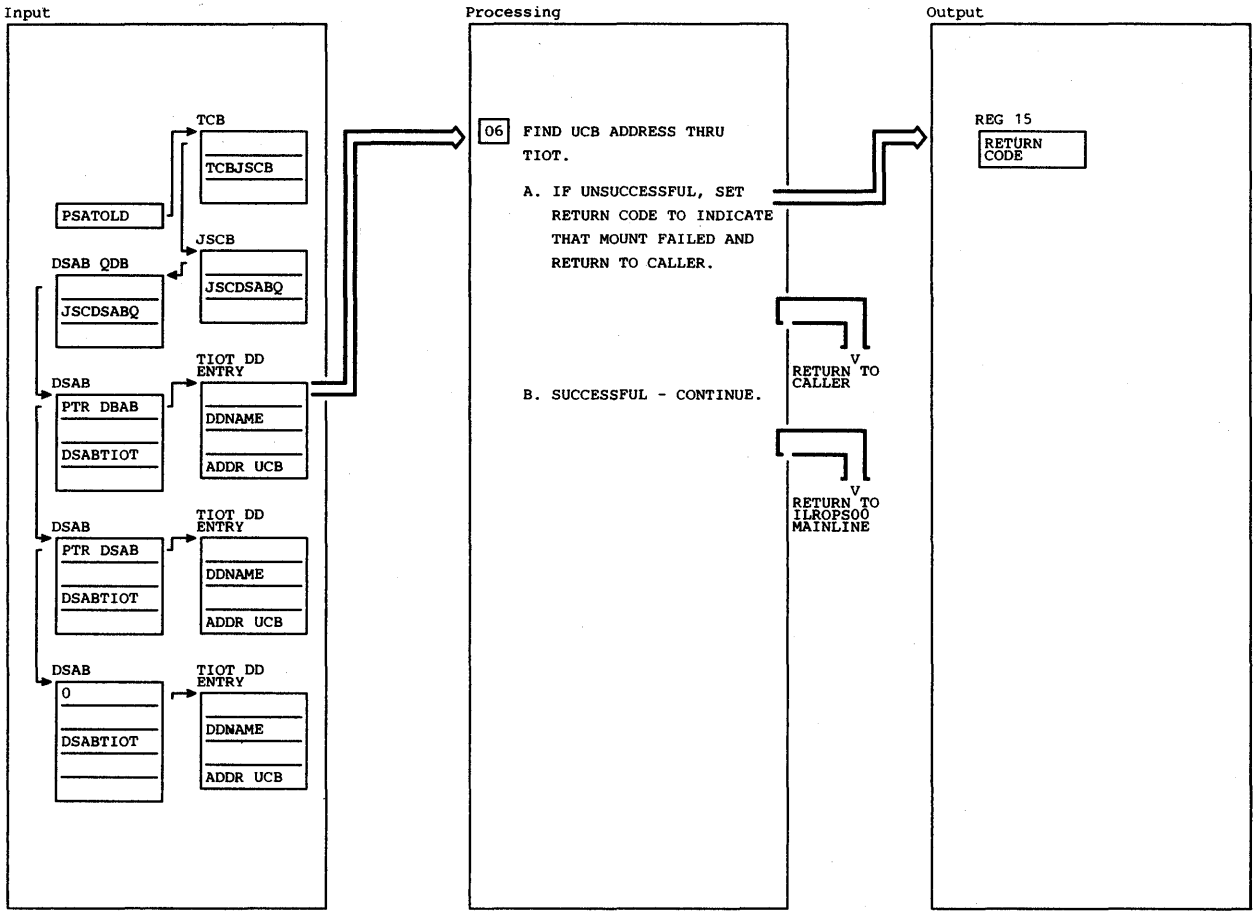
Diagram 25.32.4 DYNALLO (Part 1 of 3)



Notes	Routine	Label	Ref
04 INVOKE DYNALLOC MACRO TO ALLOCATE A DATA SET.	SVC99		
05 IF DYNAMIC ALLOCATION FAILS, SET REG 15 TO 8 AND RETURN TO CALLER.			

Notes	Routine	Label	Ref

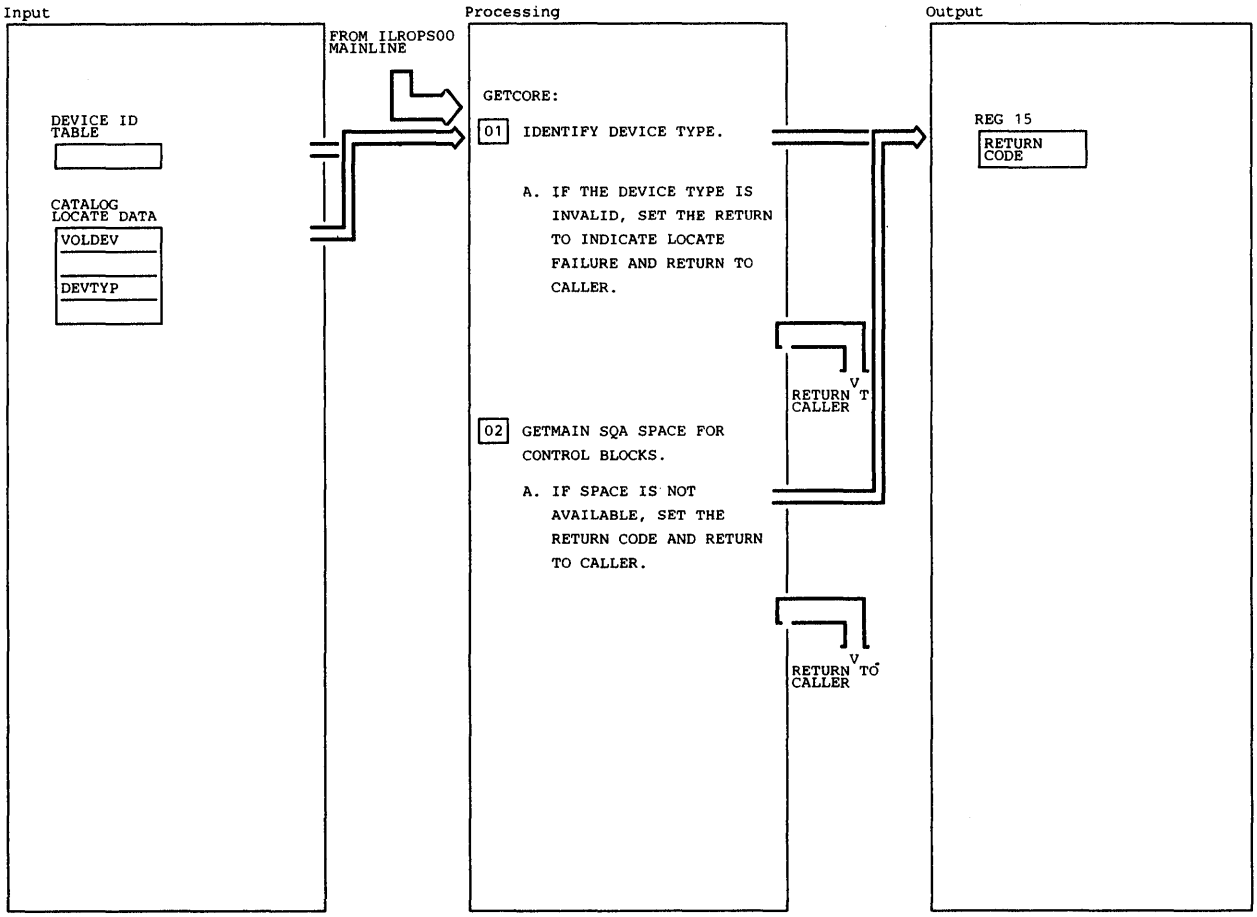
Diagram 25.32.4 DYNALLO (Part 2 of 3)



Notes	Routine	Label	Ref
06 THE UCB ADDRESS IS NOT RETURNED BY DYNAMIC ALLOCATION SO IT MUST BE OBTAINED BY SEARCHING TIOTS WITH THE RETURNED DD NAME. IF THE DD NAME CANNOT BE FOUND IN TIOTS SET REG 15 TO 8 AND RETURN TO CALLER. IF SUCCESSFUL, THEN CONTINUE.			

Notes	Routine	Label	Ref

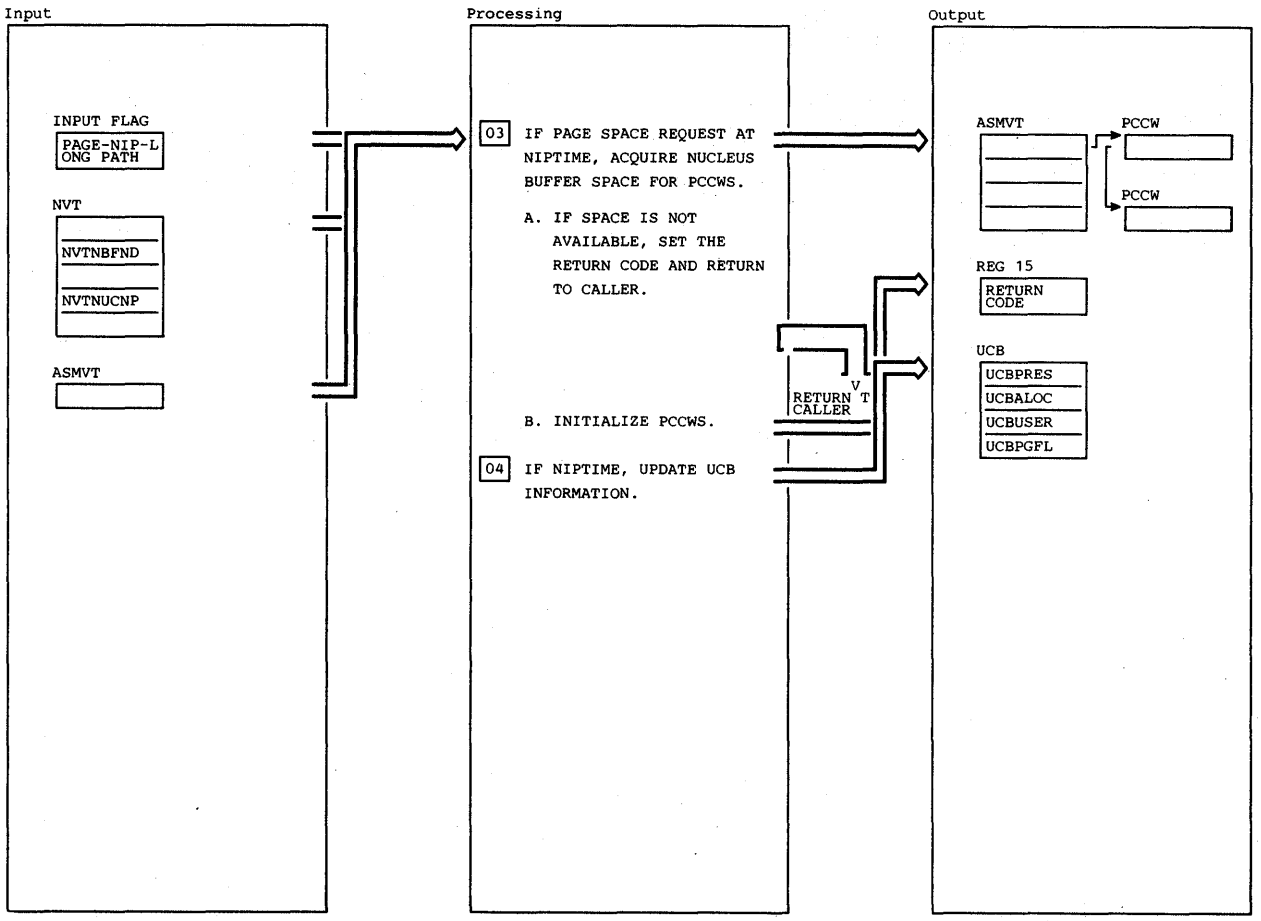
Diagram 25.32.4 DYNALLO (Part 3 of 3)



Notes	Routine	Label	Ref
01 SEARCH A TABLE OF VALID DEVICE TYPE FOR A MATCH WITH THE DEVICE TYPE RETURNED BY THE CATALOG LOCATE. IF A MATCH IS NOT FOUND, PUT A 12(LOCATE FAIL) IN REG 15 AND RETURN TO CALLER.			
02 CALCULATE THE TOTAL SIZE FOR IORB-IOSB-SRB, EDB AND LPMB. ISSUE A CONDITIONAL GETMAIN. IF REQUESTED SPACE IS NOT AVAILABLE, SET REG 15 TO 16 (NOT ENOUGH SQA SPACE) AND RETURN TO CALLER.	SVC120		

Notes	Routine	Label	Ref

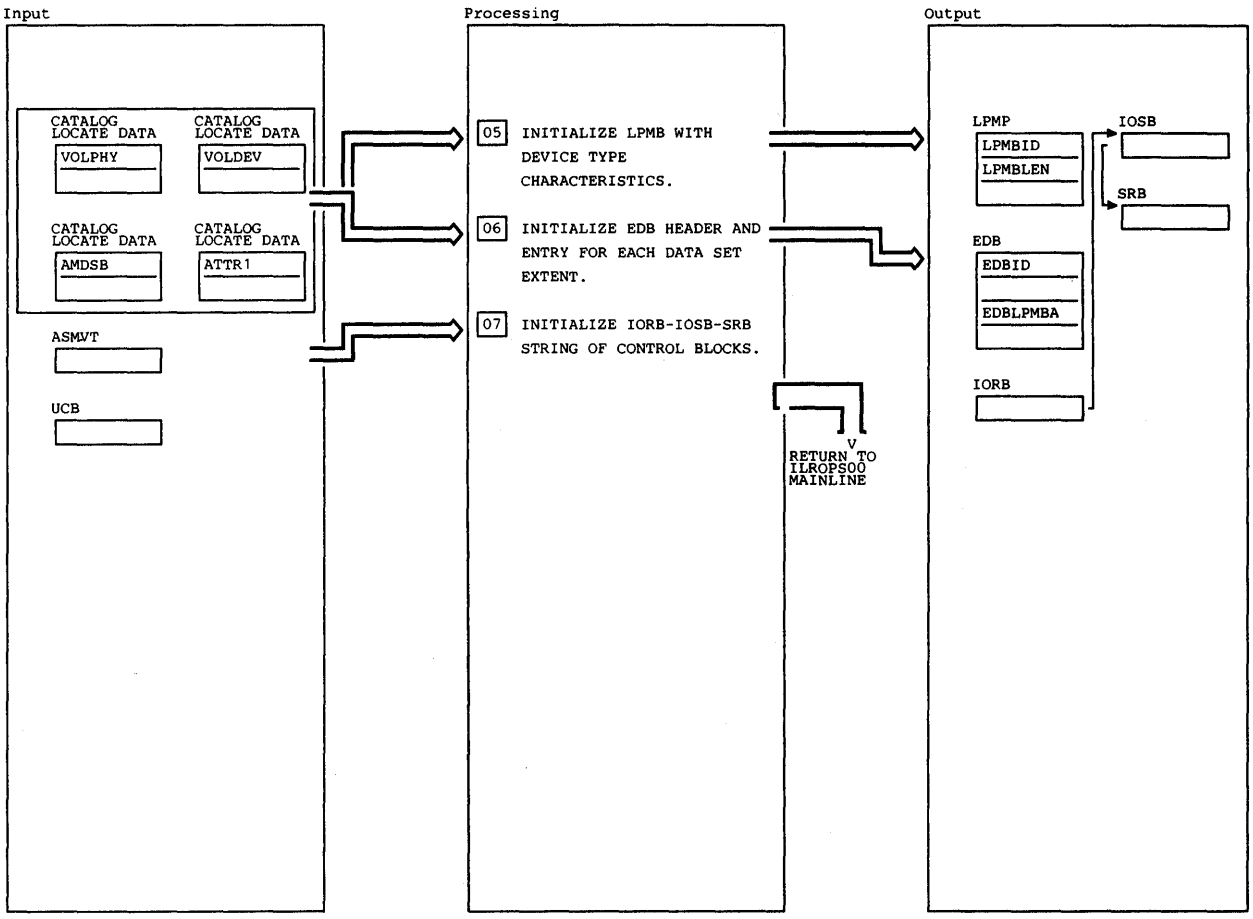
Diagram 25.32.5 GETCORE (Part 1 of 3)



Notes	Routine	Label	Ref
03 CALCULATE THE NUMBER OF PCCWS FOR THIS DEVICE TYPE AND CHECK THAT THERE IS ENOUGH SPACE FOR PCCWS IN THE NUCLEUS BUFFER. IF THERE IS NOT ENOUGH SPACE, SET REG 15 TO 20 (NUCLEUS BUFFER DOES NOT HAVE ENOUGH SPACE) AND RETURN TO CALLER. INITIALIZE THE CCW STRING WITH SEEK, SET SECTOR, SEARCH, ID, TIC, READ/WRITE NOP. CHAIN THE PCCWS TOGETHER.			
04 THE UCB INFORMATION IS UPDATED TO INCREMENT USER COUNT, AND TO MARK IT AS A PAGE SPACE AND PERMANENTLY RESIDENT.			

Notes	Routine	Label	Ref

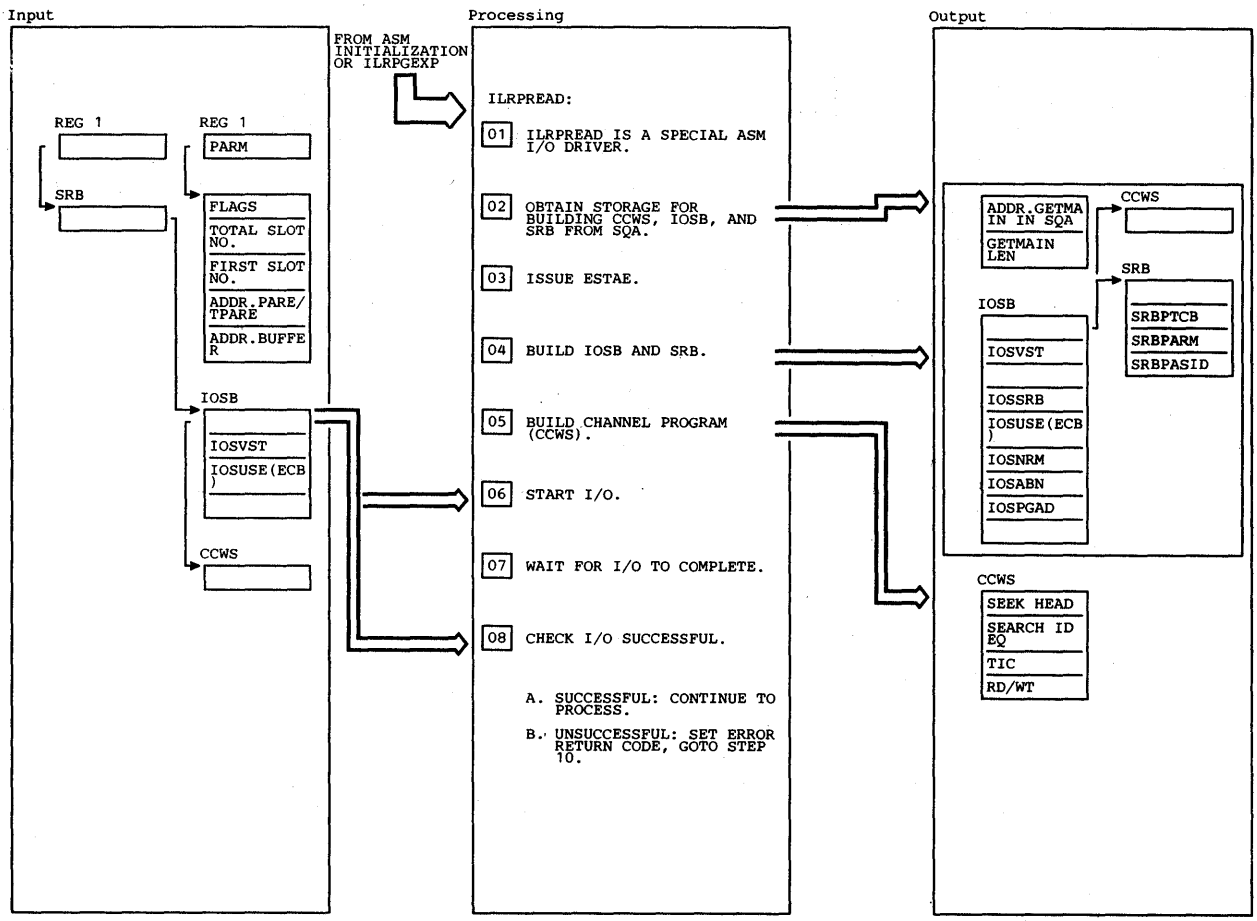
Diagram 25.32.5 GETCORE (Part 2 of 3)



Notes	Routine	Label	Ref
05 THE LPMB (LOGICAL TO PHYSICAL MAPPING BLOCK) FIELDS INITIALIZED ARE: ID, LENGTH TRACKS PER ALLOCATION UNIT, TRACKS PER CYLINDER, BLOCKS PER TRACK, BLOCK SIZE, BYTES PER TRACK, BYTES PER ALLOCATED UNITS, THE TRACK OVERFLOW AND RPS DEVICE FLAGS IF APPLICABLE.			
06 INITIALIZE RDB (EXTENT DEFINITION BLOCK) HEADER FIELDS: ID, LENGTH, NUMBER OF EXTENTS, LPMB ADDRESS. FOR EACH EXTENT INITIALIZE: LPMB ADDRESS, EXTENT NUMBER, STARTING TRACK, LOW RBA, HIGH RBA, TRACK OVERFLOW.			
07 AN IORB-IOSB-SRB STRING WILL BE INITIALIZED FOR EACH IORB REQUIRED. THE IORB FIELDS INITIALIZED ARE: ID, NUMBER OF IORB'S, RPS FLAG, IOSB ADDRESS, AND CHAIN FIELD TO NEXT IORB OR ZERO. THE INITIALIZED FIELDS IN IOSB ARE: DRIVER ID, I/O FILE MASK, IORB ADDRESS, SRB ADDRESS, I/O TERMINATION ADDRESS, NORMAL AND ABNORMAL END APPENDAGE. SRB			

Notes	Routine	Label	Ref
FIELDS INITIALIZED ARE: ID, AND NON-QUIESCEABLE PRIORITY. THE SRB IS POINTED TO BY IOSB WHICH IS POINTED TO BY IORB.			

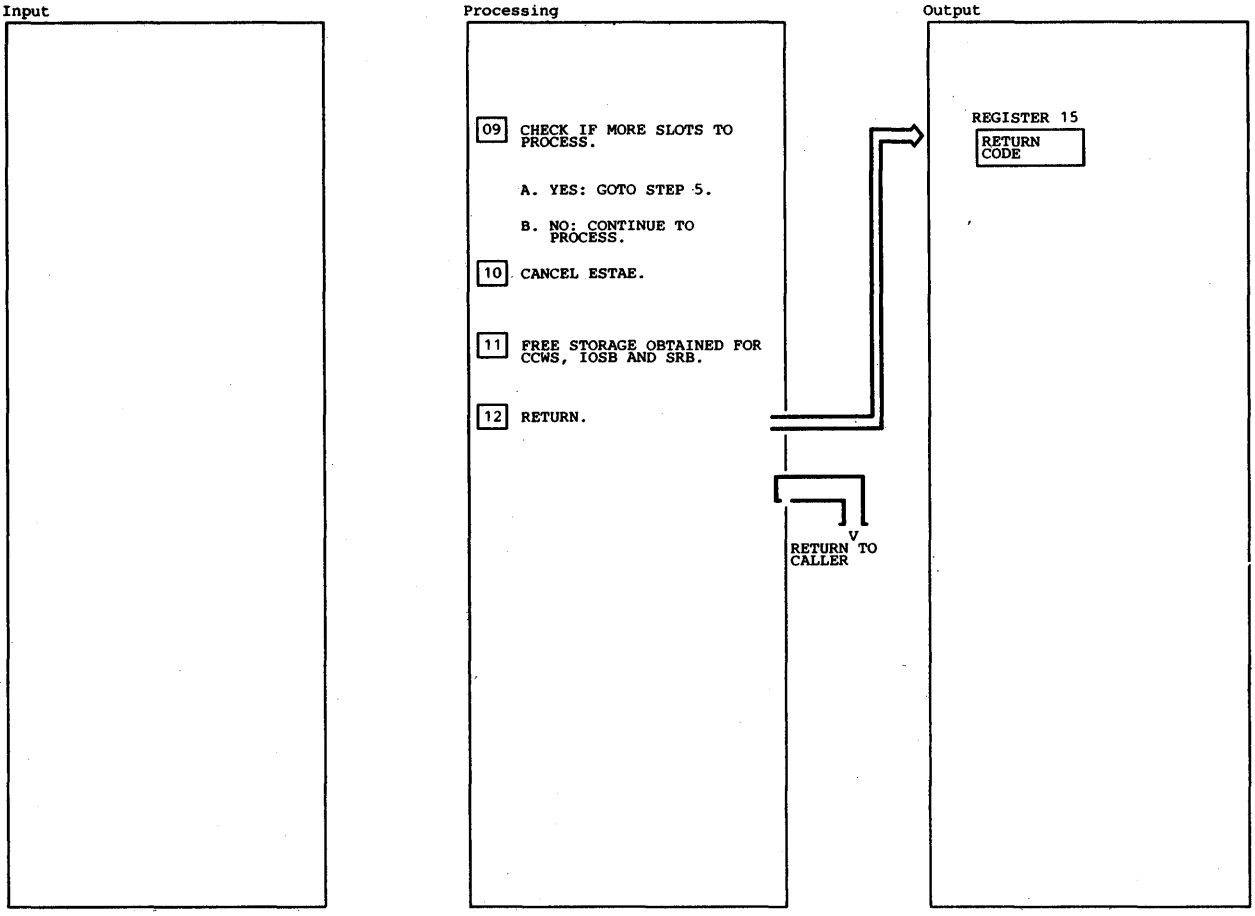
Diagram 25.32.5 GETCORE (Part 3 of 3)



Notes	Routine	Label	Ref
01 ILRPREAD IS AN I/O DRIVER THAT READS AND WRITES SLOTS CONTAINING CONTROL BLOCKS AND OTHER INFORMATION NEEDED BY ASM. SYSTEM INITIALIZATION READS AND WRITES ILRTPARB, ILRQSRCD, AND RECORD 0 TIMESTAMPS. ILRPGEXP READS AND WRITES ILRTPARB (THE TPARTBLE). THESE CONTROL BLOCKS ARE NEEDED FOR QUICK AND WARM STARTS TO OCCUR.			
02 ENTRY IS FROM ASM INITIALIZATION ROUTINES (ILRASRIM, ILRQSRIT, ILRTMI00) OR FROM PAGE EXPANSION (ILRPGEXP). STORAGE IS OBTAINED FROM SQA FOR BUILDING CCWS, IOSB AND SRB. THE CCW AREA OBTAINED IS LARGE ENOUGH TO READ/WRITE TEN SLOTS. IF GETMAIN FAILS, A CODE OF 8 IS RETURNED.	GETMAIN		
03 ESTABLISH ESTAE FOR ABNORMAL TERMINATION.	ESTAE	ESTAEXIT	25.33.3
04 THE ADDRESSES OF NORMAL END APPENDAGE, ABNORMAL END APPENDAGE, I/O TERMINATION ROUTINES, AND SRB ADDRESS ARE PUT INTO THE IOSB. THE ROUTINES ARE SECONDARY ENTRY POINTS IN ILRPREAD. THE ADDRESSES OF THE TCB AND IOSB ARE PUT INTO THE SRB. THE ASID (ADDRESS SPACE IDENTIFIER) IS ALSO PUT INTO THE SRB. SINCE THE TCB AND ASID AFFINITY ADDRESSES ARE IN THE IOSB, RTM PROCESSING CLEANS UP ANY ILRPREAD RESOURCES AND OUTSTANDING I/O UPON ERRORS OCCURRING IN IOS OR ILRPREAD APPENDAGE ROUTINES.			
05 CREATE THE CHANNEL PROGRAM FOR READS OR WRITES - MAXIMUM TEN SLOTS WITHIN A CYLINDER. THESE CONSIST OF SEEK HEAD, SEARCH ID EQUAL, TIC AND READ/WRITE. IF AN ERROR OCCURS DURING CONVERTING SLOT NUMBER TO REAL SEARCH ADDRESS THE CONTROL WILL IS			

Notes	Routine	Label	Ref
PASSED TO 10.			
06 CALL START I/O FOR READ/WRITE.	STARTIO		
07 WAIT FOR I/O TO COMPLETE. THE I/O COMPLETION ROUTINE WILL POST ECB WHEN IT GETS CONTROL.	WAIT	PREADNRM PREADABN PREADTRM	25.33.1 25.33.1 25.33.2
08 CHECK I/O COMPLETION. IF IT IS NOT SUCCESSFUL, THE CONTROL WILL BE PASSED TO STEP 10.			

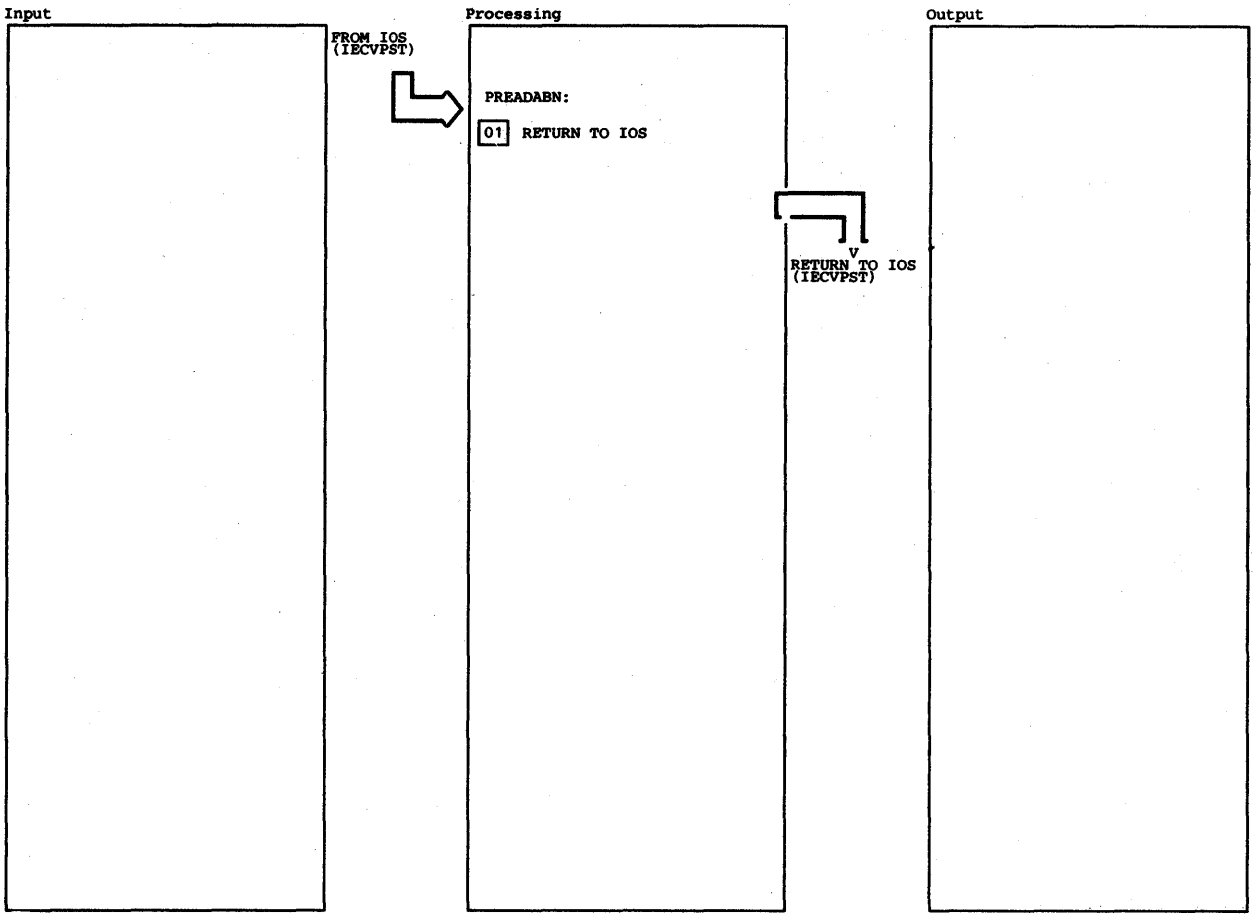
Diagram 25.33 ILRPREAD (Part 1 of 2)



Notes	Routine	Label	Ref
09 IF THE I/O COMPLETED SUCCESSFULLY, THE NUMBER OF REQUESTS REMAINING IS CHECKED. IF MORE SLOTS ARE TO BE PROCESSED, REPEAT FROM STEP 5.			
10 CANCEL THE ESTAE.	ESTAE		
11 IF THE REQUESTED READS OR WRITES ARE COMPLETED, ALL THE STORAGE OBTAINED FOR CCWS, IOSB, AND SRB WILL BE FREED.		FREEMAIN	
12 IF READ/WRITE IS SUCCESSFUL, A RETURN CODE OF ZERO WILL BE RETURNED. OTHERWISE A RETURN CODE OF FOUR FOR READ/WRITE/CONVERT ERROR OR A RETURN CODE OF EIGHT FOR SPACE NOT AVAILABLE WILL BE RETURNED.			

Notes	Routine	Label	Ref

Diagram 25.33 ILRPREAD (Part 2 of 2)



Notes	Routine	Label	Ref
01 FOR NORMAL AND ABNORMAL APPENDAGES, CONTROL IS PASSED BACK TO IOS IMMEDIATELY.			

Notes	Routine	Label	Ref

Diagram 25.33.1 PREADABN (Part 1 of 1)

This blank leaf represents pages 5-367 - 5-483 which were deleted by Supervisor Performance # 2.

Index

INDEX

OS/VS2
System Logic Library
Volume 5

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

Cut or Fold Along Line

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. (Elsewhere, an IBM office or representative will be happy to forward your comments.)

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.

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)

OS/VS2 System Logic Library Volume 5 (S370-36) Printed in U.S.A. SY28-0765-0