LY28-1200-4
File No. S370-36

Program Product

MVS/Extended Architecture System Initialization Logic

MVS/System Product:

JES3 Version 2	**5665-291**
JES2 Version 2	**5740-XC6**

IBM

Fifth Edition (June, 1987)

This is a major revision of LY28-1200-3. See the Summary of Amendments following the Contents for a summary of the changes made to this manual.

This edition applies to Version 2 Release 2.0 of MVS/System Product (5665-291 and 5740-XC6) and to all subsequent releases until otherwise indicated in new editions or Technical Newsletters. The previous edition still applies to Version 2 Release 1.7 of MVS/System Product (5665-291 and 5740-XC6) and may be ordered using the temporary order number LT00-2129. Changes are made periodically to the information herein; before using this publication in connection with the operation of IBM systems, consult the latest IBM System Bibliography, GC20-0001, for the editions that are applicable and current.

References in this publication to IBM products or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product in this publication is not intended to state or imply that only IBM's product may be used. Any functionally equivalent product may be used instead.

Publications are not stocked at the address given below. Requests for 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, Information Development, Department D58, Building 921-2, PO Box 390, Poughkeepsie, N.Y. 12602. IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

PREFACE

This publication describes the internal logic and organization of the system initialization process. The publication is intended for persons who are involved in modifying or debugging system initialization programs. For additional internal information about the use and operation of the MVS system, refer to the publication System Programming Library: Initialization and Tuning.

HOW THIS PUBLICATION IS ORGANIZED

This publication contains six chapters, five appendixes, and an index. Following, is a synopsis of the information contained in each section:

- Introduction -- a summary of system initialization functions and internal characteristics; maps of real and virtual storage.
- Data Areas -- descriptions of data areas used during IPL and NIP processing; references to microfiche publications.
- Diagnostic Aids -- Information that can be used for diagnosing problems in system initialization programs.
- Program Organization -- a description of module-to-module flow during system initialization.
- Method of Operation -- a functional approach to system initialization using both diagrams and text.
- Module Descriptions -- an alphameric list of module descriptions, including operation, entry, and exit.
- Appendix A -- brief descriptions of IPL macro instructions.
- Appendix B -- brief descriptions of NIP macro instructions.
- Appendix C -- brief descriptions of IPL service routines.
- Appendix D -- brief descriptions of NIP service routines.
- Appendix E -- a list of the abbreviations and acronyms used in this publication, with their meanings.

ASSOCIATED PUBLICATIONS

The following publications are referenced in this publication or should be used as corequisite reading:

- MVS/Extended Architecture Common VTOC Access Facility Diagnosis Reference, SY26-3929
- MVS/Extended Architecture Data Areas, LYB8-1191
- MVS/Extended Architecture Debugging Handbook, LC28-1164, LC28-1165, LC28-1166, LC28-1167, and LC28-1168.
- MVS/Extended Architecture Diagnostic Techniques, SY28-1199
- MVS/Extended Architecture JES2 Logic, LY24-6008-0
- MVS/Extended Architecture JES3 Logic, LY24-6007
- MVS/Extended Architecture Message Library: System Codes, GC28-1157
- MVS/Extended Architecture Message Library: System Messages, GC28-1376 and GC28-1377
- MVS/Extended Architecture Open/Close/EOV Logic, LY26-3892
- MVS/Extended Architecture Operations: System Commands, GC28-1206
- MVS/Extended Architecture Overview, GC28-1348
- MVS/Extended Architecture Installation: System Generation, GC26-4148.
- MVS/Extended Architecture System Logic Library Master Table of Contents and Index, SY28-1600
- MVS/Extended Architecture System Programming Library: Initialization and Tuning, GC28-1149
- MVS/Extended Architecture System Programming Library: System Management Facilities, GC28-1153
- MVS/Extended Architecture System Programming Library: System Macros and Facilities; GC28-1150 and GC28-1151
- MVS/Extended Architecture System Programming Library: SYS1.LOGREC Error Recording, GC28-1162
- MVS/Extended Architecture Virtual Storage Access Method (VSAM) Logic, LY26-3907
- OS/VS2 Mass Storage System Extensions Logic: MSS Communicator (MSSC), LY35-0038
- Resource Access Control Facility (RACF) General Information Manual, GC28-0722

CONTENTS

FIGURES

Summary of Amendments for LY28-1200-4 for MVS/System Product Version 2 Release 2.0

This edition contains changes to support of MVS/System Product Version 2 Release 2.0. The following changes include:

- 3 new IRIMs loaded by IEAIPL00
  - IEAIPL30
  - IEAIPL40
  - IEAIPL41
- Order of loads of IRIMs by IEAIPL00
- New modules
  - IEAVNP18
  - IEAVNP20
  - IEAVNP25
  - IEAVNPST
  - IEEMB888
  - IEEMB889
  - IEFAB4IE
  - IEFPPT
  - IEFRCSTP
- Deleted modules
  - IEFJESNM
  - IEFJSSNT
  - IEAVNP01
  - IEAVNP07
- Changed modules

IEAIPL00	IEAVNP13	IEAVVINT
IEAIPL01	IEAVNP16	IEEVIPL
IEAIPL02	IEAVNPA1	IEFAB41E
IEAIPL03	IEAVNPA6	IEFJSBLD
IEAIPL05	IEAVNPS5	IEFJSINT
IEAIPL99	IEAVRTOD	IEFJSIN2
IEAVNP05	IEAVN701	ISGNASIM
IEAVNP09		

Summary of Amendments
for LY28-1200-3
for MVS/System Product Version 2 Release 1.7

This edition contains changes to support of MVS/System Product Version 2 Release 1.7. The following changes are included:

- NIP Control and Service Routine (IEAVNIPM)
- A new module: Reconfiguration Installed Resources RIM - entry point IEAVNP27
- Changes to support the Vector Facility Enhancement include:
  - Updates to the NIP Control and Service Routine (IEAVNIPM).
  - The addition of prologue format documentation for IEAVNP00, an existing module that changed in this release.

Editorial and maintenance changes include:

- Reformatting of Figure 1-1 System Initialization Summary.
- Updates to the Master Scheduler Initialization diagram in support of TSO/E Release 2.

Summary of Amendments
for LY28-1200-2
for MVS/System Product Version 2 Release 1.3

This major revision contains editorial and technical changes in support of extended storage and service processors. These changes include two new modules:

- IEAIPL10
- IEAIPL20

Module descriptions have been added for:

IARMFRIM	IARMIIPL	IARMNRIM,
IARMSRIM	IARMTRIM	IARMURIM.

The following modules contain technical updates:

IEAIPL00	IEAIPL06	IEAIPL99,
IEAVNIP0	IEAVNIPM	IEAVNP02,
IEAVNPD8	IEAVRTOD	IEFJSINT.

SECTION 1. INTRODUCTION

System initialization is the process that prepares the control program and its environment to handle work. The process begins when the system operator activates the LOAD function for the processor and ends when the operating system is ready for work.

The system initialization process has three phases:

- initial program loader (IPL)
- nucleus initialization program (NIP)
- master scheduler initialization.

For an overview of the virtual storage areas initialized by the LOAD function and the three phases of system initialization, see Figure 1-1.

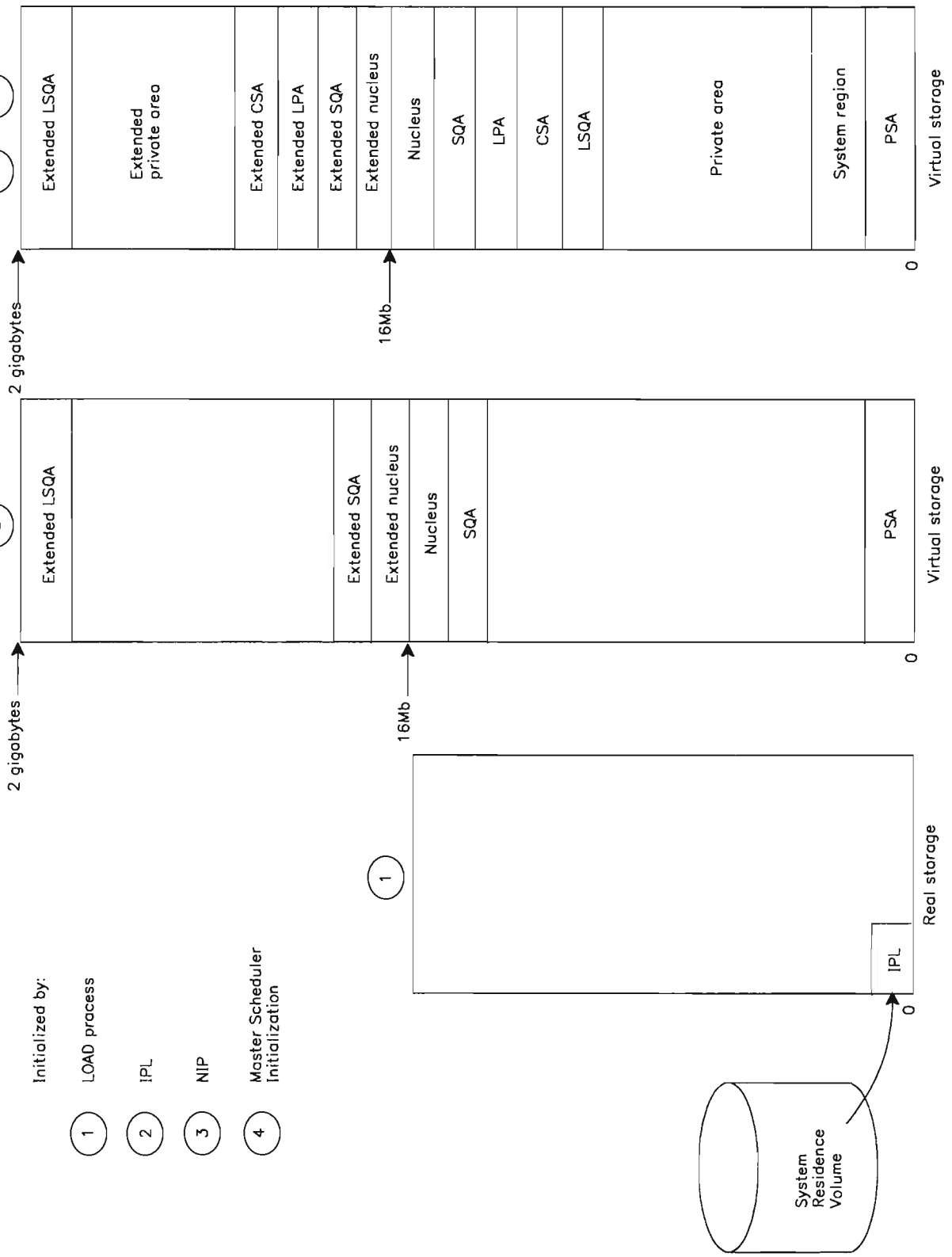


Figure 1-1. System Initialization Summary

INITIAL PROGRAM LOADER (IPL)

The system initialization process consists of three phases. The initial program loader (IPL) is the first phase and includes the resource initialization modules (IRIMS).

When the operator activates the LOAD function of the processor, the service processor loads the IPL control module, IEAIPL00, into real storage starting at location 0.

IEAIPL00 clears to zeroes each 4K frame of online real storage above the IEAIPL00 program and sets each corresponding storage key to zero. IEAIPL00 clears storage up to the storage address limit it obtains from the service processor. It then searches the system residence volume (SYSRES) for the SYS1.NUCLEUS data set, which contains modules needed for system initialization. IEAIPL00 passes control to IPL resource initialization modules (IRIMs) that initialize the system resources that IPL is responsible for.

IEAIPL00 provides the services required by the IRIMs. It services page faults and segment faults by assigning frames of real storage to each page of virtual storage that the IRIMs request. It also supports services via the supervisor call (SVC) instruction.

IPL SERVICE ROUTINES

IPL SVC 0	ISVCXDAP executes a direct access channel program.
IPL SVC 1	ISVCWAIT enters a disabled wait state.
IPL SVC 2	ISVCDAT switches translation mode.
IPL SVC 3	ISVCEXIT provides an exit from SVCFLIH.
IPL SVC 4	ISVCPGFX backs virtual storage with real frames.
IPL SVC 5	ISVCFIND reads the directory entry of a member of SYS1.NUCLEUS.
IPL SVC 6	ISVCLOAD loads a module into real storage.
IPL SVC 7	ISVCSTOR allocates contiguous frames.
IPL SVC 8	ISVCCNVT converts a relative SYS1.NUCLEUS address (TTR) to an absolute volume address (CCHHR).
IPL SVC 9	ISVCSSCH starts the IPL subchannel and then checks its status.
IPL SVC 10	ISVCSYNC builds and loads a PSW.
IPL SVC 11	ISVCCSEG builds segments by building page tables.

IPL RESOURCE INITIALIZATION MODULES (IRIMS)

In addition to IEAIPL00, IPL is made up of resource initialization modules that initialize the various resources that IPL is responsible for. The IPL control module, IEAIPL00, controls the loading and deleting of the IRIMs. IEAIPL01 determines the order of the loading of the IRIMS. The following lists each IRIM in the order of invocation and gives a brief description of its function:

IEAIPL10 Invokes the SCP INFO functions of the service processor. (IPL IRIM)

IEAIPL20 Validates the system storage and places all valid frames on the available frame queue. (IPL IRIM)

IEAIPL30 Initializes the IPL WTO facility service in the IPL work space then IPL-WTOs to the system console. (IPL IRIM)

IEAIPL40 Identifies the device support modules. IEAIPL40 is an I/O supervisor (IOS) IRIM.

IEAIPL41 Determines the DAT-on nucleus to be loaded, and determines the control section in all the modules that are going to be loaded into the nucleus (IEANUCOx). (IPL IRIM)

IEAIPL05 Builds a nucleus map (NUCMAP). (IPL IRIM)

IEAIPL02 Loads the DAT-off nucleus into real storage, reads the control sections of all of the modules being loaded into the nucleus region and resolves external references between these modules. (IPL IRIM)

IEAIPL04 Allocates storage for the page frame table, the system queue area (SQA), the extended SQA (ESQA), and the extended local system queue area (ELSQA) for the master scheduler address space. IEAIPL04 is the virtual storage management (VSM) IRIM.

IEAIPL06 Initializes the page frame table, the RSM internal table (RIT), the RSM address space block (RAB), and the RAB extension (RAX) for the master scheduler address space. IEAIPL06 is the real storage management (RSM) IRIM.

IEAIPL07 Resolves the AMODE bits of the nucleus-resident SVC and extended SVC router (ESR) tables. IEAIPL07 is the supervisor control IRIM.

IEAIPL03 Builds the resident ERP table and sets the DDT pointer in each UCB. Initializes the unit control block (UCB) for the IPL device. IEAIPL03 is an I/O supervisor (IOS) IRIM.

IEAIPL99 Copies the LPA device support module list (built by IEAIPL40) into the extended system queue area (ESQA), moves the IPL Message Queue into the system queue area (SQA), copies IPL information into common storage areas, initializes storage that is configured offline, initializes any remaining storage not as yet initialized as preferred and non-preferred storage, builds the available frame queues, and establishes addressability for the nucleus initialization program (NIP). (IPL IRIM)

NUCLEUS INITIALIZATION PROGRAM (NIP)

The nucleus initialization program (NIP) is the second phase of the system initialization process. NIP controls the initialization of system resources, but the resource initialization modules (RIMs) actually perform the work. NIP and the RIMs initialize real and virtual storage, processors, and devices, provide access to various system data sets, and initialize hardware recovery facilities, the generalized trace facility, and the resource managers in the supervisor component. In the course of their processing, NIP and the RIMs create the master scheduler address space in which they operate.

The NIP modules are:

IEAVNIP0 Prepares an environment in the master scheduler address space for execution of the RIMs. IEAVNIP0 loads the main NIP control module IEAVNIPM.

IEAVNIPM Loads, passes control to, and deletes the RIMs. Once loaded, it remains in storage until the completion of the second phase of system initialization. IEAVNIPM loads and passes control to the last NIP module, IEAVNIPX.

IEAVNPC1 Selects the NIP console and identifys it to IEAVNIPM for initial operator communication.

IEAVNP76 Initializes SYS1.LOGREC.

IEAVNP03 Checks the syntax of the system parameters and places them in PARMTAB. It processes the SYSP, LNK, APF and LNKAUTH parameters and opens the system data set SYS1.PARMLIB and the data sets specified as part of the LNKLST concatenation. IEAVNP03 is a NIP RIM.

IEAVNPX1 Performs initial NIP cleanup and prepares for the creation of additional address spaces. IEAVNPX1 is a NIP RIM.

IEAVNIPX Removes from storage the remaining NIP modules and any programs that have been loaded and passes control to master scheduler initialization.

RESOURCE INITIALIZATION MODULES (RIMS)

NIP is made up of NIP modules and resource initialization modules (RIMs) that initialize the various system resources that NIP is responsible for. NIP's main control routine, IEAVNIPM, controls the loading and deleting of the RIMs.

The following lists each RIM and gives a brief description of its function:

- **Service Processor** — initializes the control blocks that the service processor CALL SVC uses to communicate with the service processor. (IEAVNPE6)
- **Recovery termination management (RTM)** — initializes RTM control blocks. (IEAVNPA6, IEAVNPD6)
- **Machine check handler (MCH)** — initializes the machine check handler control blocks. (IEAVNP06)
- **Reconfiguration** — calls IEAVRTOD to initialize the time-of-day (TOD) clock for the IPL processor and IEEVCPR to bring online the non-IPL processors. (IEAVNP00, IEAVNP27)
- **Input/output supervisor (IOS)** — initializes IOS control blocks and unit control blocks (UCBs) for I/O devices in the configuration. (IEAVNPA2, IEAVNPB2, IEAVNF2)

- **Nucleus initialization process** — opens system data sets, merges system parameters, initializes the NIP console, and performs clean up processing for the nucleus initialization program itself. (IEAVNPC1, IEAVNP03, IEAVNPX1, IEAVNIPX)

- **Virtual storage access method (VSAM)** — opens the system catalog and VSAM data sets for NIP, performs LOCATE processing for NIP, closes the system catalog at the end of NIP processing. (IEAVNP11, IEAVNP1B)

- **Logrec initialization** — initializes SYS1.LOGREC. (IEAVNP76)

- **Real storage management (RSM)** — initializes the real storage management and extended storage management, and processes the REAL, VRREGN, and RSU system parameters. (IEAVNPE8, IEAVNPD8)

- **Global resource serialization** — initializes control blocks in SQA and creates a system address space named GRS for global resource serialization. (IEAVNP23/ISGNCBIM, IEAVNP33/ISGNTASC)

- **Auxiliary storage management (ASM)** — opens page and swap data sets; initializes ASM control blocks; during quick or warm start, recovers the pageable link pack area. (IEAVNP04/ILRASRIM and IEAVMP14/ILRASRM1)

- **Virtual storage management (VSM)** — initializes control blocks necessary for controlling the system queue area and the common service area. (IEAVNPA8, IEAVNPB8)

- **Supervisor control** — initializes the address space vector table with the maximum number of address spaces permitted in the system (IEAVNP09) and may prompt the operator to set the TOD clock (IEAVNP20).

- **Contents supervision** — initializes the pageable link pack area (PLPA), and updates the LPA with installation-specified modifications. (IEAVNP05)

- **Event notification facility (ENF)** — initializes the ENF control blocks. (IEAVNP47)

- **System resources manager (SRM)** — initializes SRM tables; initializes the value of the system automatic priority group (APG); builds control blocks for monitoring logical I/O channel use and for storing the installation performance specification (IPS) and the installation control specification; and sets parameters for SRM from the IEAOPTxx SYS1.PARMLIB member. (IEAVNP10, IEAVNP1F)

- **Dumping services** — initializes the system address space required for dumping volatile data, processes the DUMPDS command, processes the post SVC dump exits, and initializes the SVC dump data areas, the SYS1.DUMP data set(s), and the default options for the contents of SYSABEND, SYSMDUMP, and SYSUDUMP. (IEAVNP57, IEAVNPD1, and IEAVNPD2)

- **Cross memory** — builds and initializes the program call/authorization (PC/AUTH) tables that are used for cross memory operations. (IEAVNPF5)

- **System trace** — creates the system trace address space and initializes the trace function. (IEAVNP51)

- **Communications task** for the CON system parameter — initializing the MCS consoles and some Communication Task parameters (such as hardcopy log, write-to-operator buffers, etc) found in SYS1.PARMLIB. (IEAVNPA1).

- **Data management** — builds a table of authorized appendage routine names to be used by data management during OPEN processing. (IEAVNP16)

- **Master scheduler initialization interface** — sets up an interface with master scheduler initialization by passing system parameters that will be used later during master scheduler initialization. (IEAVNP13, IEAVNP18, IEAVNP20)
- **Generalized trace facility (GTF)** — initializes the interface between GTF and monitor call (MC) processing. (IEAVNP17)
- **Scheduler Interface to the General Parmlib Scan Routines** —initializes a parameter list (IEEZB819) and invokes the parmlib scan routine (IEEMB888) to process the SCHEDxx, SYS1.PARMLIB member.
- **Mass storage system communicator (MSSC)** — initializes the software support for the mass storage system (MSS) and prepares the MSS hardware for mounting and demounting mass storage volumes. (A mass storage volume consists of the two magnetic cartridges that contain the contents of a 3330 DASD volume.) (IEAVNP19)
- **Volume attribute allocation processing** — initializes UCBs with use and mount status information about volumes. (IEAVNP15/IEAVAP00)

NIP SERVICE ROUTINES

NIP service routines provide NIP modules and RIMs with functions that are frequently required, or they simulate functions that are not available in the system during NIP. NIP service routines are all contained in the IEAVNIPM load module. The following lists the NIP service routines and the functions each performs:

NIPMOUNT Ensures that required direct access or tape volumes are online and available for further NIP processing.

NIPOPEN Performs the OPEN function of creating and initializing a DEB to describe a specified data set or concatenation of data sets.

NIPPRMPT Prompts the operator for required system parameters that have not been specified or that have been invalidly specified.

NIPSENSE Interprets sense data after an I/O error occurs.

NIPSWAIT Places the system in a disabled wait state.

NIPTIME Provides the time of day in decimal for time-stamping messages, or provides a fullword binary value that reflects the number of hundredths of seconds elapsed since IEAVNIPM was first entered.

NIPUCBFN Locates a UCB.

NIPWTO Writes messages to the master operator console.

NIPWTOR Writes messages to the master operator console and reads the reply.

NIPWTOR2 (1) Waits for the completion of an operator's reply associated with an earlier NIPWTOR message, (2) frees a SQA reply buffer, or (3) frees the master-console-image buffer.

MASTER SCHEDULER INITIALIZATION

Master scheduler initialization is the third and final phase of the system initialization process. It performs initialization functions for the master scheduler's address space. First, it loads routines required by the system: system-initiated cancel, SWA management, and resource management routines. It builds the control blocks that describe each defined subsystem, to the subsystem interface and the subsystem allocation sequence table (SAST). It performs master trace initialization, if required. It performs final initialization for the communications task. It attaches the security initialization routine, allows it to process, and detaches it when complete.

Next, master scheduler initialization attaches and initializes tasks that remain as permanent system tasks after system initialization. These tasks include the missing-interrupt handler (MIH), the IPL/outage recorder, and three recovery termination tasks (the asynchronous recording task, the address space termination task, and the SVC dump task for the master scheduler's address space). Master scheduler initialization calls IOS to initialize dynamic pathing and establish the IOS storage manager as an event notification facility (ENF) listener. This portion of master scheduler initialization processes the automatic commands that NIP has obtained from SYS1.PARMLIB, sets up the last phase of ASM initialization, and links to the subsystem initialization routines whose names appear in SYS1.PARMLIB.

Finally, master scheduler initialization processes the START command for the job entry subsystem (creating a new address space), and prepares the system log for recording. The system is now initialized and ready to receive work in the form of system commands and user jobs.

Master scheduler initialization consists of three load modules located in SYS1.LINKLIB: IEEVIPL, IEEMB860, and IEEVWAIT. IEEVIPL receives control from NIP to begin the initialization of master scheduler's address space. IEEVIPL calls routines to perform various initialization functions, then passes control via the initiator to IEEMB860. The initiator interprets the JCL contained in SYS1.LINKLIB (MSTJCLxx member) in preparation for attaching IEEMB860. Therefore, IEEMB860 and its subroutines have access to those system data sets defined by the JCL.

When IEEVWAIT receives control from IEEMB860 via XCTL, it performs its initialization functions, processes the START command for the job entry subsystem, and waits for notification to process system commands. IEEVWAIT represents the permanent system task known as the master scheduler.

SYSTEM COMPONENT ADDRESS SPACE INITIALIZATION

A system component can execute in the address space of a program that requests its services, in its own address space, or partially in both the requesting address space and its own address space. By creating its own address space to hold some or all of its data and executable code, a system component can reduce the amount of storage it requires in the common area.

If a system component has its own address space, that address space must be created and initialized or be capable of handling requests before any other address space requests the component's services. The services of some components must be available at the completion of system initialization (master scheduler initialization and JES initialization). There are two general ways in which a component can create its own address space, only one of which can be used before system initialization is complete:

- By means of a START command (issued by the operator or by the system itself via SVC 34). For example, START TCAM results in the creation and initialization of the TCAM address space. The START command can be issued only after system initialization is complete. Because the START command can be used to start any program that has a procedure in SYS1.PROCLIB, component address spaces created via START are not included in the count of system component address spaces provided by the DISPLAY A,ALL command.

- By means of IEEMB881, the system component address space create routine. IEEMB881 can be invoked during NIP initialization, during master scheduler initialization, during JES initialization, or after system initialization is complete. The following components use IEEMB881 to create their own address spaces during NIP processing, master scheduler initialization, or JES3 initialization:

  - Program call/authorization (the PC/AUTH address space)
  - System trace (the TRACE address space)
  - Allocation (the ALLOCAS address space)
  - Global resource serialization (the GRS address space)
  - Dumping services (the DUMPSRV address space)
  - Communications task (the CONSOLE address space)
  - System Management Facility (SMF address space)
  - JES3 (the JES3AUX address space)

Note that the master scheduler's address space, which is the first address space in the system, and the primary JES address space are specifically created by NIP and the master scheduler, respectively, without use of the generalized processing provided by IEEMB881. The count of system component address spaces provided by DISPLAY A,ALL is the count of address spaces created via IEEMB881. Address spaces built by a component using IEEMB881 are sometimes referred to as system address spaces.

Figure 1-2 is a virtual storage map showing the master scheduler address space, the system address spaces, the job entry subsystem address spaces, the LLA address space, the SMF address space, the optional TCAM address space, and a user address space.

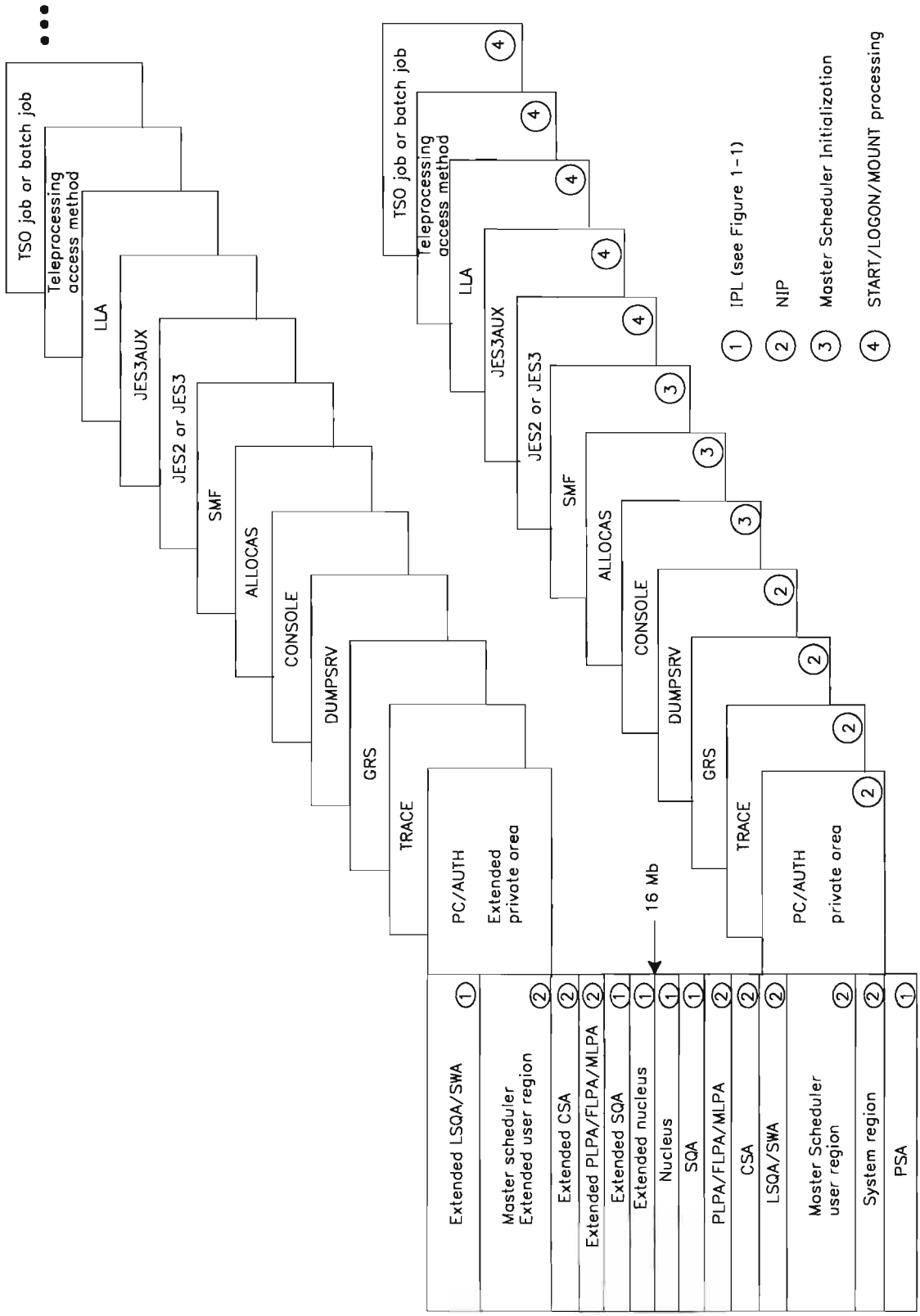


Figure 1-2. Virtual Storage Layout for Multiple Address Spaces

CREATION OF SYSTEM COMPONENT ADDRESS SPACES VIA IEEMB881

IEEMB881 receives control from one of the following modules to build a command scheduling control block (CSCB) representing the request for the specified address space.

- Allocation address space create (IEFAB4I0) - The ALLOCAS address space
- Communications task address space create routine (IEAVN700) - the CONSOLE address space
- Program call/authorization RIM (IEAVNPF5) - the PC/AUTH address space
- Global resource serialization trigger address space creation (IEAVNP33/ISGNTASC) - the GRS address space
- Dumping services address space RIM (IEAVNP57) - the DUMPSRV address space
- Local, global CTC initialization routine (IATINM3) - the JES3AUX address space
- System trace address space create routine (IEAVETAC) - the TRACE address space
- System Management Facility initialization routine (IEEMB820 - the SMF address space

When an address space is created via a START command, IEE0803D builds a CSCB. IEEMB881 fills in fields in the CSCB based on input passed by the caller.

The process of creating and initializing the component address space represented by the CSCB is primarily a process of building control blocks that:

- Define the address space
- Enable the address space to provide the component's services
- Enable the address space to use system services

Figure 4-15 illustrates the module flow for creating an address space via IEEMB881. The following topics describe the roles of the modules involved.

DEFINING THE ADDRESS SPACE

The modules invoked to define a system component address space are the same modules invoked to define any address space in the system. They are:

- IEAVEMRQ (memory request), which allocates an ASID and builds the ASCB for the new address space
- IEAVEMCR (memory create) which calls IEAVGCAS to build page and segment tables for the new address space
- IEAVEMIN (memory initialization), which builds control blocks necessary for executing tasks in that address space
- IEAVXMIN (cross-memory initialization), which connects the address space to cross memory control blocks.

The diagram "Obtaining a New Virtual Memory" in the section "Command Processing" of System Logic Library describes the processing of these modules, including other modules called by them.

PROVIDING THE COMPONENT'S SERVICES

The address space initialization necessary to provide a component's services is specific to that component's address space. IEEMB881 allows its caller to provide the name of a specialized initialization routine. (For address spaces created via the START command, the procedure in SYS1.PROCLIB does any required specialized initialization.) The initialization routines provided by the components that use IEEMB881 are:

- Program call/authorization: IEAVXMAS
- System trace: IEAVETAI
- Global resource serialization: ISGNASIM
- Dumping services: IEAVTSAI
- Communications task: IEAVN701
- Allocation: IEFHB4I1
- SMF: IEEMB820
- JES3: IATINXM

With the exception of JES3, the component's initialization routines execute before or during master scheduler initialization. As a result, these routines can use only those system services that have completed their initialization. (This restriction applies to all routines that execute during master scheduler initialization.) Routines that require a service that might not be initialized can use the event notification facility (ENF), provided it has been initialized, to wait for the initialization of the required service. For example, ISGNASIM uses ENF to wait for the availability of timer and console services.

Each initialization routine initializes its address space according to its component's own requirements. This includes creating and initializing control blocks, loading code, and making its data and code accessible by other address spaces via cross-memory instructions.

USING SYSTEM SERVICES

Two processes initialize address spaces to allow them to use system services:

- RCT (region control task) processing
- STC (started task control) processing

RCT processing always occurs for component address spaces created via IEEMB881. RCT establishes itself as the first task in the new address space.

IEEPRWI2, the first STC module, always receives control but the STC processing that occurs depends on the action taken by the component's initialization routine. (When creating an address space via a START command, IEEPRWI2 does not call a specific initialization routine and all of STC processing takes place.) The component's initialization routine, after doing component-specific initialization, has these choices:

- It can return to IEEPRWI2 to complete STC processing. If it does, the component must provide a procedure in SYS1.PROCLIB or use IEESYSAS.
- It can return to IEEPRWI2 to terminate the address space.
- It can enter a never-ending wait. It does not return to IEEPRWI2 and no further STC processing occurs.

Address spaces differ in their ability to use system services:

- Limited Function Address Space

  If the specific initialization routines provided by the components that use IEEMB881 enter a wait state, pending master scheduler initialization, STC does no additional

address space initialization. Thus, the functions that component address spaces can perform are limited. Components with limited function address spaces cannot:

- Allocate data sets.
- Read JCL procedures from SYS1.PROCLIB.
- Allocate a SYSOUT file through the job entry subsystem.
- Use some system services because the components frequently run in cross memory mode. System Macros and Facilities describes the restrictions placed on an address space that runs in cross memory mode.

- Full Function Address Space

  If, after a component completes its own initialization, it returns to IEEPRWI2 and completes STC processing, the address space is fully initialized. Such an address space is called a full function address space.

  The component creating a full function address space does not need to provide a procedure in SYS1.PROCLIB. If specified to IEEMB881, a common system address space procedure, IEESYSAS will invoke a specified program to run in the address space.

  For example, if a full function address space called FFA is to be started using module IEEMB899, the component would ordinarily need to supply a procedure of the following form:

```
//FFA       PROC
            EXEC          PGM=IEEMB899
```

  The procedure will be invoked as follows:

```
//IEESYSAS        JOB
//FFA             EXEC       IEESYSAS
```

  The procedure, IEESYSAS, consists of the following statements:

```
//IEESYSAS         PROC           PROG=IEFBR14
//                 EXEC           PGM=&PROG
```

All system component address spaces are limited function address spaces except the dumping services address space, DUMPSRV and the SMF address space.

SUBSYSTEM INTERFACE INITIALIZATION

Subsystems are specified in the IEFSSNxx members of SYS1.PARMLIB at system initialization. The master scheduler base initialization module (IEEVIPL) gives control to the subsystem interface initialization module (IEFJSINT). IEFJSINT calls the subsystem service routine (IEFJSBLD) to:

- Build a subsystem communication vector table (SSCVT) for the master subsystem.
- Build and initialize the subsystem vector table (SSVT) for the master subsystem.
- Build the subsystem allocation sequence table (SAST) for later use in allocating subsystem data sets.

IEFJSINT then returns control to IEEVIPL.

The job entry subsystem builds and initializes its own SSVT when the system is initialized. All other subsystems must do likewise. A subsystem can be initialized as follows:

- By being started (for example: START JES2) or,

- By having an initialization routine specified in the IEFSSNxx SYS1.PARMLIB record.

Additional subsystem initialization is performed by module IEFJSIN2, which is invoked by IEEMB860. IEFJSIN2 calls the PARMLIB read routine (IEEMB878) to read each record from the IEFSSNxx members. All records read are passed to the positional parse routine (IEEMB882). IEFJSIN2 also calls IEFJSBLD to perform the following:

- Link to the subsystem initialization routines specified in the subsystem name table.
- Build a SSCVT for each unique subsystem specified in the IEFSSNxx members and link to the initialization routine if any is specified on the PARMLIB record.
- Build a hash table containing SSCVT addresses for use in processing SSI requests.
- Rebuild the SAST if additional subsystems were defined in SYS1.PARMLIB.

IEFJSBLD and IEFJSIN2 call the subsystem initialization message writer (IEFJSIMW) to issue operator messages. The text of the messages is contained in IEFJSIMM.

SYSTEM DATA SETS USED BY SYSTEM INITIALIZATION

IPL uses the following system data set:

- SYS1.NUCLEUS, which is a required data set that contains the resident nucleus and the IPL modules. SYS1.NUCLEUS resides on the system residence volume.

NIP uses the following system data sets:

- SYS1.NUCLEUS, which is a required data set that contains the NIP modules. SYS1.NUCLEUS resides on the system residence volume.
- SYS1.LOGREC, which is a required data set that is used for recording hardware, software, and I/O errors. If not cataloged, SYS1.LOGREC resides on the system residence volume. NIPOPEN service routine performs open processing for SYS1.LOGREC.
- SYS1.SVCLIB, which is a required data set that resides on the system residence volume. SYS1.SVCLIB is an authorized program library that contains type 3 and type 4 SVC routines used by NIP. NIPOPEN service routine performs open processing for SYS1.SVCLIB.
- SYS1.LINKLIB, which is an authorized program library that contains system and user programs that are referred to by the XCTL, ATTACH, LINK, and LOAD macro instructions. SYS1.LINKLIB also contains an assembler-language processor, a linkage editor, the utility programs, and the service aid programs. SYS1.LINKLIB is a required data set that resides on a direct access volume. The LNKLST data sets are cataloged in the system catalog. NIPOPEN service routine performs open processing for SYS1.LNKLIB and concatenates user-specified data sets to it to open the data sets specified as part of the LNKLST concatenation.
- System catalog, which contains pointers to all cataloged data sets in the system control program. The system catalog is a required data set that resides on a direct access volume. The VSAM-catalog-open RIM, IEAVNP11, and the VSAM-data-set-open RIM, IEAVNP1A, perform VSAM open processing for the system catalog; the VSAM-catalog-close RIM, IEAVNP1B, performs VSAM close processing for the system catalog.

- SYS1.PARMLIB, which contains both IBM-supplied and user-supplied lists of system parameter values that serve as input to system initialization. SYS1.PARMLIB is a required data set that resides on a direct access volume. If it does not reside on the system residence volume, SYS1.PARMLIB is cataloged in the system catalog. The NIPOPEN service routine performs open processing for SYS1.PARMLIB.

- Page data sets, which are required VSAM data sets that reside on direct access volumes and are cataloged in the system catalog. The ASM RIMs, IEAVNP04/ILRASRIM and IEAVNP14/ILRASRM1, use an ASM open routine (ILROPS00) to perform open processing for the page data sets.

- Swap data sets, which are optional VSAM data sets that reside on direct access volumes and are cataloged in the system catalog. The ASM RIM, IEAVNP14/ILRASRM1, uses an ASM open routine (ILROPS00) to perform open processing for the swap data sets.

- SYS1.LPALIB, which contains modules that are loaded into the pageable link pack area. The contents supervisor RIM, IEAVNP05, loads modules from SYS1.LPALIB and the user-specified data sets that are concatenated to SYS1.LPALIB (the LPALST). SYS1.LPALIB is required only for cold start system initialization. The LPALST data sets reside on direct access volumes and must be cataloged in the system catalog.

- SYS1.DUMPxx, which is an optional data set that contains system dumps used to record areas of virtual storage in case of system task failures. SYS1.DUMPxx resides on either a direct access volume or a magnetic tape volume. It is cataloged only if it resides on a direct access volume. If the installation has specified dump data sets using the DUMP system parameter, the recovery termination management SVC dump RIM, IEAVTSDD, builds a dump data set queue (SDDSQ) to contain the name (DASD) or unit address (tape) of available SYS1.DUMPxx data sets.

Master scheduler initialization uses the following system data sets:

- SYS1.LOGREC, which contains information relating to the current system initialization. During master scheduler initialization, the missing-interrupt handler issues an initialization record for this data set.

- SYS1.LINKLIB (at the head of the LNKLST concatenation), which contains some of the master scheduler initialization modules. The member MSTJCLxx contains the card images for the JCL that defines data sets for the system and for TSO.

- SYS1.STGINDEX, which contains ASM mapping tables for the pages of virtual input/output (VIO) data sets that must be saved across job steps and between system initializations. SYS1.STGINDEX is a data set that resides on a direct access volume. SYS1.STGINDEX is cataloged in the system catalog. An ASM module (ILRTMI00) opens SYS1.STGINDEX during master scheduler initialization.

- SYS1.MANn, which is the SMF recording data set. During SMF initialization, the SMF writer opens the data sets specified in PARMLIB member SMFPRMxx.

- One or more RACF data set(s) (on systems with the RACF program product, Program No. 5740-XXH), which contain descriptions of each RACF-defined user of resource. They are used by RACF for user verification and authorization checking. For more information about RACF, see the Resource Access Control Facility (RACF) General Information Manual.

The following data sets are defined in the MSTJCL00 member of SYS1.LINKLIB (MSTJCL00, supplied by IBM, is the default master

JCL member). These data sets are available during the last part of master scheduler initialization.

- SYS1.PROCLIB, which contains standard JCL procedures for the system.
- SYS1.PARMLIB, which contains various lists of system parameters.
- SYS1.UADS, which is a user attribute data set that contains a description of each TSO user. It is used by LOGON scheduling to verify LOGON commands.
- SYS1.BRODCAST, which contains the broadcast messages for TSO.
- Internal readers, which are two internal reader data sets (STCINRDR and TSOINRDR) that are defined by the master scheduler's JCL. They are the means by which started task control passes JCL to a job entry subsystem.

Dump Analysis and Elimination (DAE) uses the following data set after initialization:

- SYS1.DAE, which contains records describing previous problems that have occurred on the system. These problems resulted in either an SVC dump or an SYSMDUMP dump being requested. The data set can be created after SYSGEN, using IEFBR14, which allocates space. The DAEALLOC member of SYS1.SAMPLIB contains JCL that an installation can execute to create the SYS1.DAE data set.

ERROR HANDLING DURING SYSTEM INITIALIZATION

Before system recovery routines are available, IPL and NIP provide diagnostic support for hardware and software errors.

There are certain program checks that IPL expects to experience and recover from. If, however, an unexpected program check occurs, IPL places the system in a wait state with a code of X'019'.

NIP attempts to detect unrecoverable system errors that occur during NIP processing and diagnose the errors with appropriate messages or wait state codes.
To detect errors, NIP provides trap routines that stop the initialization process at the time the error occurs. NIP traps the following unrecoverable system errors:

- Requests for ABEND (SVC 13)
- Machine checks
- Requests for entry to recovery termination management
- Unexpected program checks
- Requests for type 3 and type 4 SVC routines prior to the link pack area being available

NIP provides diagnostic information about the above errors by placing appropriate wait state codes in the wait state PSW.

NIP attempts to initialize the system regardless of errors made by the installation in specifying system parameters. If NIP encounters an incorrectly specified system parameter, it prompts the operator for a correct value. If the operator cannot correct the specification, the operator may specify that NIP use system defaults. Most parameters have default values either in a member of SYS1.PARMLIB or in NIP code. After NIP processes the operator's response to the SPECIFY SYSTEM PARAMETERS message, the SYS1.PARMLIB defaults are no longer available.

In cases when defaults do not exist, the operator may not cancel the parameter.

To aid recovery if an I/O error occurs during paging, ASM creates two copies of the pageable common area when a duplex page data set is provided. (The pageable common area contains the pageable link pack area and its modifications, and the common service area.)

ASM places one copy on the PLPA or common page data set and the other on the duplex data set. If the copy on the PLPA or common page data set cannot be read ASM attempts to read the copy on the duplex data set. If successful, the system continues processing without interruption.

If the mass storage system communicator (MSSC) cannot initialize at NIP or master scheduler initialization time, two messages are issued to the operator. One message indicates the reason MSSC cannot initialize. The other requires that the operator respond, requesting that the initialization continue without the mass storage system.

If master scheduler initialization fails or abnormally terminates, an ESTAE routine performs error processing and, in some cases, retry processing. If master scheduler initialization cannot continue, the system will enter a wait state after error messages are issued. (Error messages can be issued only if the communications task has been initialized.)

USER CONTROL OF INITIALIZATION

Before system initialization, the installation customizes the control program to both the data processing requirements and the machine configuration of the installation. The system generation process obtains standard and optional IBM-supplied components from distribution libraries, then combines them with user-written components from user data sets. The system generation process places the resultant system control program in system data sets on system volumes. During Input/Output generation (IOGEN) the installation defines the devices through the I/O Configuration Program. The MVS Configuration Program defines these devices to the system and creates the EDT table during EDTGEN. When these phases are complete, the system is ready for initialization.

During system initialization, the installation can affect initialization processing by selecting the type of IPL, specifying system parameters, and communicating with the system through the operator interface.

TYPES OF IPL

Page data sets are opened and initialized according to the type of IPL - cold, quick, or warm:

- Cold Start: Any IPL that loads (or reloads) the PLPA, and does not preserve VIO data set pages. The first IPL after system generation (sysgen) is always a cold start because the PLPA is initially loaded. Subsequent IPLs are cold starts when the PLPA is reloaded either to alter its contents or to restore its contents if they were destroyed.
- Quick Start: Any IPL that does not reload the PLPA and does not preserve VIO data set pages. Instead of reloading the PLPA, NIP resets the page and segment tables to match the last-created PLPA.
- Warm Start: Any IPL that does not reload the PLPA and does preserve journaled VIO data set pages.

THE FIRST IPL AFTER SYSTEM GENERATION: At the first IPL after sysgen, NIP automatically loads the PLPA from the LPALST data sets. The page data sets for this IPL are those named in SYS1.PARMLIB member IEASYS00 plus any specified by the operator. These page data sets are those that were specified in sysgen DATASET macros that contained the PAGEDSN keyword or the DUPLEXDS,NAME keyword combination. The operator need not enter the CLPA (create link pack area) parameter in response to the SPECIFY SYSTEM PARAMETERS message to load the PLPA. NIP does it automatically.

After the first IPL, the SYS1.LOGREC initialization routine, IFCDIP00, must be run to initialize SYS1.LOGREC. This routine must also be run whenever SYS1.LOGREC is reallocated.

AN IPL AT WHICH THE PLPA IS RELOADED: The PLPA needs to be loaded three times:

- At the first IPL after sysgen, when NIP loads it automatically.
- At an IPL after the installation has added or modified one or more modules in SYS1.LPALIB, or in an LPALST data set concatenated to SYS1.LPALIB, has tested the alteration, and now wants to put the replacement module(s) in the PLPA.
- At an IPL after the PLPA page data set has been damaged (and is therefore unusable and its contents must be restored).

To reload the PLPA from LPALST, the operator enters CLPA (create link pack area) as one of his responses to the SPECIFY SYSTEM PARAMETERS message. Reloading the PLPA should not be a common occurrence. It should be done only when necessary because the associated I/O slows down the IPL and because previously existing VIO data set pages are deleted. (For further information on the loading of the PLPA, see the CLPA parameter in the description of the IEASYSxx parmlib member.)

AN IPL AFTER POWER-UP: The IPL performed after power-up is called a quick start, because the PLPA from the previous IPL can be used without reloading from LPALST. For a quick start, the CVIO system parameter is specified; VIO data set pages are purged, page data sets added (optionally reserved for non-VIO paging), and swap data sets replaced. The operator, or the IEASYSxx member of SYS1.PARMLIB, can add additional page data sets by specifying the PAGE parameter (with or without specifying the NONVIO system parameter) and can replace swap data sets by specifying the SWAP parameter.

AN IPL AFTER A SYSTEM CRASH: Provided that the operator does not enter the CLPA or CVIO system parameters, the operator can warm start the system after a system crash. Existing journaled VIO data set pages are retained by the auxiliary storage manager for continued use. The specified parmlib parameter list (IEASYSxx) would not include the CVIO or CLPA system parameters.

Any definitions of existing page data sets as non-VIO local page data sets are preserved. Also the operator can define a local page data set that previously was used for VIO paging as a non-VIO local page data set. During system operation, the VIO pages on the newly-designated non-VIO local page data set will migrate to any local page data set used for VIO paging. An installation can remove a local page data set that was designated on the previous IPL as a non-VIO local paging data set. Removing a local page data set prior to a warm start requires that the local page data set contain no VIO pages. If the local page data set contains VIO pages, then the warm start is changed into a quick start.

SYSTEM PARAMETERS

System parameters provide the installation with the means to tailor or tune a system to meet the installation's requirements. Following is a brief description of the system parameters:

- APF indicates the members (IEAAPFxx) of SYS1.PARMLIB that contain authorized data set names.
- APG specifies the number of the automatic priority group for use by the system resources manager (SRM), unless nullified by the IPS.
- CLOCK indicates the member (CLOCKxx) of SYS1.PARMLIB which prompts the operator to initialized the TOD clock and specifies the difference between the local time and GMT.
- CLPA indicates that NIP should load the pageable link pack area with the modules contained in the LPALST data sets.
- CMB specifies the I/O device classes for which measurement data is to be collected, in addition to the DASD and tape device classes.
- CMD indicates the member or members (COMMNDxx) of SYS1.PARMLIB that contain commands to be issued automatically during master scheduler initialization.
- CON indicates the CONSOLxx member of SYS1.PARMLIB that contains the multiple console support (MCS) console definition and various communication task parameters.
- CSA specifies the number of 1K blocks of virtual storage to be reserved for the common service area (CSA) and extended CSA (ECSA).
- CVIO indicates that the pages of previously-created virtual input/output (VIO) data sets should be deleted from external page storage
- DUMP specifies whether SYS1.DUMP data sets for SVC dump reside on direct access or tape devices.
- DUPLEX specifies the duplex paging data set to be used. The DUPLEX parameter is optional and should be used only if CLPA is specified. For quick and warm starts, a duplex paging data set from the previous CLPA IPL is used; the DUPLEX parameter is ignored.
- FIX indicates one or more members (IEAFIXxx) of SYS1.PARMLIB that contain names of modules to be placed in the fixed link pack area
- GRS indicates whether the system being initialized is to start a global resource serialization complex, join an existing global resource serialization complex, or not be part of a complex.
- GRSCNF indicates the member (GRSCNFxx) of SYS1.PARMLIB that contains the information that describes the systems to be part of a global resource serialization complex.
- GRSRNL indicates one or more members (GRSRNLxx) of SYS1.PARMLIB that contain the resource name lists (RNLs) used by the global resource serialization complex.
- ICS indicates the member (IEAICSxx) of SYS1.PARMLIB that contains the installation control specification for the system resources manager (SRM).
- IOS indicates the member (IECIOSxx) of SYS1.PARMLIB that contains the IOS options for the missing interrupt handler (MIH) and for hot I/O (HOTIO).

- IPS indicates the member (IEAIPSxx) of SYS1.PARMLIB that contains the installation performance specification (IPS) for the system resources manager (SRM).
- LNK indicates one or more members (LNKLSTxx) of SYS1.PARMLIB that contain names of data sets to be concatenated to SYS1.LINKLIB.
- LNKAUTH specifies that the data sets in the LNKLST concatenation are all to be treated as APF authorized, or are only to be treated as APF authorized if specifically named in the APF table.
- LOGCLS specifies the JES output class for the log data sets.
- LOGLMT specifies the maximum number of messages for a log data set.
- LPA indicates one or more members (LPALSTxx) of SYS1.PARMLIB that contain the names of data sets to be concatenated to SYS1.LPALIB.
- MAXUSER specifies the maximum number of address spaces that may be created under normal conditions. When the system is heavily used, the value specified on the RSVSTRT parameter may permit additional started tasks.
- MLPA indicates one or more members (IEALPAxx) of SYS1.PARMLIB that contain names of modules to be placed in the modified link pack area (MLPA).
- MSTRJCL indicates the member (MSTJCLxx) of SYS1.LINKLIB that contains the JCL for starting the master scheduler address space.
- NONVIO specifies the names of local paging data sets that are not to receive VIO pages when the directed VIO function is active.
- OPI indicates whether the operator may override parameters in IEASYSxx.
- OPT indicates a member (IEAOPTxx) of SYS1.PARMLIB that contains the tuning parameters for the SRM.
- PAGE specifies the names of page data sets to be used with the exception of the duplex paging data set. See the DUPLEX parameter listed earlier.
- PAGTOTL specifies the total number of page and swap data sets that can be made available to the system for the life of the IPL.
- PAK indicates the members (IEAPAKxx) of SYS1.PARMLIB that contain lists of modules to be packed together in the PLPA.
- PURGE specifies the demounting of mass storage volumes that had been specified at the previous IPL on a specific processor. If PURGE is issued during IPL and the IPL fails, PURGE must be issued again during re-IPL or the mass storage volume inventory will be inaccurate. (For more information, see Mass Storage System Extensions Logic: MSS Communicator (MSSC).
- RDE specifies the inclusion of the reliablity data extractor feature. RDE also controls the "Enter IPL reason" prompt. For more information, see SYS1.LOGREC Error Recording.
- REAL specifies the maximum amount of 1K units of real storage that can be allocated for V=R jobs.
- RER specifies that the reduced error recover procedures for magnetic tape are to be used. This must be requested in the OPTCD parameter of a data definition (DD) statement or in

the DCB macro instruction. If RER is not specified, the system ignores all requests for reduced errror recovery.

- RSU specifies the number of processor storage units that are required for storage reconfiguration. The frames in these storage units will be designated as the nonpreferred area; they are not to be used for long-term storage if preferred area storage is available.
- RSVNONR specifies the number of entries in the address space vector table (ASVT) reserved for replacing entries marked non-reusable for the duration of the IPL.
- RSVSTRT specifies the number of entries in the address space vector table (ASVT) reserved for address spaces created in response to a START command. This helps avoid a re-IPL when there is no available entry for a critical address space.
- SCH indicates the member (SCHEDxx) of SYS1.PARMLIB that contains master scheduler parameters.
- SMF indicates the member (SMFPRMxx) of SYS1.PARMLIB that contains SMF parameters.
- SQA specifies the number of 64K blocks of the virtual system queue area (SQA) and extended SQA to be created during NIP initialization in addition to the initial three blocks of SQA and three blocks of extended SQA.
- SSN indicates one or more members (IEFSSNxx) of SYS1.PARMLIB that define subsystems to the system during master scheduler initialization.
- SVC indicates the member (IEASVCxx) that contains installation-defined SVCs.
- SWAP specifies the names of swap data sets to be used.
- SYSNAME provides the name by which a system is known to the global resource serialization complex.
- SYSP indicates one or more system parameter lists (IEASYSxx) in SYS1.PARMLIB that are to be used by NIP in addition to IEASYS00.
- VAL indicates one or more members (VATLSTxx) of SYS1.PARMLIB that contain attributes of direct access volumes.
- VRREGN specifies the default real storage region size in 1K units for V=R jobs that do not specify a region size in their JCL.

NIP and master scheduler initialization routines process the system parameters during system initialization processing. Figure 1-3 shows the relationship among the system parameters, the system components they affect, and the initialization modules that process the parameters.

System Parameter	System Component Affected	NIP Object Module
APF	Contents supervisor	IEAVNPA5
APG	System resources manager	IEAVNP10
CLOCK	Master scheduler	IEAVNP20
CLPA	Contents supervisor and	IEAVNP05
	Auxiliary storage management	ILRASRIM, ILRASRM1
CMB	System resource management	IEAVNP1F
CMD	Master scheduler initialization	IEAVNP13
CON	Communications task	IEAVNPA1
CSA	Virtual storage management	IEAVNP08
CVIO	Auxiliary storage management	ILRASRIM, ILRASRM1
	Auxiliary storage management	ILRTMI00
DUMP	Recovery termination management	IEAVTSDD
DUPLEX	Auxiliary storage management	ILRASRIM
FIX	Contents supervisor	IEAVNP05
GRS	Global resource serialization	ISGNCBIM, ISGNGRSP
GRSCNF	Global resource serialization	ISGJPARM
GRSRNL	Global resource serialization	ISGNRNLP
ICS	System resources manager	IEAVNP10
IOS	I/O supervisor	IEAVNPF2
IPS	System resources manager	IEAVNP10
LNK	NIP	IEAVNP03
LNKAUTH	NIP	IEAVNP03
LOGCLS	System log	IEAVNP13
LOGLMT	System log	IEAVNP13
LPA	Contents supervisor	IEAVNP05
MAXUSER	Supervisor control	IEAVNP09
MLPA	Contents supervisor	IEAVNP05
MSTRJCL	Master scheduler initialization	IEAVNP13
NONVIO	Auxiliary storage management	ILRASRIM, ILRASRM1
OPI	NIP	IEAVNP03
OPT	System resources manager	IEAVNP10
PAGE	Auxiliary storage management	ILRASRIM
PAGTOTL	Auxiliary storage management	ILRASRIM
PAK	Contents supervisor	IEAVNPC5
PURGE[1]	Mass storage system communicator	IEAVNP19
RDE	Data management	IEAVNP16
REAL	Real storage management	IEAVNPD8, IEAVNPE8
RER	Data management	IEAVNP16
RSU	Real storage management	IEAVNPD8
RSVNONR	Supervisor control	IEAVNP09
RSVSTRT	Supervisor control	IEAVNP09
SCH	Master scheduler	IEANVP18
SMF	System management facilities	IEAVNP13
SQA	Virtual storage management	IEAVNPA8
SSN	Subsystem interface	IEAVNP13
SVC	Supervisor Control	
SWAP	Auxiliary storage management	ILRASRIM
SYSNAME	Global resource serialization	ISGNCBIM
SYSP	NIP	IEAVNP03
VAL	Volume attribute processing	IEAVAP00
VRREGN	Virtual storage management	IEAVNPD8

Figure 1-3. Relationship Among System Parameters, System Components, and NIP Processing Modules

[1] For information on PURGE command processing, refer to Mass Storage System Extensions Logic (MSSC).

System parameters come from two sources:

- Using IEBUPDTE, the installation specifies system parameters in IEASYSxx members of SYS1.PARMLIB. If the installation places parameters in an IEASYSxx member, the operator indicates which member contains the parameters by using the SYSP= parameter in response to the SPECIFY SYSTEM PARAMETERS message. By using an IEASYSxx member for system parameters, the installation reduces the amount of operator interaction during NIP processing. For information about determining system parameter values and entering system parameters in SYS1.PARMLIB, refer to Initialization and Tuning. NIP merges parameters from IEASYS00 with parameters specified in alternate IEASYSxx members. Parameter values from the last IEASYSxx member that NIP reads override previous IEASYSxx and IEASYS00 values for those parameters.

- In response to the SPECIFY SYSTEM PARAMETERS message issued by NIP, the operator may specify system parameters. An operator-entered parameter overrides the same parameter specified in IEASYS00 or IEASYSxx, except for:

  - A parameter for which operator intervention is prohibited (OPI=NO).

  - The PAGE parameter; NIP adds the data set names entered by the operator to the data set names specified in either IEASYS00 or IEASYSxx.

OPERATOR INTERFACE DURING INITIALIZATION

IPL permits the operator to specify a nucleus other than the system-assigned nucleus (IEANUC01). IPL also allows selection of an alternate I/O configuration. For details on the procedure involved, see Operations: System Commands.

NIP establishes operator communications immediately after it initializes the master console.

1. NIP issues message 'IEA101A SPECIFY SYSTEM PARAMETERS'. The operator's reply indicates one of the following:

   - NIP should use the default list of system parameters (IEASYS00) in SYS1.PARMLIB.

   - NIP should use the list of system parameters (IEASYSxx) in SYS1.PARMLIB.

   - NIP should cancel a parameter value specified in IEASYSxx and use the system default if one exists.

   - NIP should use new system parameter values specified in the reply, overriding the values specified in IEASYSxx.

2. NIP issues message 'IEA437A SPECIFY MASTER CATALOG PARAMETER'. The operator's reply indicates one of the following:

   - NIP should use the default member SYSCATLG in SYS1.NUCLEUS to find the master catalog.

   - NIP should use an alternate member SYSCATnn in SYS1.NUCLEUS to find the master catalog.

     The operator's procedures for specifying an alternate master catalog is described in Operations: System Commands

At system generation time, the installation uses the OPI parameter to indicate whether the operator may override system parameters in IEASYSxx members of SYS1.PARMLIB. The installation may restrict operator intervention for individual system parameters in the IEASYSxx member or for the entire set

of system parameters. However, if a parameter in an IEASYSxx member is in error and the installation has restricted operator intervention for the parameter, NIP prompts the operator for a correct parameter value regardless of the installation's restriction. The procedure for restricting operator intervention during NIP is described in Initialization and Tuning

When IEAVNIPM calls IEAVNP20, TOD clock initialization allows the operator to enter the Greenwich mean time, the local time, and the local date; these are required to set the TOD clock(s), and the local time and date for the system. When a TOD clock is ready to be set to the correct time, TOD clock initialization requests that the operator enable the clock security switch.

Note: In the reply used to set the TOD clock or the local time and date, the operator can choose to specify a new value for the IPS parameter. The IPS parameter indicates the member of SYS1.PARMLIB that contains the installation performance specification for the system resource manager (SRM).

During master scheduler initialization, SMF initialization allows the operator to enter correct SMF parameters to replace those parameters that are invalid or missing.

REAL AND VIRTUAL STORAGE DURING INITIALIZATION

The initial program loader (IEAIPL00) is a stand-alone program consisting of three IPL records. It resides on cylinder 0, track 0 of the system residence volume. In preparation for loading IEAIPL00, the operator mounts the system residence volume on a direct access device. Then the operator initiates LOAD processing. In response to this initiation the hardware reads the first IPL record into real storage locations 0 through 23 (see Figure 1-4). The doubleword beginning at location 8 is a read CCW (channel command word) that causes the loading of the second IPL record at a real storage address higher than the size of the IPL text record. The transfer-in-channel (TIC) command at location 16 in the first record causes control to pass to the second record.

The second IPL record is a chain of CCWs that cause the fourth record, containing the IPL text, to be read into real storage beginning at location 0. This IPL text record overlays the first record. When the last CCW in the second record has been executed, the processor loads the doubleword beginning at location 0 as a new PSW, thus passing control to the IPL text (fourth record).

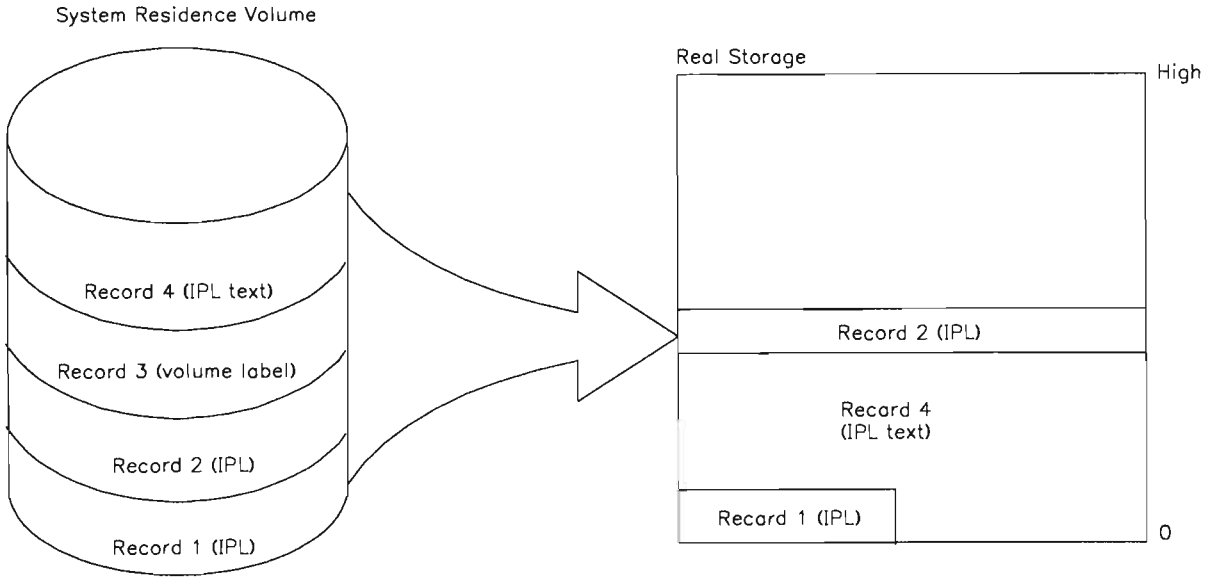


Figure 1-4. Real Storage During IPL Loading

The following storage maps describe virtual storage during the initialization process and indicate the modules that allocate storage for the major system areas and begin to initialize these areas. Figure 1-5 describes virtual storage at the conclusion of the IPL phase of initialization. Figure 1-6 describes virtual storage immediately before the NIP control program, IEAVNIPM, begins to execute. Figure 1-7 describes virtual storage at the conclusion of the NIP phase of initialization.

For other maps of virtual storage during initialization, see Figure 1-2, which shows the virtual storage layout for multiple address spaces, and Figure 5-3, which shows the virtual storage on exit from IEAIPL04.

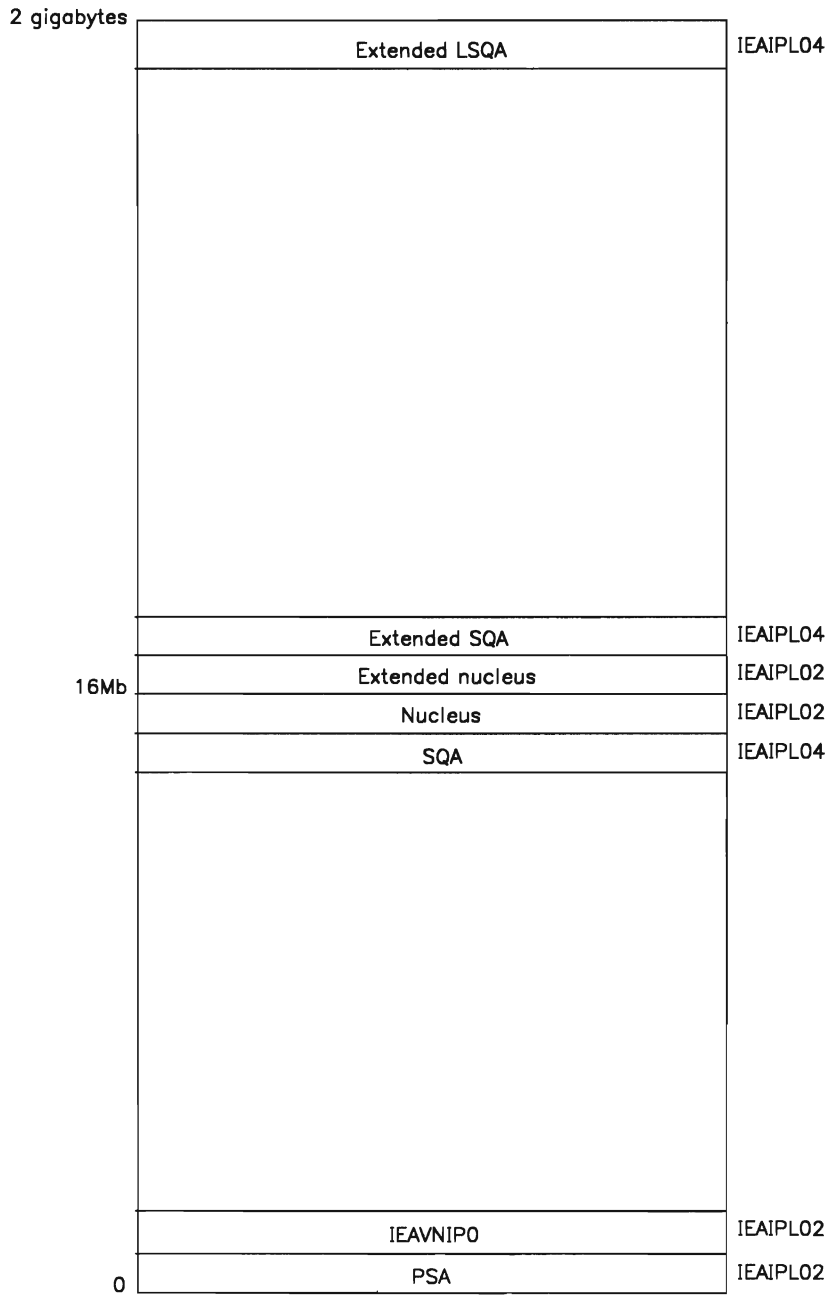


Figure 1-5. Virtual Storage on Exit from IEAIPL99

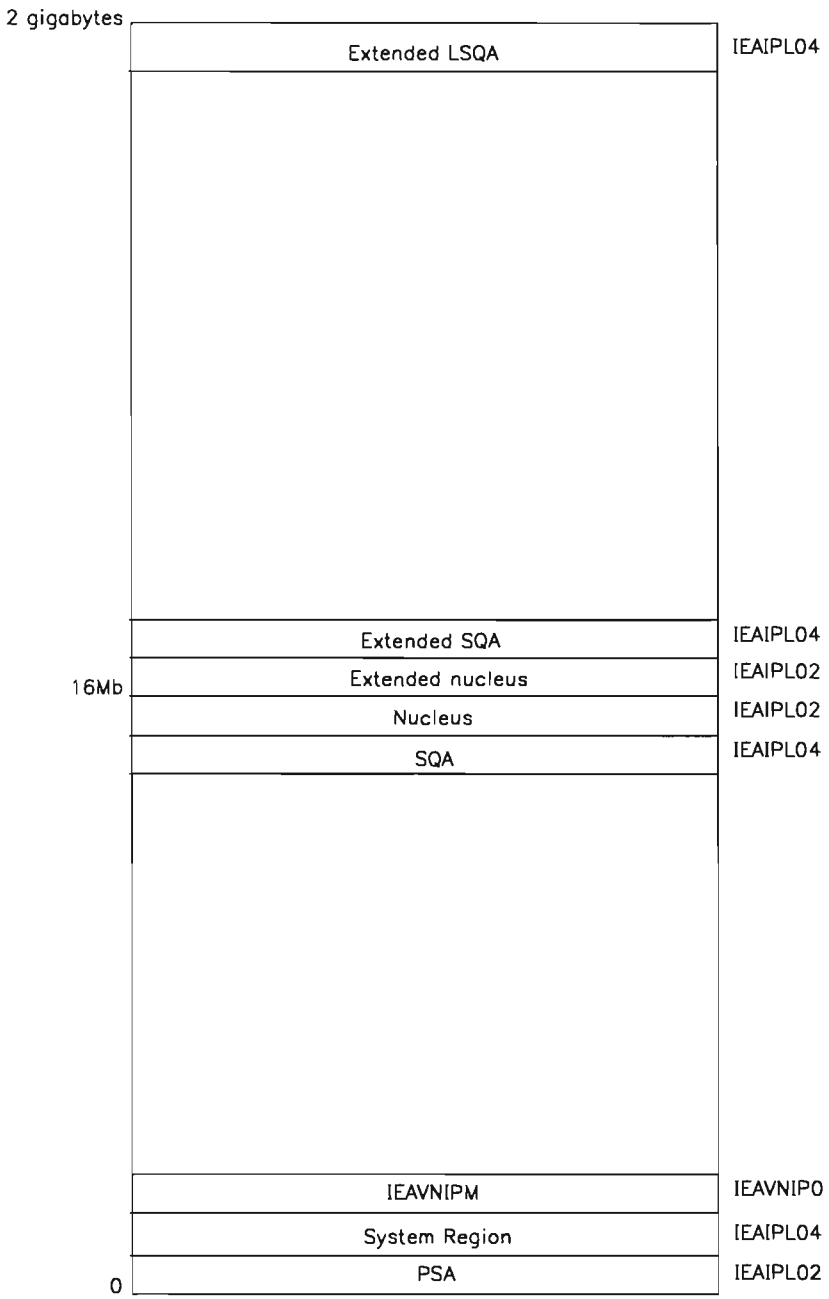


Figure 1-6. Virtual Storage on Exit from IEAVNIP0

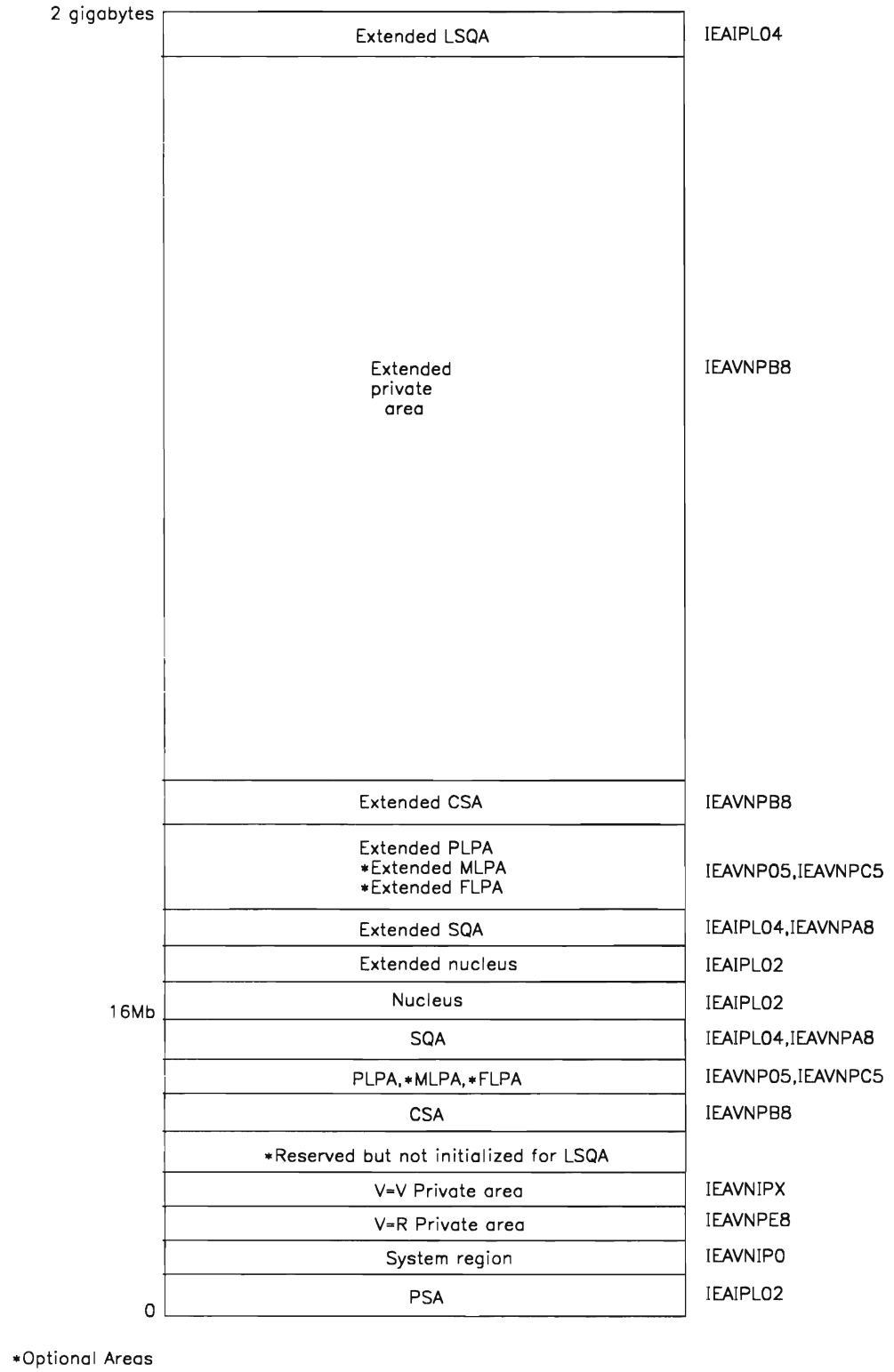


Figure 1-7. Virtual Storage on Exit from IEAVNIPX

SECTION 2. DATA AREAS

This section briefly describes the data areas used during IPL and NIP.

The following Microfiche publications contain more information on the data areas used during IPL and NIP processing.

Data Areas

- LYB8-1191 for MVS/SP V2 JES2
- LYB8-1195 for MVS/SP V2 JES3

Data Area Usage Table

- LYB8-1193 for MVS/SP V2 JES2
- LYB8-1197 for MVS/SP V2 JES3

Symbol Usage Table

- LYB8-1192 for MVS/SP V2 JES2
- LYB8-1196 for MVS/SP V2 JES3

DATA AREA DESCRIPTIONS

This section includes brief descriptions of the following data areas:

IPL Data Areas

IPL vector table (IVT)
Nucleus map (NUCMAP)
Message queue element (MQE)
Message queue header (MQH)
Nucleus load list element (NLLE)

NIP Data Areas

Extended quick start record (EQSRD)
NIP parameter address table (PARMTAB)
NIP parameter area (NIPPAREA)
NIP system parameter queue entry (SPE)
NIP vector table (NVT)
NIPMOUNT parameter list
NIPOPEN parameter list
NIPWTO message header
NIPWTOR parameter list
Quick start record (QSRCD)
Quick start record extension (XQSRD)

Master Scheduler Data Areas

Master trace table (MTT)

The layouts of the above data areas, except the system parameter queue entry are in the publication MVS/XA Debugging Handbook. The layout of the system parameter queue is in this section.

IPL Vector Table (IVT)

POINTED TO BY: Register 1 on entry to IRIMs and IEAVNIP0.

MAPPING MACRO: IHAIVT

USE: Used for communication among the IRIMs, IEAIPL00, and IEAVNIP0.

LENGTH: 336 bytes

Nucleus Map (NUCMAP)

POINTED TO BY: CVTNUCMP.

MAPPING MACRO INSTRUCTION: IEANUCMP.

USE: Describes all DAT-on nucleus csects and entry points. For details of format and order, see Diagram 3: NUCMAP Creation in Section 2.

LENGTH: Length of each entry (4 bytes) x number of entry points in the nucleus.

Message Queue Element (MQE)

POINTED TO BY: MQHIST and MQHNTH

MAPPING MACRO: IHAMQE

USE: Contains messages issued during IPL

LENGTH: 10 bytes + message length. Maximum messaga length is 125 bytes.

Message Queue Header (MQH)

POINTED TO BY: IVTMQHP during IPL and NVTMQHP during NIP

MAPPING MACRO: IHAMQH

USE: MQH is the message header for the IPL message queue.

LENGTH: 12 bytes

Nucleus Load List Element (NLLE)

POINTED TO BY: IVTNLLEF, IVTNLLEL and NLLNEXT

MAPPING MACRO: IEANLLE

USE: Contains the SYS1.NUCLEUS member name of the module and the work area for the nucleus loading amd mapping routine.

LENGTH:

Extended Quick Start Record (EQSRD)

POINTED TO BY: Field NVTQSBUF + length of QSRCD (the EQSRD is contiguous to the QSRCD).

MAPPING MACRO: ILREQSRD.

USE: Contains information about the extended pageable link pack area; used for quick start and warm start initialization.

LENGTH: 8192 bytes.

NIP Console Table (IEAVNCTxx)

POINTED TO BY:

MAPPING MACRO:

USE: Built by MVSCP. Lists terminals that NIP is allowed to use as consoles. NIP uses this list to find a NIP console, xx. The "xx" is determined by the load parameter from SYS1.PARMLIB member CONSOLxx. The default is "00."

LENGTH:

NIP Parameter Address Table (PARMTAB)

POINTED TO BY: Field NVTPTAB in the NVT.

MAPPING MACRO: IEAPPNIP.

USE: Contains addresses of system parameter values.

LENGTH:

NIP Parameter Area (NIPPAREA)

POINTED TO BY: Field NVTPAREA in the NVT.

MAPPING MACRO: IEAPPNIP.

USE: Contains workarea for reading SYS1.PARMLIB members.

LENGTH: 526 bytes.

NIP System Parameter Queue Entry (SPE)

POINTED TO BY: The first SPE is in field NVTSPE of the NVT.

MAPPING MACRO: None.

USE: Tracks SQA buffers that contain WTOR replies.

LENGTH: Variable

NIP Vector Table (NVT)

POINTED TO BY: Register 2 (RNVT).

MAPPING MACRO: IHANVT.

USE: Used for communication among NIP modules; contains information such as NIP wait state codes, pointers to nucleus control blocks, storage allocation values, NIP save areas, system status flags, PSW descriptors, and addresses of IEAVNIPM service routines.

LENGTH:

NIPMOUNT Parameter List

POINTED TO BY: Register 1 (RPARM).

MAPPING MACRO INSTRUCTION: IEAPMNIP TYPE=MOUNTPL.

USE: Passes information to the NIPMOUNT service routine.

LENGTH:

NIPOPEN Parameter List

POINTED TO BY: Register 1 (RPARM).

MAPPING MACRO INSTRUCTION: IEAPMNIP TYPE=OPENPL.

USE: Passes information to the NIPOPEN service routine.

LENGTH:

NIPWTO Message Header

POINTED TO BY: Register 1 (RPARM) or, if NIPWTOR request, NIPWTOR parameter list.

MAPPING MACRO INSTRUCTION: IEAPMNIP TYPE=HEADER.

USE: Passes information to the NIPWTO service routine.

LENGTH:

NIPWTOR Parameter List

POINTED TO BY: Register 1 (RPARM).

MAPPING MACRO INSTRUCTION: IEAPMNIP TYPE=PLIST.

USE: Passes information to the NIPWTOR service routine.

LENGTH: 16 bytes.

Quick Start Record (QSRCD)

POINTED TO BY: Field NVTQSBUF in the NVT.

MAPPING MACRO: ILRQSRCD.

USE: Contains information about the pageable link pack area; used for quick and warm start initialization.

LENGTH: 8192 bytes.

Quick Start Record Extension (XQSRD)

POINTED TO BY: QSRXQSR in the QSRCD.

MAPPING MACRO: ILRXQSRD.

USE: For quick and warm start initialization, contains PLPA slot information that must be save across IPLs.

LENGTH: 4096 bytes.

Master Trace Table (MTT)

POINTED TO BY: Field BAMTTBL in the master scheduler resident data area (MSRDA) control block. This control block is located in the nucleus and it is pointed to by field CVTMSER in the CVT.

MAPPING MACRO: IEEZB806

USE: Contains messages that are eligible for hard copy. The table is a wraparound table that is filled from the end to the beginning.

LENGTH: 16K to 999K.

SECTION 3. DIAGNOSTIC AIDS

This section contains information that can be used to diagnose problems in system initialization programs:

- TCB structure — This section gives a visual representation of the tasks that are attached by the master scheduler initialization and remain as permanent system tasks.
- IPL diagnostic techniques — This section describes some of the causes of disabled wait states.
- IEAVNIP0 diagnostic trap routines — This section describes the routines that trap hardware and software errors during IEAVNIP0, with the wait state codes used to indicate a particular error.
- IEAVNIPM diagnostic trap routines — This section describes the routines that trap hardware and software errors before the system recovery routines are available. It describes the actions taken at the time of the error, the wait state code if NIP terminates, and the values to which the trap routines are reset at the end of NIP processing.
- NIP Wait State Code X'64' — This section describes the conditions that cause a wait state code X'64' with a reason code of 5 or 9.

Additional diagnostic information can be found in the following publications:

- Message Library: System Messages, which contains message IDs and an explanation about why the message was produced.
- Message Library: System Codes, which contains wait state codes and an explanation why they are produced.
- System Programming Library: Diagnostic Techniques, which contains methods for diagnosing system problems.

TCB Structure After Initialization

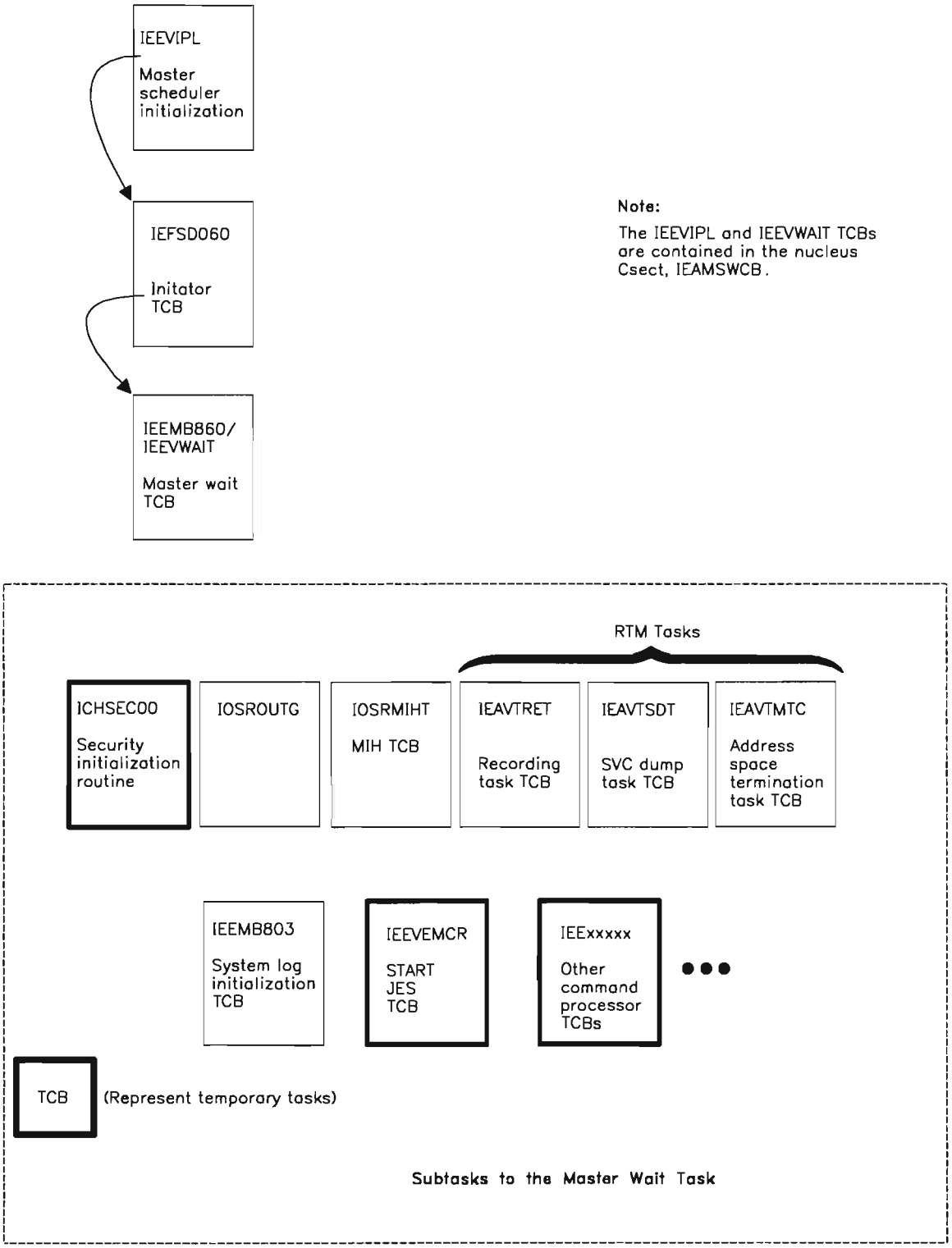


Figure 3-1. TCB Structure in the Master Scheduler Region Following System Initialization

IPL DIAGNOSTIC TRAP TECHNIQUES

The IPL component contains no explicit recovery or retry routines for program errors. Detection of a software error during IPL leaves no reasonable alternative but to enter a disabled wait state.

IPL does try to recover from a class of correctable I/O errors. Uncorrectable I/O errors, however, cause a disabled wait state.

The contents of the old PSWs are particularly useful for diagnosis of IPL errors because the wait state is entered so soon after the error occurs. A stand-alone dump is also required whenever the cause of a wait state cannot be determined from the operator's console.

There is other diagnostic information available during IPL in the following real storage locations:

Loc X'10' Pointer to the frame queue (list of good frames)
Loc X'14' Pointer to the IPL diagnostic area
Loc X'4C' Pointer to the failing CCW, if a unit check occurred
Loc X'54' Four sense bytes that describe the unit check

The IPL diagnostic area, pointed to by real location X'14', has the following format:

Hex Offset	Length	Description
+ 00	8	Name of the module most recently given control
+ 08	8	Name of SYS1.NUCLEUS member that IPL most recently tried to find
+ 10	4	Pointer to the start of the SVC stack
+ 14	4	Pointer to the current element in the SVC stack
+ 18	4	Pointer to the address of the location just beyond the end of the SVC stack
+ 1C	4	Length of an SVC stack element
+ 20	4	Virtual address of the IRIM area
+ 24	4	Real address of the IRIM area
+ 28	4	Virtual address of the IVT
+ 2C	4	Real address of the IVT
+ 30	4	Real address of the most recent SCPINFO response
+ 34	4	Address of the subchannel information block
+ 38	4	Address of the sense data (meaningful only after a unit check)

The SVC stack area is used to save status on an SVC interrupt. Each entry, which is 32 bytes long, contains:

- The SVC old PSW at the time of interrupt
- The SVC interruption code (SVC number)
- The contents of registers 1-5 at the time of interrupt

There is a pointer set to the top stack element, the entry currently in use. This pointer must be updated (by adding the length of an entry) before the status for a new interrupt can be stored on the stack. While an SVC routine is processing, the pointer indicates the stack entry that contains the saved status of the requestor of this SVC. An exception to this general processing occurs when the SVC is the EXIT SVC (SVC 3); status is not saved when this SVC is issued. Instead, IPL removes the top element from the stack and makes it current (restores the registers and loads the old PSW) so control passes back to the point at which the SVC was issued. Because of the way the stack is used, the pointer is initially set to point 32 below the bottom of the stack. IPL checks for stack overflow and stack underflow, either of which causes a disabled wait state (X'074').

An SVC stack entry has the following format:

Hex Offset	Length	Description
+ 00	8	PSW at time of interrupt
+ 08	2	Interruption code

+ 0A	2	Reserved
+ 10	20	Registers 1-5 at the time of interrupt

The nucleus map (NUCMAP) is not included in a dump unless NUCMAP is specified on a request for print dump service aid.

For a list of the SVCs that are available at IPL-time, see "Initial Program Loader" in Section 1.

There is a nucleus map lookup service, NUCLKUP, which can be used either to (1) retrieve the address and AMODE of a nucleus CSECT or entry point or (2) retrieve the name and address of the nucleus CSECT pointed to by a given address from the nucleus lookup routine, IEAVENLU. For more information on NUCLKUP, see SPL: System Macros and Facilities.

IEAVNIP0 DIAGNOSTIC TRAP ROUTINES

Because system recovery routines are not available while IEAVNIP0 processes, IEAVNIP0 provides diagnostic support for software and hardware errors by trapping the errors. The IEAVNIP0 trap routines stop the initialization process at the time the error occurs and provide diagnostic information in the form of wait state codes.

ABEND Trap (NIPABEND)

DESCRIPTION OF TRAP: IEAVNIP0 saves the ABEND SVC entry from the SVCTABLE (set up at SYSGEN time) in the NVT and replaces the entry with the address of NIPABEND. IEAVNIP0 also changes the SVC type to 1.

OPERATION: Traps requests for ABEND and loads a wait state PSW to terminate NIP.

OUTPUT: Wait state PSW contains wait state code X'30' in bits 56-63 and a completion code, if any, in bits 36-47.

EXIT TO: System wait state.

Machine Check Trap (NVTMCPSW)

DESCRIPTION OF TRAP: IEAVNIP0 saves the machine check PSW and replaces it with a PSW that points to the next sequential instruction with machine checks enabled.

OPERATION: Traps machine checks and loads a wait state PSW to terminate NIP.

OUTPUT: Wait state PSW contains wait state code X'44' in bits 56-63, and the logical address of the processor in bits 32-47.

EXIT TO: System wait state.

Recovery Termination Management Traps (NP0RTP01-15)

DESCRIPTION OF TRAP: IEAVNIP0 replaces the addresses of the routines in the RTM1 table with the addresses of trap routines NP0RTP01 through NP0RTP15.

ENTRY FROM: Recovery termination management.

OPERATION: Traps calls to recovery termination management that occur during IEAVNIP0 and loads a wait state PSW to terminate IEAVNIP0.

OUTPUT: Wait state PSW contains an index that represents the requested entry in the RTM1 table and wait state code X'64'.

EXIT TO: System wait state.

Unexpected Program Check Trap (NP0UXPC)

DESCRIPTION OF TRAP: IEAVNIP0 saves the SYSGEN program check new PSW and replaces it with a PSW that points to a BR 10 instruction; it sets register 10 to the address of NP0UXPC.

OPERATION: Traps unexpected program checks and loads a wait state PSW to terminate IEAVNIP0.

OUTPUT: Wait state PSW contains wait state code X'46' in bits 56-63.

EXIT TO: System wait state.

STAE/ESTAE (SVC 60) Trap

DESCRIPTION OF TRAP: IEAVNIP0 saves the STAE/ESTAE SVCTABLE entry in the NVT and replaces it with the address of NOP code. IEAVNIP0 also changes the SVC type to 1.

OPERATION: Traps requests for STAE or ESTAE and clears register 15.

OUTPUT: Register 15 contains zeroes.

EXIT TO: Caller of STAE or ESTAE.

IEAVNIPM DIAGNOSTIC TRAP ROUTINES

Before system recovery routines are available, IEAVNIPM provides diagnostic support for software and hardware errors by trapping the errors. The trap routines detect errors that occur during NIP, stop the initialization process at the time the error occurs, and provide diagnostic information in the form of wait state codes and messages.

The following table lists the trap routines, the action taken by NIP, and the module that provides the corresponding regular system service.

Trap Routine	Resolution	Regular System Service Begins After
ABEND	Wait State X'40' or X'30'	IEAVNIPX
Machine Check	Wait State X'044'	IEAVNP06
RTM	Wait State X'064'	IEAVNIPX
STAE/ESTAE	BR14 (No-op)	IEAVNPA6
ENQ/DEQ	BR14 (No-op)	IEAVNP23
SVC Types 3 & 4	Wait State X'065'	IEAVNP05

Because RTM has not been initialized, NIP does not issue any ABENDs or MEMTERMs, nor does it request any type of dump (SDUMP or ABDUMP). Therefore, a stand-alone dump is required to diagnose any problem for which the cause cannot be determined from the wait state code and the diagnostic data provided with it.

ABEND Trap (NIPABEND)

DESCRIPTION OF TRAP: IEAVNIPM places the address of the NIPABEND routine in the SVC table entry for the ABEND SVC (SVC 13). IEAVNIPM also changes the SVC type to 2 so that the contents supervisor RIM (IEAVNP05) will not alter the entry when it initializes type 3 and type 4 SVCs in the table. IEAVNP05 initializes the SVC 13 entry point address in an entry in the NVT; at the end of NIP processing, IEAVNIPX resets the SVC table entry from the trap routine address to the SVC 13 entry point address.

ENTRY FROM: SVC SLIH.

OPERATION: Traps ABEND requests and provides diagnostic information about the ABEND.

INPUT: Register 1 contains the ABEND code.

OUTPUT: Message IEA303W contains the ABEND code X'40'; the wait state PSW contains the ABEND code and the wait state code.

EXIT TO: NIPSWAIT service routine.

Type 3 and 4 SVC and LOCATE SVC Trap (NIPSVC)

DESCRIPTION OF TRAP: IEAVNIPM places the address of the NIPSVC routine in the SVC table entries for all type 3 and type 4 SVCs. After the LPA is constructed, the contents supervisor RIM (IEAVNP05) replaces the entries with addresses of the proper SVC routines.

Although IEAVNIPM does not treat LOCATE SVC 26 requests as errors, it does trap LOCATE requests. IEAVNIPM places the address of NIPSVC in the SVC table entry for the LOCATE SVC (SVC 26) and changes the SVC type to 2. After NIP processing is complete, IEAVNIPX resets the SVC 26 entry with the address of the LOCATE SVC routine.

ENTRY FROM: SVC SLIH.

OPERATION: Traps any type 3 and type 4 SVC request during NIP and prepares NIP for termination; traps LOCATE SVC requests and passes control to the NIP LOCATE module, IEAVNP12.

INPUT: Register 5 points to the RB that contains the SVC code.

OUTPUT: For type 3 and type 4 SVC requests (except LOCATE), field NPMFLWSC of the PSW structure (NPMWTPSW) in IEAVNIPM contains wait state code X'65' and field NPMIDPSW contains the SVC code.

EXIT TO: For type 3 and type 4 SVC requests, return to NIPSWAIT service routine; for LOCATE SVC (SVC 26) requests, return to issuer of LOCATE.

Recovery Termination Management Trap (NPRTMTAB)

DESCRIPTION OF TRAP: IEAVNIPM replaces the address of the recovery termination management table (RTM1) -- located in field CVTBTERM of the CVT -- with the address of the table in NPRTMTAB. IEAVNIPX resets the trap prior to the end of NIP.

OPERATION: NPRTMTAB consists of a table of addresses that point to code that diagnoses each entry to recovery termination management for software recovery. NPRTMTAB traps all entries to recovery termination management.

INPUT: Register 1 contains the entry code identifier.

OUTPUT: Field NPMIDPSW of the PSW structure (NPMWTPSW) in IEAVNIPM contains a number identifying the RTM1 entry called; field NPMFLWSC contains wait state code X'64'.

EXIT TO: NIPSWAIT service routine.

DESCRIPTION OF WAIT STATE CODE X'64'

A wait state code of X'64' indicates that an attempt was made to enter RTM during NIP. A reason code is also supplied with this wait state code (PSW bits 44 to 51). The most common codes encountered are 5 (SVC error) and 9 (program check).

Wait state code X'64' with a reason code of 5 occurs when an SVC instruction is issued in an invalid mode (such as SRB or locked mode). However, the cause is usually not locked mode, because it is common for system routines to issue an ABEND (SVC 13) while holding a lock. Therefore, the first thing to do with a wait state code of X'64', with a reason code of 5, is to determine if an SVC 13 was the last SVC issued (look at SVC OLD PSW or trace table).

If SVC 13 was the last SVC, the R1 field in the trace table contains the ABEND code and the R15 field contains the reason code. The ABEND code might give some clue as to which SVC issued the ABEND. If so, examination of the trace table can show what module issued the SVC that caused the ABEND. It may be that the input to that SVC was the cause of the error.

If a wait state code of X'64' with a reason code of 5 occurs and SVC 13 was not the last SVC issued, then the error is a true SVC error. It could be that a module is erroneously issuing an SVC while holding a lock. However, it is also possible that some routine that was previously in control returned to its caller without releasing a lock. In this case, the SVC-issuer is correct.

A wait state code of X'64' with a reason code of 9 indicates that a program check (other than a resolvable translation exception) has occurred. The program check OLD PSW indicates the type of exception and where it occurred. The registers at the time of the exception are saved, with X'48' placed in the LCCA. If the error was a translation exception (program interrupt code of X'10' or X'11'), location X'90' contains the address that could not be translated. The trace table also contains this information.

SECTION 4. PROGRAM ORGANIZATION

This section describes the physical organization of the object modules that perform system initialization functions. Figure 4-1 through Figure 4-15 show the control flow among the object modules during system initialization.

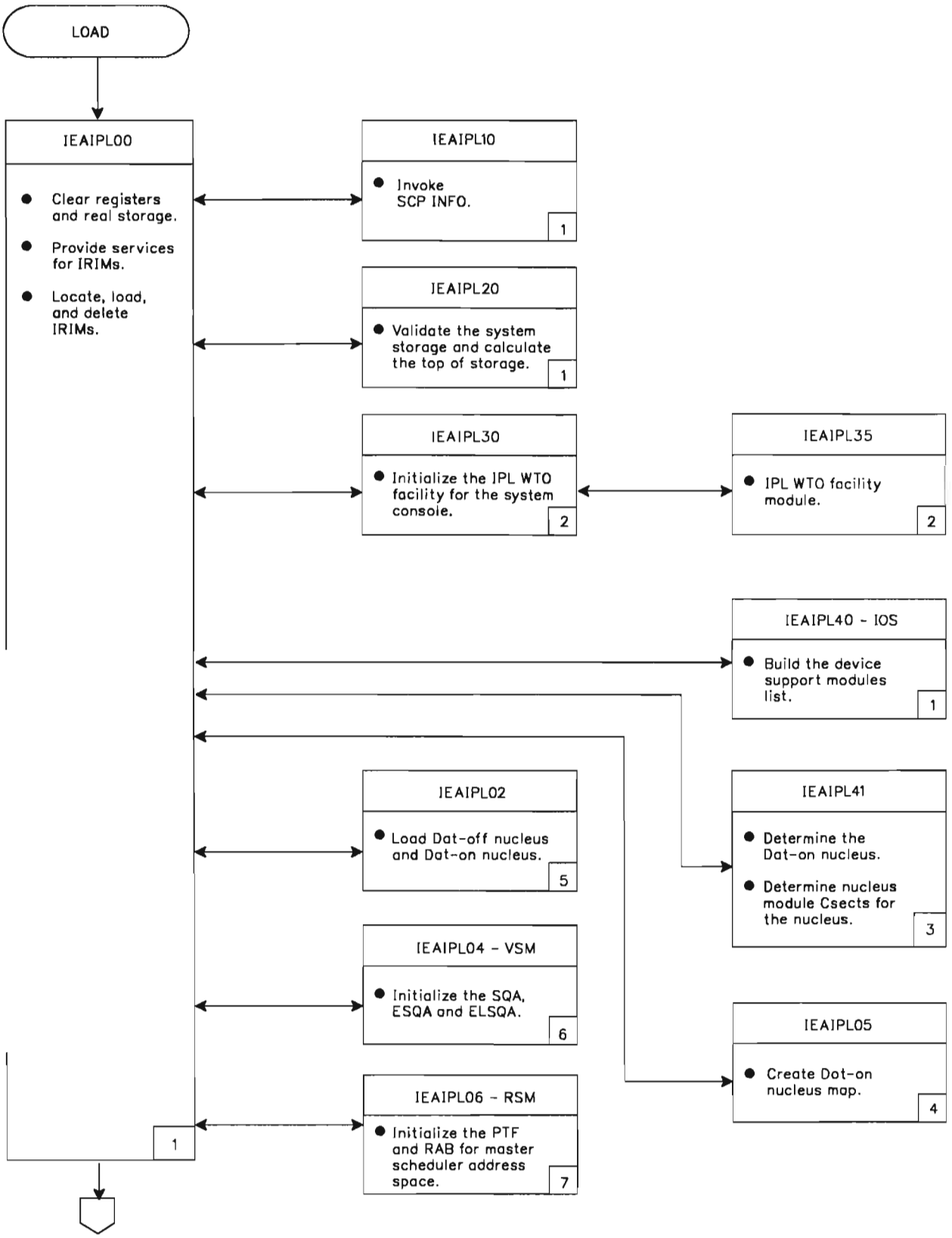


Figure 4-1 (Part 1 of 13). System Initialization Module Flow

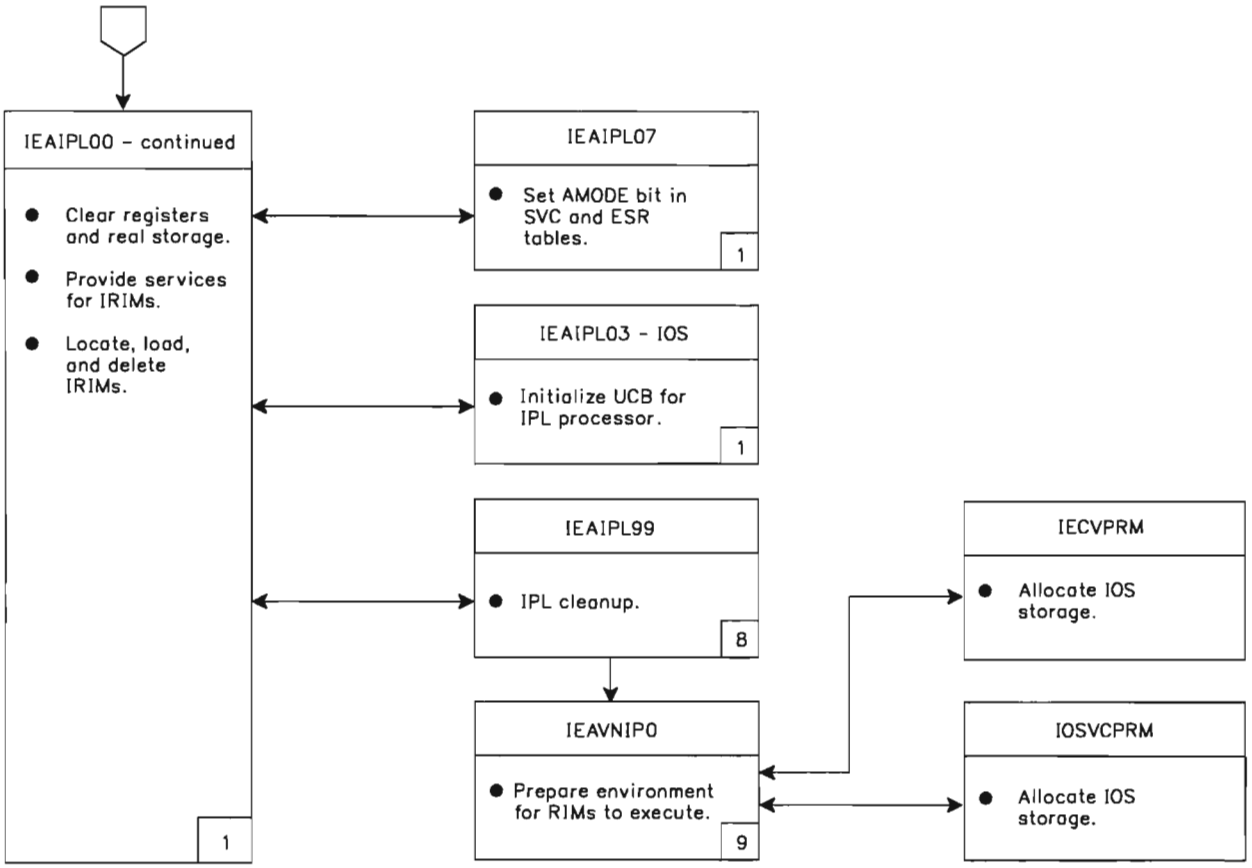


Figure 4-1 (Part 2 of 13). System Initialization Module Flow

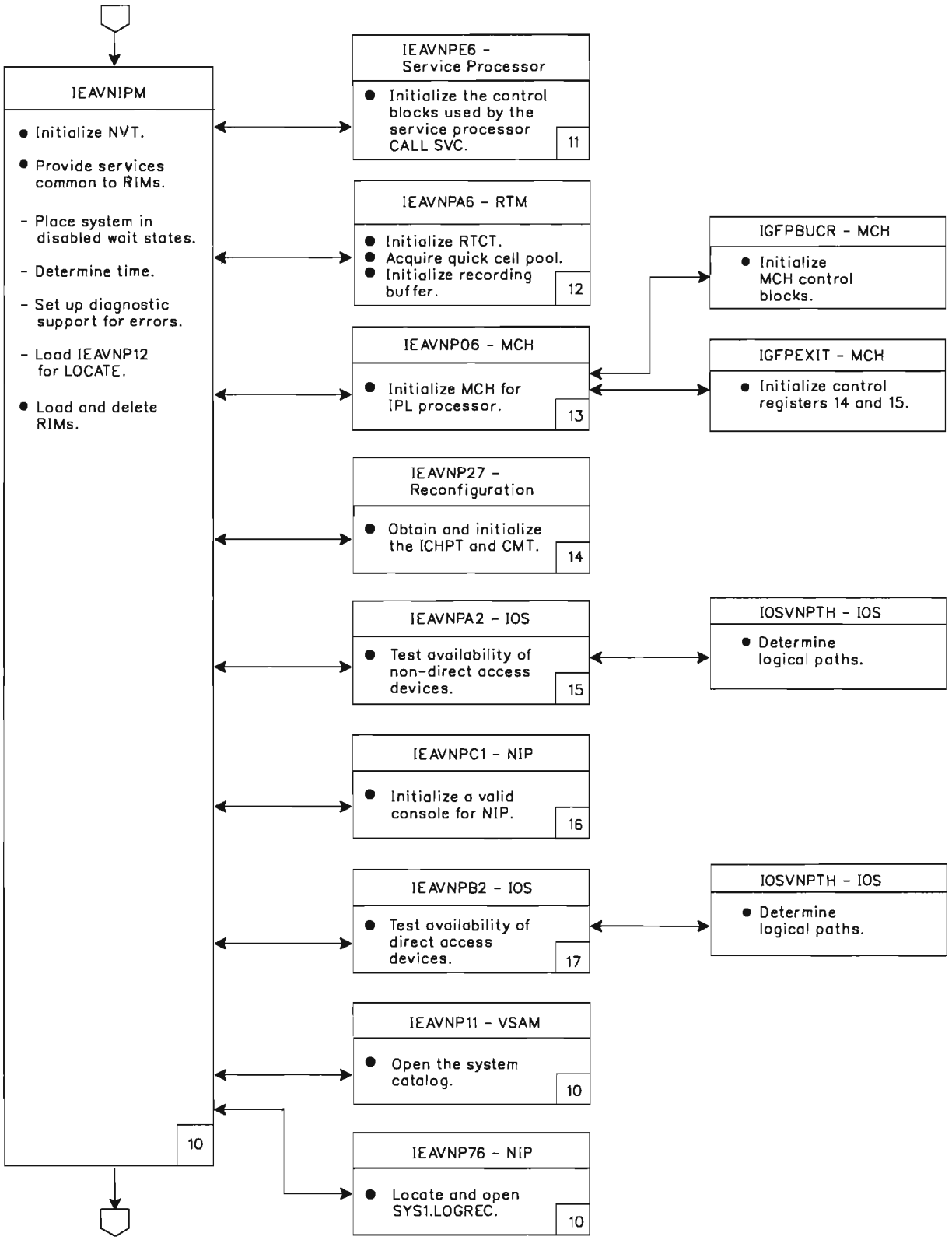


Figure 4-1 (Part 3 of 13). System Initialization Module Flow

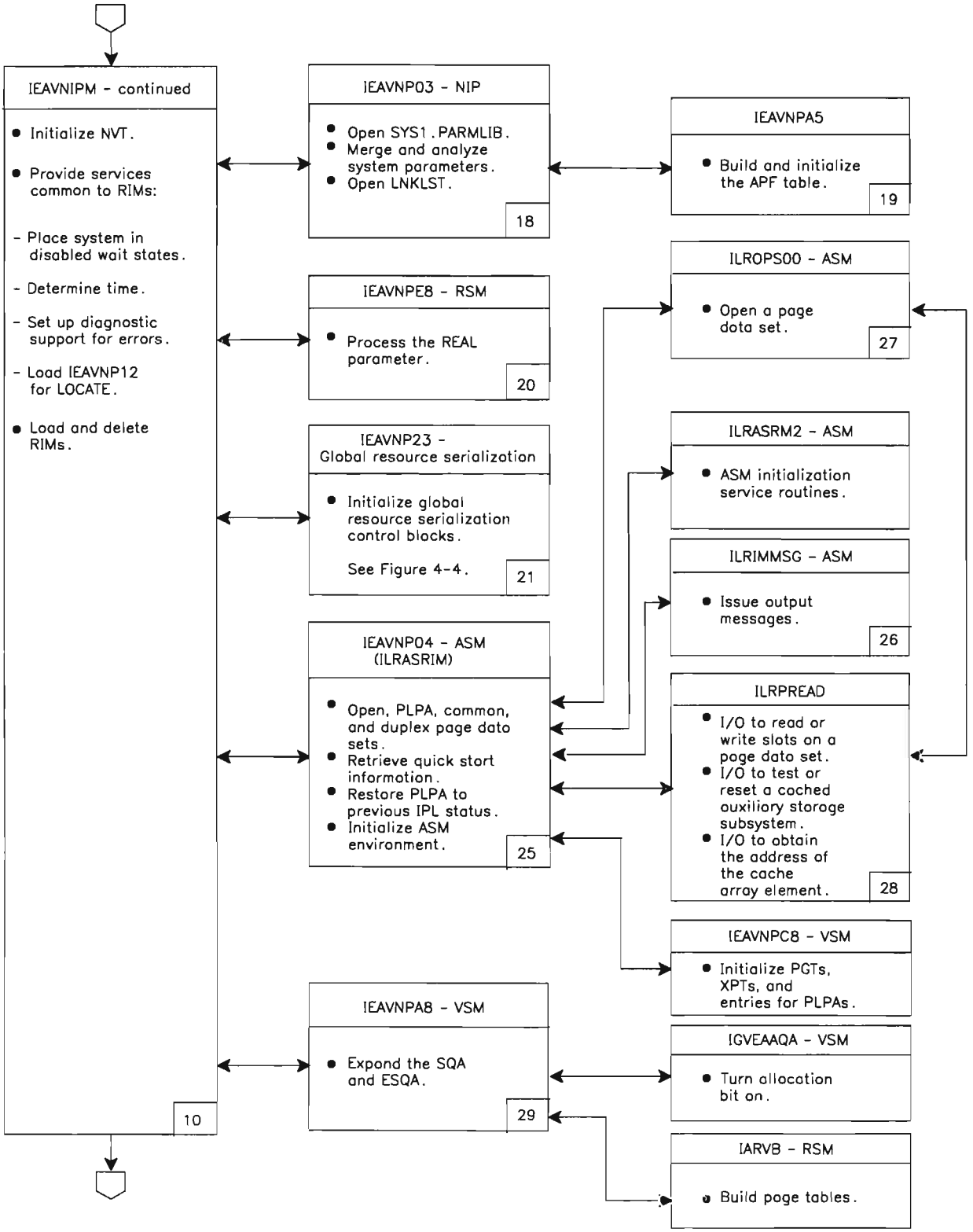


Figure 4-1 (Part 4 of 13). System Initialization Module Flow

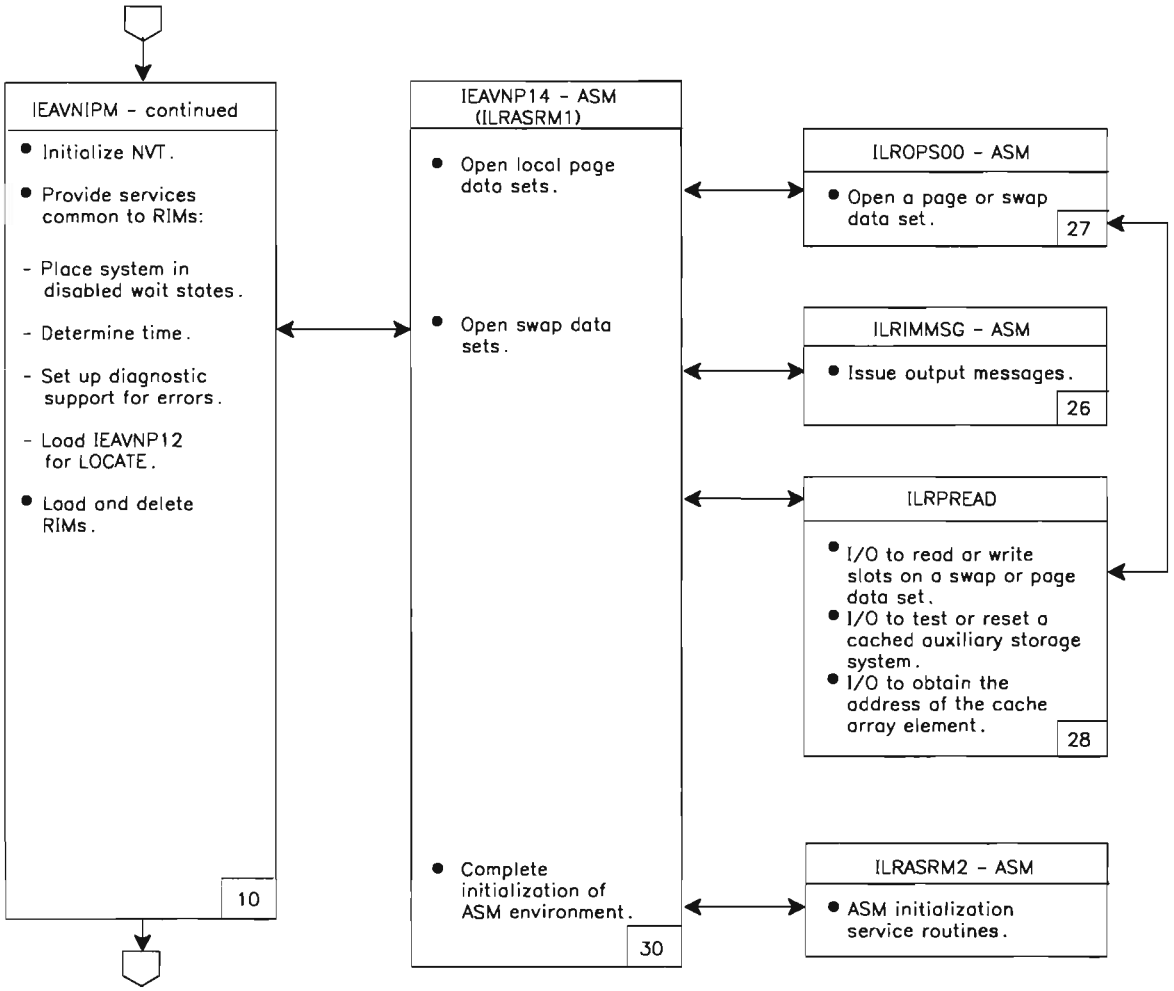


Figure 4-1 (Part 5 of 13). System Initialization Module Flow

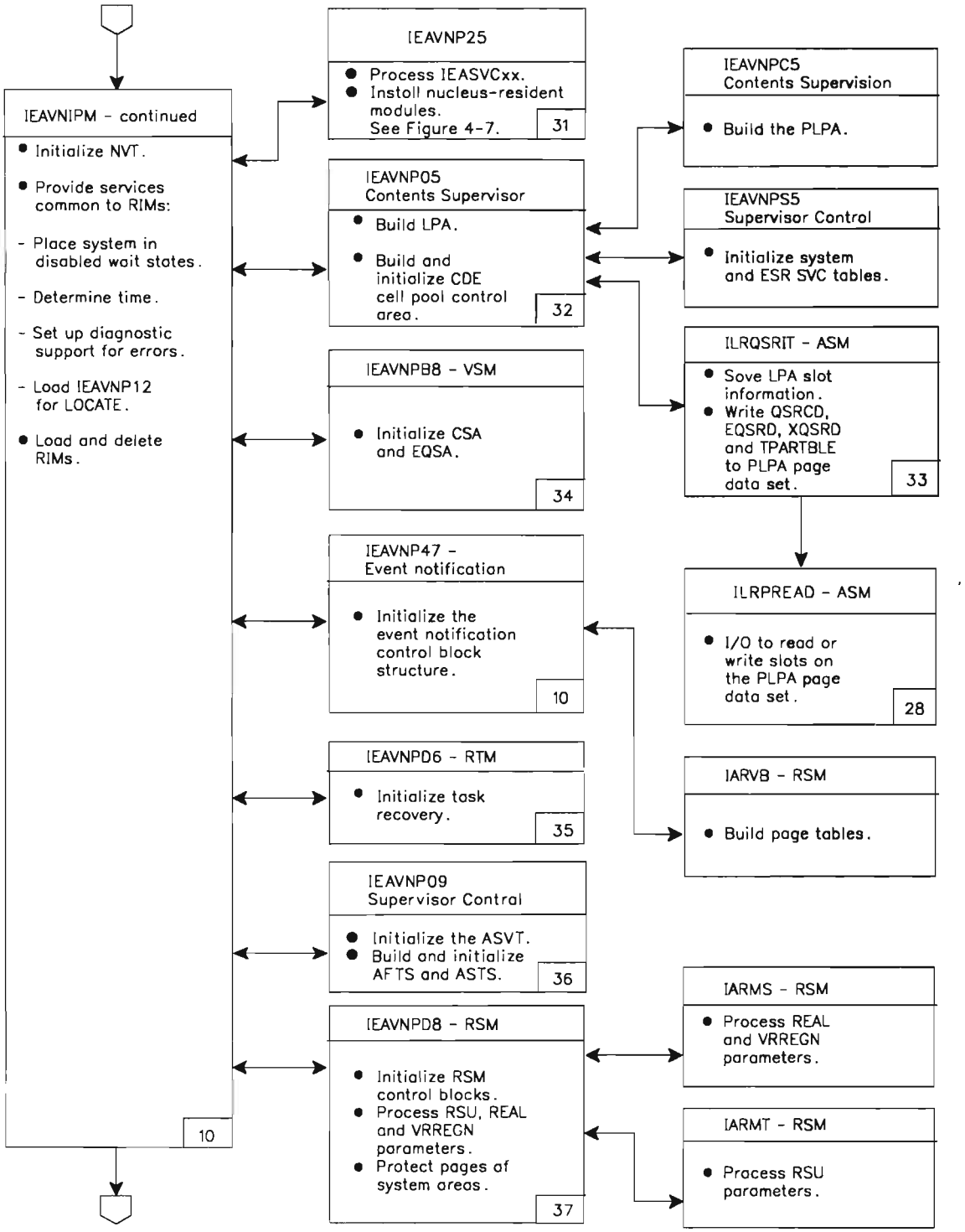


Figure 4-1 (Part 6 of 13). System Initialization Module Flow

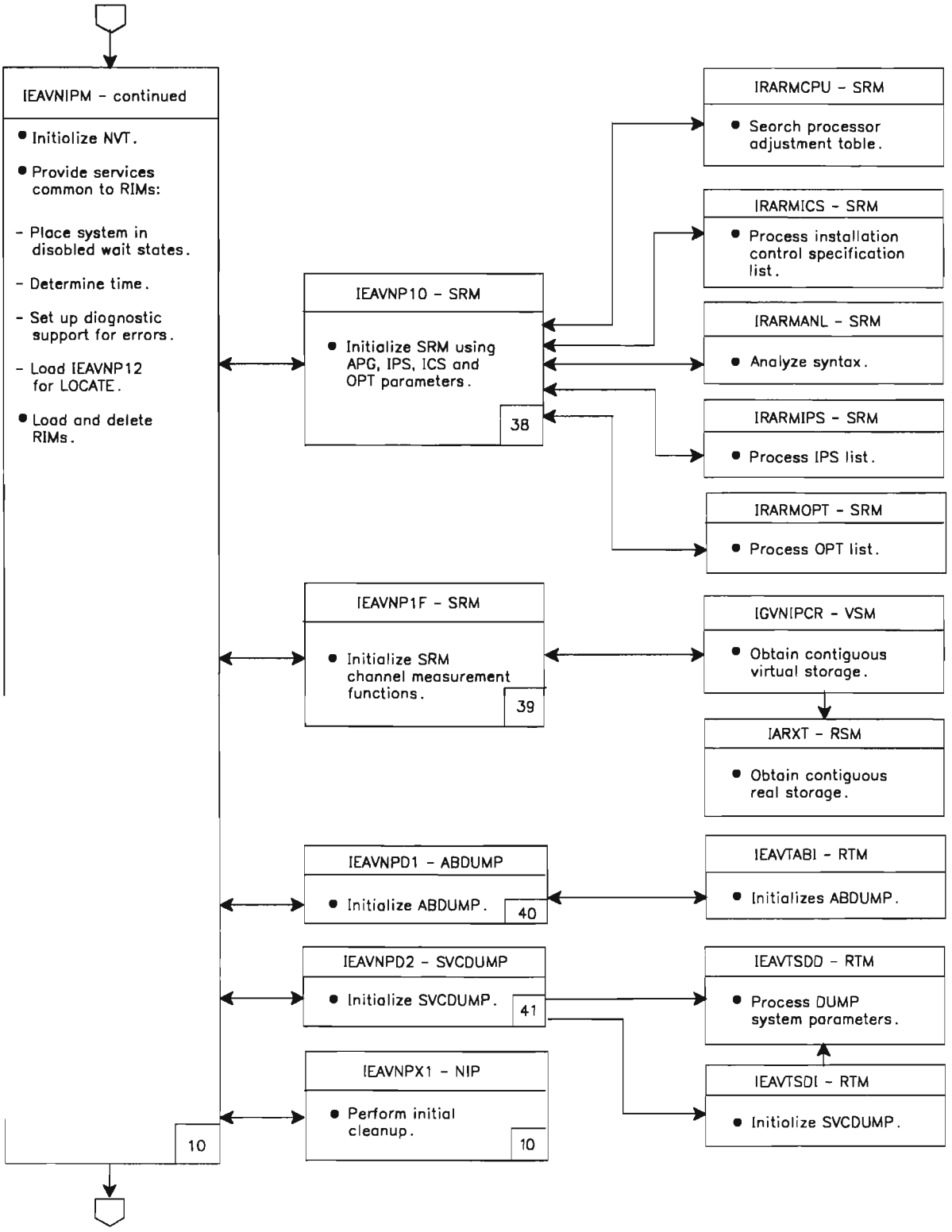


Figure 4-1 (Part 7 of 13). System Initialization Module Flow

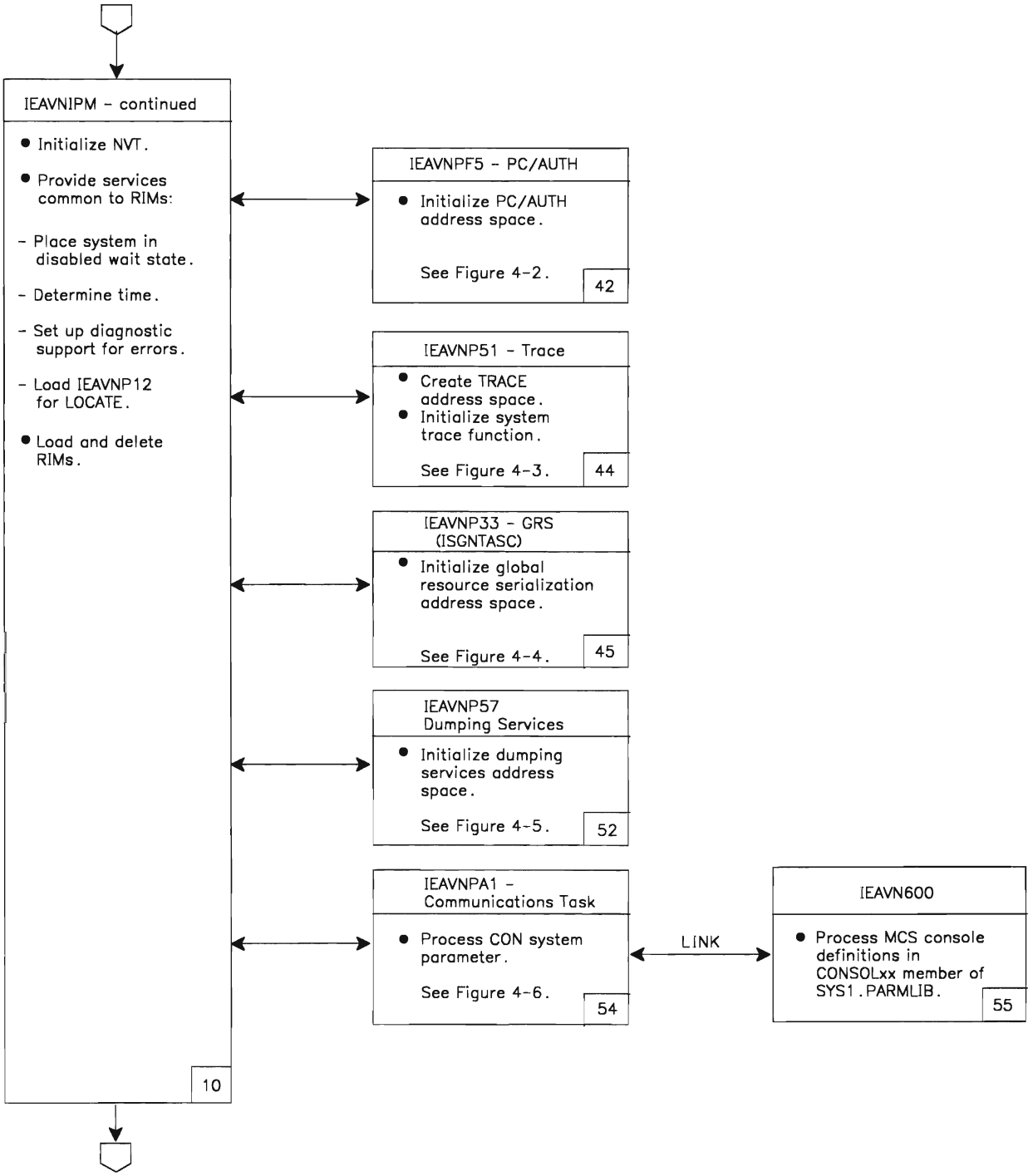


Figure 4-1 (Part 8 of 13). System Initialization Module Flow

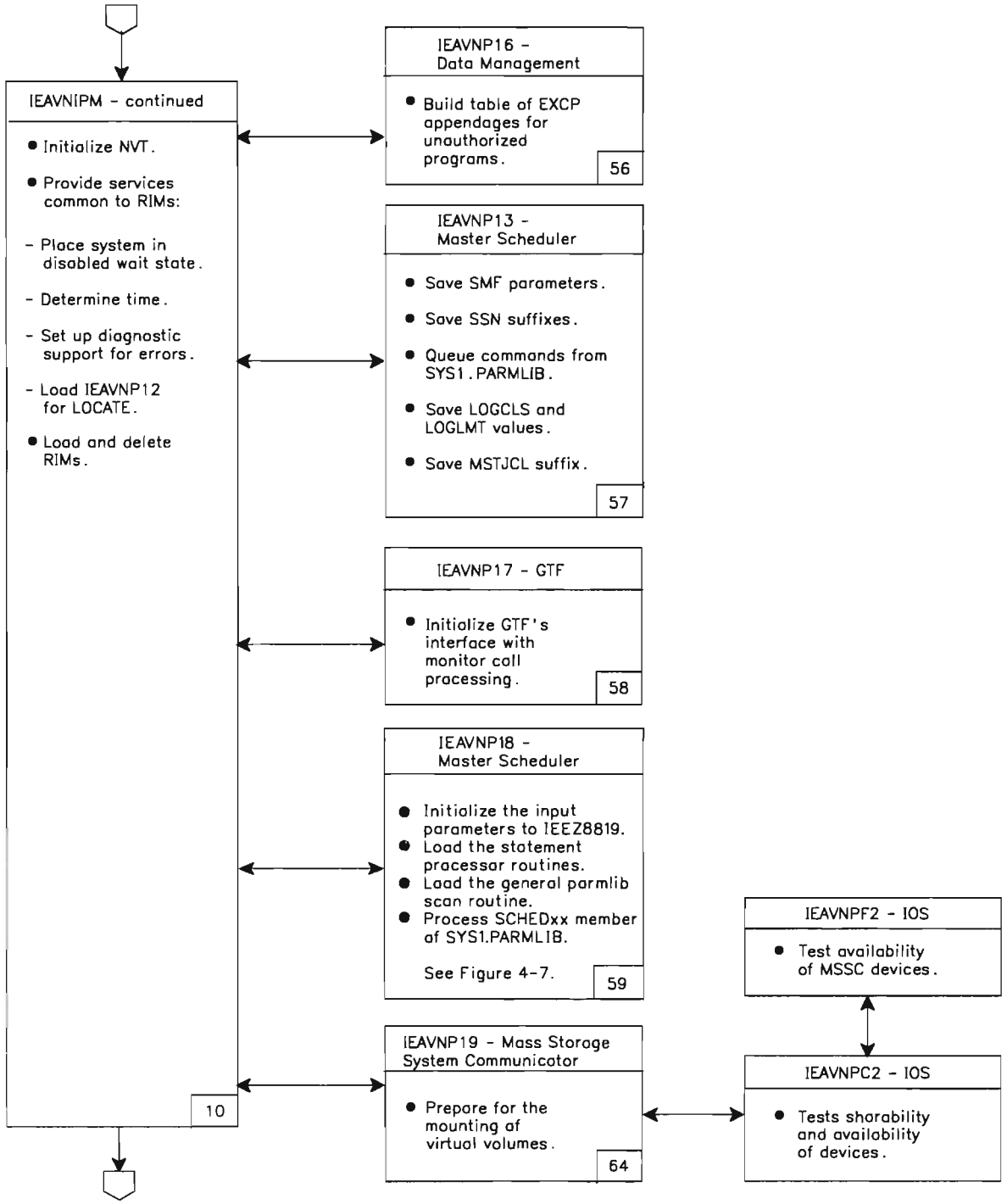


Figure 4-1 (Part 9 of 13). System Initialization Module Flow

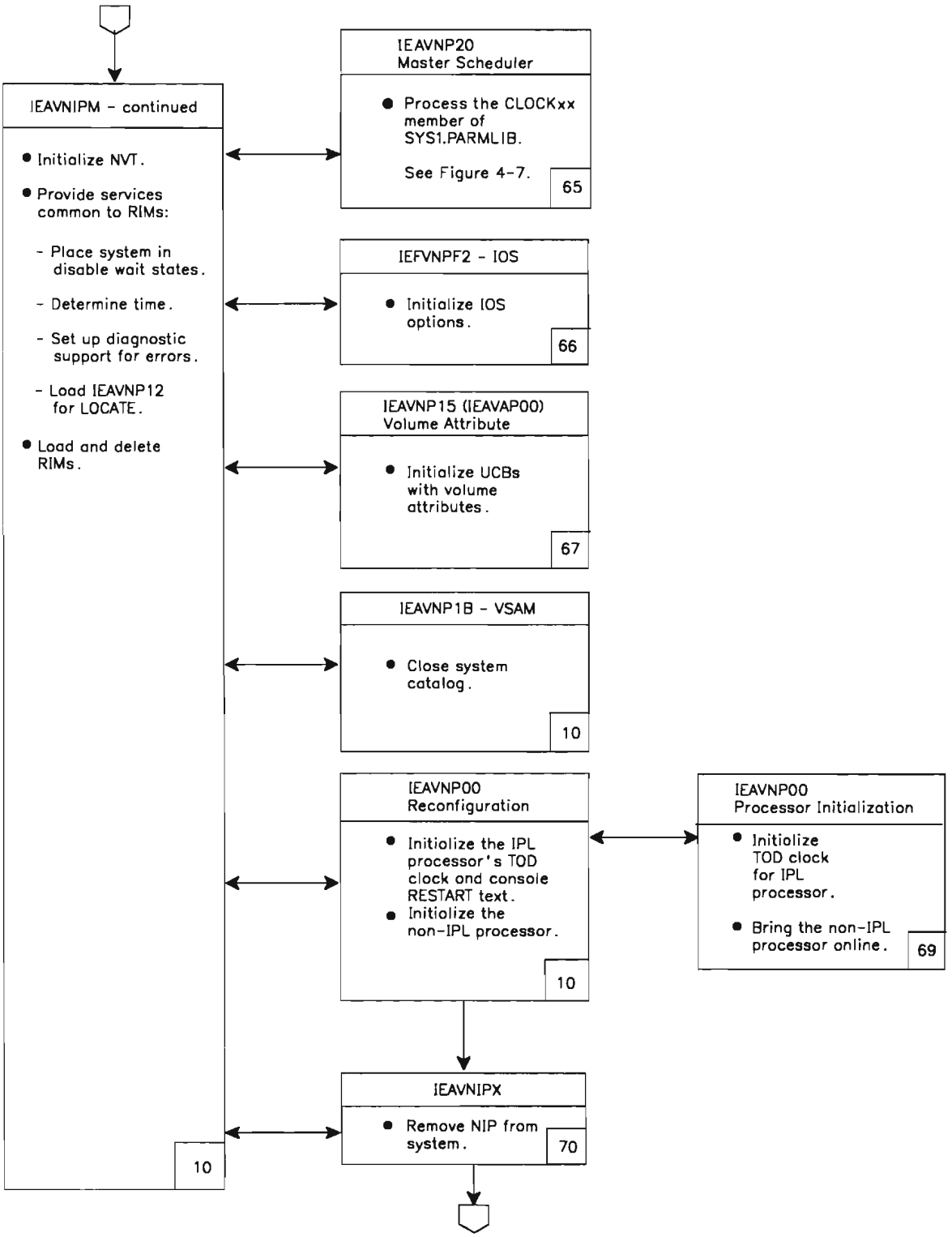


Figure 4-1 (Part 10 of 13). System Initialization Module Flow

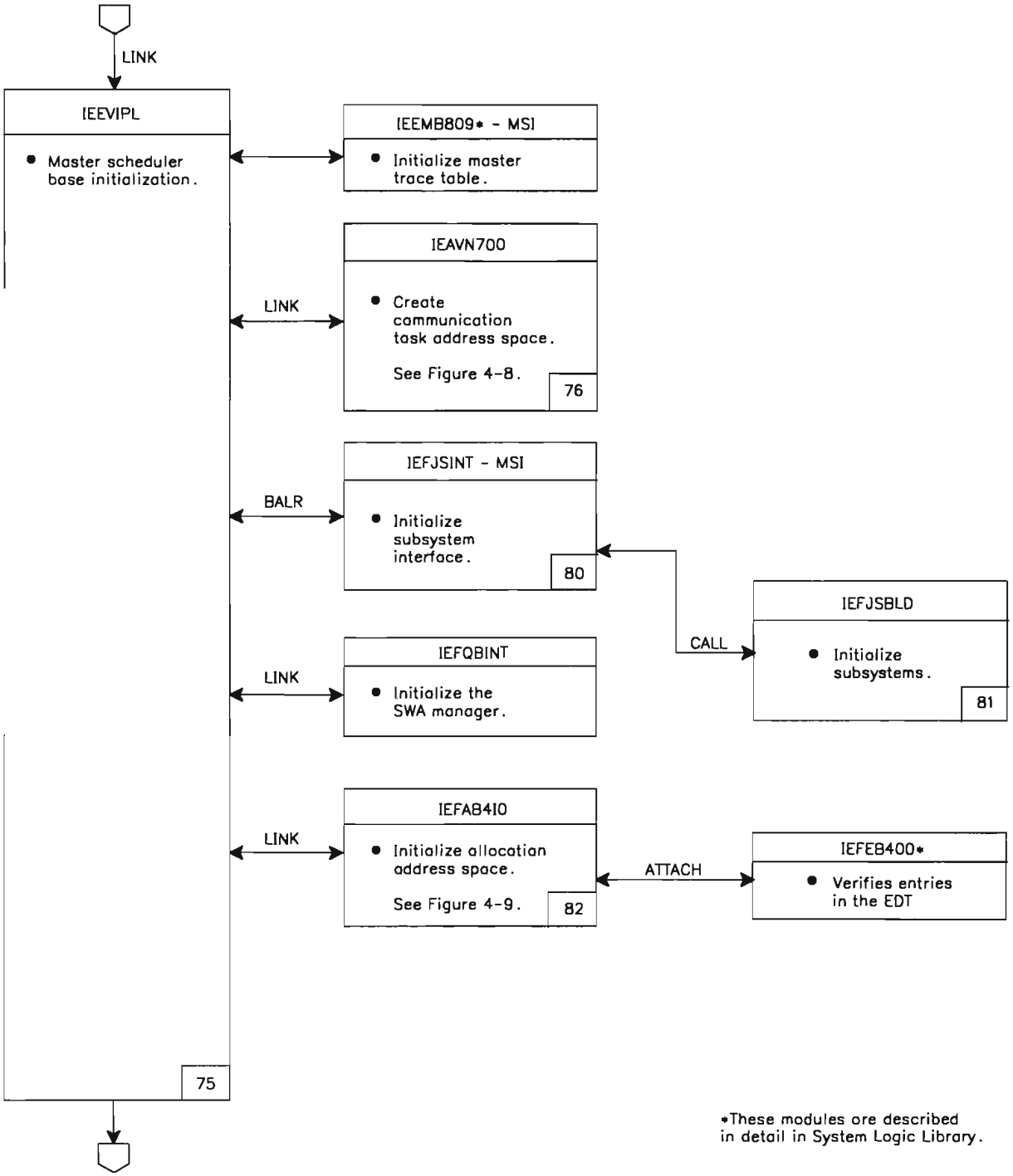


Figure 4-1 (Part 11 of 13). System Initialization Module Flow

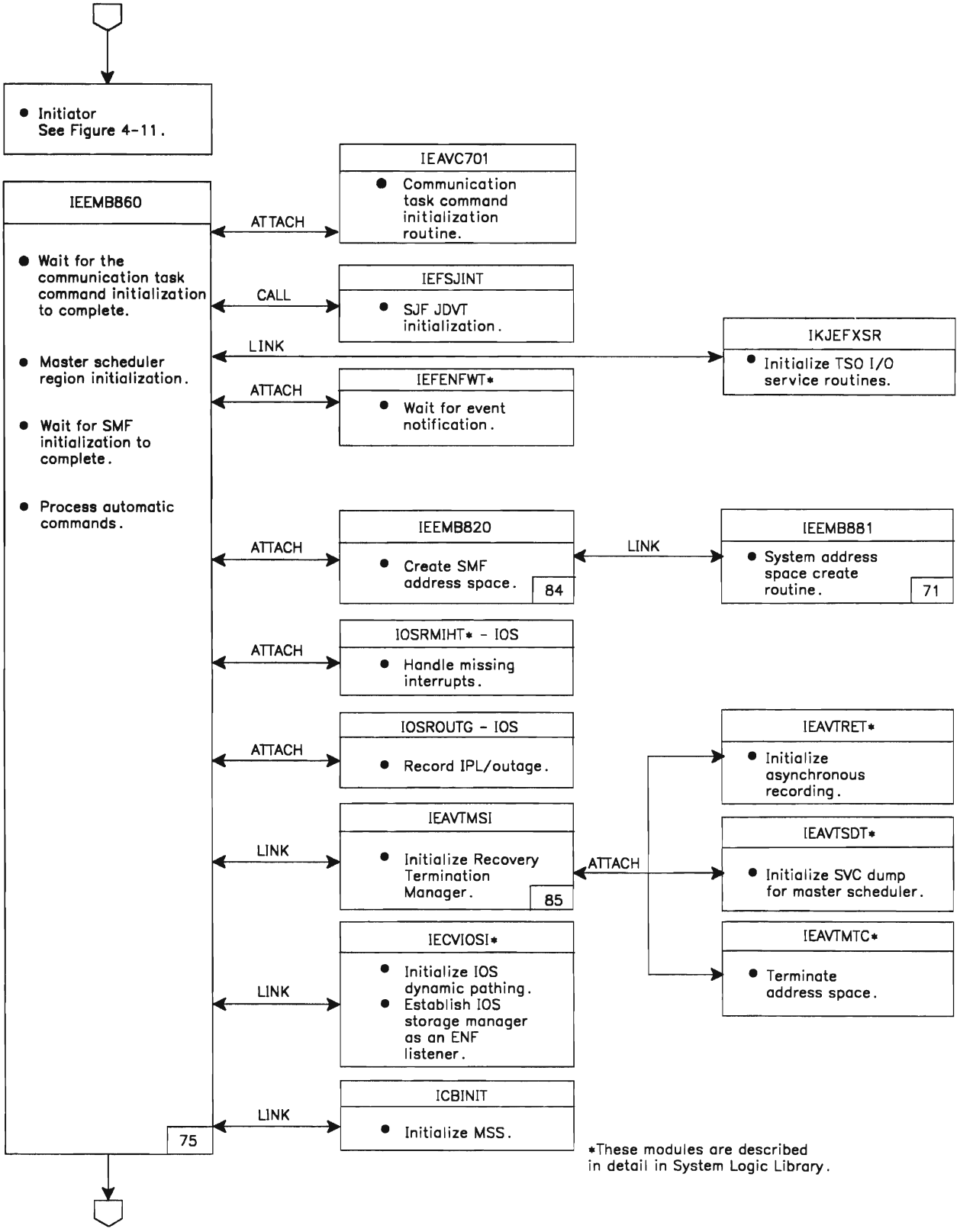


Figure 4-1 (Part 12 of 13). System Initialization Module Flow

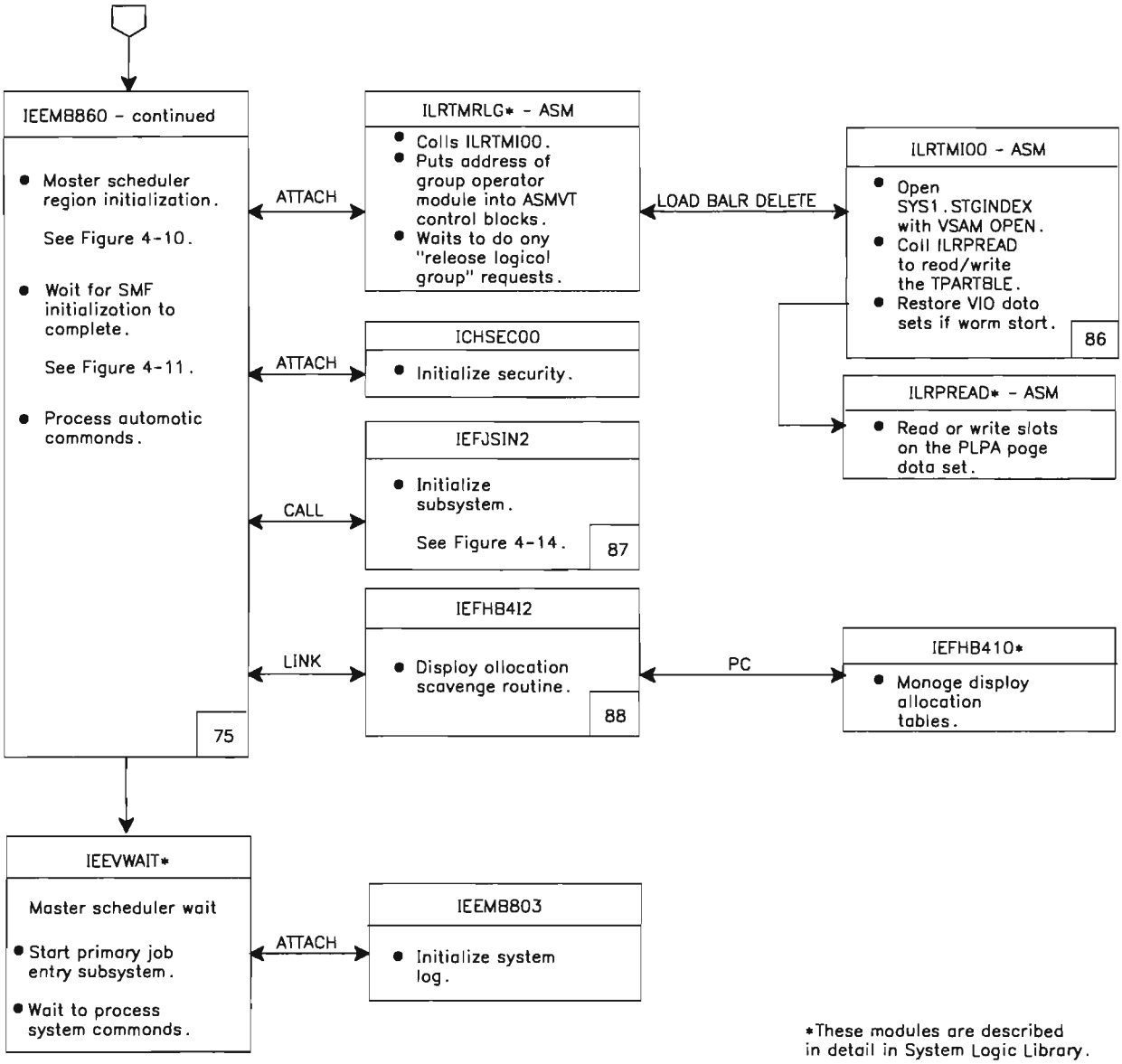


Figure 4-1 (Part 13 of 13). System Initialization Module Flow

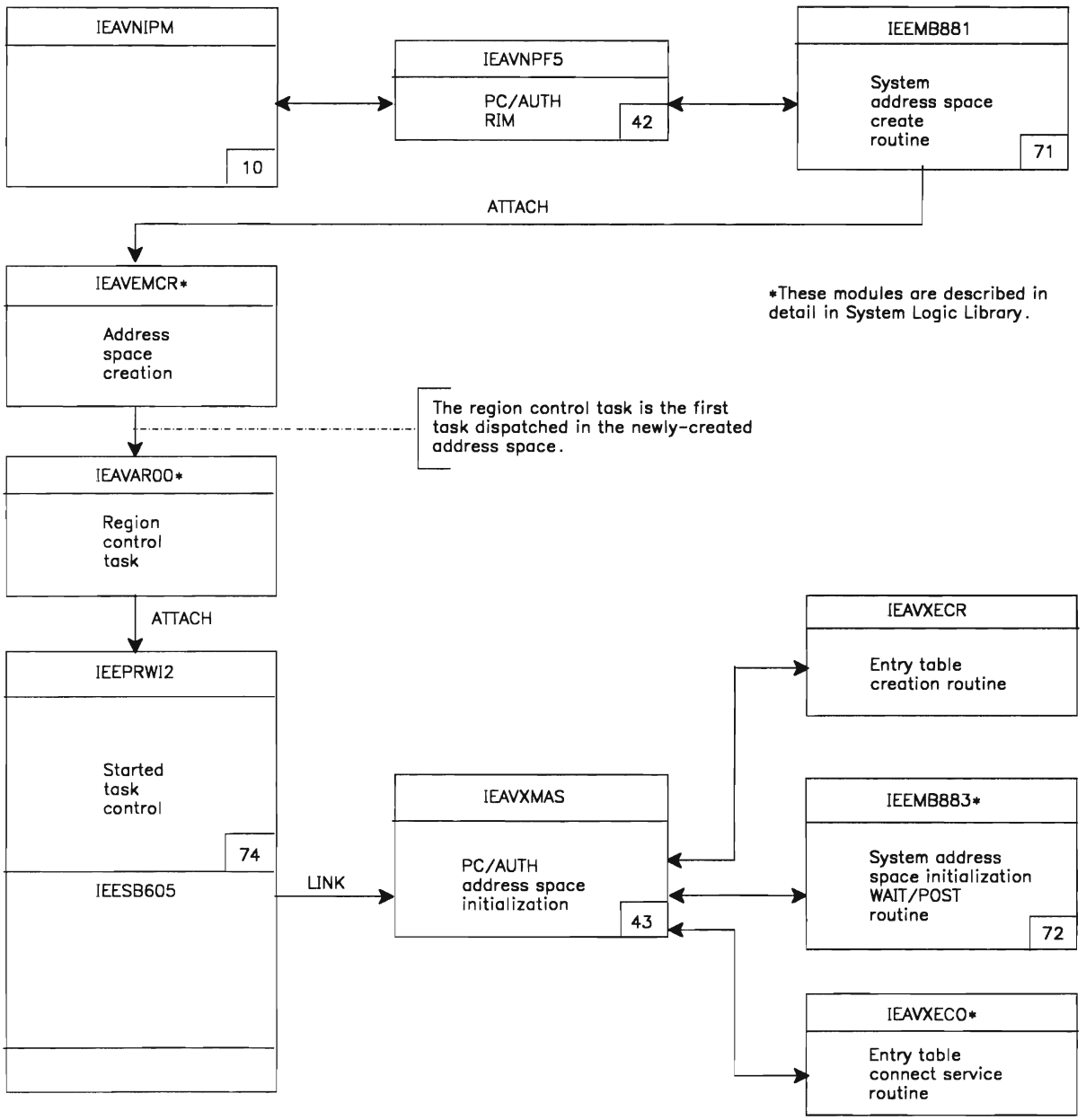


Figure 4-2. Initializing the Program Call/Authorization Address Space

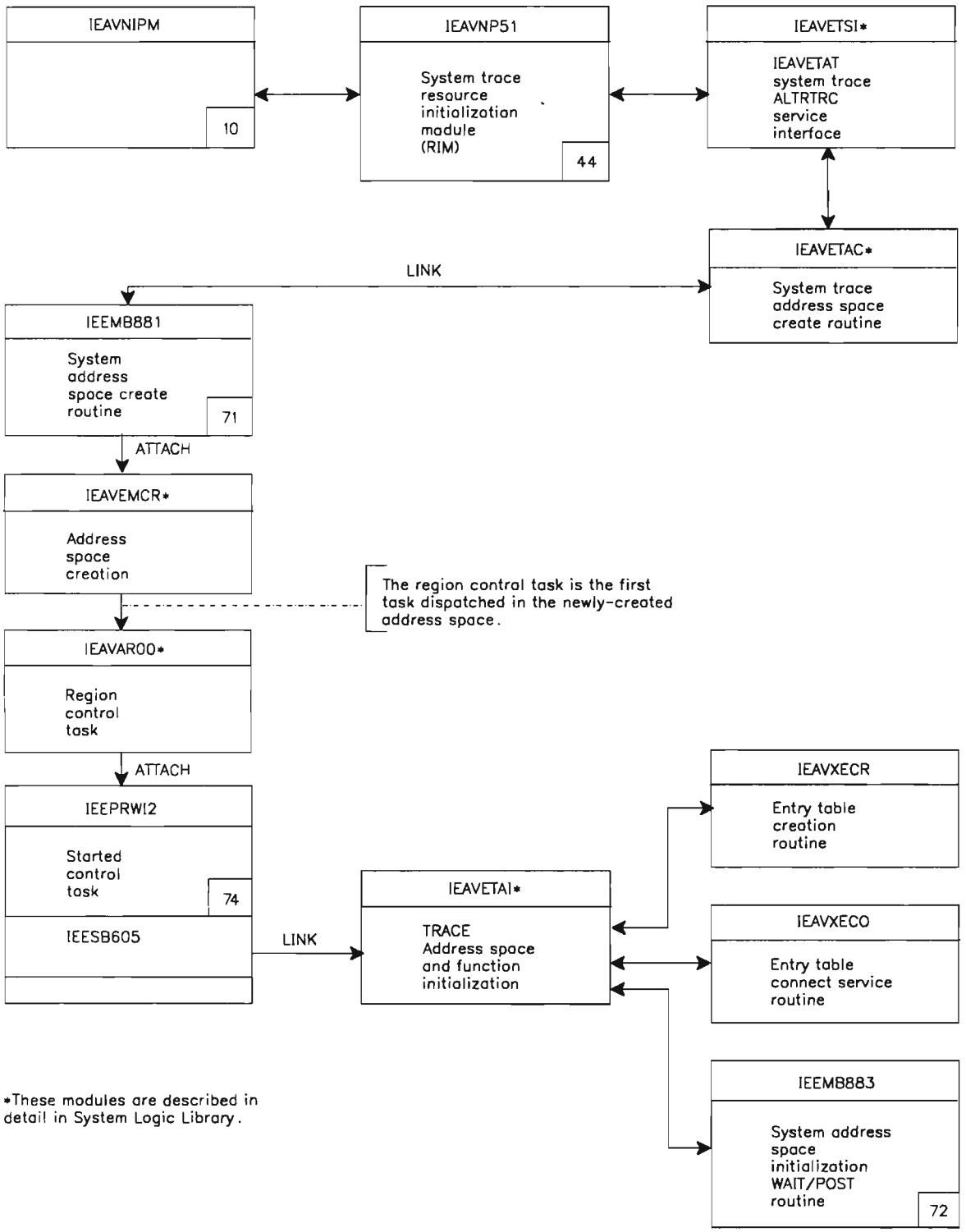


Figure 4-3. Initializing the System Trace Address Space

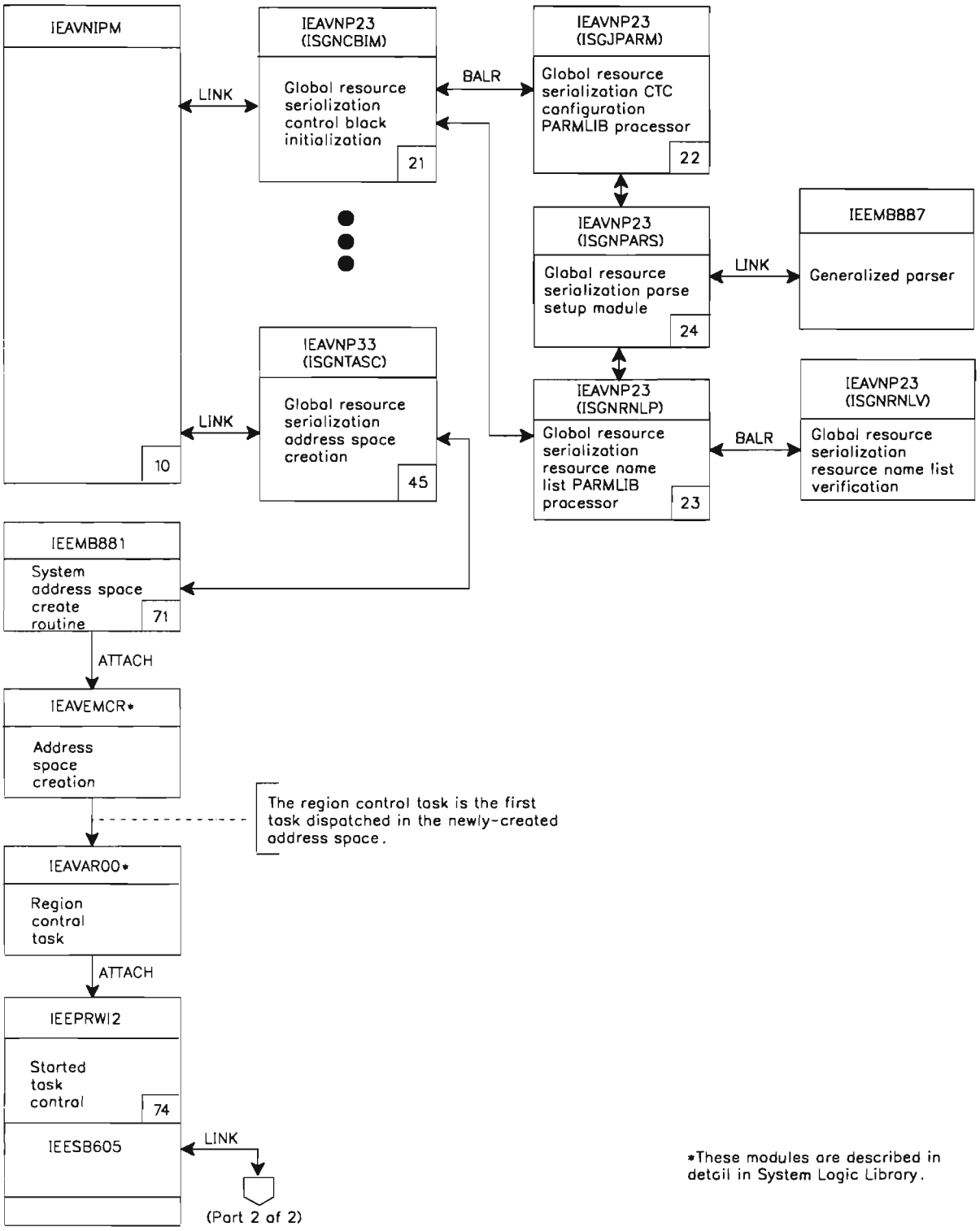


Figure 4-4 (Part 1 of 3). Initializing the Global Resource Serialization Address Space

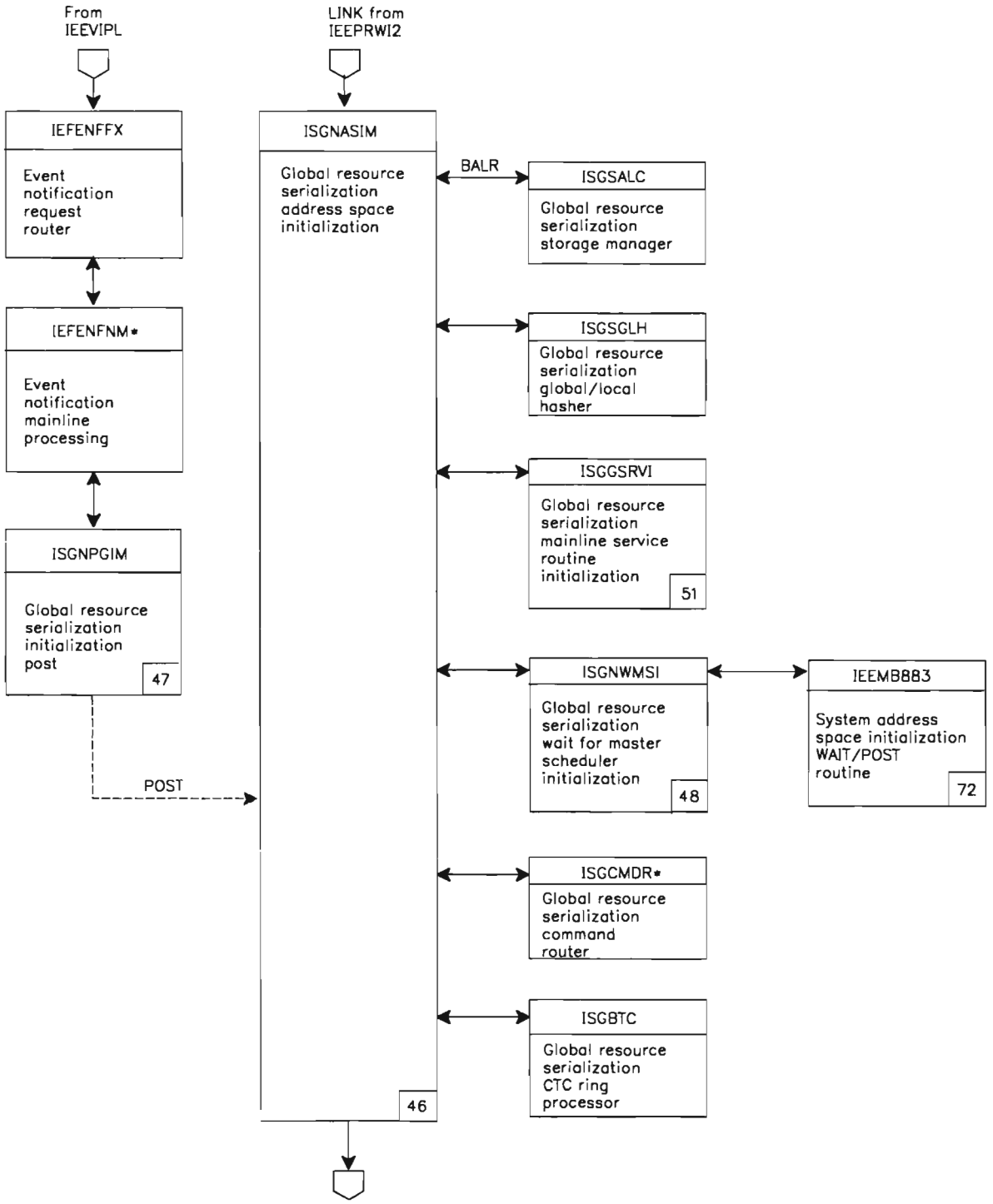


Figure 4-4 (Part 2 of 3). Initializing the Global Resource Serialization Address Space

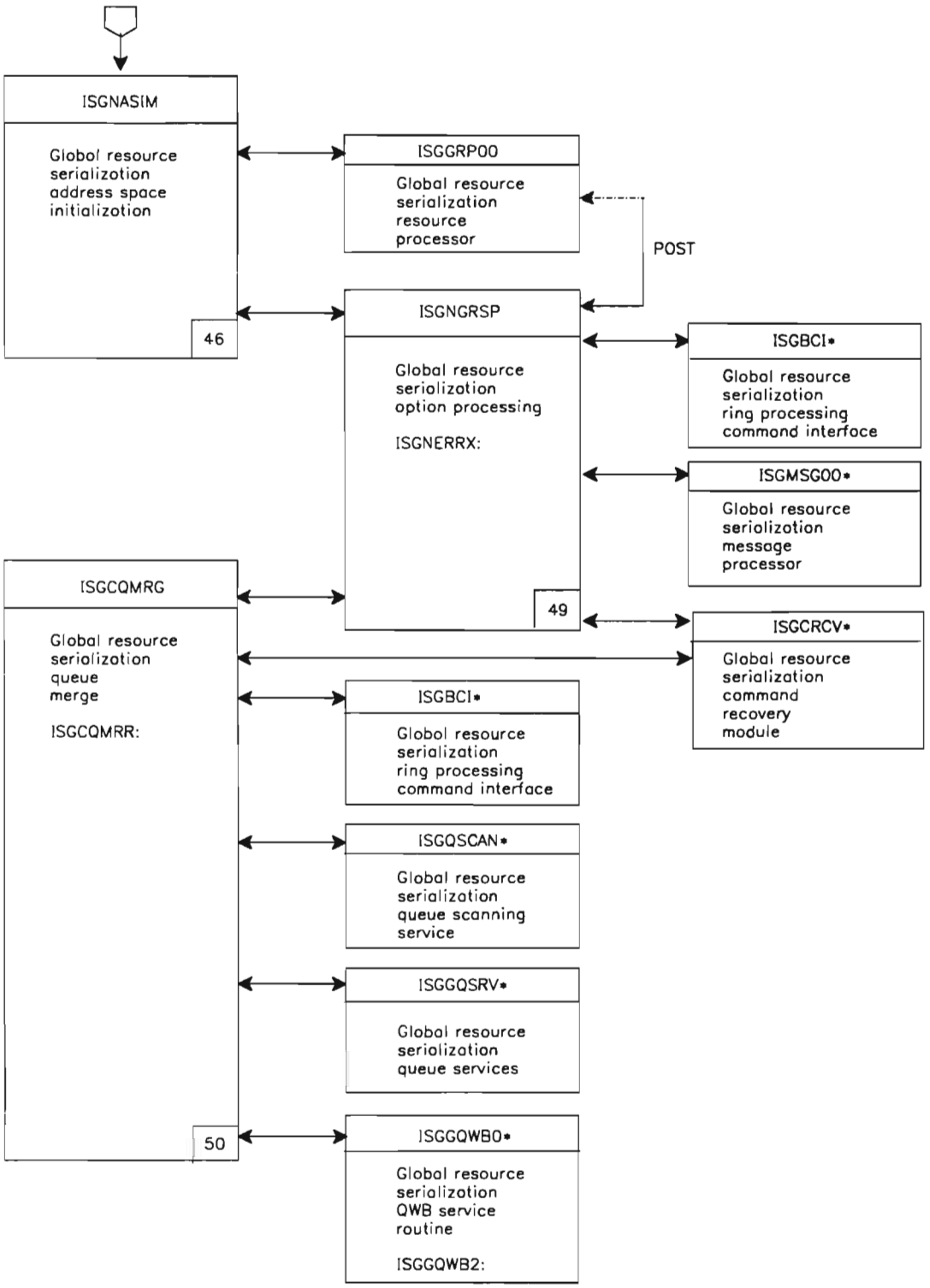


Figure 4-4 (Part 3 of 3). Initializing the Global Resource Serialization Address Space

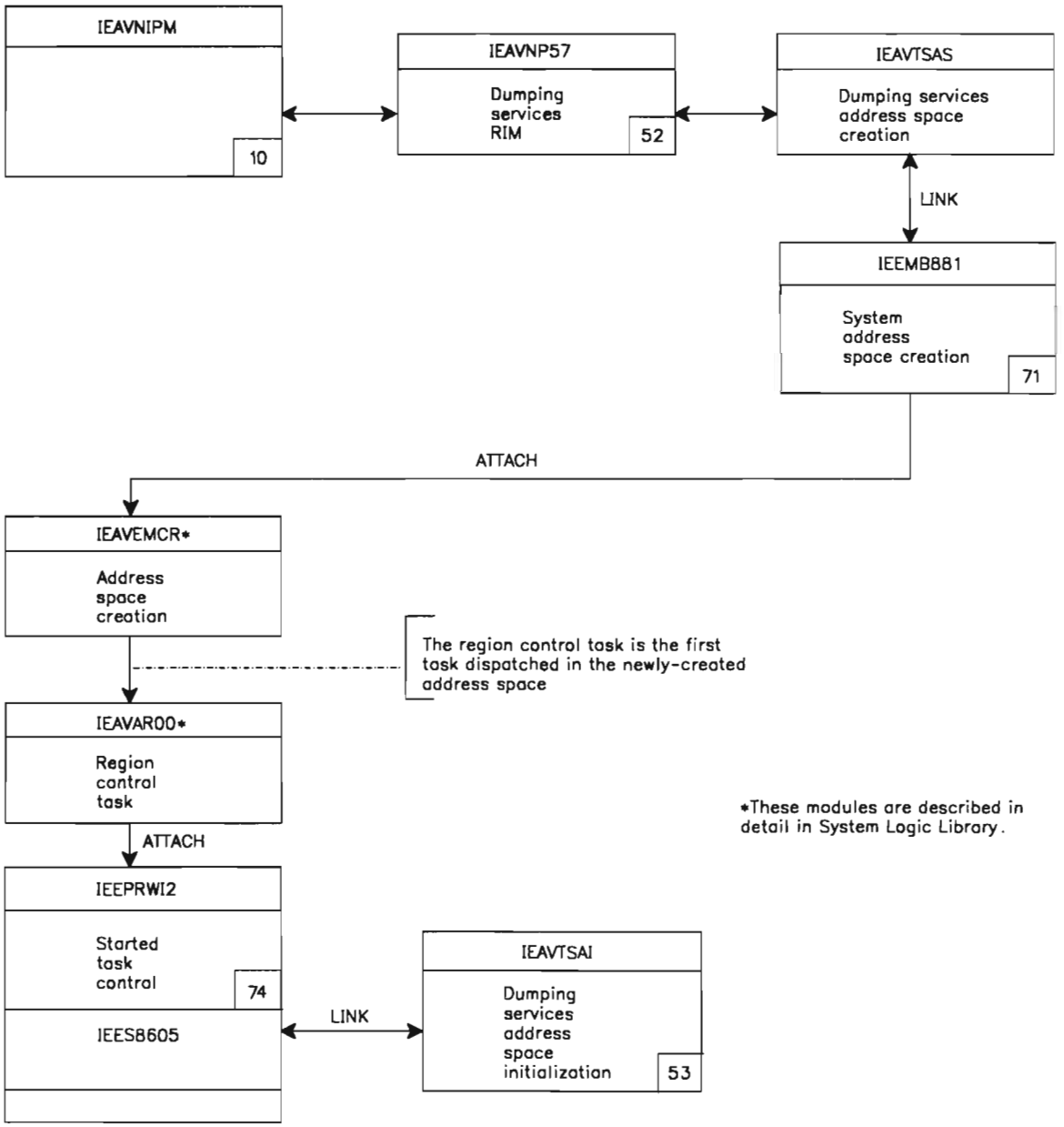


Figure 4-5. Initializing the Dumping Services Address Space

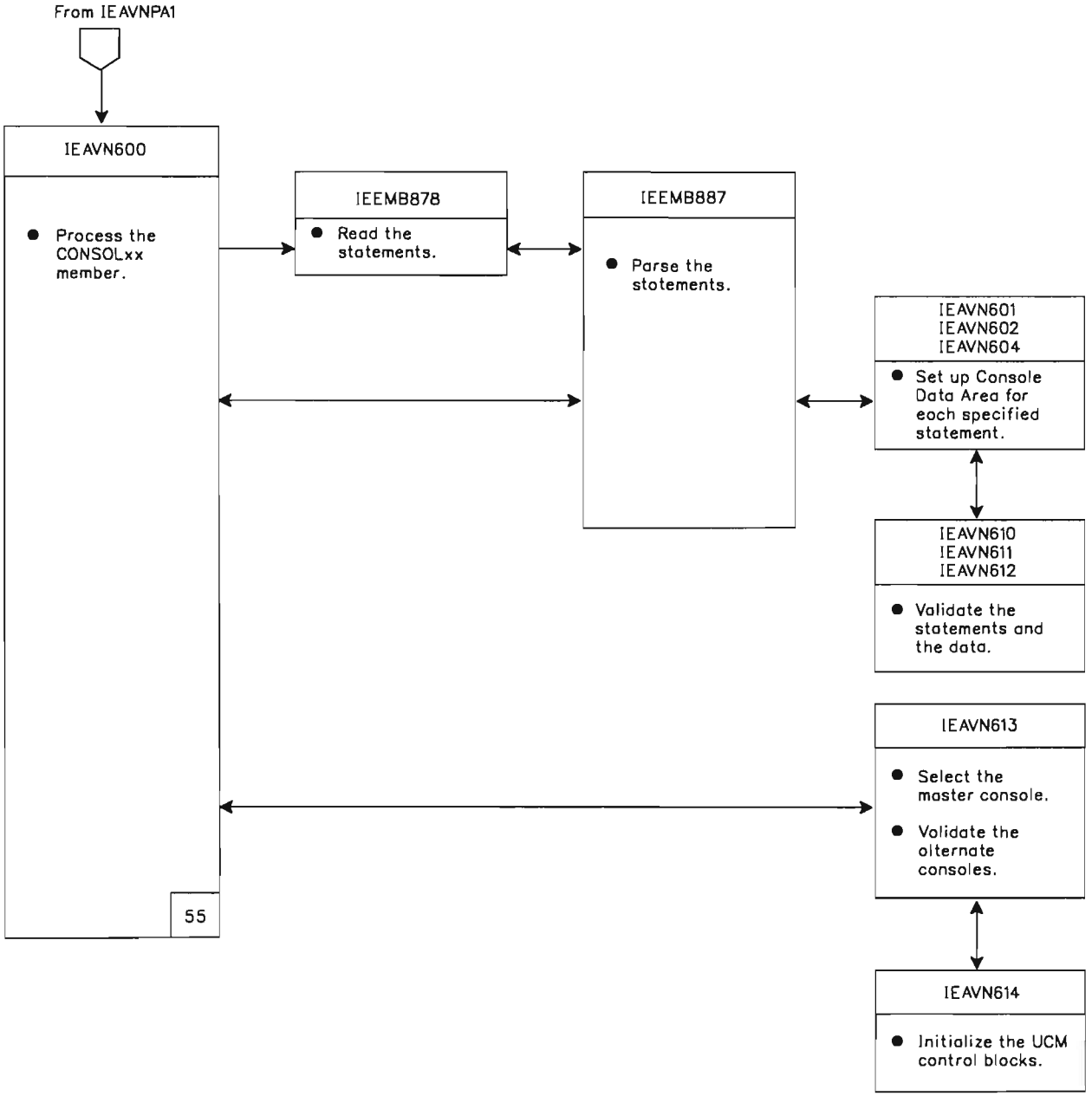


Figure 4-6. Processing the CONSOLxx member of SYS1.PARMLIB

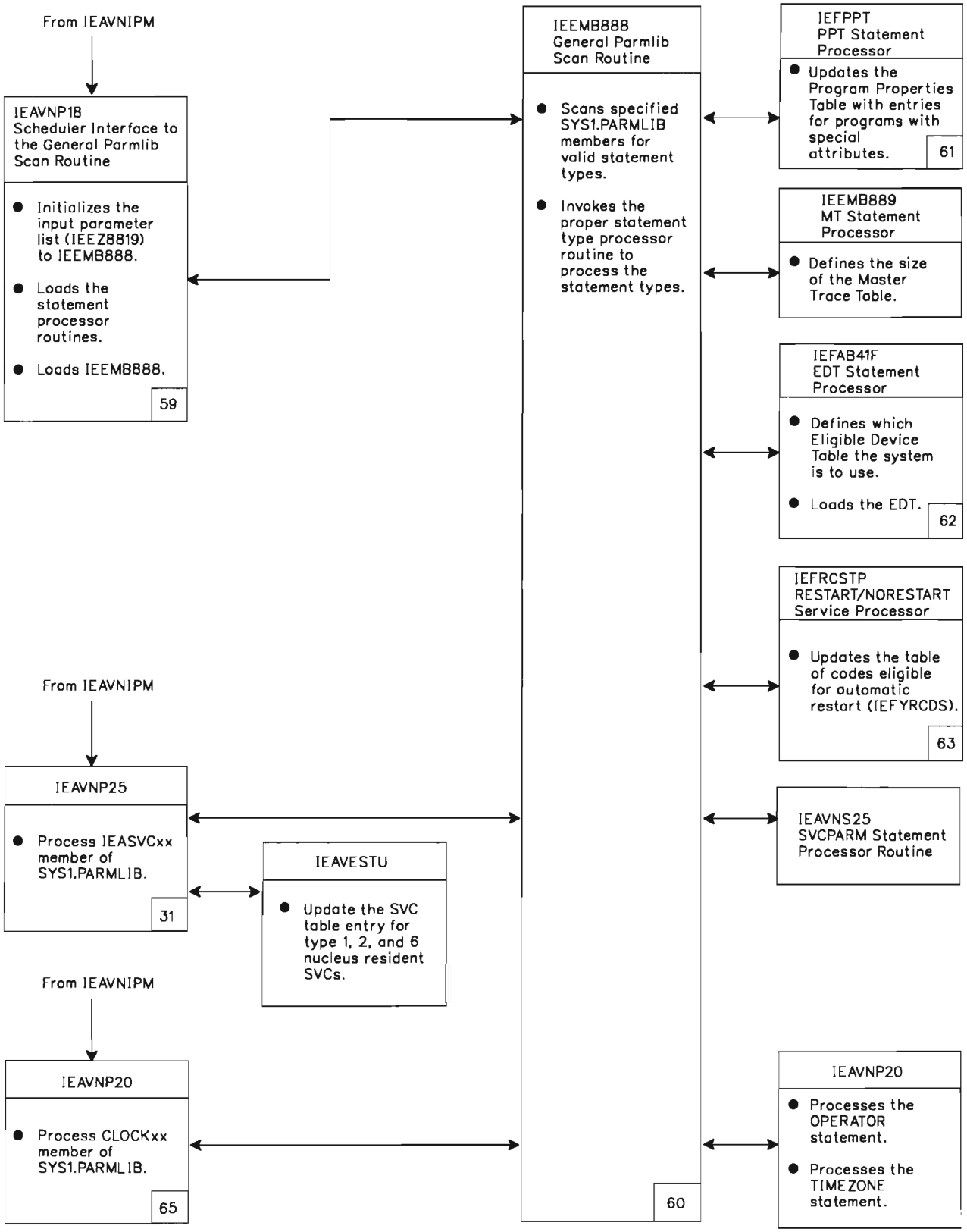


Figure 4-7. Processing the SCHEDxx, IEASVCxx, and CLOCKxx members of SYS1.PARMLIB

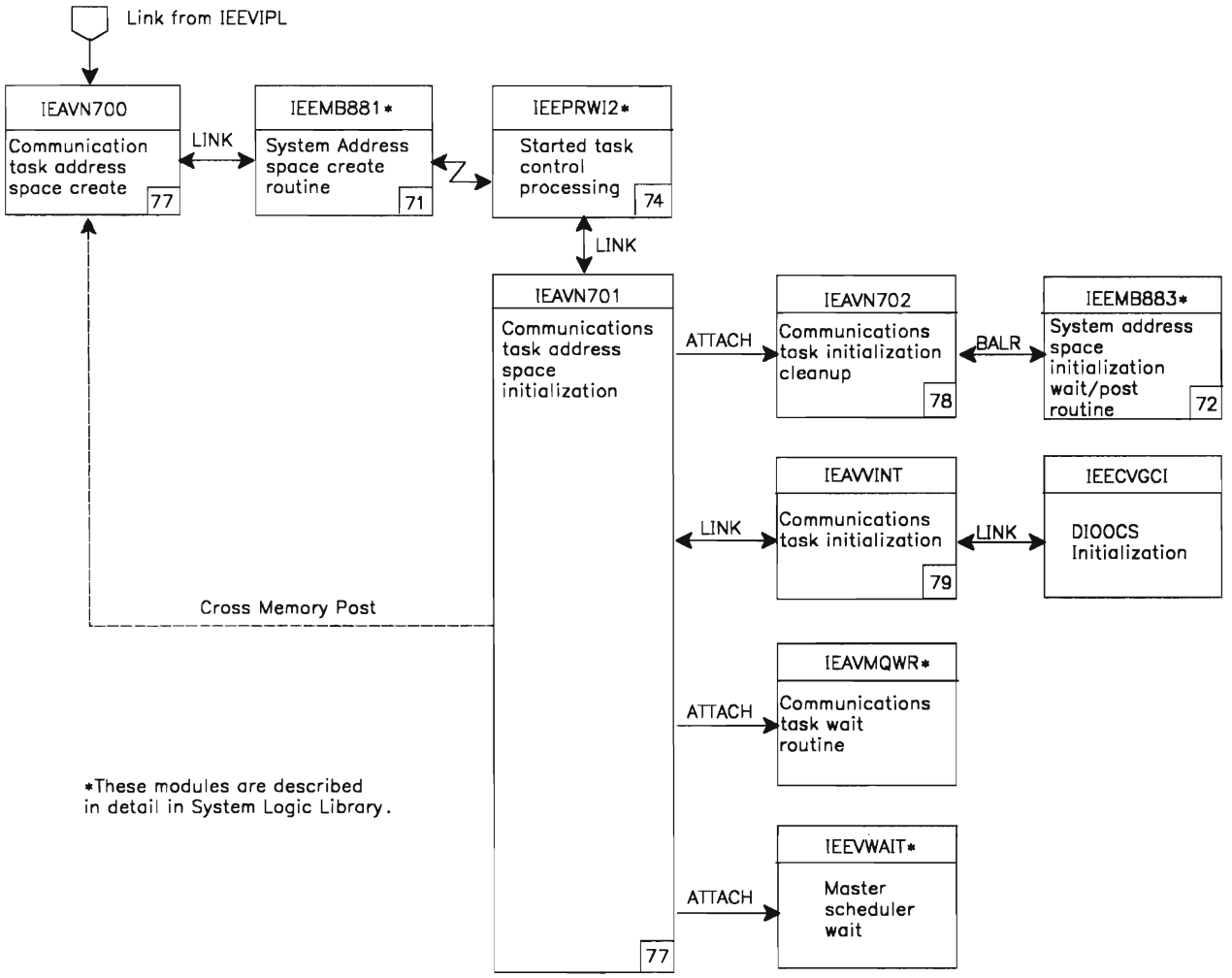


Figure 4-8. Initializing the Communications Task Address Space

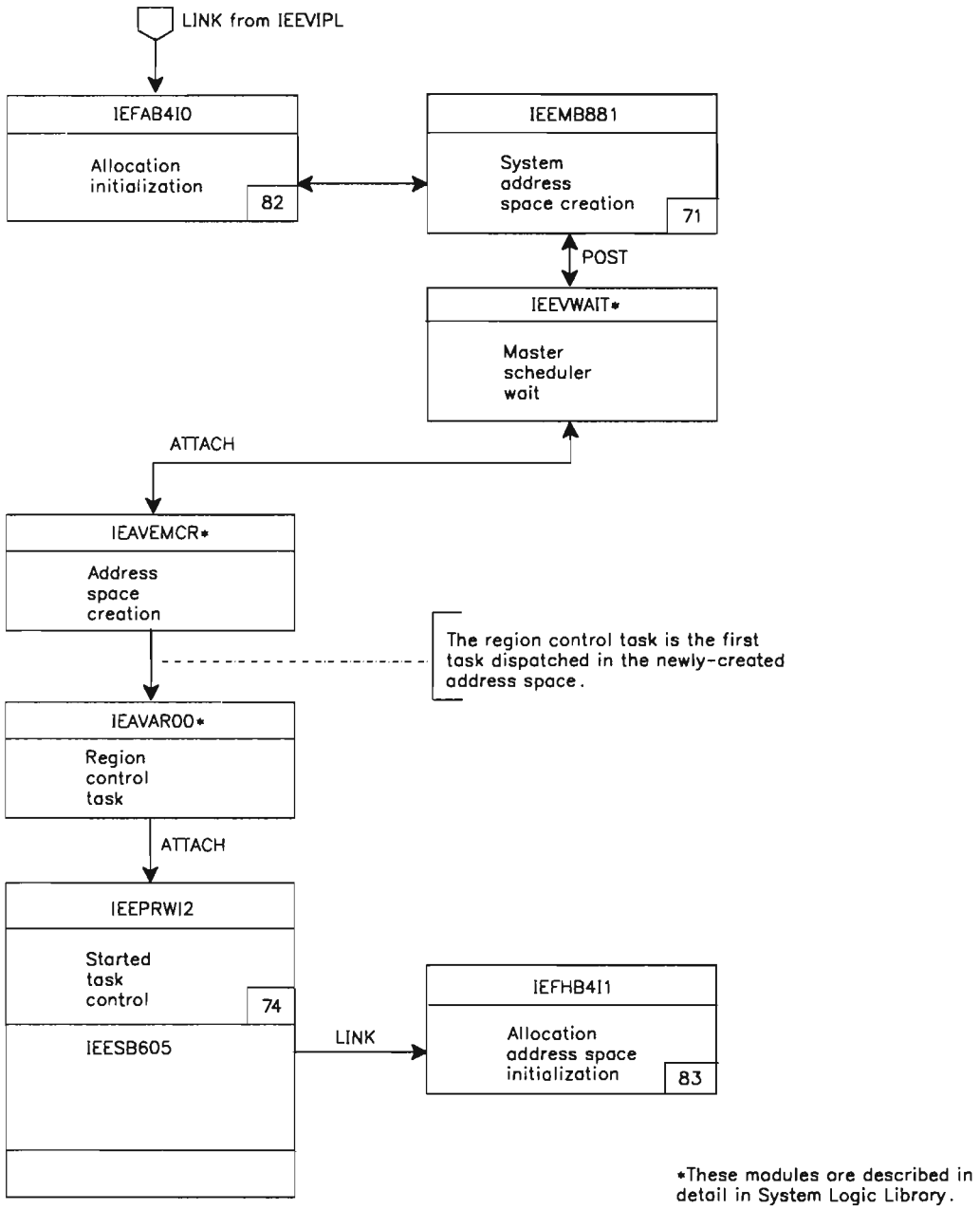


Figure 4-9. Initializing the Allocation Address Space

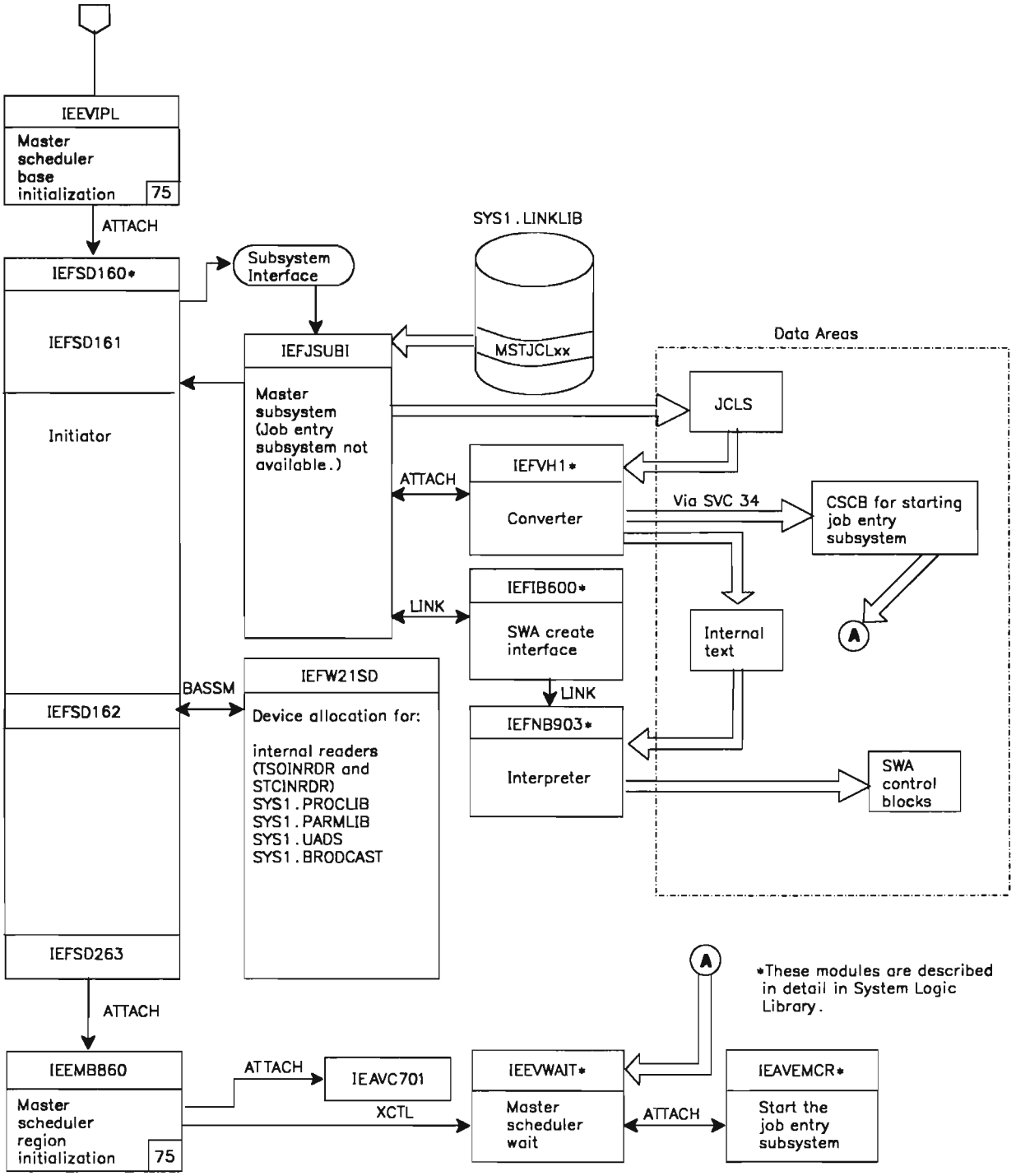
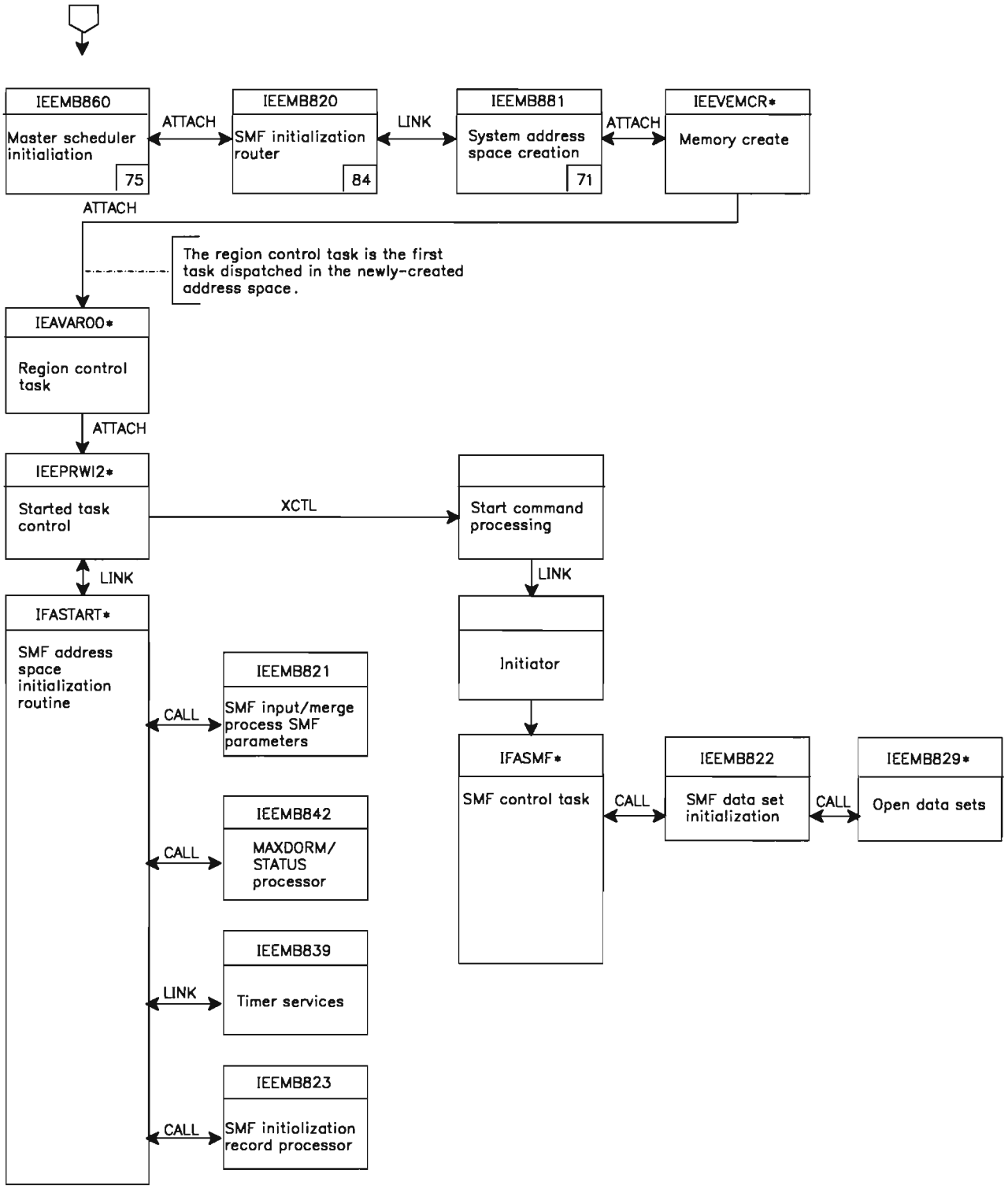


Figure 4-10. Initiation of the Master Scheduler



*These modules are described in detail in System Logic Library.

Figure 4-11. Initializing the System Management Facilities Address Space

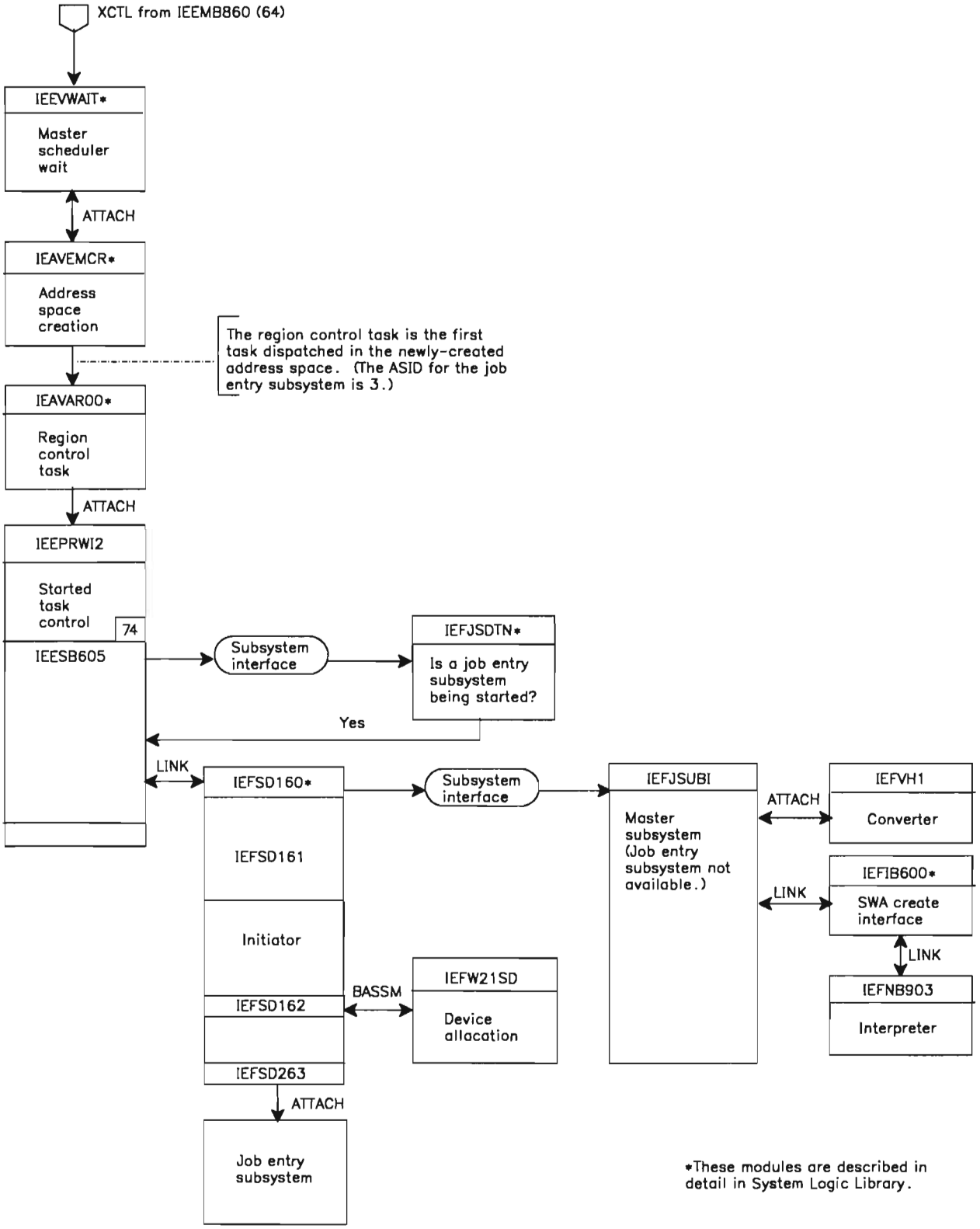


Figure 4-12. Initiation of the Job Entry Subsystem

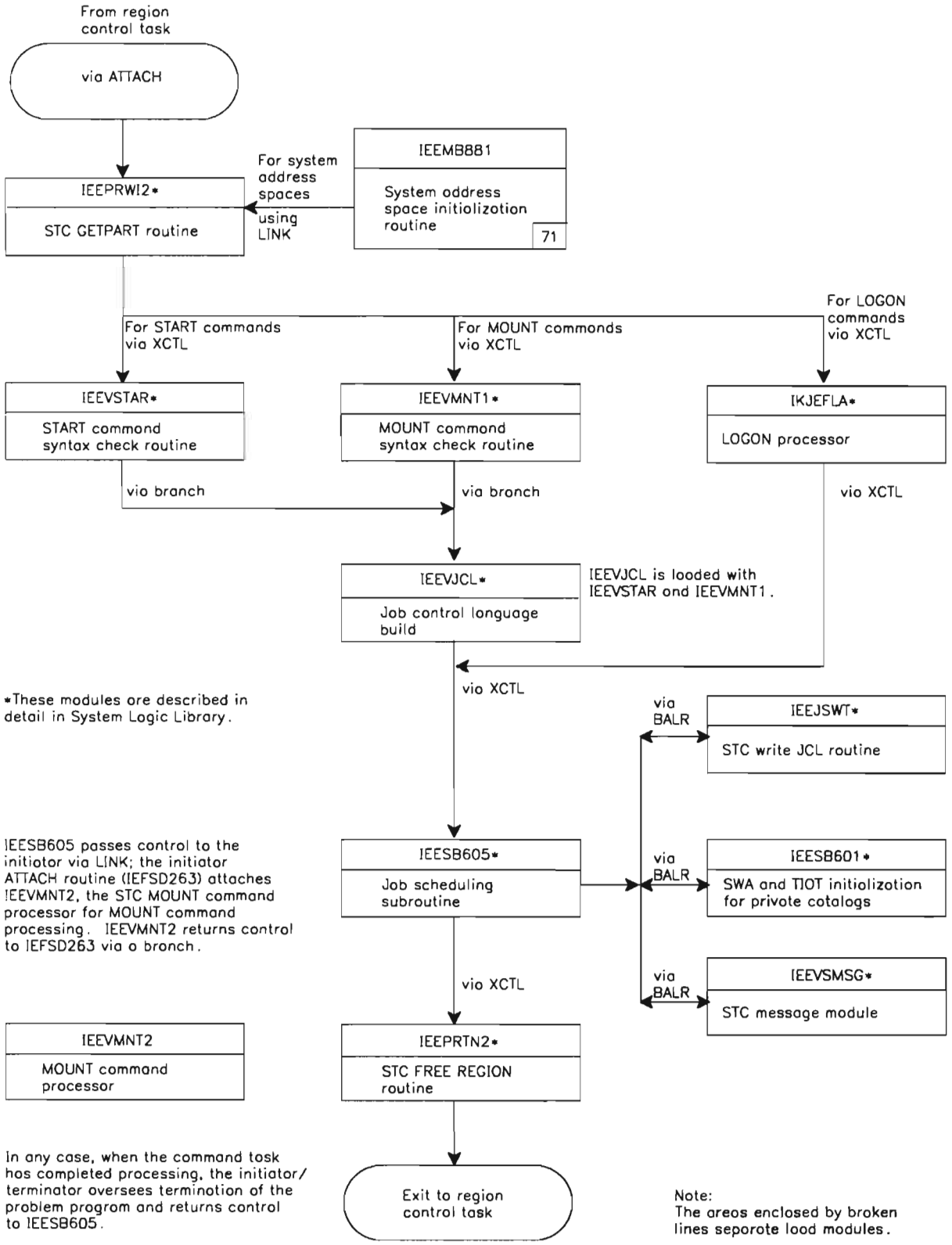


Figure 4-13. Started Task Control (STC) Module Flow

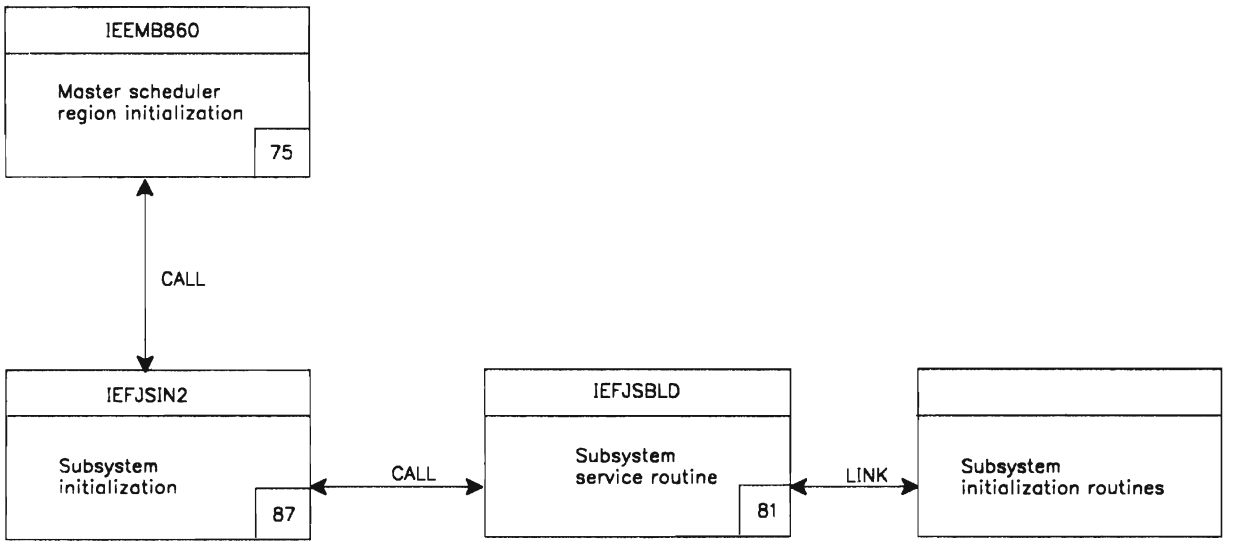
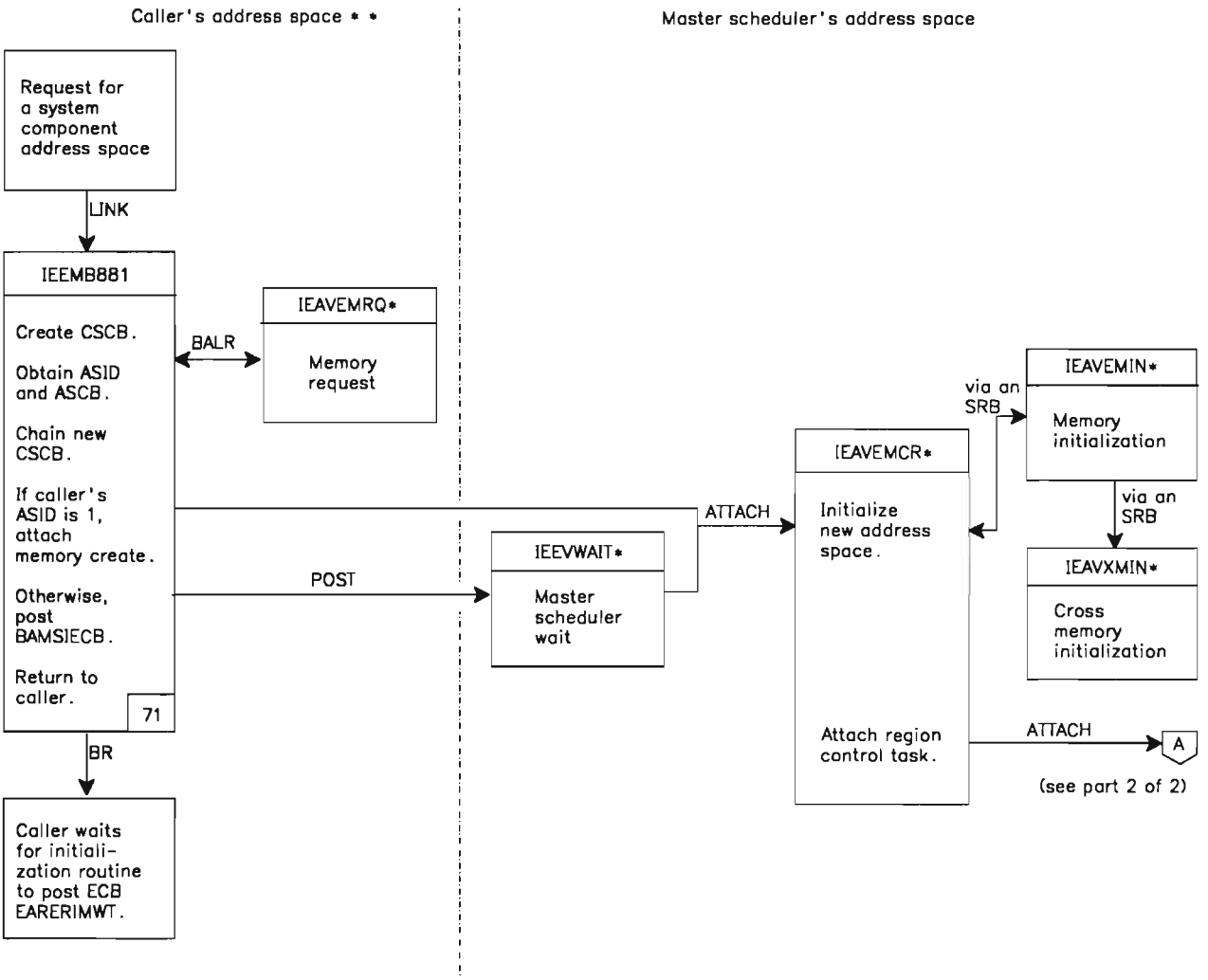


Figure 4-14. Initializing Subsystems



** When creating the JES3AUX address space, the caller's address space is the JES3 address space. When creating the PC/AUTH, ALLOCAS, GRS, or DUMPSRV address space, the caller's address space is the master scheduler's address space (ASID=1).

*These modules are described in detail in System Logic Library.

Figure 4-15 (Part 1 of 2). Initializing a System Component Address Space Using IEEMB881

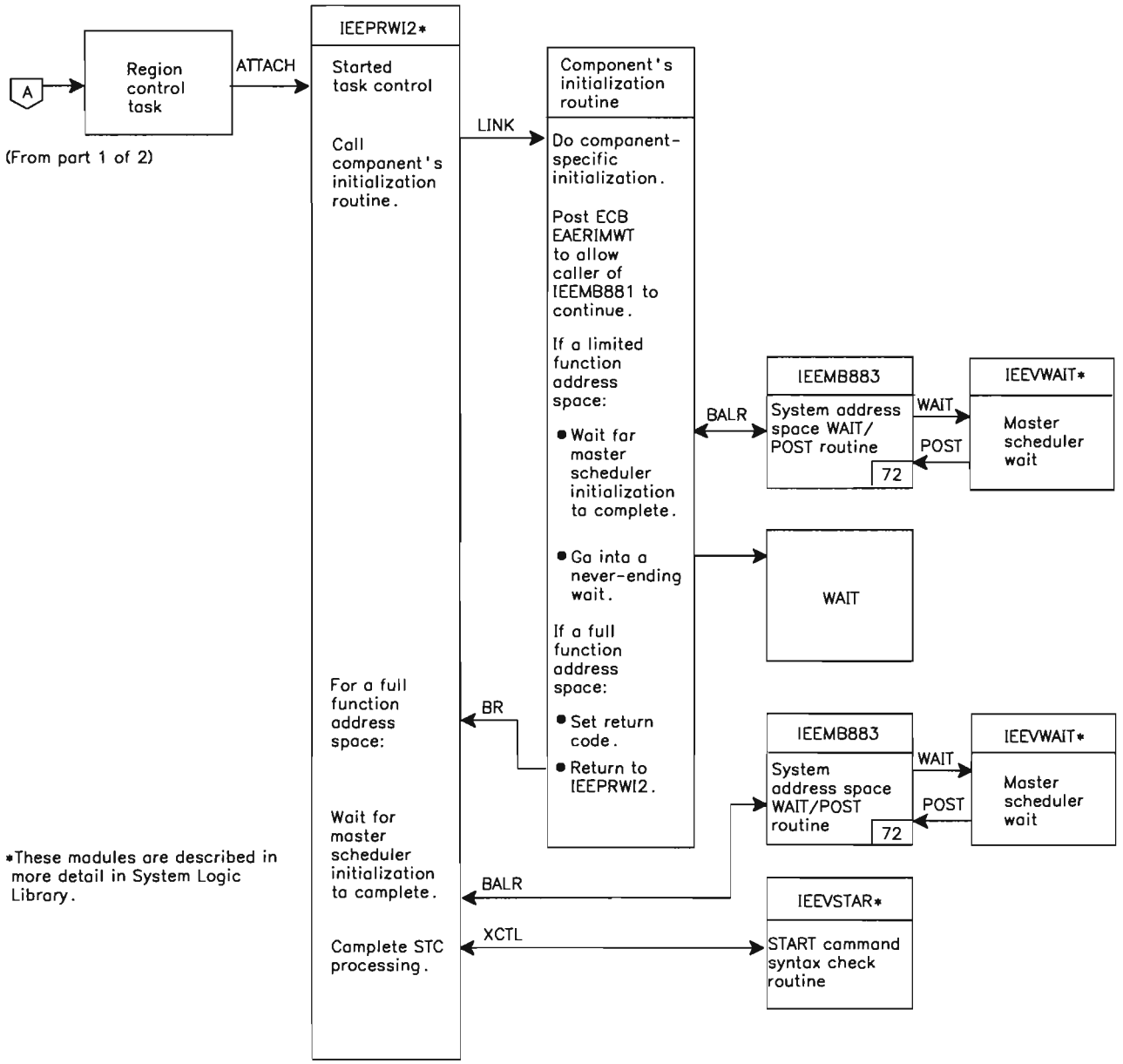


Figure 4-15 (Part 2 of 2). Initializing a System Component Address Space Using IEEMB881

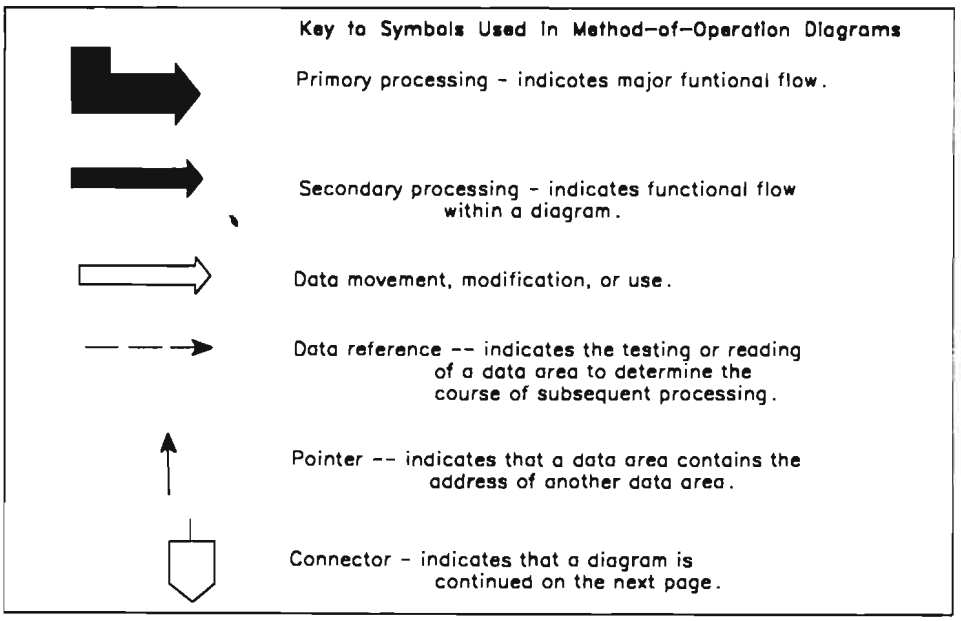
SECTION 5. METHOD OF OPERATION

This section uses diagrams and text to describe the functions performed during system initialization. There are two types of diagram formats: HIPO and prologue. The diagrams are presented, as closely as possible, in their order of invocation and emphasize functions performed rather than the program logic and organization. Program logic and organization is described in "Section 4: Program Organization."

HIPO 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" box. The object module name and labels in the extended description point to the code that performs the function.

The following figure shows the symbols used in HIPO diagrams. The relative size and the order of fields in control block illustrations do not always represent the actual size and format of the control block.

Figure 5-1. Key to HIPO Diagrams



The prologue format diagrams contain detailed information that is broken down into four different headings. The four headings and the topics they document are:

Module Description, which includes:

- Descriptive name
- Function (of the entire module)
- Entry point names, which includes:
  - Purpose (of the entry point)
  - Linkage
  - Callers
  - Input
  - Output
  - Exit normal
  - Exit error, if any
- External references, which includes:
  - Routines
  - Data areas, if any
  - Control blocks
- Tables
- Serialization

Module Operation, which includes:

- Operation, which explains how the module performs its function.

- Recovery operation, which explains how the module performs any recovery.

Diagnostic aids, which provide information useful for debugging program problems; this includes:

- Entry point names
- Messages
- Abend codes
- Wait state codes
- Return codes for each entry point. Within each entry point, return codes might be further categorized by exit-normal and exit-error.
- Entry register contents for each entry point
- Exit register contents for each entry point

Logic Diagram, which illustrates the processing of the module, the input it uses, the output it produces, and the flow of control. The following figure illustrates the graphic symbols and format used in the prologue format diagrams.

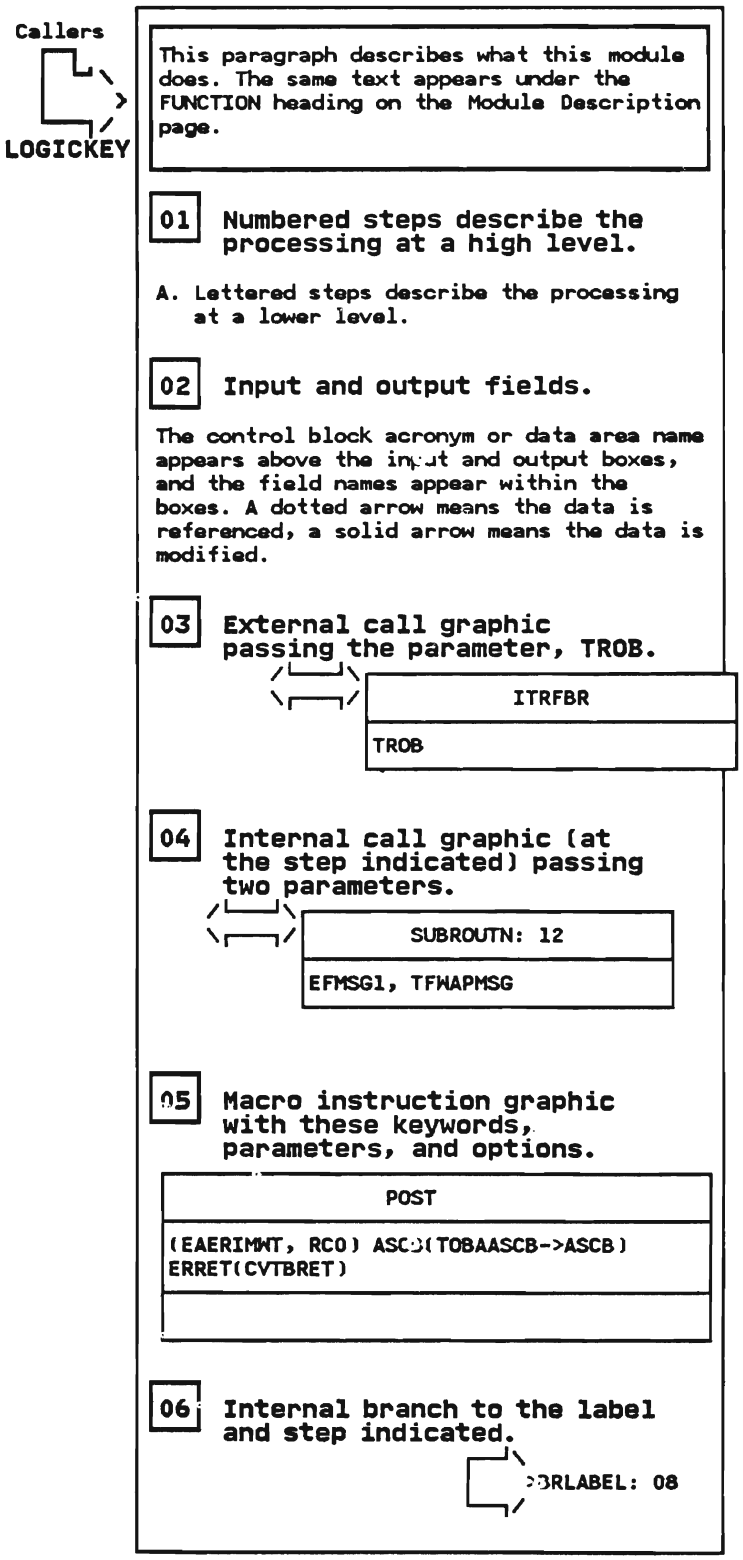


Figure 5-2 (Part 1 of 2). Key to Logic Diagrams

LOGICKEY - Key to the Logic Diagrams STEP 07

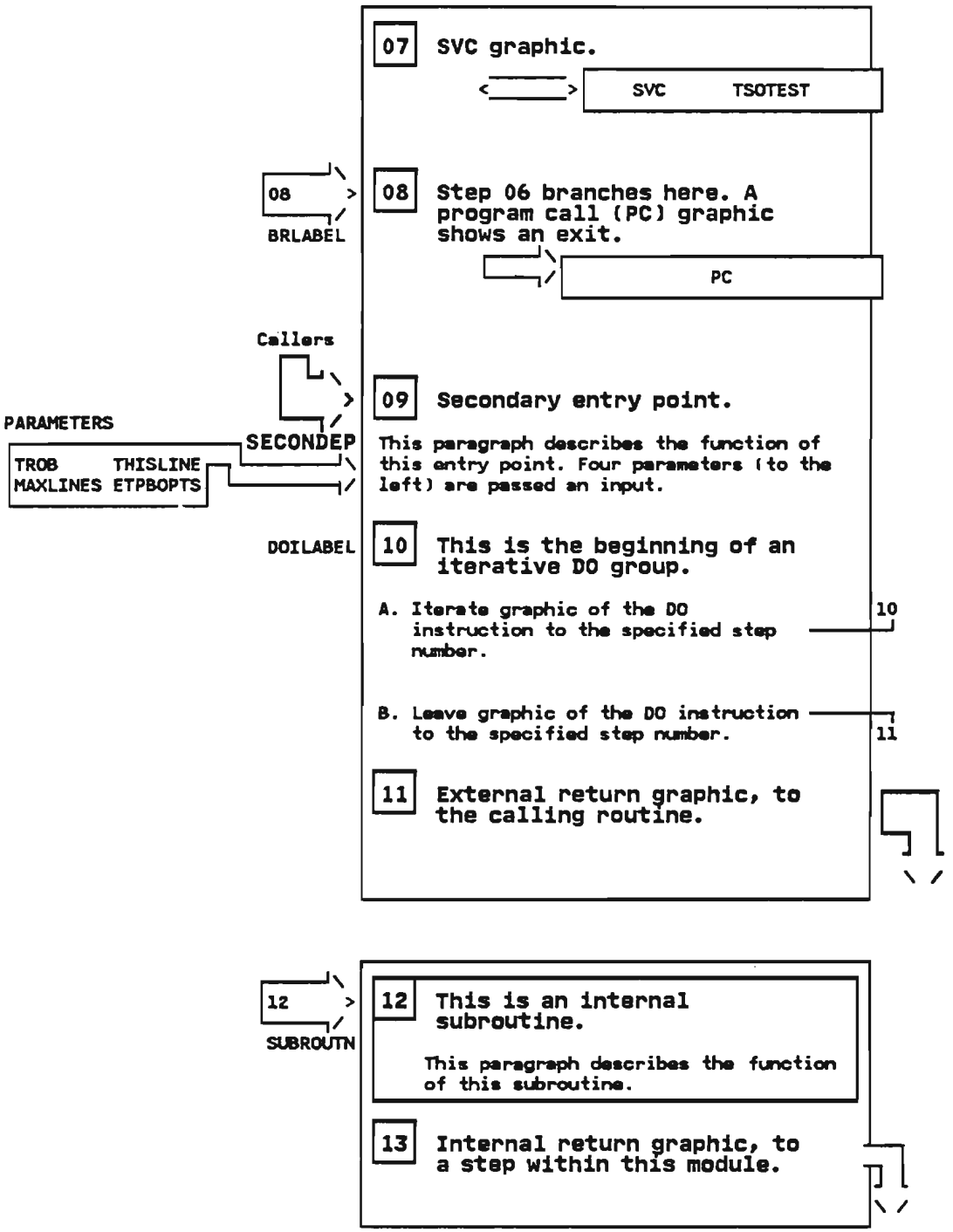


Figure 5-2 (Part 2 of 2). Key to Logic Diagrams

The diagrams in Section 5 are grouped according to three phases of initialization: IPL, NIP, and master scheduler initialization. Diagrams 1-8 provide details about the IPL phase of system initialization. Diagrams 9-70 provide details about the NIP phase of system initialization including details about the system component address space initialization that occurs during NIP processing. Diagrams 71-88 provide details about the master scheduler initialization phase.

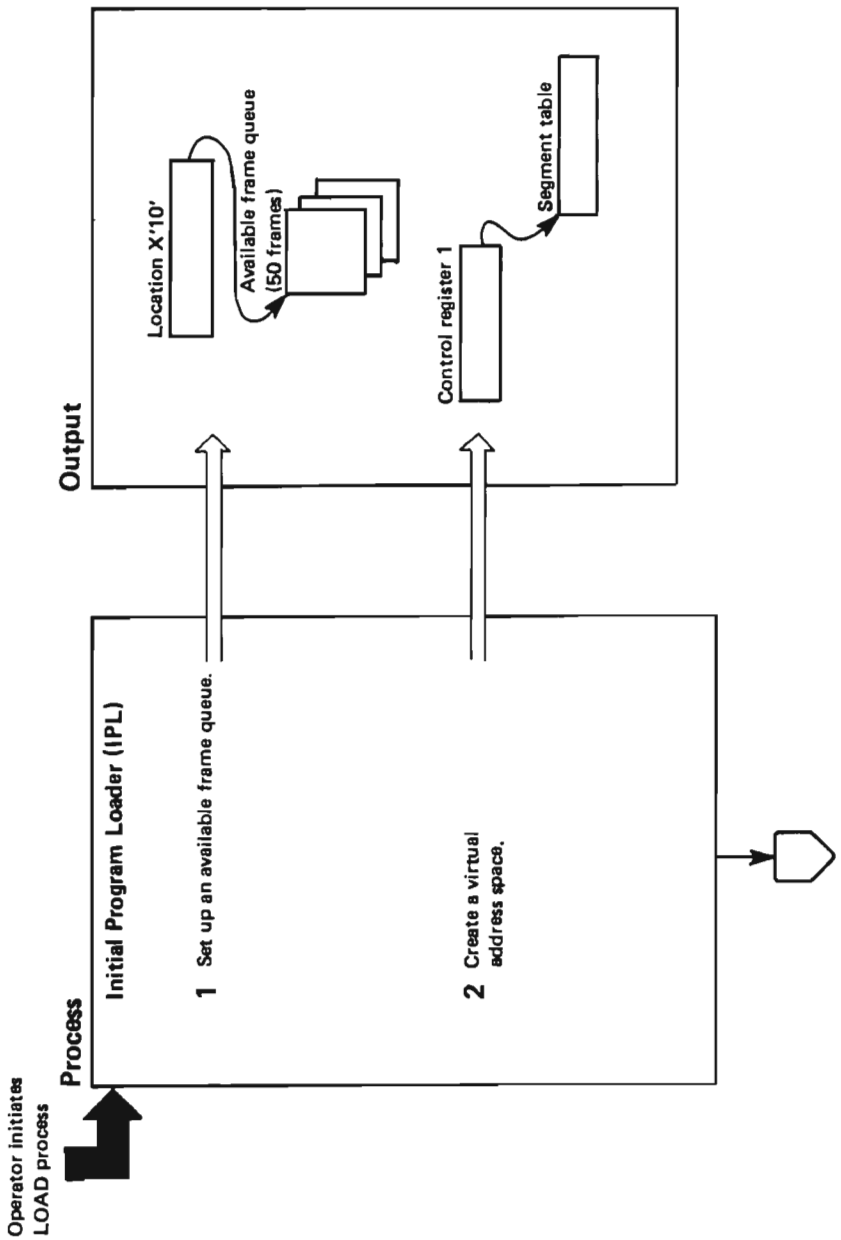


Diagram 1. Initial Program Loader (IEAIPL00) Part 1 of 18

Diagram 1. Initial Program Loader (IEAIPL00) Part 2 of 18

Module	Label	Extended Description	Module	Label

The first phase of system initialization is the initial program loader (IPL), which begins execution when the operator initiates the LOAD process. Module IEAIPL00 prepares the virtual environment in which the IPL resource initialization modules (IRIMs) will execute. It loads and calls each IRIM and provides common services for the IRIMs through IPL SVCs. The IPL SVCs are described in detail later in this extended description under "Processing IPL SVC Requests".

1 IEAIPL00 tests each frame of real storage from location 4K up until it obtains fifty (50) available frames. IEAIPL00 issues a Test Block (TB) instruction to test each frame. The TB instruction returns a condition code that identifies the state of the frame, and IEAIPL00 sets fields accordingly. The following table shows the condition code and the state of the frame that the code represents:

Condition Code	*State of the Frame*
0	good
1	defective
3	offline

Note: The TB instruction does not actually set condition code 3. IEAIPL00 simulates this condition code when an addressing exception occurs on a TB instruction.

When IEAIPL00 finds an available frame, it clears the frame's storage key to zero and chains the frame to an available frame queue.

2 IEAIPL00 uses the page backing service, ISVCPGFX (IPL SVC 4), to back a page for the IPL workspace and a page for IRIM. Two megabytes of virtual storage are allocated as IPL work space. This work space will contain the IPL vector table, tables that IEAIPL02 will build, and the IRIMs as they execute. The default load frame parameters will be saved in IVTPARMD.

IEAIPL00

Diagram 1. Initial Program Loader (IEAIPL00) Part 3 of 18

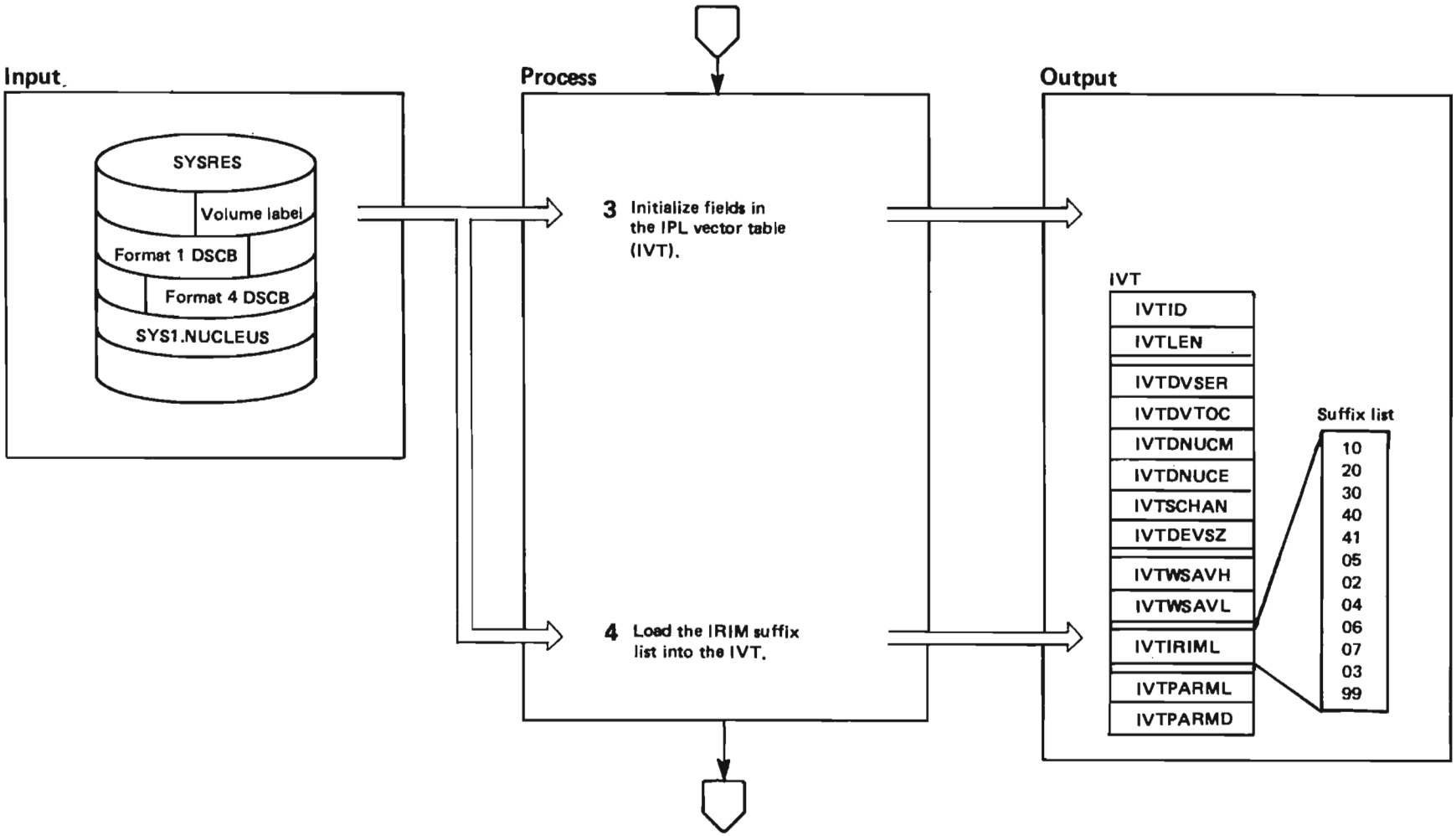


Diagram 1. Initial Program Loader (IEAIPL00) Part 4 of 18

Extended Description	Module	Label

3 IEAIPL00 issues ISVCXDAP (IPL SVC 0) to access the IPL volume on SYSRES and reads the following:

- The volume label of the IPL device to determine the serial number and VTOC address
- The format 4 DSCB of the VTOC to determine the number of cylinders on the volume and the number of tracks per cylinder
- The format 1 DSCB for the SYS1.NUCLEUS data set to determine the start and end addresses of the extent of this data set

IEAIPL00 copies this data into the IVT, which is the communication area for the IRIMs, IEAIPL00, and IEAVNIP0.

4 IEAIPL00 issues the module loading service, IPL SVC 6, to load module IEAIPL01 into the IVTIRIML field in the IVT. IEAIPL01 contains a suffix list, a list of two-character IRIM identifiers that IEAIPL00 will append to the root "IEAIPL" to form IRIM names. This list represents the IRIMs that IEAIPL00 will call during the first phase of system initialization. The entry point of IEAIPL01 identifies the start of the list; two bytes of binary zeroes identify the end of the list. Thus the suffix list reads as follows, with the function of the corresponding modules indicated in parentheses:

10 (Invoke SCP INFO)
20 (Validate the system storage and calculate the top of storage)
30 (Initialize the IPL WTO facility for the system console)
40 (Build the device support modules list)
41 (Determine the DAT-on nucleus and the nucleus module Csects for the nucleus)
05 (Create the DAT-on nucleus map)
02 (Load Dat-off and Dat-on nucleus)
04 (Initialize the VSM control blocks)
06 (Initialize the RSM control blocks)
07 (Update the AMODE bit for the SVC and ESR table entries)
03 (Initialize the UCB for the IPL processor)
99 (IPL cleanup in preparation for invoking NIP)

Extended Description	Module	Label

Passing Control to IRIMs

Beginning with '02', the first entry in the suffix list in the IVT, IEAIPL00:

- Constructs the full name of the IRIM to be invoked by appending the identification number to 'IEAIPL'.
- Reads the directory entry for the IRIM by issuing ISVCFIND (IPL SVC 5).
- Clears the frame into which the IRIM will be loaded.
- Loads the IRIM into storage, using the module loading service ISVCLOAD (IPL SVC 6). Address constants within the IRIM are relocated with respect to the virtual address of the IRIM area.
- Issues ISVCSYNC (IPL SVC 10) to call the IRIM at its linkage editor-assigned entry point in supervisor state, key 0, disabled, and with address translation on in 31-bit addressing mode. Registers contain the following:
  - Register 1 – virtual address of the IVT
  - Register 13 – address of an 18-word save area
  - Register 14 – address of ISVCEXIT (IPL SVC 3) which the IRIM must issue to return to IEAIPL00
  - Register 15 – entry point address of the IRIM.

Each IRIM called, except for the last one, returns control to IEAIPL00, which gets the next IRIM ID in the suffix list and repeats the loop. The last IRIM, IEAIPL99, passes control to IEAVNIP0 instead of returning to IEAIPL00.

If one of the following errors occurs, IEAIPL00 loads wait state code X'075' in PSW bits 52-63 and a reason code in PSW bits 40-43.

Error	*Reason Code*
IEAIPL01 was not found.	1
IEAIPL01 is larger than the IVTIRIML field.	2
IEAIPL99 returned to IEAIPL00	3
An IRIM other than IEAIPL01 was not found.	4
The IRIM is larger than a page.	5

Diagram 1. Initial Program Loader (IEAIPL00) Part 5 of 18

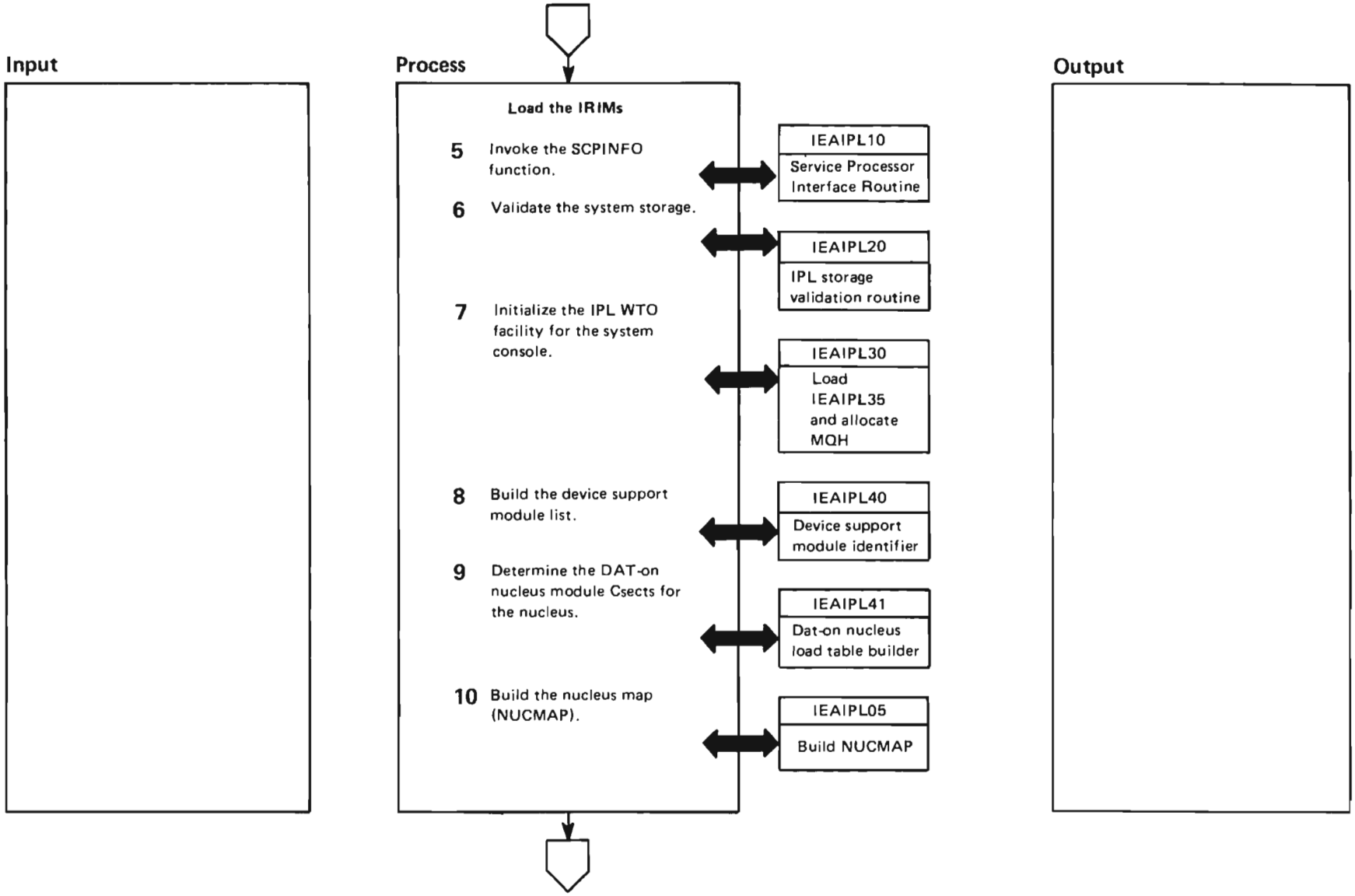


Diagram 1. Initial Program Loader (IEAIPL00) Part 6 of 18

Extended Description	Module	Label
5 IEAIPL10 issues a SCP INFO service call instruction to obtain the storage address range constant, the storage address increment constant, and the load parameter from the service processor. The load parameter will be moved into IVTPARML. If the service call is not supported on the processor, IEAIPL10 issues an MSSF SCP INFO diagnosis command. If either of the following problems occur, IEAIPL10 enters a wait state: • The service processor is busy or not active, or the MSSF diagnose command is rejected. • IEAIPL10 has made three unsuccessful attempts to access a service processor through either command. If processing is successful, IEAIPL10 returns to IEAIPL00.	IEAIPL10	
6 IEAIPL00 calls IEAIPL20 to validate the system storage and place all valid frames on the available frame queue. IEAIPL20 calculates the top of storage, and switches dynamic address translation (DAT) off to access the frames. IEAIPL20 queues all valid frames on the available frame queue, then switches DAT on again. IEAIPL20 then moves the segment table for the master scheduler address space above the 16-megabyte line, and returns to IEAIPL00.	IEAIPL20	
7 IEAIPL00 calls IEAIPL30 to block load the DAT-off nucleus into real storage and scatter load the DAT-on nucleus into virtual storage backed by real frames. IEAIPL30 also initializes the DAT-off to DAT-on linkage table, which establishes a communications area for the DAT-on and DAT-off nucleus.	IEAIPL30	
8 IEAIPL00 calls IEAIPL40 to build a nucleus map (NUCMAP), which resides directly above the DAT-on nucleus.	IEAIPL40	
9 IEAIPL00 calls the input/output supervisor (IOS) IRIM, IEAIPL41, to locate and initialize the unit control block (UCB) for the IPL device. After the hardware IPL sequence is completed, all subchannels except for the IPL subchannel are disabled. The IOS IRIM obtains from the subchannel the device number (SCHDEVNO) that the operator specified as the number of the IPL device and searches the UCB chain to find the UCB for the device. If the IOS IRIM cannot find the UCB, or if the UCB is not for a DASD or is for a non-base exposure of a multiple exposure device, the IOS IRIM puts the system in a wait state with a code of X'031'. Otherwise, the IOS IRIM initializes the following fields in the UCB:	IEAIPL41	

Field	*Initialization value*
UCBSID	Identifier of the subchannel
UCBPRES	Permanently resident flag (set to '1' B)
UCBSYSR	System residence flag (set to '1' B)
UCBVOLI	Volume serial number of the system residence volume
UCBVTOC	TTR of the VTOC for the system residence volume
UCBCHAN	Address of the channel used in the IPL procedures
UCBLPM	Logical path mask
UCBNOCON	Not-connected flag field (set to '0'B which indicates that the device is connected)

IEAIPL03 also modifies the IPL subchannel to initialize the SCHIP field with the address of the UCB for the IPL device.

10 IEAIPL00 calls IEAIPL41 to build a nucleus map (NUCMAP from the information contained in the nucleus load list, CESD entries and relocation tables. The NUCMAP is built in the ready only extended section of the nucleus region.	IEAIPL05	

Diagram 1. Initial Program Loader (IEAIPL00) Part 7 of 18

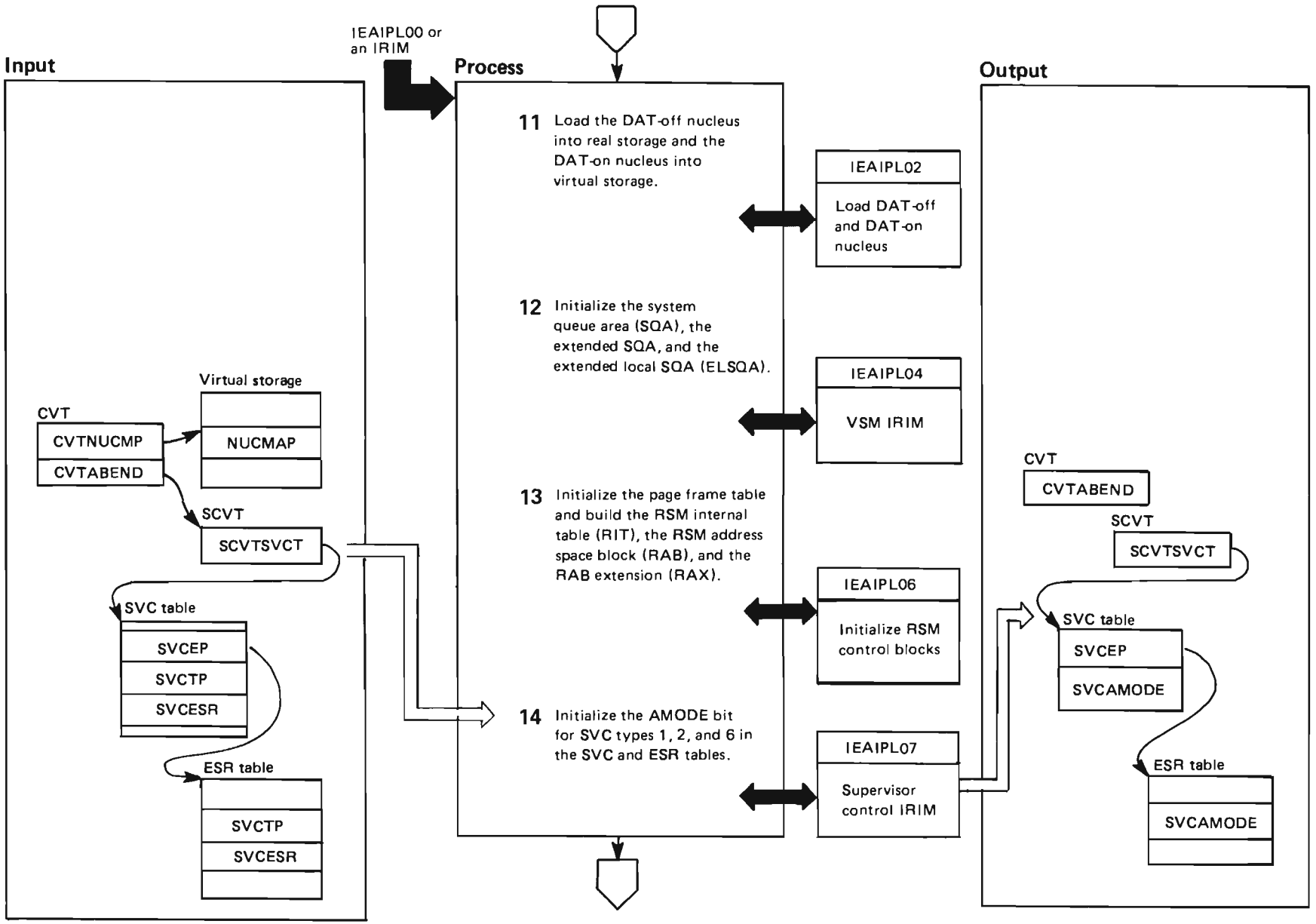


Diagram 1. Initial Program Loader (IEAIPL00) Part 8 of 18

Extended Description	Module	Label
11 IEAIPL00 calls IEAIPL02 to block-load the DAT-off nucleus into real storage and scatter-load the DAT-on nucleus into virtual storage backed by real frames. It resolves external references in the modules that are loaded into the DAT-on nucleus.	IEAIPL02	
12 IEAIPL00 calls the virtual storage management (VSM) IRIM, IEAIPL04, to allocate storage for the system queue area (SQA), the extended SQA (ESQA), and the extended local SQA (ELSQA). It builds VSM control blocks in these areas.	IEAIPL04	
13 IEAIPL00 calls the real storage management (RSM) IRIM, IEAIPL06, to initialize the page frame table. IEAIPL06 also builds the RSM internal table (RIT), the RSM address space block (RAB), and the RAB extension (RAX) for the master scheduler address space.	IEAIPL06	
14 IEAIPL00 calls the supervisor control IRIM, IEAIPL07, to update the AMODE bit of the nucleus-resident SVC entries located in the SVC and extended SVC router (ESR) tables. These tables contain one eight-byte entry for each SVC routine. IEAIPL07 sets the AMODE bit for the SVC entries (types 1, 2, and 6) that reside in the DAT-on nucleus. A later routine (IEAVNPS5) will set the AMODE bit for the SVCs (types 3 and 4) that will reside in the LPA. IEAIPL07 loops successively through the SVC and ESR table entries looking for SVC types 1, 2, and 6. SVCESR='0'B identifies entries in the SVC table, SVCESR='1'B identifies entries in the ESR table, and SVCTP identifies the type of SVC. • For non-ESR entries in the SVC table of type 1, 2, and 6 (SVCTYPE), IEAIPL07 uses the nucleus map service NUCKLUP to find the AMODE of the CSECT containing the entry point pointed to by the SVC entry. IEAIPL07 then sets theAMODE bit in the SVC table equal to the AMODE bit of the entry in the NUCMAP.	IEAIPL07	
• For ESR entries of types 1, 2, and 6, IEAIPL07 determines the address of the table and the number of entries in the table. For each entry in the ESR table, IEAIPL07 uses the nucleus map service NUCLKUP to find the AMODE of the CSECT containing the entry point pointed to by the ESR entry. IEAIPL07 then sets the AMODE bit in the ESR table equal to the AMODE bit of the entry in the NUCMAP. If NUCLKUP does not find a name in NUCMAP to match the name of the SVC or ESR table entry, then IEAIPL07 issues a disabled wait state code of X'077'. For a description of the NUCLKUP service routine, see *SPL: System Macros and Facilities*.		

Diagram 1. Initial Program Loader (IEAIPL00) Part 9 of 18

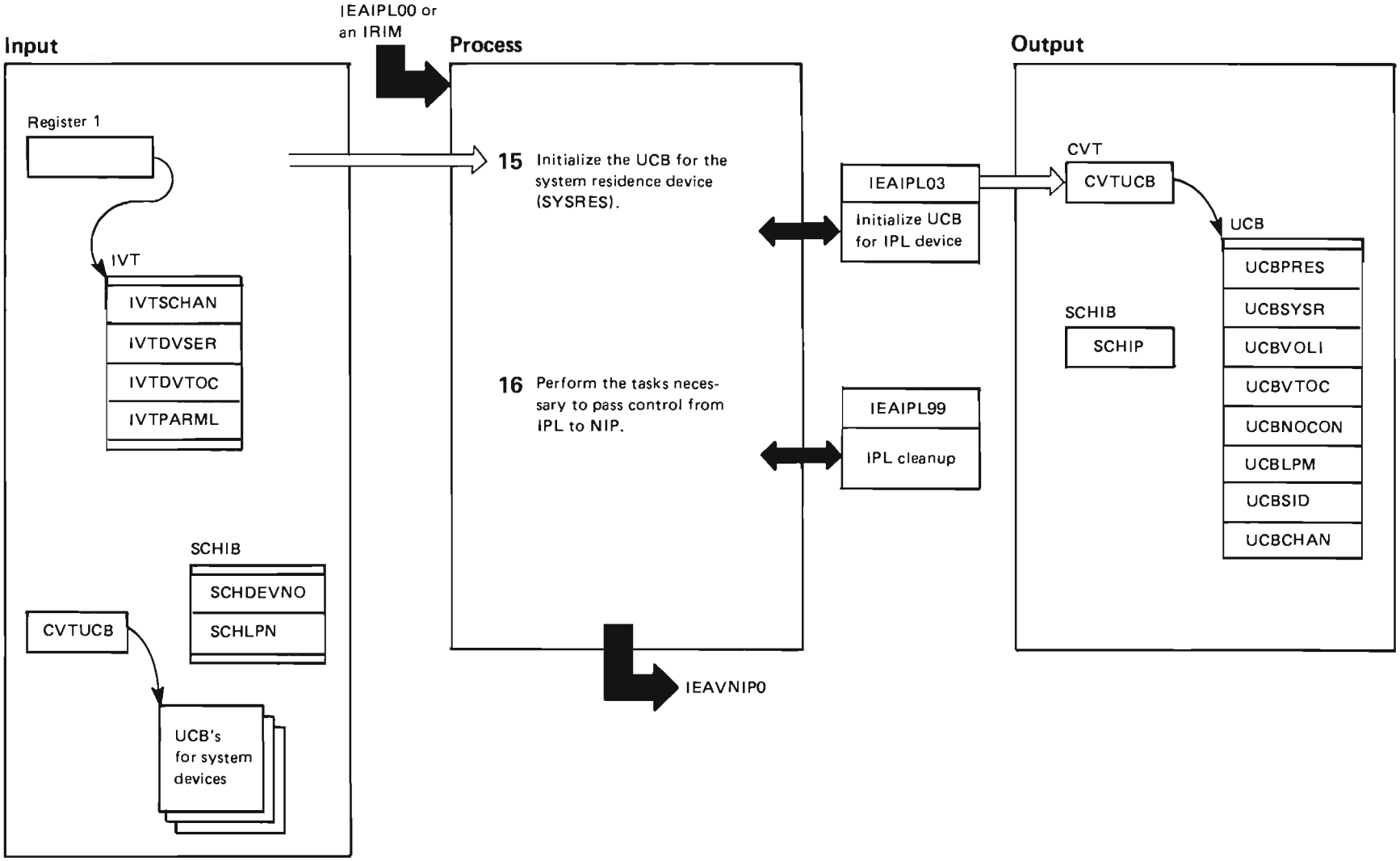
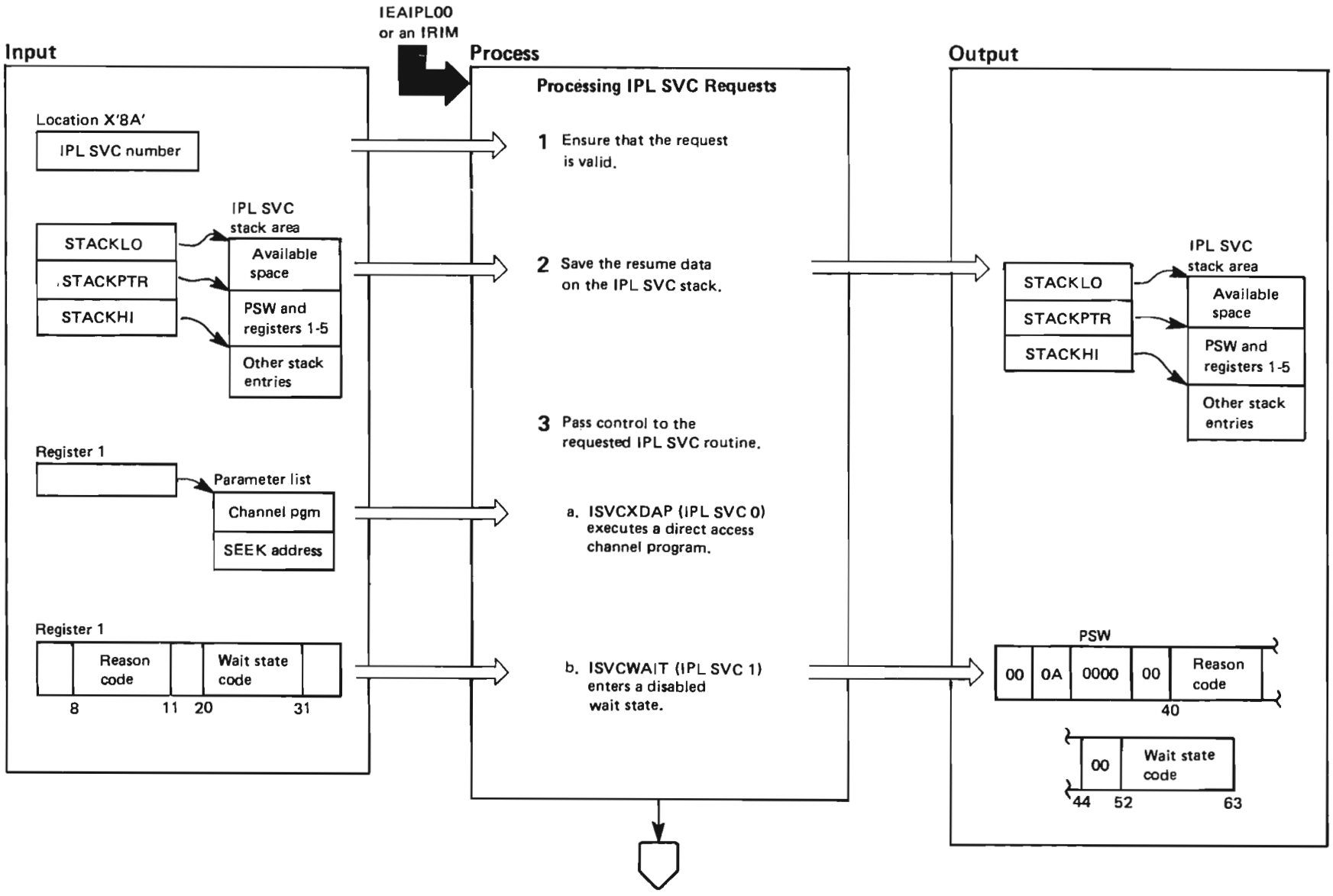


Diagram 1. Initial Program Loader (IEAIPL00) Part 10 of 18

Exstended Description	Module	Label
15 IEAIPL00 calls the input/output supervisor (IOS) IRIM, IEAIPL03, to locate and initialize the unit control block (UCB) for the IPL device.	IEAIPL03	
After the hardware IPL sequence is completed, all subchannels except for the IPL subchannel are disabled. The IOS IRIM obtains from the subchannel the device number (SCHDEVNO) that the operator specified as the number of the IPL device and searches the UCB chain to find the UCB is not for a DASD or is for a non-base exposure of a multiple exposure device, the IOS IRIM puts the system in a wait state with a code of X'031'. Otherwise, the IOS IRIM initializes the following fields in the UCB:		
Field — *Initialization value* UCBSID — Identifier of the subchannel UCBPRES — Permanently resident flag (set to '1' B) UCBSYSR — System residence flag (set to '1' B) UCBVOLI — Volume serial number of the system residence volume UCBCHAN — Address of the channel used in the IPL procedures UCBLPM — Logical path mask UCBNOCON — Not-connected flag field (set to '0' B which indicates that the device is connected)		
16 IEAIPL00 calls IEAIPL99 to clean up the IPL environment in preparation for passing control to NIP, the second phase of system initialization. IEAIPL99 places IPL information into common storage so the NIP can access it, initializes storage management, and establishes the available frame queues. It establishes addressability for NIP and frees the storage occupied by the IPL modules and workarea.	IEAIPL99	

Diagram 1. Initial Program Loader (IEAIPL00) Part 11 of 18



Extended Description	Module	Label
Whenever this module or an IRIM issues an IPL SVC instruction, the hardware loads the address of IEAIPL00's IPL SVC interruption handler (entry point SVCFLIH within this module) from the IPL SVC new PSW. As a result, SVCFLIH gets control to process the SVC request.		
1 If the IPL SVC number is one that has not been defined, SVCFLIH loads wait state code X'074' and reason code 1 in the PSW.	IEAIPL00	SVCFLIH
2 When processing all IPL SVC instructions except IPL SVC 3, SVCFLIH saves the information required to properly resume the interrupted program in a 32-byte entry in a IPL SVC stack. Each entry contains: • The SVC old PSW at the time of the interruption • The IPL SVC number • The contents of registers 1 through 5 at the time of the interruption		
An IPL SVC 3 instruction needs no stack entry because the issuer of the IPL SVC 3 instruction is requesting return to the routine that issued the IPL SVC at the top of the stack. The necessary resume data is already stored there.		
Note that maintaining the SVC stack allows IPL SVC routines to issue other IPL SVCs: that is, IPL SVC requests can be nested. When nesting exceeds the dimensions of the stack, SVCFLIH enters a wait state code of X'074' in the PSW.		
3 To route control to the requested IPL SVC subroutine, SVCFLIH issues the IPL SVC number as an index into an internal branch table. If the IPL SVC number is undefined, SVCFLIH loads wait state code X'074' in the PSW. Each SVC subroutine is described below:		SVCROUTE
a. ISVCXDAP (IPL SVC 0) ISVCXDAP starts a channel program to execute a direct access channel program to the IPL volume. The channel program must contain only format-1 CCWs and should not contain SEEKs or use data chaining. The initial SEEK CCW is prefixed to the requestor's channel program by this service. All addresses in the channel program must be real addresses.		XDAPSVC
b. ISVCWAIT (IPL SVC 1) ISVCWAIT puts a processor in a disabled wait state. It stores the contents of register 1 in the instruction address section of a PSW that has the wait bit on and that is disabled for I/O and external interruptions. Because this action causes the system to be placed in a disabled wait state, ISVCWAIT does not return to its caller.		WAITSVC

Diagram 1. Initial Program Loader (IEAIPL00) Part 13 of 18

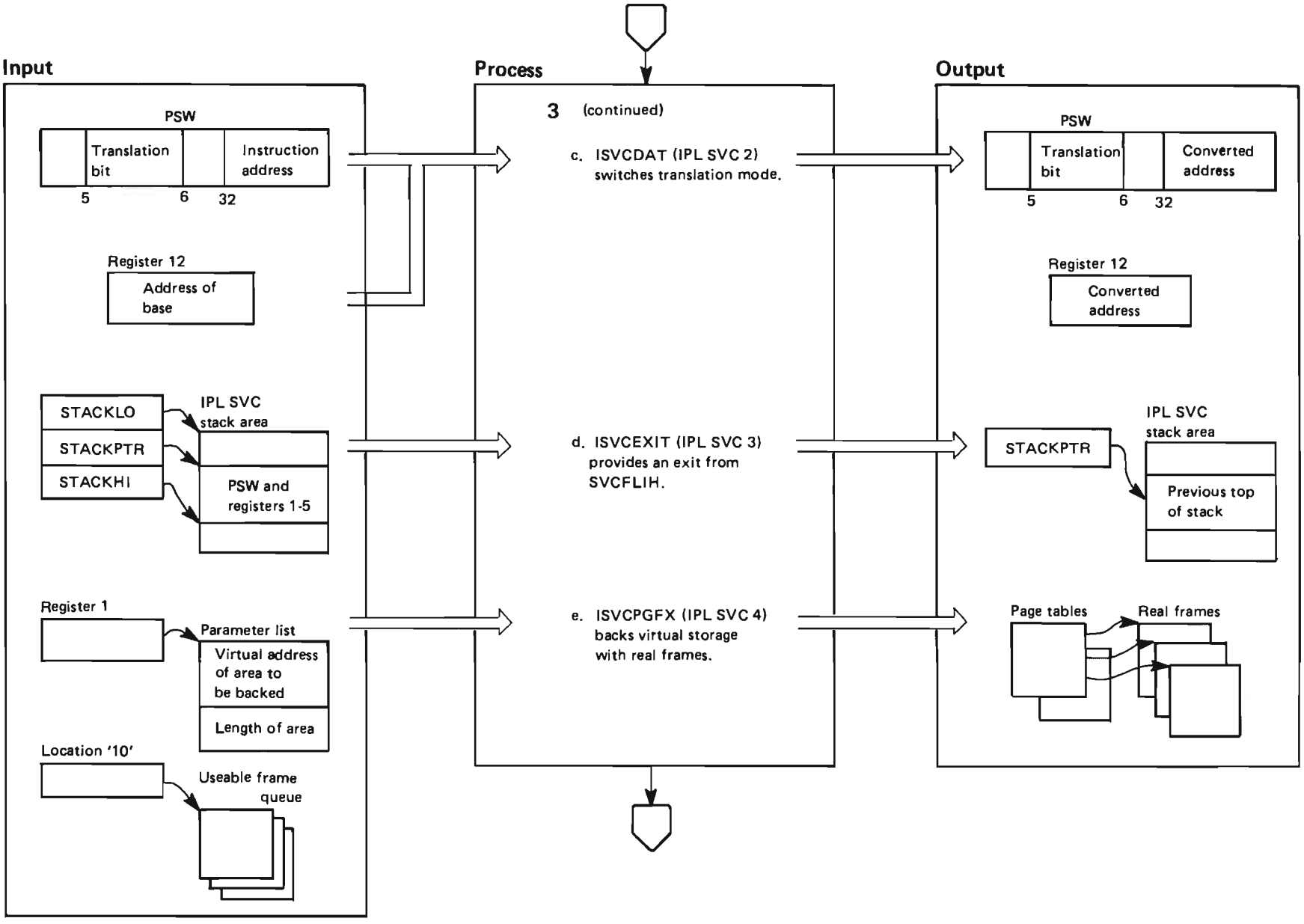


Diagram 1. Initial Program Loader (IEAIPL00) Part 14 of 18

Extended Description	Module	Label
3 (continued)		
c. ISVCDAT (IPL SVC 2) ISVCDAT switches translation mode. The translation bit in the system mask is turned on (meaning DAT-on) or off (meaning DAT-off) and the instruction address in the PSW is converted from real to virtual (or virtual to real) as is the address in register 12, which must be the IRIM base register. It uses IPL SVC 3 to return to the caller.		DATSVC
d. ISVCEXIT (IPL SVC 3) ISVCEXIT returns control to the issuer of the SVC represented by the top element of the SVC stack. To do so, ISVCEXIT: • Makes the top element of the SVC stack available • Restores register 1-5 with the values they had when the top element was created • Loads the PSW that had been saved in the top element		EXITSVC
e. ISVCPGFX (IPL SVC 4) ISVCPGFX backs virtual storage with real frames. If the page is the first within a segment, ISVCPGFX issues IPL SVC 11 to create a page table for the segment and connect the page table to the segment tables. It then backs each page by obtaining a frame and setting the appropriate page table entry to point to it. When all pages containing any part of the area have been backed, ISVCPGFX uses IPL SVC 3 to return to the caller. The real storage obtained might not be contiguous.		PGFXSVC

Diagram 1. Initial Program Loader (IEAIPL00) Part 15 of 18

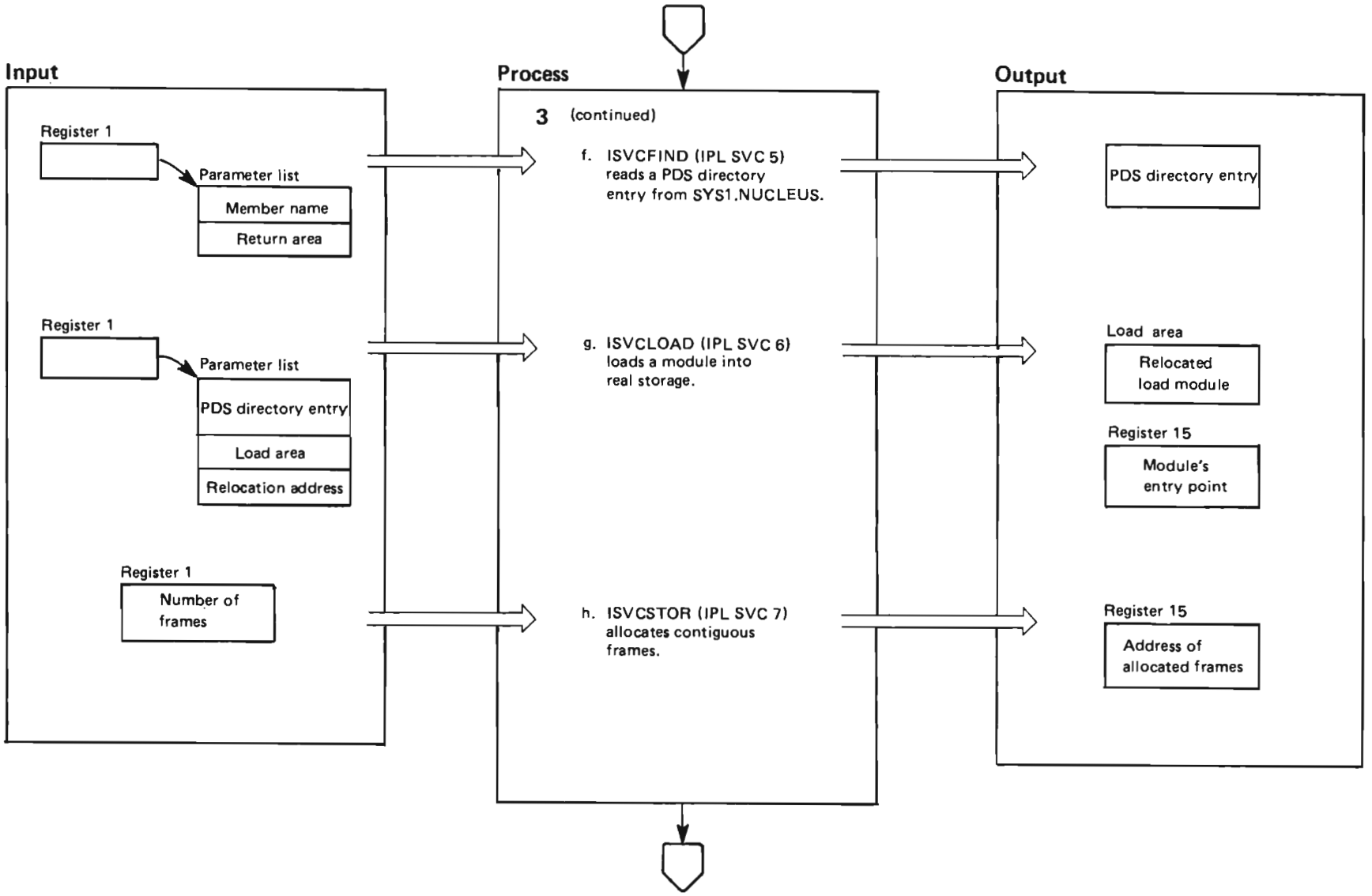


Diagram 1. Initial Program Loader (IEAIPL00) Part 16 of 18

Extended Description	Module	Label
3 (continued)		
f. ISVCFIND (IPL SVC 5) ISVCFIND reads a partitioned data set directory entry for a specified member of SYS1.NUCLEUS. It issues IPL SVC 0 to read the appropriate directory block from SYS1.NUCLEUS and scans it for the requested member name. If it locates an entry for the requested member, it copies the entry into the specified area and sets a return code of 0. Otherwise, it sets a return code of 4. It uses IPL SVC 3 to return to the caller.		FINDSVC
g. ISVCLOAD (IPL SVC 6) ISVCLOAD loads a module from SYS1.NUCLEUS into real storage. ISVCLOAD uses information in the directory entry to read the first text record into real storage at the offset indicated by the linkage editor. Thereafter, is uses information in a control record to read the text record that follows. To resolve address constants within a text record, ISVCLOAD uses the RLD record(s) following each text record in conjunction with the logical address passed as the third parameter. If ISVCLOAD finds an address constant shorter than four bytes, it loads a wait state code of X'076' in the PSW. After successfully loading the module, ISVCLOAD puts the module's entry point into register 15 and returns to its issuer using IPL SVC 3.		LOADSVC
h. ISVCSTOR (IPL SVC 7) ISVCSTOR allocates contiguous real storage. The address of the contiguous area is returned in register 15. The area returned will contain zeroes. ISVCSTOR loads a wait state code of X'070' in the PSW if the requested number of contiguous frames cannot be found.		MSSFCALL

Diagram 1. Initial Program Loader (IEAIPL00) Part 17 of 18

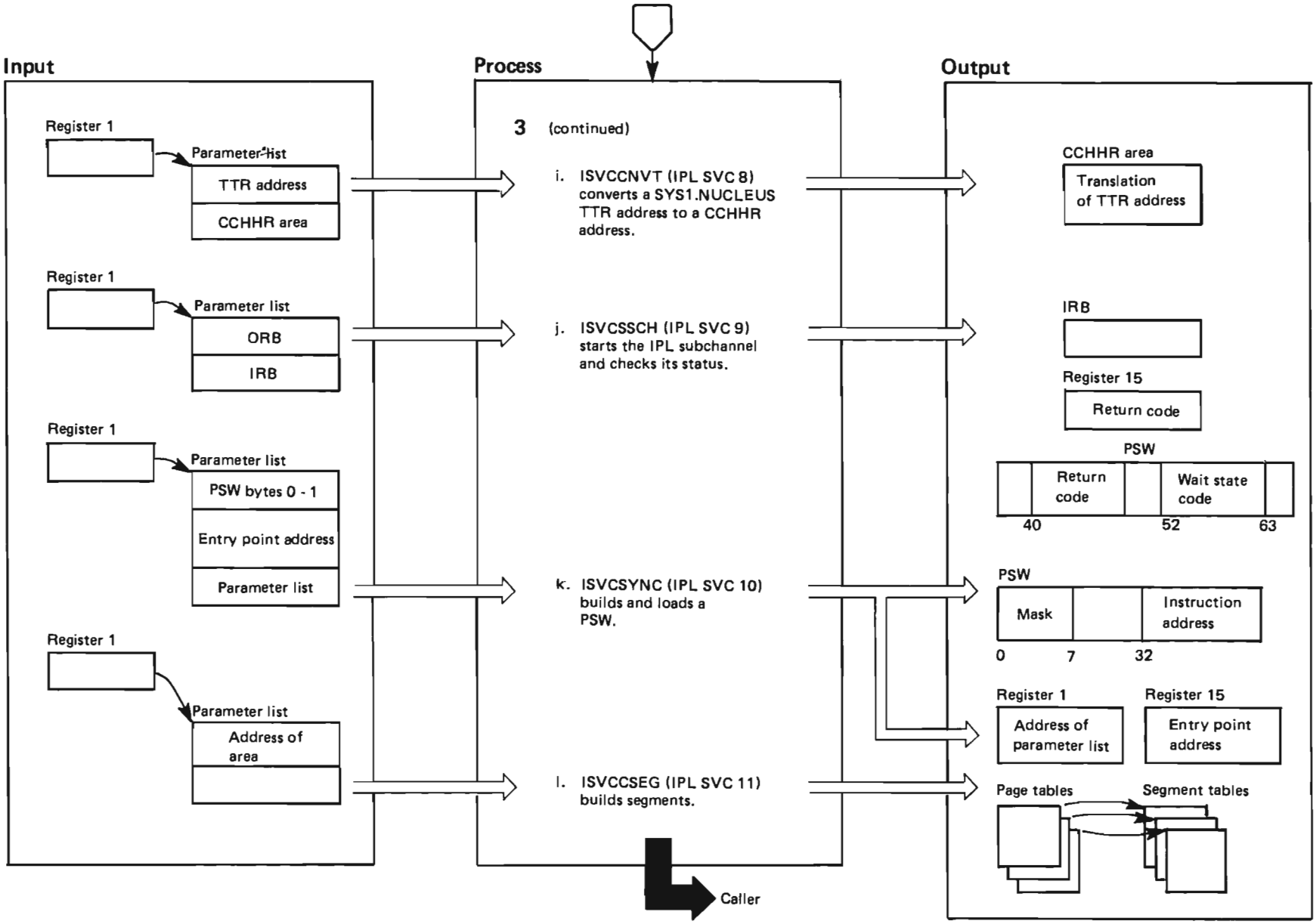


Diagram 1. Initial Program Loader (IEAIPL00) Part 18 of 18

Extended Description	Module	Label
3 (continued)		
i. ISVCCNVT (IPL SVC 8) ISVCCNVT converts the TTR address (the relative address of the record within the data set) of a SYS1.NUCLEUS member to a CCHHR address (the actual address of the record on a DASD). To compute the CCHHR address, ISVCCNVT uses the data set's starting CCHHR address (taken from the format 1 DSCB for the data set), the number of tracks per cylinder (taken from the format 4 DSCB), and the TTR address passed as a parameter.		CNVTSVC
j. ISVCSSCH (IPL SVC 9) ISVCSSCH starts a channel program, waits for the I/O to complete, and determines the subchannel status. It uses the operation request block (ORB) as an operand on the Start Subchannel instruction. If the subchannel starts successfully, ISVCSSCH uses IPL SVC 10 to put the processor in a wait state enabled for I/O interruptions. After an I/O interruption from an IPL device, ISVCSSCH retrieves the subchannel status by issuing a Test Subchannel instruction. The instructions' operand is the interrupt response block (IRB), which was passed as the second parameter. If ISCVSSCH does not detect an exception, ISVCSSCH returns a code of 0. If a unit check occurs, ISCVSSCH returns a code of 4. If any other exception occurs, ISCVSSCH causes the processor to enter a disabled wait state with a wait state code of X'003', X'074' with reason code 4, or X'074' with reason code 5. Unless processing results in a disabled wait state, ISVCSSCH returns to its issuer.		SSCHSVC
k. ISVCSYNC (IPL SVC 10) ISVCSYNS builds and loads a PSW. ISVCSYNC constructs the PSW from the input, loads the parameter list address in register 1 and the entry point address in register 15, then loads the PSW. Either the requested routine receives control, or the system enters an enabled wait state. IEAIPL00 uses ISVCSYNC to pass control to IRIMs and to load enabled wait PSWs. If an IRIM issues an IPL SVC 3 after an IPL SVC 10, control will return to the issuer of SVC 10.		SYNCHSVC
l. ISVCCSEG (IPL SVC 11) ISVCCSEG creates segments by building page tables for all segments containing any part of the specified area and connecting these tables to the segment table. The parameter list consists of two words: the starting virtual address of the area for which ISVCCSEG is to build page tables and the length of the area. Segments that already exist are ignored.		CSEGSVC
Recovery Processing: Processing Program Check Interruptions When a program check occurs in this module or in an IRIM, the hardware loads the address of the program check interruption handler (entry point PCFLIH within this module) from the program check new PSW. As a result, PCFLIH receives control to process the interruption. If a page or segment exception occurs, PCFLIH issues an IPL SVC 4 to back the page containing the failing address, then returns control to the point of interruption. If an addressing exception occurs, PCFLIH examines the instruction causing the exception. If it is a test block instruction, PCFLIH sets the condition code field in the old program check PSW to 3, informing the issuer of the instruction that the block of storage being tested is not online. PCFLIH returns to the point of interruption. When the instruction causing the address exception is not the test block instruction, or when a type of segment exception, or address exception occurs, PCFLIH loads a wait state code of X'019' in the PSW and enters a disabled state.		PCFLIH

Diagram 2. Allocate IPL Message Queue Header (IEAIPL30)/Load IEAIPL35 Part 1 of 6

```
IEAIPL30 - MODULE DESCRIPTION

DESCRIPTIVE NAME: IRIM that loads IEAIPL35, the IPL WTO facility module.

FUNCTION:
 Bring IEAIPL35 into storage and allocate the IPL Message
 Queue Header (MQH).

ENTRY POINT: IEAIPL30

 PURPOSE: Main entry point of the module.

 LINKAGE: LPSW.

 CALLERS: IEAIPL00.

 INPUT: IVT - IPL Vector Table. (Pointed to by register 1 on entry.)

 OUTPUT:
    Module IEAIPL35 loaded into the IPL workspace.
    IPL Message Queue Header allocated from the IPL
    workspace.

 EXIT NORMAL: Return to caller via register 14.

    OUTPUT: See above.

 EXIT ERROR: Disabled wait state.

EXTERNAL REFERENCES:

 ROUTINES: IEAIPL00 SVC Routines.

 CONTROL BLOCKS:
    Common name  Macro Name   Description
    -----------  ----------   -----------
    IVT          IHAIVT       IPL vector table
    MQH          IHAMQH       IPL message queue header
    PDS          IHAPDS       Partitioned data set directory entry
    PGTE         IARPGTE      Page table entry.
```

Diagram 2. Allocate IPL Message Queue Header (IEAIPL30)/Load IEAIPL35 Part 2 of 6

IEAIPL30 - MODULE OPERATION

. Obtain sufficient virtual storage for the IPL Message Queue Header (MQH). This storage is taken from the IPL workspace.

. Save the virtual address of the MQH in the IPL Vector Table (IVT).

. Obtain sufficient contiguous virtual storage for IEAIPL35. This is also taken from the IPL workspace, and is obtained such that the first byte is on a page boundary.

. Obtain sufficient contiguous real storage for IEAIPL35. Note: ISVCLOAD requires that the real storage into which a module is loaded be contiguous. Thus, the virtual storage used to hold IEAIPL35 will be backed by the contiguous real frames obtained here.

. Set up the page table so that the virtual storage that will be used to hold IEAIPL35 is mapped on to the contiguous real storage obtained above. The page table entries are located by attempting to load the real address of each (virtual) page that will be used to load IEAIPL30. All of the page table entries for these pages are invalid, having not yet been backed by (real) frames. When the execution of a LRA instruction cannot be completed due to an invalid page table entry, the real address of the invalid entry is returned. The page table entry so located can thus be filled in with the next (real) frame address and marked as valid.

. Issue ISVCLOAD to load IEAIPL35 into the real storage obtained.

. Save the entry point address of IEAIPL35 in the IVT.

Diagram 2. Allocate IPL Message Queue Header (IEAIPL30)/Load IEAIPL35 Part 3 of 6

IEAIPL30 - DIAGNOSTIC AIDS

ENTRY POINT NAME: IEAIPL30

MESSAGES: None

ABEND CODES: None

WAIT STATE CODES:

Wait state Code	Reason Code	Description
X'055'	X'005'	The PDS entry for module IEAIPL35 in SYS1.NUCLEUS could not be located.
X'072'	None	The amount of storage needed to either allocate the IPL Message Queue Header or load IEAIPL35 exceeds the amount available in the IPL workspace.
X'074'	X'006'	IEAIPL30 has received an unexpected return code from a Load Real Address (LRA) instruction.

RETURN CODES: None

REGISTER CONTENTS ON ENTRY:

```
Register     0 - Irrelevant
Register     1 - Address of the IVT
Registers 2-12 - Irrelevant
Register    13 - Address of a standard save area
Register    14 - Return address
Register    15 - Entry point address
```

REGISTER CONTENTS ON EXIT:

EXIT NORMAL:

Registers 0-15 - Restored to values present on entry.

Diagram 2. Allocate IPL Message Queue Header (IEAIPL30)/Load IEAIPL35 Part 4 of 6

```
IEAIPL35 - MODULE DESCRIPTION

DESCRIPTIVE NAME: IPL WTO (Write To Operator) facility module.

FUNCTION:
 Places a message passed by the caller on the IPL/NIP
 message queue.

ENTRY POINT: IEAIPL35

 PURPOSE: Main entry point of the module.

 LINKAGE: BALR via PL/S macro IPLWTO.

 CALLERS: IPL resource initialization modules (IRIMs).

 INPUT:
    IEAIPL35 is passed the address of a parameter list
    in register one.  The parameter list contains:
    . Address of the message text.
    . Length (in bytes) of the message text.
    . Flag indicating whether this is a Write To Operator (WTO)
      or a Write To Log (WTL).
    . Address of the IPL Vector Table (IVT).

 OUTPUT: None

 EXIT NORMAL: Return to caller via register 14.

 EXIT ERROR:

    OUTPUT:
       . System is placed in a wait state with
         wait state code X'072'.

 EXIT ERROR:

    OUTPUT:
       . System is placed in a wait state with
         wait state code X'074', reason code X'007'.

EXTERNAL REFERENCES:

 ROUTINES: None

 CONTROL BLOCKS:
    Common name/Use    Macro Name   Description
    ---------------    ----------   -----------
    IVT      (R,W)     IHAIVT       IPL Vector Table.
    MQE      (C,R,W)   IHAIMQ       IPL/NIP Message Queue Element.
    MQH      (R,W)     IHAIMQ       IPL/NIP Message Queue Header.
    NWTOHDR (C,R,W)    IEAPMNIP     NIP parameter list mappings.

    Legend - C=Create, R=Read, W=Write, D=Destroy
```

Diagram 2. Allocate IPL Message Queue Header (IEAIPL30)/Load IEAIPL35 Part 5 of 6

IEAIPL35 - MODULE OPERATION

At IPL time (which is defined to be before IEAVNIP0 gets control) console communications are unavailable. Therefore, IEAIPL35 does not actually write to the console. Instead, messages issued at IPL time are saved in the IPL/NIP Message Queue (IMQ). The messages so saved are written to the NIP console by IEAVNIPM after the NIP console has been initialized.

A detailed description of the processing done by IEAIPL35 follows.

. Verify the input parameters. If any errors in the passed parameters are found, IEAIPL35 loads a disabled wait state with wait state code X'074' and reason code X'007'. Otherwise, the parameters are valid, so the next step is performed.

. If there is not enough space left in the IPL workspace to hold an IPL/NIP Message Queue Element (MQE), then IEAIPL35 loads a disabled wait state with wait state code X'072'. Otherwise, there is enough room, so the next step is performed.

. Obtain enough virtual storage from the IPL workspace to contain one MQE, and initialize the new MQE. This initialization includes copying the message text from the input parameter list to the new MQE and setting the proper bit in the NIPWTO parameter list (which is contained in the MQE) when the caller has requested a WTL. (For a WTO, the message will eventually be written, via NIPWTO, to the NIP console and recorded in the NIP hardcopy buffer. For a WTL, the message is only recorded in the NIP hardcopy buffer - it is not written to the NIP console.) The new MQE is chained on to the IPL/NIP Message Queue.

. IEAIPL35 restores registers and returns to its caller.

Diagram 2. Allocate IPL Message Queue Header (IEAIPL30)/Load IEAIPL35 Part 6 of 6

IEAIPL35 - DIAGNOSTIC AIDS

ENTRY POINT NAME: IEAIPL35

MESSAGES: None

ABEND CODES: None

WAIT STATE CODES:

Wait state Code	Reason Code	Description
X'072'	None	The IPL workspace has been exhausted.
X'074'	X'007'	The input to IEAIPL35 is invalid.

RETURN CODES: None

REGISTER CONTENTS ON ENTRY:

```
Register     0 - Irrelevant
Register     1 - Address of IEAIPL35 parameter list.
Registers 2-12 - Irrelevant
Register    13 - Address of a standard save area
Register    14 - Return address
Register    15 - Entry point address
```

REGISTER CONTENTS ON EXIT:

EXIT NORMAL:

```
  Registers 0-15 - Restored to values present on entry.


  Register      0 - Unpredictable.
  Register      1 - Input to ISVCWAIT.
  Registers   2-5 - Unpredictable.
  Register      6 - Address of IEAIPL35's parameter list.
  Registers  7-11 - Unpredictable.
  Register     12 - IEAIPL35 base address.
  Register     13 - IEAIPL35 save area address.
  Registers 14-15 - Unpredictable.


  Register      0 - Unpredictable.
  Register      1 - Input to ISVCWAIT.
  Registers   2-5 - Unpredictable.
  Register      6 - Address of the invalid parameter list.
  Registers  7-11 - Unpredictable.
  Register     12 - IEAIPL35 base address.
  Register     13 - IEAIPL35 save area address.
  Registers 14-15 - Unpredictable.
```

Diagram 3. Build DAT-On Nucleus (IEAIPL41) Part 1 of 3

```
IEAIPL41 - MODULE DESCRIPTION

DESCRIPTIVE NAME: DAT-ON nucleus load table builder

FUNCTION:
 Build DAT-ON nucleus load tables.

ENTRY POINT: IEAIPL41

 PURPOSE: See FUNCTION entry

 LINKAGE: LPSW

 CALLERS: IEAIPL00

 INPUT:
    IVT - IPL vector table
    PDS directory
    CESD records from SYS1.NUCLEUS in the format described in
         the declare for 'CESDREC'

 OUTPUT:
    Updates nucleus load list, CESD list, and relocation
    factor table.

 EXIT NORMAL: Return to caller via register 14

 EXIT ERROR: Disabled wait state

EXTERNAL REFERENCES:

 ROUTINES: IEAIPL00 SVC routines

 CONTROL BLOCKS:
    IVT - IPL vector table
    PDS2 - Pds directory entry
    NLLE - Nucleus load list element
    CESD - Composite external symbol dictionary

TABLES:
 Relocation table (mapped 1 to 1 with CESD list)
  - Address table (load address of each control section)
  - RLF table (relocation factor of each control section)
 CESD table
```

Diagram 3. Build DAT-On Nucleus (IEAIPL41) Part 2 of 3

IEAIPL41 - MODULE OPERATION

1. Determine which DAT-ON nucleus is to be loaded.

2. Construct the module name of this dat on-nucleus. Builds a nucleus load list element for the DAT-ON nucleus, and chains it to the nucleus load list created by IEAIPL40.

3. For each element in the nucleus load list does the following:

   - Reads the associated pds directory for each module.
   - Reads all associated composite external symbol dictionary (CESD) entries.
   - Accumulate a list of all ENTRY-POINT names

4. The entry point names are checked to verify that no duplicate entry points exist. If duplicate names are found, messages are issued and a non-restartable wait-state is entered.

5. Creates an entry in the nucleus load list element for each module containing the starting address and a pointer to the relocation factor for each control section.
   This is done by determining the size of each of four sections of the nucleus (read/write,read only, read only extended, & read/write extended, and calculating the displacement from the beginning of the section in which a module resides.

Diagram 3. Build DAT-On Nucleus (IEAIPL41) Part 3 of 3

IEAIPL41 - DIAGNOSTIC AIDS

ENTRY POINT NAME: IEAIPL41

MESSAGES:

```
IEA086W - xxxxxxxx not found in SYS1.NUCLEUS
IEA087W - ENTRY-POINT xxxxxxxx appears in yyyyyyyy
          and zzzzzzzz
IEA088W - Nucleus requires more storage below 16 mb
          than is available
IEA091I - Nucleus X selected
```

ABEND CODES: None

WAIT STATE CODES:

```
          All are non-restartable.
X'025' = An entry-point appears in more than
          one module being loaded into DAT-ON
          nucleus.
X'055' = A module was not found in SYS1.NUCLEUS
          Reason code 2.
X'071' = Virtual storage below 16 mb has been
          exhausted.
X'072' = IRIM work space exhausted.
```

RETURN CODES:

EXIT NORMAL:

0

REGISTER CONTENTS ON ENTRY:

Reg 1 - Address of IPL vector table

REGISTER CONTENTS ON EXIT: Irrelevant

Diagram 4. Nucleus Map Creation (IEAIPL05) Part 1 of 3

```
IEAIPL05 - MODULE DESCRIPTION

DESCRIPTIVE NAME: DAT-ON nucleus map builder

FUNCTION:
 Build DAT-ON nucleus map

ENTRY POINT: IEAIPL05

 PURPOSE: Main ENTRY-POINT of the module

 LINKAGE: LPSW from IEAIPL00

 CALLERS: IEAIPL00

 INPUT:
    IVT - IPL vector table (pointed to by register 1 on entry)
    Nucleus Load List (pointed to IVTNLLEF)

 OUTPUT:
    Nucmap - Nucleus map stored at the top of read-only
             extended nucleus

 EXIT NORMAL: Return to caller via register 14

EXTERNAL REFERENCES:

 ROUTINES: None

 CONTROL BLOCKS:
    IVT  - IPL vector table
    NLLE - Nucleus load list element
    NUCMENT - Nucleus map entry
    PDS2 - Pds directory entry

TABLES:
 Nucleus Load List
 CESD table
 Relocation table
 Nucleus map
```

Diagram 4. Nucleus Map Creation (IEAIPL05) Part 2 of 3

IEAIPL05 - MODULE OPERATION

1. Access the following tables:
   Nucleus load list (NLLEs)
   - CESD entries
   - Relocation table

   and use this information to calculate the offset from label to end of a csect and create an entry for the nucleus map.

2. Setup pointers to the nucleus map and sort the map entries in ascending order of the address fields.

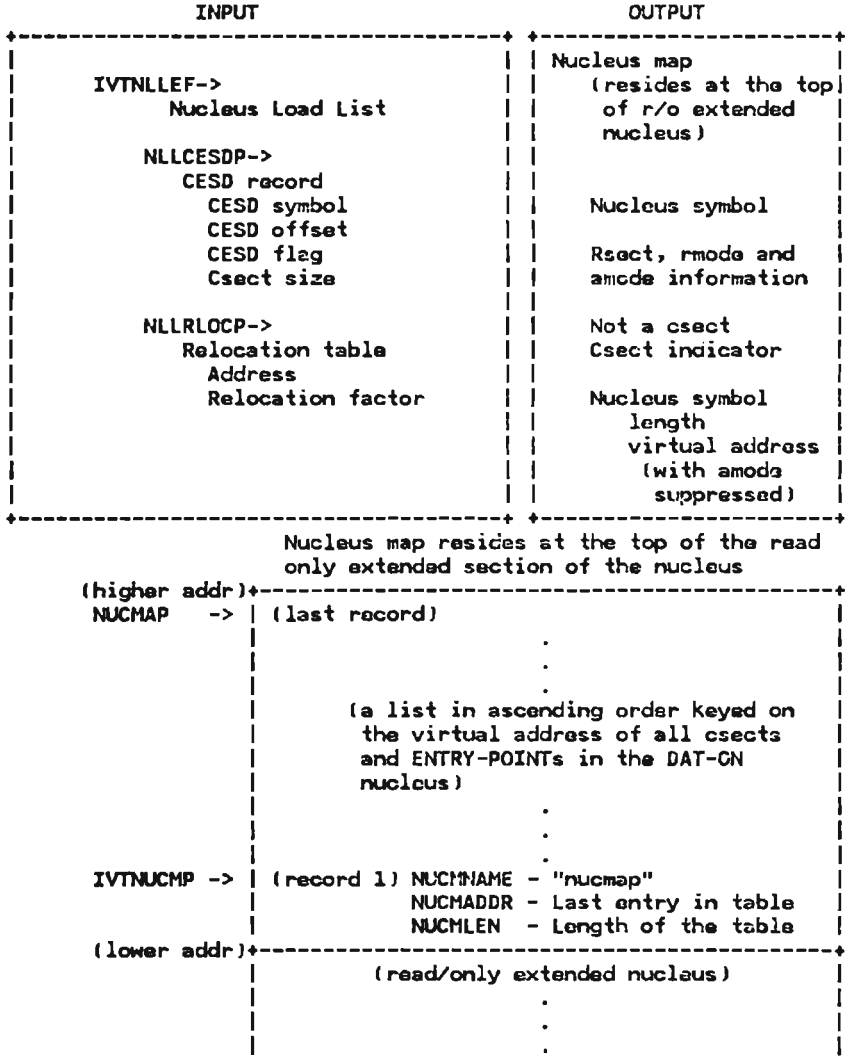


Diagram 4. Nucleus Map Creation (IEAIPL05) Part 3 of 3

IEAIPL05 - DIAGNOSTIC AIDS

ENTRY POINT NAME: IEAIPL05

MESSAGES: None

ABEND CODES: None

WAIT STATE CODES: None

RETURN CODES:

EXIT NORMAL:

0

REGISTER CONTENTS ON ENTRY:

Reg 1 - Address of the IPL vector table.

REGISTER CONTENTS ON EXIT: Irrelevant

Diagram 5. Load DAT-Off Nucleus and DAT-On Nucleus (IEAIPL02) Part 1 of 3

```
IEAIPL02 - MODULE DESCRIPTION

DESCRIPTIVE NAME: Load DAT-OFF and DAT-ON nuclei

FUNCTION:
 Loads Dat-off nucleus modules into contiguous real
 storage and scatter loads the Dat-on nucleus into
 virtual storage.  Also establishes addressability
 between the Dat-on and the Dat-off nuclei by
 processing entries in the Dat-off nucleus linkage
 table.

ENTRY POINT: IEAIPL02

 PURPOSE: See FUNCTION

 LINKAGE: LPSW from IEAIPL00

 CALLERS: IEAIPL00

 INPUT:
    IVT - IPL vector table (pointed to by register 1 on entry)
    nucleus load list (pointed to by IVTNLLEF)

 OUTPUT:
    Dat-off nucleus in real storage
    Dat-on nucleus in virtual storage
    Dat-on/Dat-off nuclei linkage table
    Initialized CVT fields:
      CVTDOFFE -- Dat-off nucleus end address
      CVTDOFFS -- Dat-off nucleus start address
      CVTERWNE -- Extendad read/write nucleus end address
      CVTERWNS -- Extended read/write nucleus start address
      CVTNUCLS -- Selected nucleus id
      CVTNUCMP -- Nucleus map address
      CVTPROD  -- System id
      CVTRONE  -- Read/only nucleus end address
      CVTRONS  -- Read/only nucleus start address
      CVTRWNE  -- Read/write nucleus end address
      CVTRWNS  -- Read/write nucleus start address
      CVTSVPRC -- Service processor status
      CVTIOCID -- I/O configuration identifier

 EXIT NORMAL: Return to IEAIPL00

 EXIT ERROR: Disabled wait state (via IPL isvcwait)

EXTERNAL REFERENCES:

 ROUTINES: IEAIPL00 SVC routines

 CONTROL BLOCKS:
    CCW      - Channel command word
    CVT      - Communications vector table
    IHADATOF - DAT-ON/dat-off nuclei LINKAGE table
    IVT      - IPL vector table
    NLLE     - Nucleus load list element
    PDS2     - Pds directory entry

TABLES:
 .  Relocation table (mapped 1 to 1 with CESD list)
    - Address table (load address of each control section)
    - Rlf table (relocation factor of each control section)
 .  IEAVEDAT - Dat-off/Dat-on nucleus LINKAGE table
```

Diagram 5. Load DAT-Off Nucleus and DAT-On Nucleus (IEAIPL02) Part 2 of 3

IEAIPL02 - MODULE OPERATION

- Set name of Dat-off module to be loaded
- Read PDS directory entry for that member
- Allocate enough contiguous real frames from the IPL frame queue to hold the module
- Load the module into that storage
- Loop through the Nucleus Load List and read each member into virtual storage. Each control section within each module is read into the storage location specified by the relocation table entry corresponding to the CESD list entry for that control section
- The nucleus map is used to resolve external symbol references as the modules are read into virtual storage.
- Load modules into storage locations set up by IEAIPL41.
- Initialize the Dat-on to Dat-off nucleus linkage table
- Place start and end addresses of the Dat-off, read-write, read-only, and extended read-write nuclei into the IVT
- Place I/O configuration identifier and system product information from the IVT into the CVT.

Diagram 5. Load DAT-Off Nucleus and DAT-On Nucleus (IEAIPL02) Part 3 of 3

IEAIPL02 - DIAGNOSTIC AIDS

ENTRY POINT NAME: IEAIPL02

MESSAGES:

IEA086W - xxxxxxxx not found in SYS1.NUCLEUS
IEA089W - xxxxxxxx has more than 1 csect but was not link editied with sctr option
IEA090W - Load real address failed during IEAIPL02 processing
IEA092I - Warning: Unresolved external reference yyyyyyyy in module xxxxxxxx
IEA093I - Module xxxxxxxx contains unresolved weak external reference yyyyyyyy - Written to the system log only.

ABEND CODES: None

WAIT STATE CODES:

X'054' = A member that was to be loaded into the DAT-ON nucleus contained more than one csect and was not link edited in scatter format.

X'055' = The selected nucleus member data set was not found in SYS1.NUCLEUS. In this module, may occur only if the dat-off nucleus member was not found.

X'074' = Reason code 6 - Invalid address returned from a load real address instruction

RETURN CODES: None

REGISTER CONTENTS ON ENTRY:

Reg 1 - Address of the IPL vector table.

REGISTER CONTENTS ON EXIT: Irrelevant

Diagram 6. Virtual Storage Management IPL Initialization (IEAIPL04) Part 1 of 12

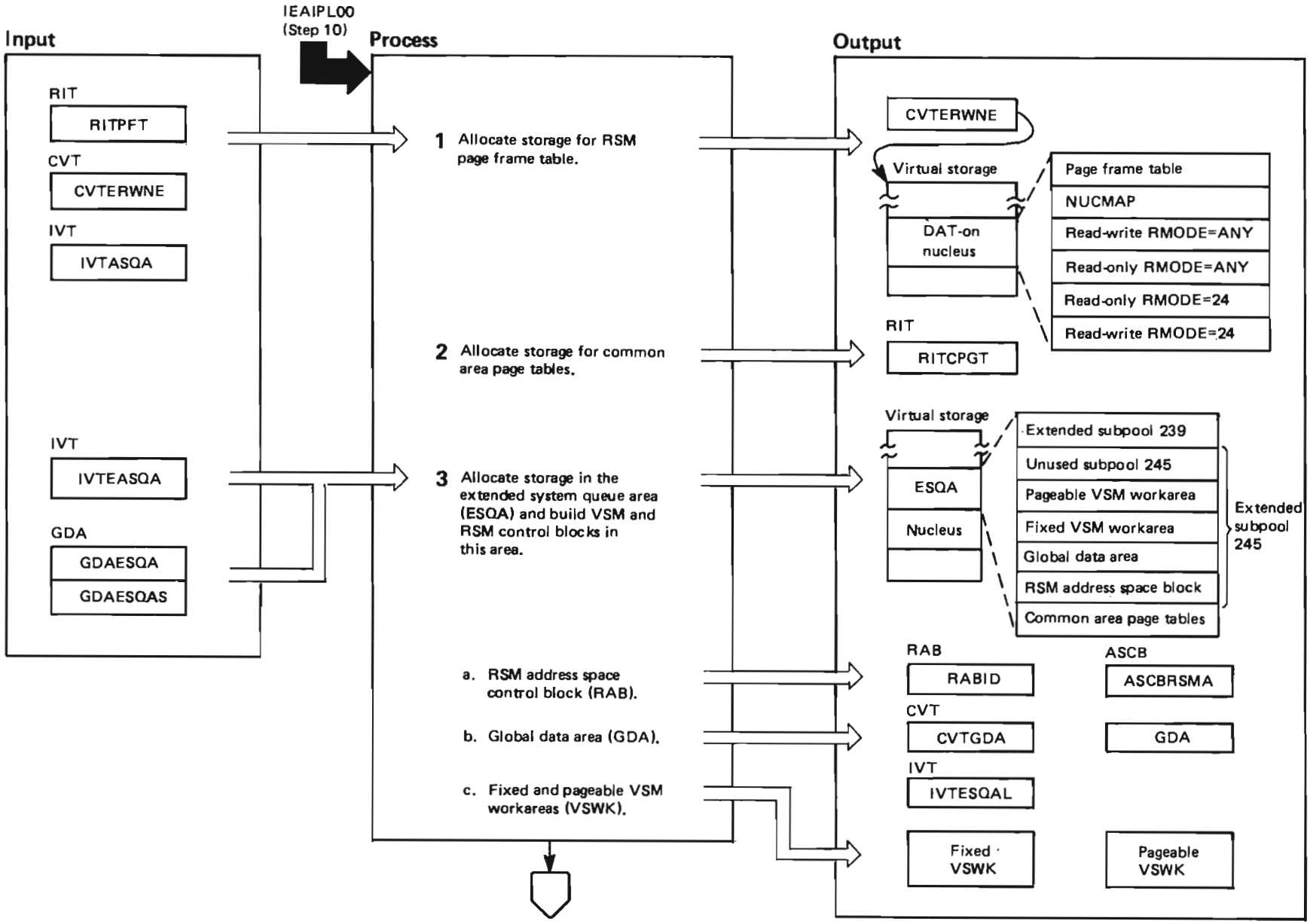


Diagram 6. Virtual Storage Management IPL Initialization (IEAIPL04) Part 2 of 12

Extended Description	Module	Label
IEAIPL00 calls the virtual storage management (VSM) IRIM, IEAIPL04, to allocate storage for the system queue area (SQA), the extended SQA (ESQA), and the extended local SQA (ELSQA). It builds the control blocks that are required by VSM and real storage management (RSM). A map of virtual storage upon exit from IEAIPL04 is included at the end of this diagram as Figure 5-3.		
1 The VSM IRIM defines space immediately above the DAT-on nucleus for the RSM page frame table. It adjusts the pointer to the end of the extended read-write nucleus (CVTERWNE) so that the page frame table is included in the nucleus and places a pointer to the page frame table in the RSM internal table (RIT). RSM will initialize the RSM page frame table in IEAIPL06.	IEAIPL04	
2 The VSM IRIM allocates eight megabytes of virtual storage immediately above the RSM page frame table for the common area page tables. In IEAIPL06, RSM will initialize the required number of page tables and, in IEAVNPB8, VSM frees the storage that RSM does not need for the common area page tables. This storage becomes part of the extended subpool 245.		
3 The VSM IRIM allocates 320K of storage for extended subpools 239 and 245 (160K for each). The VSM IRIM places no data in subpool 239. It allocates storage for the following control blocks in extended subpool 245: a. The RSM address space control block (RAB) RSM will initialize this control block in IEAIPL06. IEAIPL04 sets the address of the RAB in the ASCB and initializes the RAB's control block ID.		
b. The global data area (GDA) The VSM IRIM calculates the temporary start address of the extended CSA (GDAECSA) by adding the size of the ESQA (GDAESQAS) to the start address of the ESQA (GDAESQA). It initializes a field in the CVT that points to the GDA and adjusts a field (EVTESQAL) to reflect the size of ESQA. The VSM IRIM initializes the following fields in the GDA (addresses are virtual unless otherwise stated): *Field initialized* — *Initialization value* GDAID — 'GDA' GDAESQAS — Size of the ESQA GDAECSA — Address of the ECSA GDAESQA — Address of the ESQA GDASPTT — Address of subpool translation table GDAWRKAP — Address of workarea for global pageable subpools GDAWRKA — Address of workarea for global fixed subpools GDAPGTAD — Address of common page tables GDAPGTSZ — Size of common page tables System components cannot issue a FREEMAIN against the common page tables. c. Two VSM work areas (VSWK) These are permanent work areas that VSM uses to process storage requests for global fixed subpools and global pageable subpools. Within each VSWK is a dynamic work area from which VSM obtains storage for its own variables. The VSM IRIM initializes these fields in the VSWKs: *Field initialized* — *Initialization Value* VSWKID — 'VSWK' VSWKGDA — Address of the GDA VSWKTOP — Address of the top of the stack VSWKSADR — Address of the stack area		

Diagram 6. Virtual Storage Management IPL Initialization (IEAIPL04) Part 3 of 12

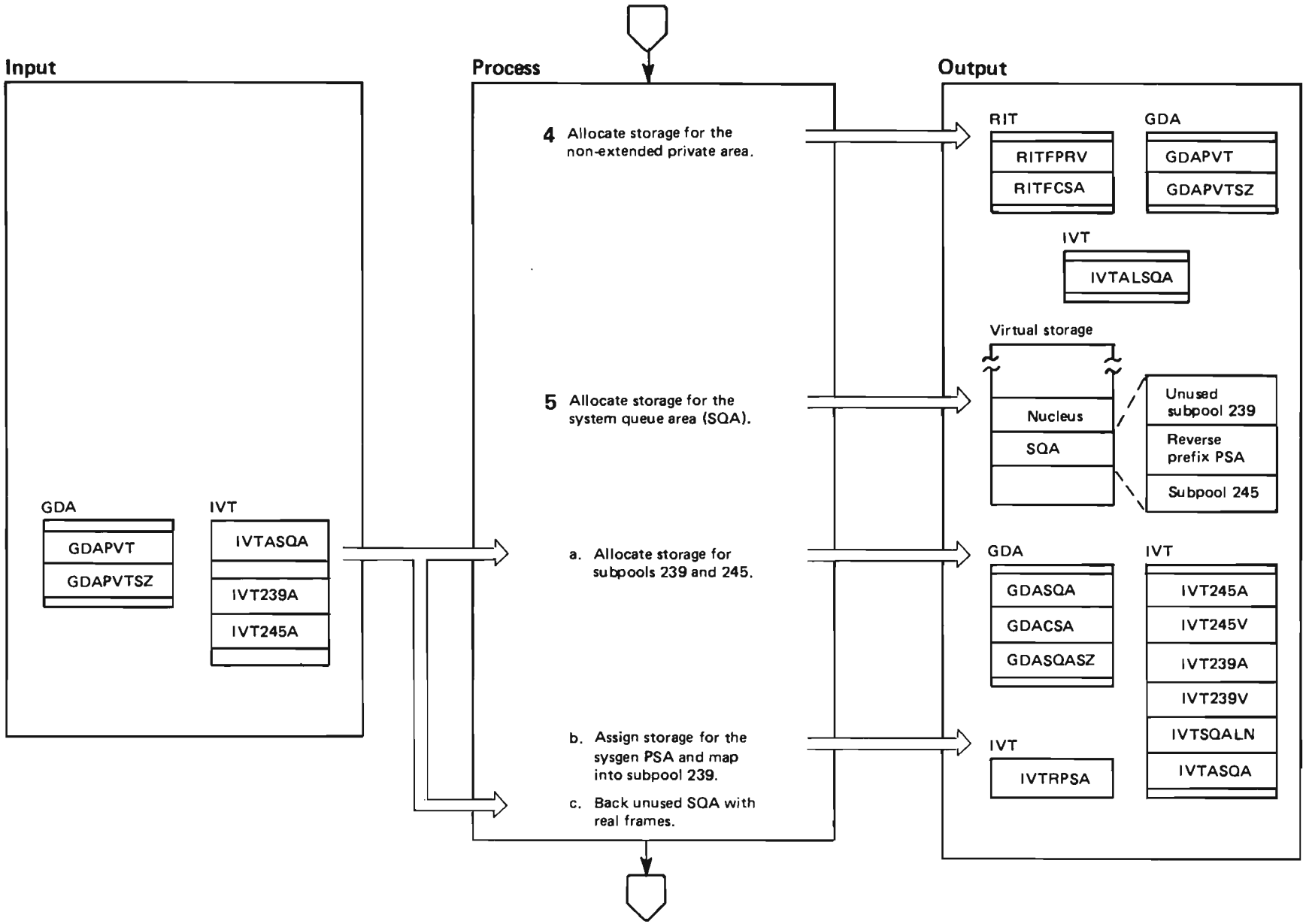


Diagram 6. Virtual Storage Management IPL Initialization (IEAIPL04) Part 4 of 12

Extended Description	Module	Label
4 The VSM IRIM allocates two megabytes of virtual storage for the non-extended private area. The IRIM initializes fields in the GDA and the RIT that contain address and size information about the private area.		
5 The VSM IRIM allocates 256K of storage just below the nucleus for the non-extended subpools 239 and 245 128K for each). The VSM IRIM initializes it from the top down as follows:		
a. The VSM IRIM allocates 128K of non-extended subpool 239 starting at the start address of the DAT-on nucleus (CVTRWNS) and allocating down 128K. It places no data in this area. The VSM IRIM then allocates 128K down from this point for subpool 245. IVTASQA points to the beginning of subpool 245, which is also the start address for the SQA. By adding the address of the start of the private area (GDAPVT) to the size of the private area (GDAPVTSZ) and comparing the sum with IVTASQA, the VSM IRIM can determine if the SQA overlaps the area that the VSM IRIM previously assigned to the private area. (See step 4). If there is an overlap, the VSM IRIM calls ISVCWAIT (IPL SVC 1) which puts the processor in a wait state with a code of X'71'.		
b. The VSM IRIM calls ISVCCSEG (IPL SVC 11) to create the page tables for the absolute zero PSA at the start of subpool 239. This PSA will be used as the reverse prefix PSA. The VSM RIM updates the page table entry to point to absolute zero and maps the PSA that was initialized during system generation (also called the SYSGEN) PSA) to the start address of subpool 239. It sets IVTRPSA to point to this area.		MAPVIRT
c. The IRIM calls ISVCPGFX (IPL SVC 4) to back the unassigned non-extended SQA (both subpools 239 and 245) with real storage frames. The VSM IRIM sets the fetch-protected storage key for subpool 239. It also initializes fields in the GDA that contain the size and address of the non-extended SQA and the start address of the non-extended CSA.		BACKPAGE SETKEY

Diagram 6. Virtual Storage Management IPL Initialization (IEAIPL04) Part 5 of 12

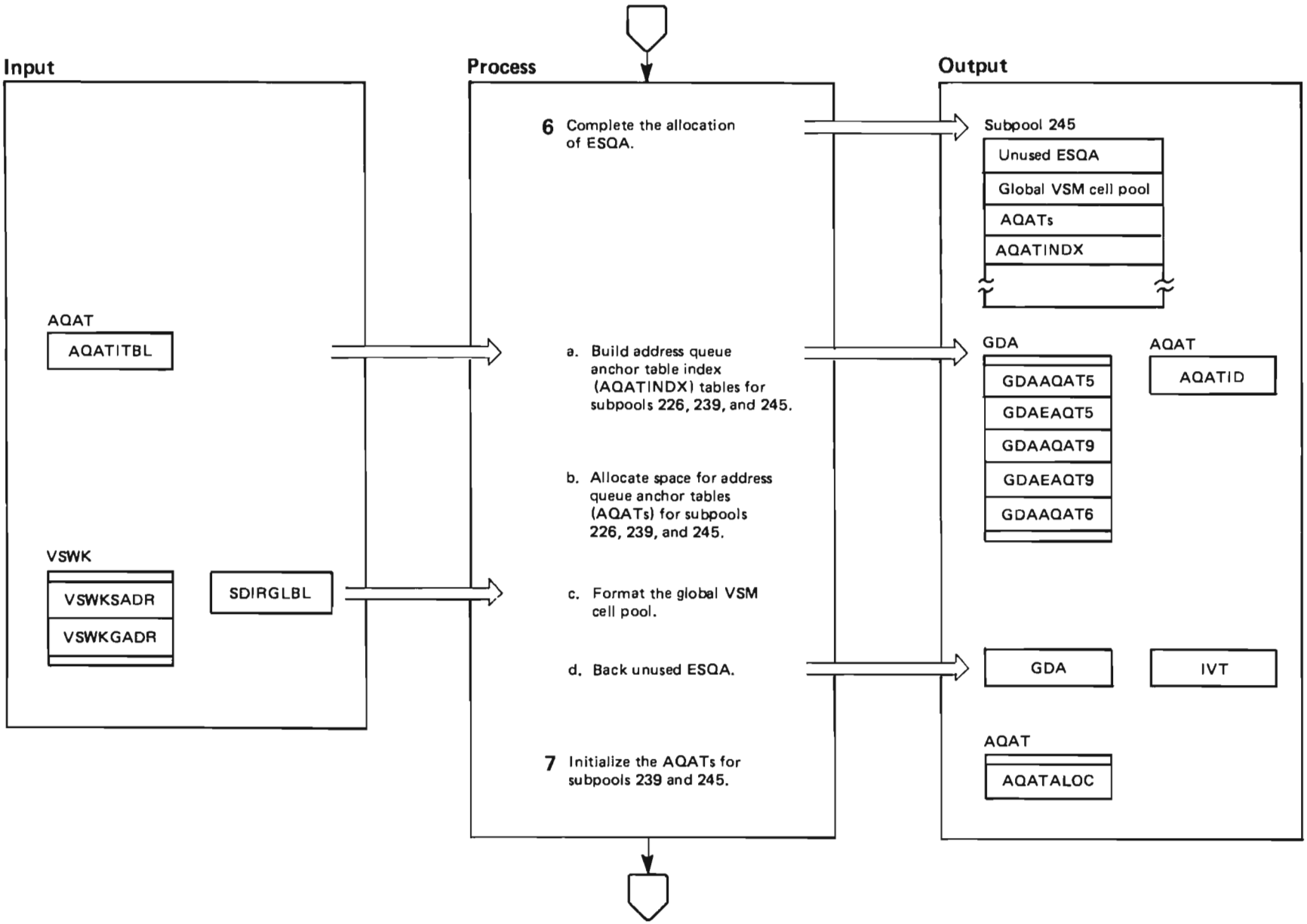


Diagram 6. Virtual Storage Management IPL Initialization (IEAIPL04) Part 6 of 12

Extended Description	Module	Label
6 The VSM IRIM continues the initialization of extended subpool 245 that it started in Step 3. It allocates storage above the pageable VSM workarea as follows:		
a. The address queue anchor table index (AQATINDX) tables for subpools 226, 239, and 245. These tables contain pointers to the address queue anchor tables (AQATs) that the VSM IRIM will allocate in Step 6b. For descriptions of AQATs and AQATINDXs, see Volume 1 of *Debugging Handbook*.		INITINDX
b. Address queue anchor tables (AQATs) for SQA and ESQA The VSM IRIM allocates space for the AQATs. The AQATs describe the allocation of virtual storage in the ESQA and SQA. The IRIM will initialize these AQATs in Step 7.		
c. The global VSM cell pool In order to format the VSM cell pool, the VSM IRIM first calls the service routine NUCLKUP to obtain the address of the formatting routine IGVVSFMT. It then calls IGVVSFMT to format the VSM cell pool.		

Extended Description	Module	Label
d. Unused ESQA area The VSM IRIM backs the unused ESQA area with real frames and sets the fetch-protected storage key for extended subpool 239. The VSM IRIM sets fields to point to ESQA areas as follows:		BACKPAGE SETKEY

Field initialized	*Initialization Value*
GDASMAD	*Start of global storage management area
GDASMSZ	Size of global storage management area
IVTE239A	Address of unused extended subpool 239
IVTE239V	Size of unused extended subpool 239
IVTE245A	Address of unused extended subpool 245
IVTE245V	Size of unused extended subpool 245

*The global storage management area includes the RSM address space block and the VSM cell pool and all areas in between. System components cannot issue a FREEMAIN macro instruction for storage in this area.

7 The VSM IRIM initializes AQATs for the extended and non-extended subpools 239 and 245. The VSM IRIM calls the service routine, IGVAAQAT, to set the AQAT allocation bits (AQATALOC) to indicate that 192K of storage for the SQA is allocated.		INITAQAT

Diagram 6. Virtual Storage Management IPL Initialization (IEA1PL04) Part 7 of 12

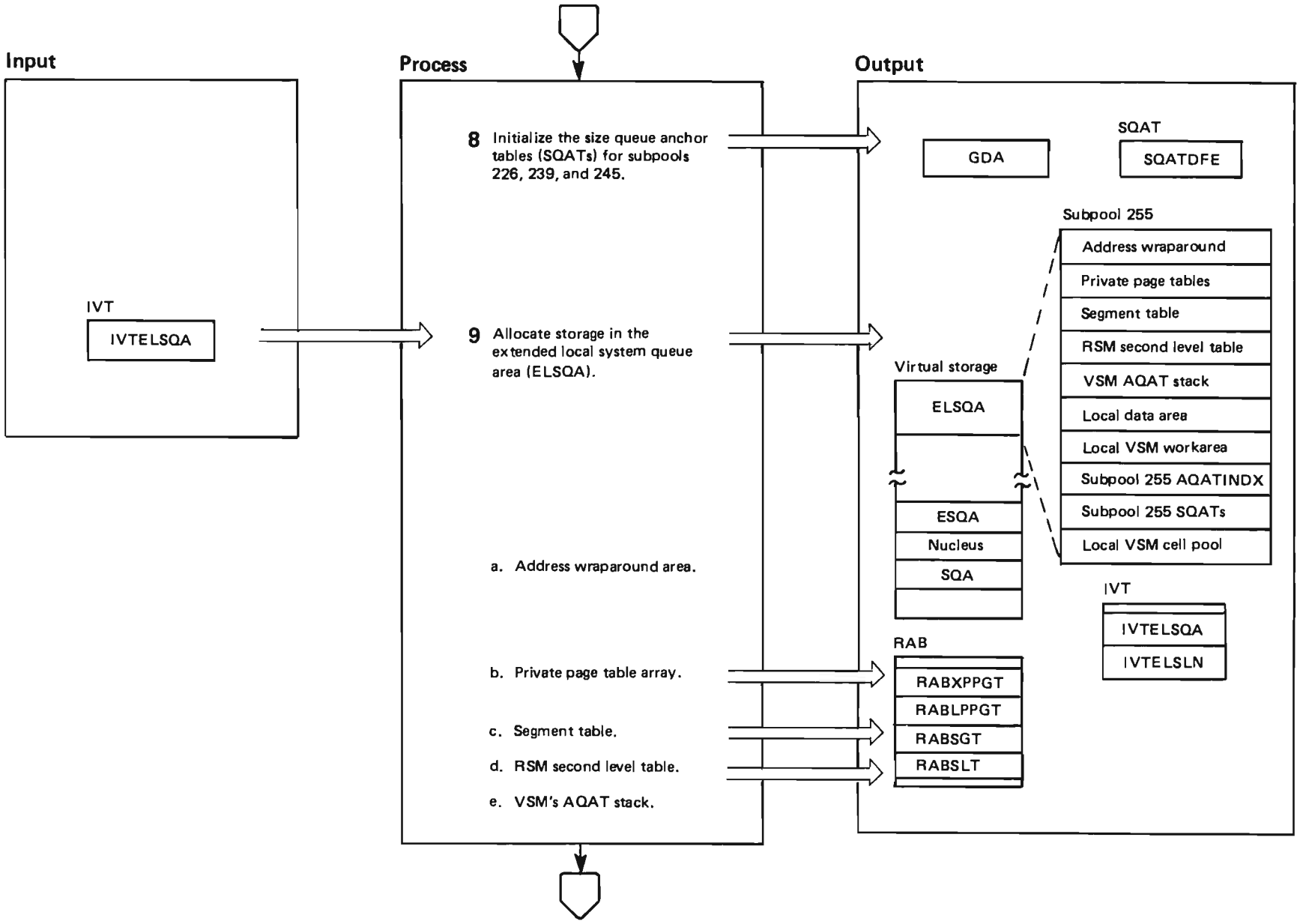


Diagram 6. Virtual Storage Management IPL Initialization (IEAIPL04) Part 8 of 12

Extended Description	Module	Label
8 The VSM IRIM initializes the size queue anchor tables (SQATs) in the read-only nucleus for the following areas: • Non-extended subpool 226 • Extended and non-extended subpools 239 • Extended and non-extended subpools 245 For a description of SQATs, see Volume 1 of *Debugging Handbook.* The IRIM builds dummy double free elements (DFEs), sets the address of the dummy DFEs in the SQAT entries, and enqueues the dummy DFEs on the DFE address and and size queues. It then initializes the following fields in the GDA:		INITSQAT
Field initialized — *Initialization Value* GDAADFxx — Address of the first DFE on the DFE address queue GDAADLxx — Address of the last DFE on the DFE address queue GDASZFxx — Address of the first DFE on the DFE size queue GDASZLxx — Address of the last DFE on the DFE size queue GDAEADFy — Addr of the first DFE on the extended address queue GDAEADLy — Address of the last DFE on the extended address queue GDAESZFy — Address of the first DFE on the extended size queue GDAESZLy — Address of the last DFE on the extended size queue GDASQAT6 — Address of the subpool 226 SQAT GDASQAT9 — Address of the subpool 239 SQAT GDAESQT9 — Address of the extended subpool 239 SQAT GDASQAT5 — Address of the subpool 245 SQAT GDAESQT5 — Address of the extended subpool 245 SQAT (xx is 45, 39 or 26 for subpools 245, 239, and 226) (y is 5 or 9 for extended subpools 245 and 239)		QDFE

Extended Description	Module	Label
9 The VSM IRIM allocates storage for the extended local system queue area (ELSQA) beginning at the two gigabyte line and assigning downward. It initializes fields in the IVT giving the size and address of the ELSQA. The VSM IRIM assigns storage for the following: a. Address wraparound area The IRIM assigns 4K bytes of storage to be unused. A program check will occur if this area is referenced. b. Private area page table area The IRIM assigns eight megabytes of storage to this area. The IRIM initializes fields in the RAB with the addresses of the extended and non-extended private area page table areas. c. Segment table The IRIM assigns two contiguous 4K frames of storage for this area. The IRIM initializes fields in the RAB with the address of the segment table. d. RSM second level table (SLT) The IRIM initializes fields in the RAB with the address of the RSM second level table. e. Reserve area for future AQATs (208K) The IRIM assign 208K of storage to this area. Because the LSQA size is dynamic, the VSM IRIM provides a stack from which VSM can draw storage for AQATs as needed.		

Diagram 6. Virtual Storage Management IPL Initialization (IEAIPL04) Part 9 of 12

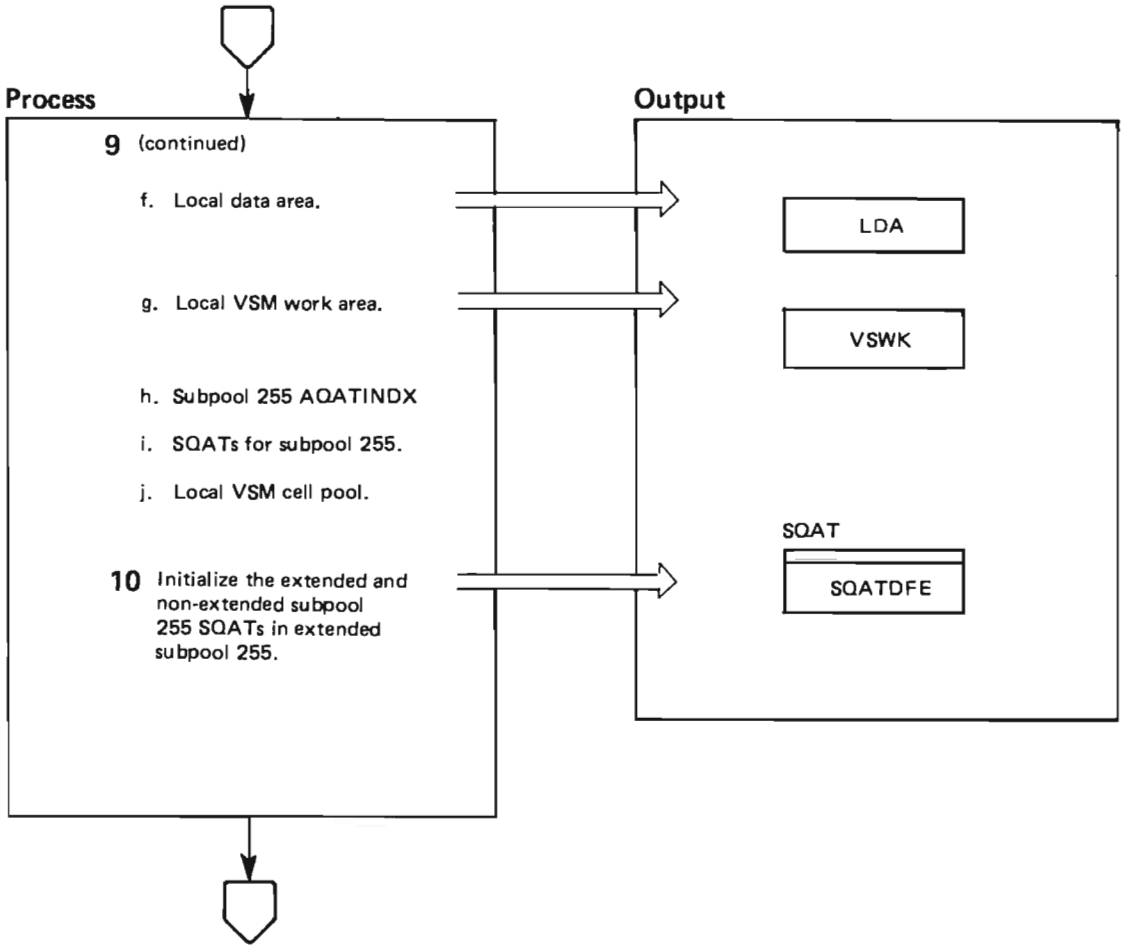


Diagram 6. Virtual Storage Management IPL Initialization (IEAIPL04) Part 10 of 12

Extended Description	Module	Label
9 (continued)		
f. Local data area		
The IRIM initializes the following fields in the LDA:		
Field initialized — *Initialization value*		
LDAID — 'LDA'		
LDAAQTST — address of the reserve area for AQATs		
LDAAQTAD — address of the reserve area for AQATs		
LDAASCB — address of the ASCB		
LDAWRKA — address of the local VSM workarea		
LDAEAQAT — address of the ELSQA AQATINDX		
LDAAQAT — address of the LSQA AQATINDX		
LDASQAT — address of the LSQA SQAT		
LDAADF — address of the first DFE on LSQA address queue		
LDAADL — address of the last DFE on LSQA address queue		
LDASZF — address of the first DFE on LSQA size queue		
LDASLZ — address of the last DFE on LSQA size queue		
LDAESQAT — address of the ELSQA SQAT		
LDAEADF — address of the first DFE on ELSQA address queue		
LDAEADL — address of the last DFE on ELSQA address queue		
LDAESZF — address of the first DFE on ELSQA size queue		
LDAESZL — address of the last DFE on ELSQA size queue		
LDASMAD — address of the storage management area		
LDASMSZ — size of the storage management area		
LDAELIM — V=V region limit for extended storage		
LDAEVVRG — V=V high amount for extended storage		
LDALIMIT — V=V region limit for storage below the line		
LDAVVRG — V=V high amount for storage below the line		
g. Local VSM work area (VSWK)		
The local VSM workarea is a permanent workarea that VSM uses when it processes storage requests for local subpools. Within the VSM workarea is a dynamic stack area from which VSM obtains storage for its own variables. The VSM IRIM initializes the following fields in the VSWK:		
Field Initialized — *Initialization value*		
VSWKID — 'VSWK'		
VSWKLDA — address of LDA		
VSWKGDA — address of GDA		
VSWKSADR — address of VSM stack area		
VSWKTOP — address of top of VSM stack area		
VSWKCELA — address of VSM cell pool head		
h. Extended subpool 255 AQATINDXs		
i. Subpool 255 SQATs		
The IRIM copies the SQATs for ESLQA and LSQA from the read-only nucleus into extended subpool 255.		
j. Local VSM cell pool		
THE IRIM calls the service routine, IGVVSFMT, to format the local VSM cell pool.		
10 The VSM IRIM initializes the extended and non-extended subpool 255 SQATs in extended subpool 255. The IRIM builds dummy DFEs, sets the address of the dummy DFEs in the SQAT, and enqueues the dummy DFEs on the DFE address and size queues.		INITSQAT QDFE

Diagram 6. Virtual Storage Management IPL Initialization (IEAIPL04) Part 11 of 12

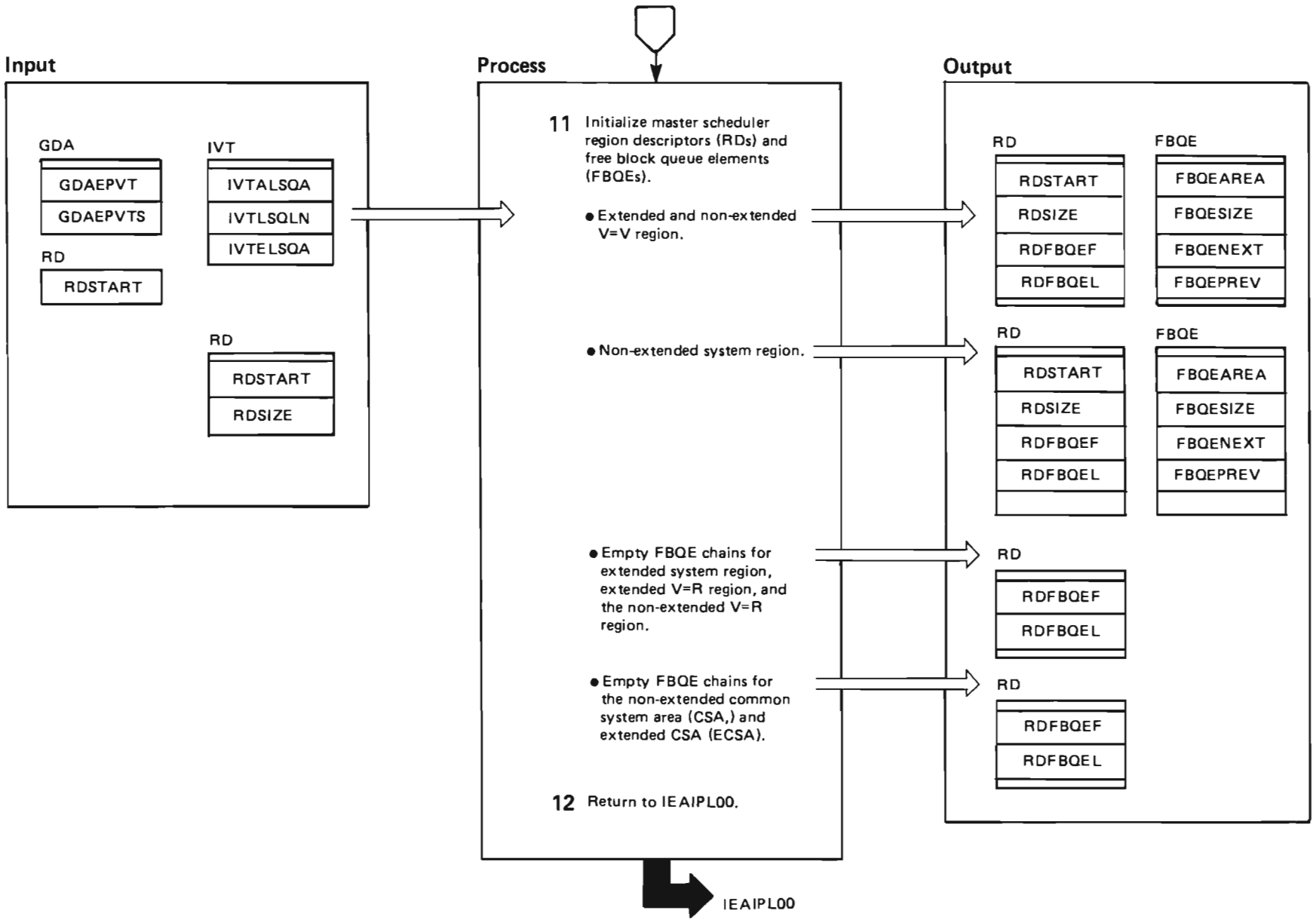


Diagram 6. Virtual Storage Management IPL Initialization (IEAIPL04) Part 12 of 12

Extended Description	Module	Label
11 The VSM IRIM initializes master scheduler region descriptors (RDs) and free block queue elements (FBQEs): • The IRIM initializes the non-extended and extended V=V region RDs. It also initializes and enqueues FBQEs that describe the unused storage in the non-extended and extended V=V region. • The IRIM initializes RDs for the non-extended system region. It also initializes and enqueues an FBQE that describes the entire non-extended system region. • The IRIM initializes RDs to describe empty FBQE chains for the extended system region, the V=R region, and the extended V=R region. (The RD start address and size fields are zero.) These regions do not exist above the 16 megabyte line. • The IRIM initializes RDs describing empty FBQE chains for non-extended and extended CSA. There is no CSA or ECSA until IEAVNPB8 processes the CSA parameter. **12** The IRIM sets a return code of zero and returns to IEAIPL00.		

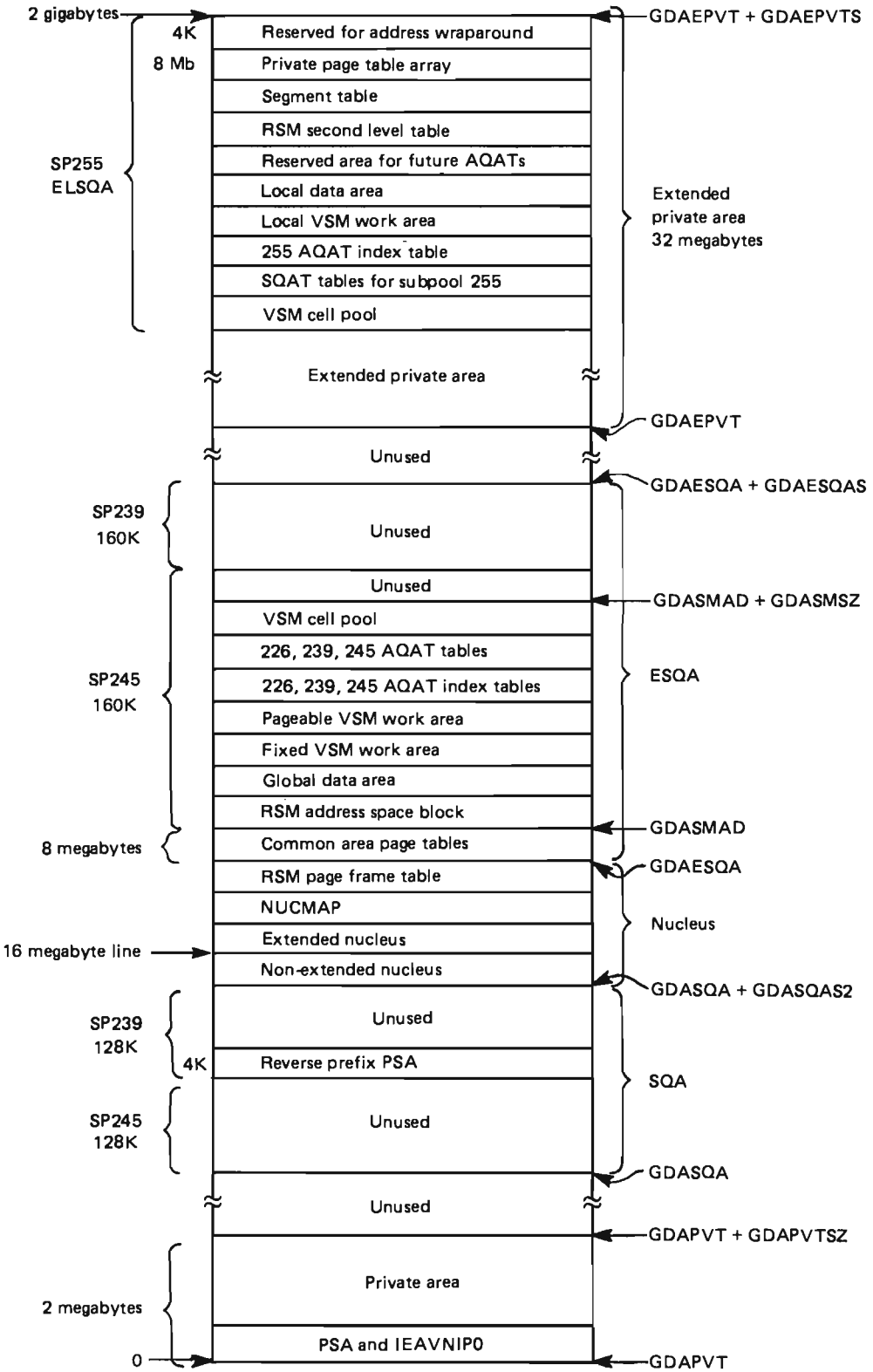


Figure 5-3. Virtual Storage on Exit from IEAIPL04

Diagram 7. Real Storage Management IPL Resource Initialization (IEAIPL06) Part 1 of 6

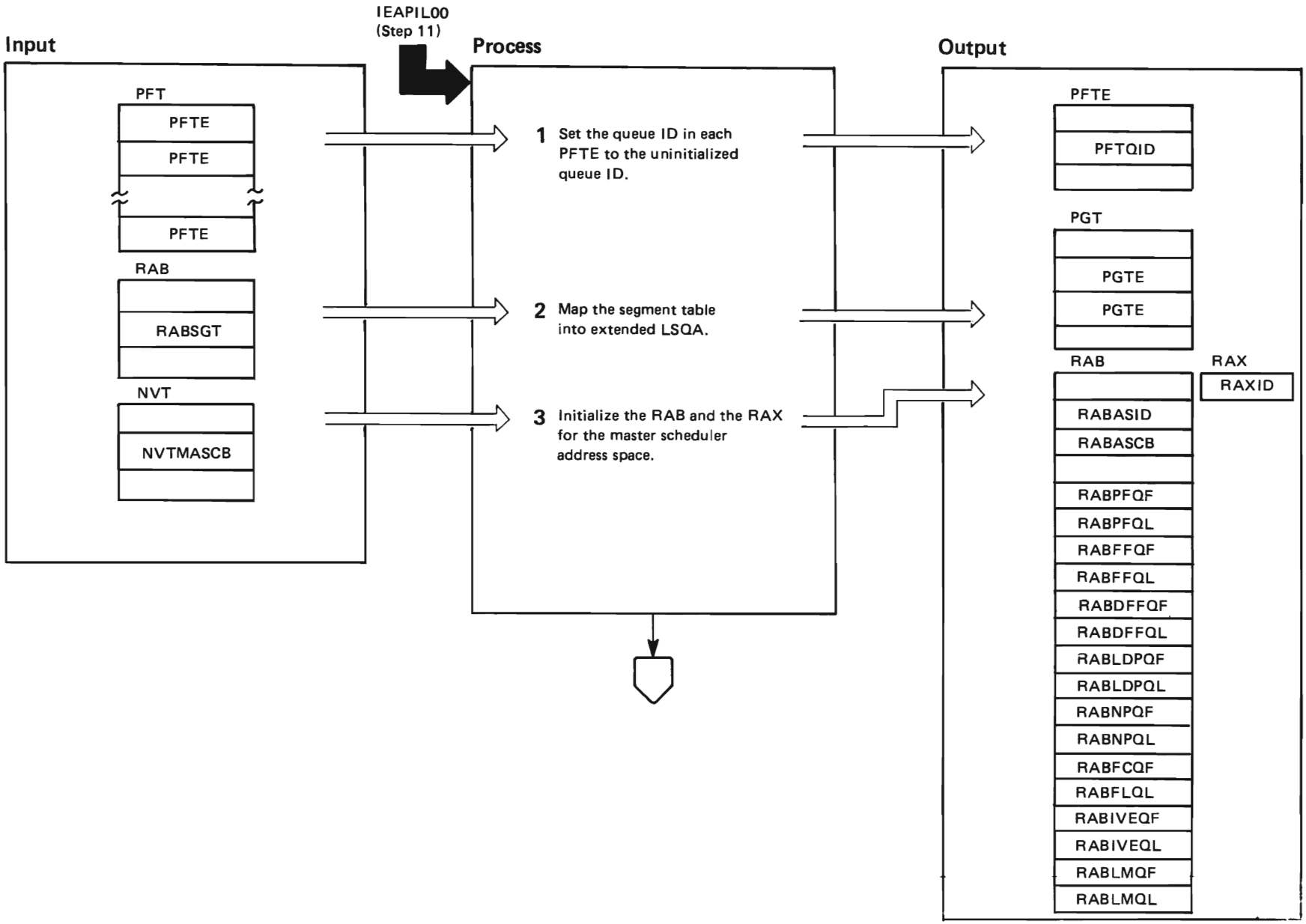


Diagram 7. Real Storage Management IPL Resource Initialization (IEAIPL06) Part 2 of 6

Extended Description	Modules	Label
The real storage manager (RSM) IRIM, IEAIPL06, initializes the page frame table. IPL has already used some frames for the nucleus, the PSA, the NIP module (IEAVNIP0), the system queue area (SQA), the extended SQA (ESQA), the extended local system queue area (ELSQA), and the hardware system area (HSA). IEAIPL06 accounts for all these frames by filling in the page frame table entries (PFTEs) to reflect the current use of the frames. IEAIPL06 also initializes fields in the RSM internal table (RIT), the RSM address space block (RAB), and the RAB extension (RAX) for the master scheduler address space.		
1 IEAIPL06 sets the queue ID in each page frame table entry (PFTQID) to the initial value X'FE' to indicate that each frame is unqueued and uninitialized.	IEAIPL06	IARMI
2 IEAIPL06 maps the valid page tables for the master scheduler address space into extended LSQA by computing the address of the page table entry for each valid segment. It puts these addresses into the page table to describe the page tables themselves and places the addresses of the frames backing the segment table into the page table to describe the segment table.		
3 IEAIPL06 initializes the RSM address space block (RAB) and the RAB extension (RAX) for the master scheduler address space. IEAIPL06: a. Places the address of the ASCB and the ASID in the RAB. b. Makes each frame queue and page control block (PCB) queue related to the RSM address space circular. IEAIPL06 makes the queues circular by placing the address of the queue trailer fields of the RAB. c. Initializes the RAX and places the address of the RAX in the master scheduler ASCB.		

Diagram 7. Real Storage Management IPL Resource Initialization (IEAIPL06) Part 3 of 6

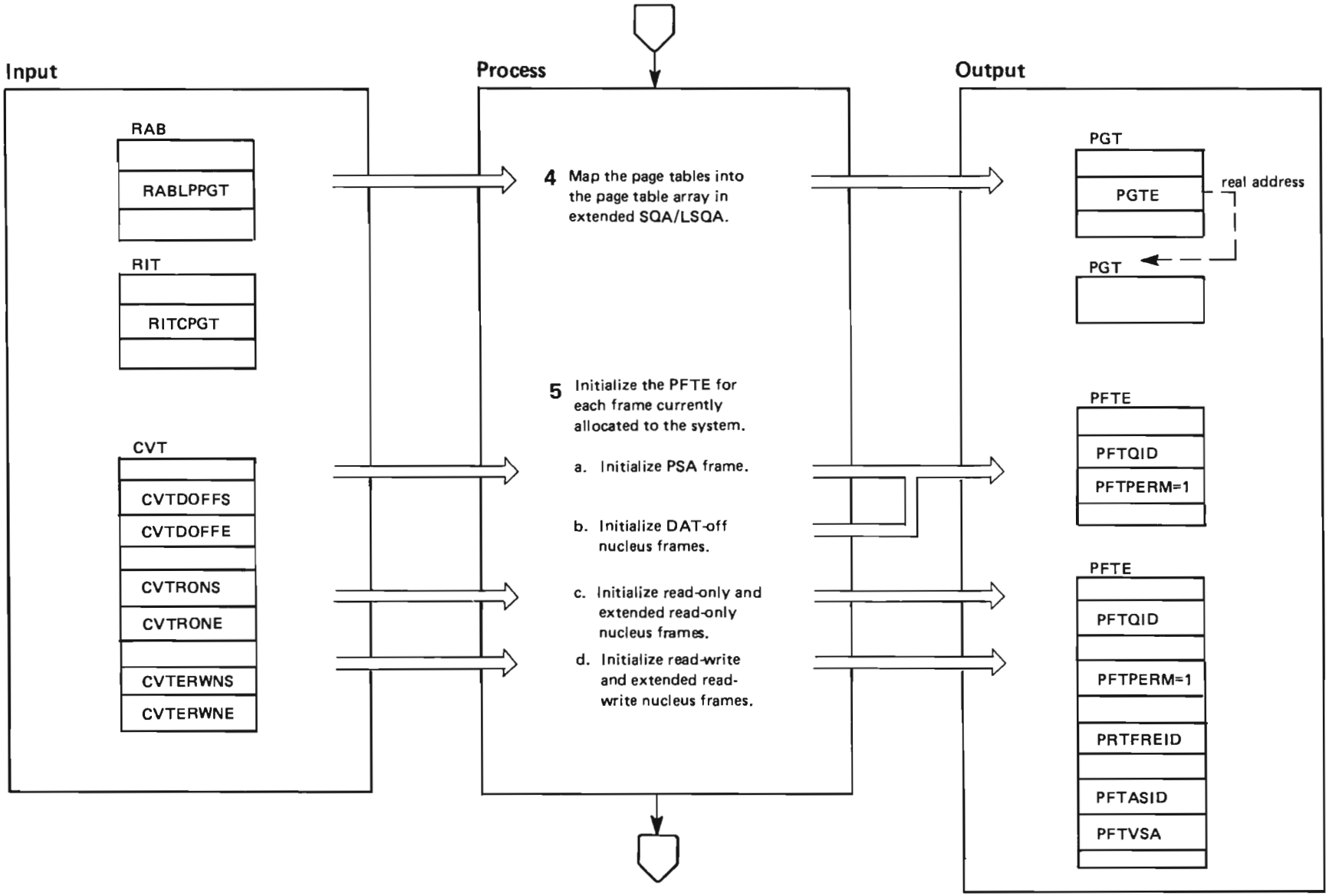


Diagram 7. Real Storage Management IPL Resource Initialization (IEAIPL06) Part 4 of 6

Extended Description	Module	Label

4 The RSM IRIM maps each page table for a valid segment (SGTINV=0) into virtual storage by placing the real address of the frame backing the page table in the PFTE for the page table page. If the segment is common storage, it locates the page table in the page table array in the extended SQA (ESQA); if the segment is private storage, it locates the page table in the master scheduler extended LSQA.

5 IEAIPL06 initializes the PFTEs for each frame currently in use as follows:

a. The PSA: IEAIPL06 marks the frame as permanently resident (PFTPERM=1), places a unique ID (X'FF') in PFTFREID to indicate that the frame is not to be freed, and sets the PSA queue ID in the PFTE (PFTQID=X'F5').

b. The DAT-off nucleus: IEAIPL06 marks each frame as permanently resident (PFTPERM=1), places a unique ID (X'FF') in PFTFREID to indicate that the frame is not to be freed, and sets the DAT-off queue ID in the PFTE (PFTQID=X'F0').

c. The read-only and extended read-only nucleus: IEAIPL06 marks each frame as permanently resident (PFTPERM=1), places a unique ID (X'FF') in PFTFREID to indicate that the frame is not to be freed, and sets the ASID value for the common area (PFTASID), the virtual address (PFTVSA), and the read-only and extended read-only nucleus queue ID in the PFTE (PFTQID=X'F1').

d. The read-write and the extended read-write nucleus frames: IEAIPL06 marks each frame as permanently resident (PFTPERM=1), sets PFTASID to the ASID value for the common area, initializes the virtual address, places a unique ID (X'FF') in PFTFREID to indicate that the frame is not to be freed, and places the read-write and extended read-write nucleus queue ID in the PFTE (PFTQID=X'F2').

Note that the page frame table and NUCMAP are part of the read-write nucleus.

Diagram 7. Real Storage Management IPL Resource Initialization (IEAIPL06) Part 5 of 6

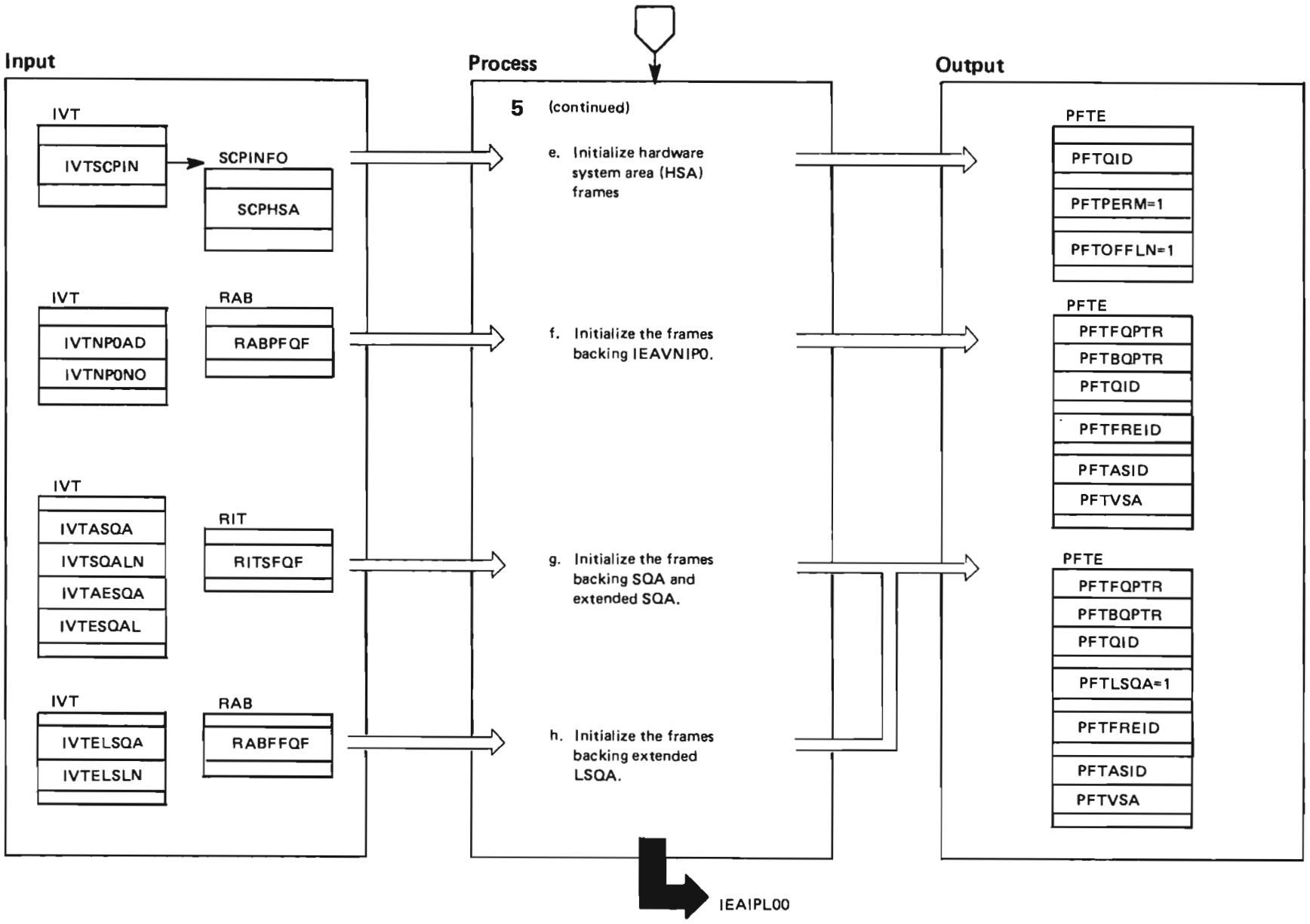


Diagram 7. Real Storage Management IPL Resource Initialization (IEAIPL06) Part 6 of 6

Extended Description	Module	Label

5 (continued)

e. The hardware system area (HSA): IEAIPL06 marks each frame as permanently resident (PFTPERM=1), offline (PFTOFFLN=1), places a unique ID (X'FF') in PFTFREID to indicate that the frame is not to be freed, and sets the HSA queue ID in the PFTE (PFTQID=X'F4').

f. The frames that back the module IEAVNIP0: IEAIPL06 sets the virtual address, PFTASID, to the ASID value for the master scheduler address space, the pageable frame queue (PFQ) ID in the PFTE (PFTQID=X'81'), and places the frames backing IEAVNIP0 on the PFQ for the master scheduler address space.

g. The SQA: IEAIPL06 sets the LSQA/SQA flag, the virtual address, the PFTASID to the ASID value for the common area, the ID of the queue to which the frame returns when it is freed, and the SQA frame queue (SFQ) ID in the PFTE (PFTQID=X'21'). IEAPIL06 places the frames backing the SQA on the SFQ.

h. The extended LSQA for the master scheduler address space: IEAIPL06 sets the LSQA/SQA flag, the virtual address, the PFTASID to the ASID value for the master scheduler address space, the ID of the queue to which the frame returns when it is freed, and the SFQ ID in the PFTE (PFTQID=X'82'). IEAIPL06 places the frames backing the extended LSQA on the fixed frame queue for the master scheduler eddress space.

Diagram 8. IPL Cleanup IRIM (IEAIPL99) Part 1 of 6

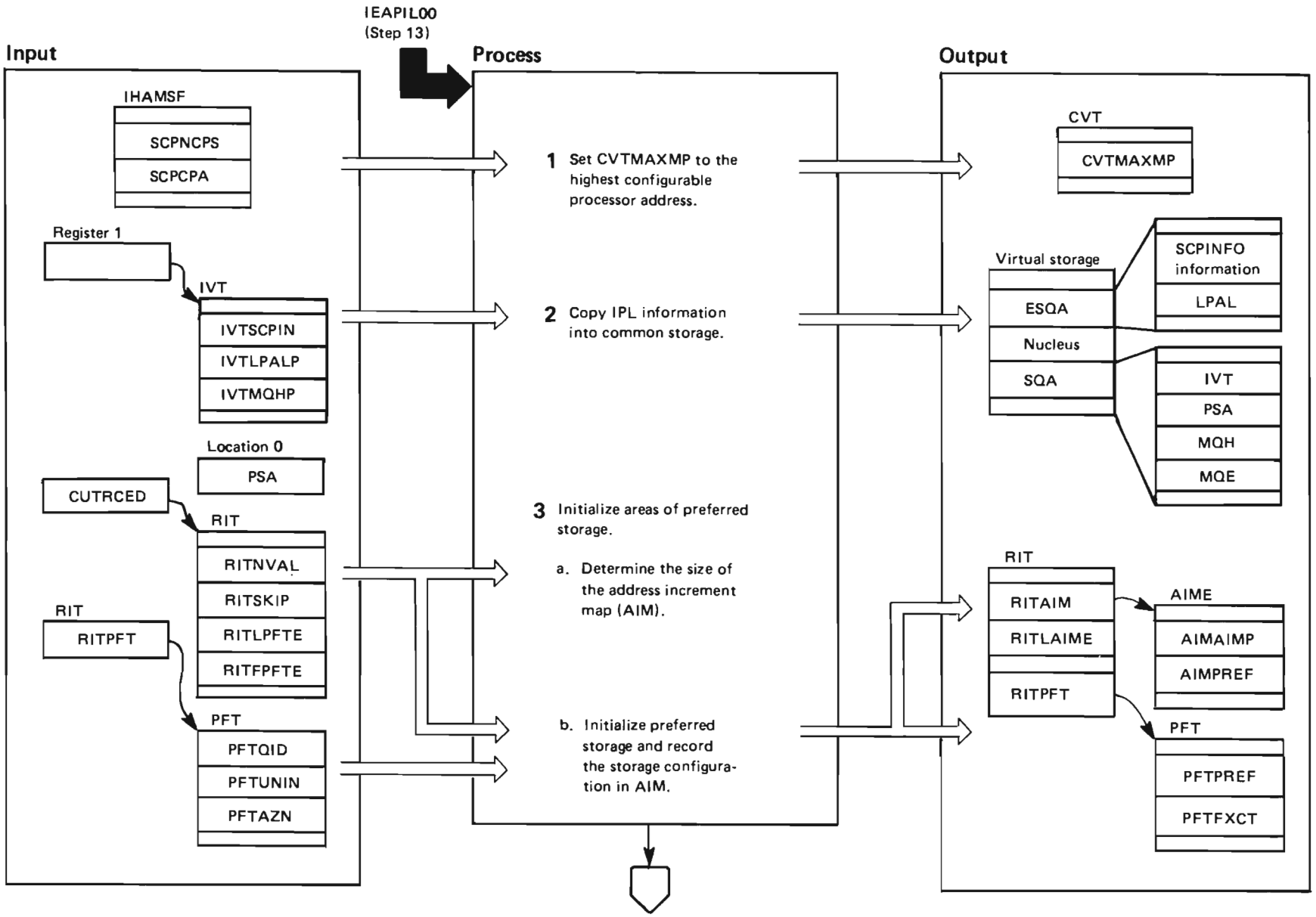


Diagram 8. IPL Cleanup IRIM (IEAIPL99) Part 2 of 6

Extended Description	Module	Label

To prepare to pass control to NIP, IEAIPL99 performs a series of miscellaneous tasks. It places IPL information, LPA device support module list and IPL message queue into common storage so that NIP can access it. It initializes storage for the RSM and establishes the available frame queues. It establishes addressability for NIP and removes the IPL modules and workarea from storage. For a map of virtual storage exit from IEAIPL99.

1 IEAIPL99, scans through the service processor SCPINFO configurable processor data to find the highest configurable processor address (SCPCPA) and places this address in CVTMAXMP.

2 IEAIPL99 allocates storage in common areas and copies the following IPL information so that NIP can access it:

- Service processor SCPINFO information is copied into the extended system queue area (ESQA) (subpool 245).
- The PSA that was initialized at system generation (the sysgen PSA) is copied into the non-extended SQA (subpool 245).
- The IPL vector table (IVT) is copied into the non-extended SQA (subpool 245).
- The device support module list (LPAL) is copied into ESQA.
- The message queue header (MQH) and message queue element (MQE) are copied into SQA.

Extended Description	Module	Label

3 IEAIPL99 establishes the address ranges of preferred storage (storage that is set aside for long-term use). It allocates and initializes the RSM address increment map (AIM). The AIM indicates which address increments of real storage contain only non-preferred frames and which ones contain a combination of preferred and non-preferred frames. There is one AIM entry (AIME) for each address increment; an AIME consists of a header record and an entry for each band in the address increment.

a. IEAIPL99 uses the service processor SCPINFO information in the RSM internal table (RIT) to calculate how much storage is needed for the AIM. It obtains storage in the extended SQA (subpool 245) for the AIM.

b. IEAIPL99 scans the page frame table entries (PFTEs) for frames that are currently assigned. At this time in the initialization process, all frames that are currently assigned are considered preferred (long time use). IEAIPL99 initializes storage as follows:

   - If any frame in an address increment is currently used (PFTQID¬=PFTUNIN) but is not the PSA (PFTQID=PFTAZN), IEAIPL99 marks the AIME to indicate that some storage in that address increment is preferred (AIMAINP=0).
   - When an address increment contains some preferred frames (AIMAINP=0), IEAIPL99 creates an initial preferred storage area by marking all frames in that address increment preferred (PFTPREF=1). It then sets a field (AIMPREF=1) to indicate that some frames within the band are preferred.

Diagram 8. IPL Cleanup IRIM (IEAIPL99) Part 3 of 6

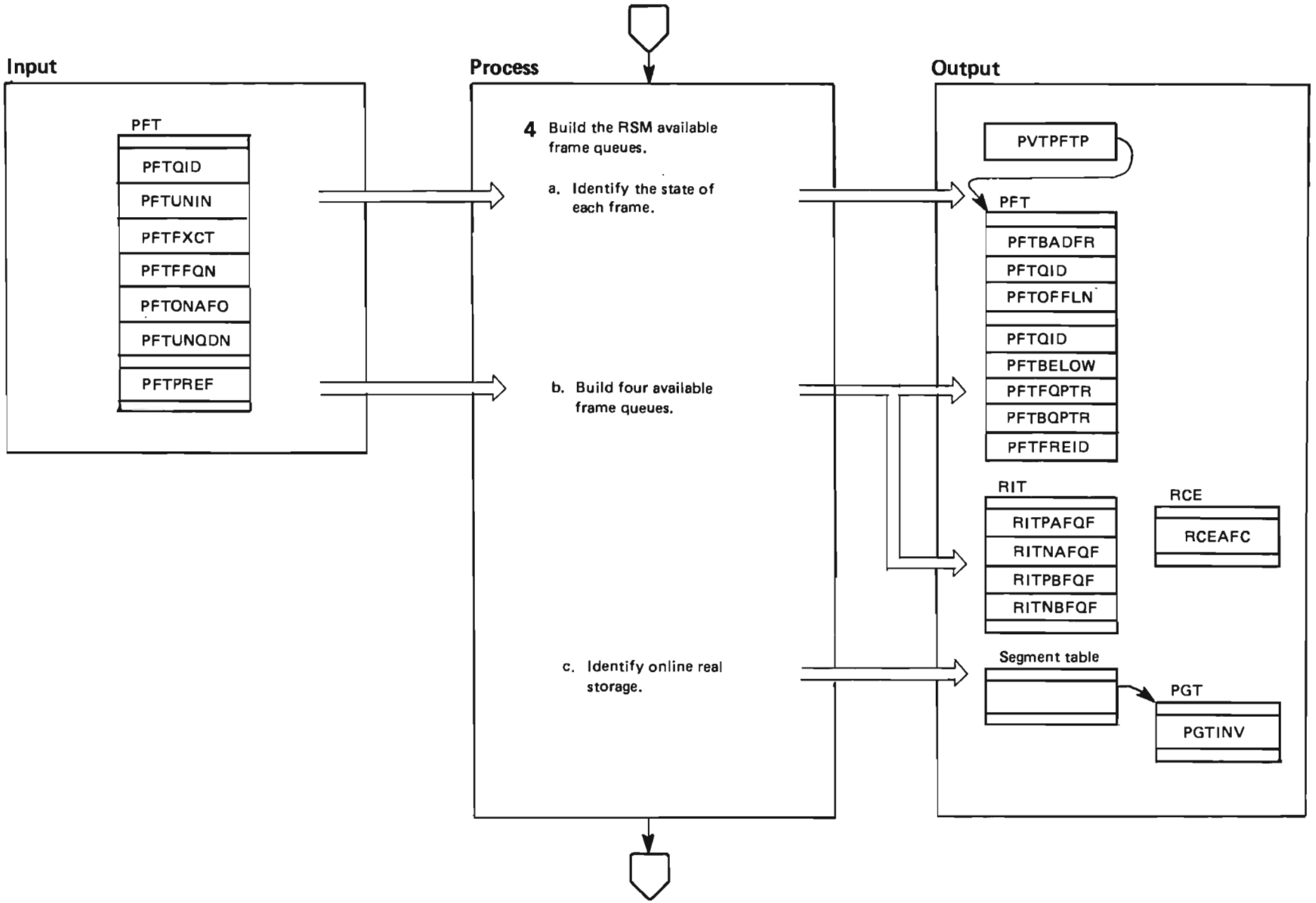


Diagram 8. IPL Cleanup IRIM (IEAIPL99) Part 4 of 6

Extended Description	Module	Label

4 Having accounted for all preferred storage, IEAIPL99 builds the available frame queues for RSM. It begins by marking available both the frame that contains IEAIPL99 and the frame that holds IEAIPL99's page table. It dequeues the frame for the module's page table from the fixed frame queue and marks invalid the corresponding PGTE (PGTINV) and decreases the fix count in the PFTE (PFTFXCT).

a. IEAIPL99 uses the test block (TB) instruction to retest all uninitialized frames. The TB instruction returns a condition code that identifies the state of the frame, and IEAIPL99 sets fields accordingly. The following table shows the condition code, the state of the frame that the code represents, and the field IEAIPL99 initializes to indicate the state of the frame:

Condition Code	*State of the Frame*	*Field Set*
0	good	none
1	defective	PFTBADFR PFTOFFLN PFTQID PFTFREID PFTBELOW
3 (see note)	offline	PFTBADFR PFTOFFLN PFTQID PFTFREID PFTBELOW

Note: Condition code 3 is not actually set by the TB instruction. IEAIPL00 similates this condition code when addressing exception occurs on a TB instruction.

Extended Description	Module	Label

b. IEAIPL99 scans again through the PFTEs to locate available frames. An available frame is an uninitialized good frame. For each available frame, it:

- Checks the address of the frame associated with the PFTE to determine whether the frame resides above or below the 16 megabyte line
- Checks the PFTE to determine if the frame is preferred or non-preferred

Depending on the results of these checks, IEAIPL99 builds four RSM available frame queues by placing the frame on the appropriate queue. The four available frame queues are:

preferred, above the 16Mb line
preferred, below the 16Mb line
non-preferred, above the 16Mb line
non-preferred, below the 16Mb line

IEAIPL99 sets pointers in the RIT to the available frame queue headers. It also sets the field RCEAFC to indicate the count of the total available frames.

IEAIPL99 sets a field in the PFT (PFTFREID) to indicate which available frame queue the frame should return to after it is freed. This queue should be the same as the queue to which the frame is currently assigned.

c. If any frame in a band within an address increment is marked online (PFTOFFLN=0), IEAIPL99 marks that band to indicate that some storage in that band is online (AIMOFF=0).

Diagram 8. IPL Cleanup IRIM (IEAIPL99) Part 5 of 6

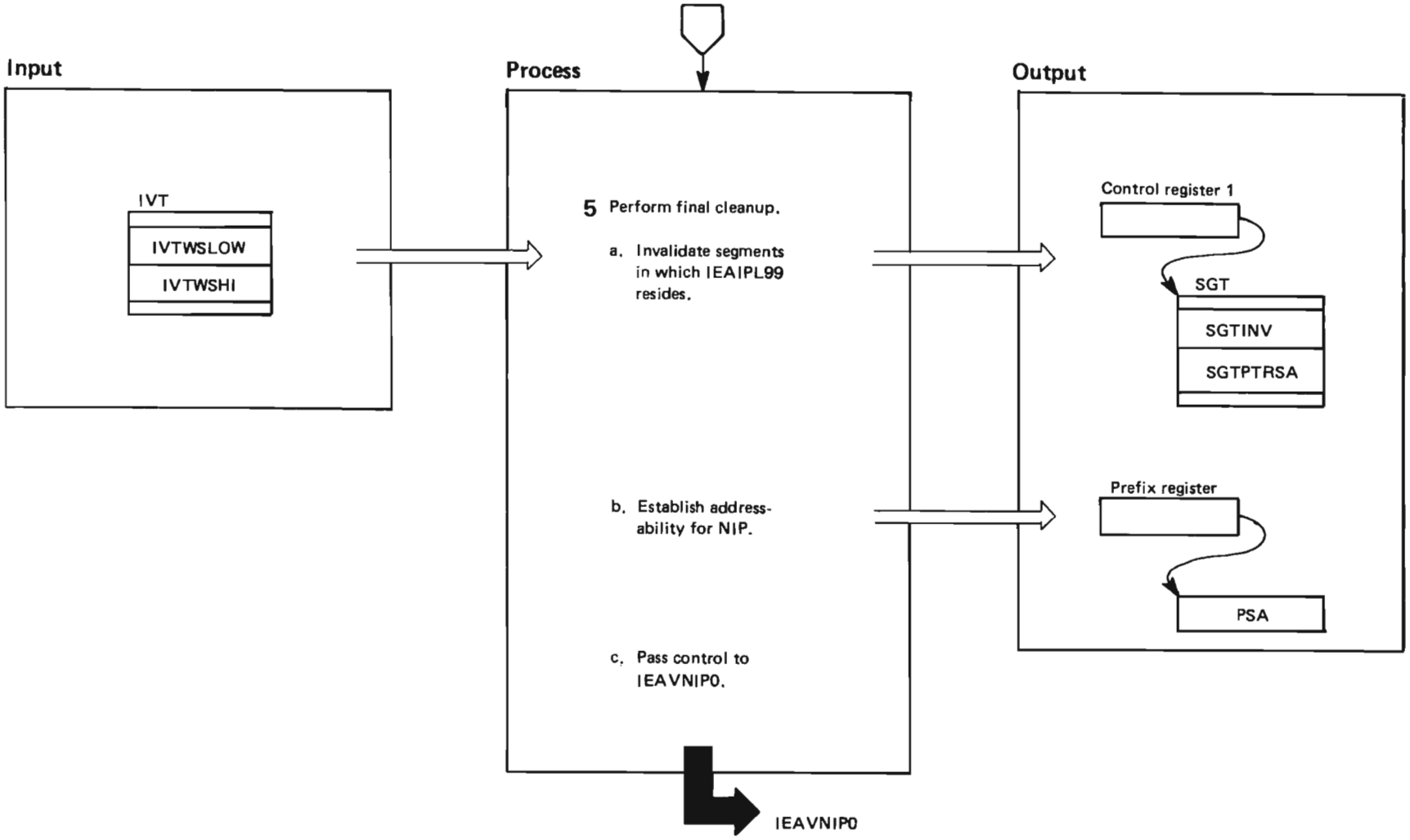


Diagram 8. IPL Cleanup IRIM (IEAIPL99) Part 6 of 6

Extended Description	Module	Label
5 IEAIPL99 performs the final cleanup needed before control passes to the second phase of the system initialization process.		
a. Because the IPL workspace is no longer useful, IEAIPL99 invalidates the segment table entries that map this space.		
b. IEAIPL99 sets the page table entry for virtual zero to real zero and the prefix register to point to the PSA that was initialized at system generation. This action allows the interrupt handlers to get control when interrupts occur.		
c. IEAIPL99 issues a load PSW instruction, which passes control to IEAVNIP0.		

Diagram 9. IPL/NIP Interface Routine (IEAVNIP0) Part 1 of 16

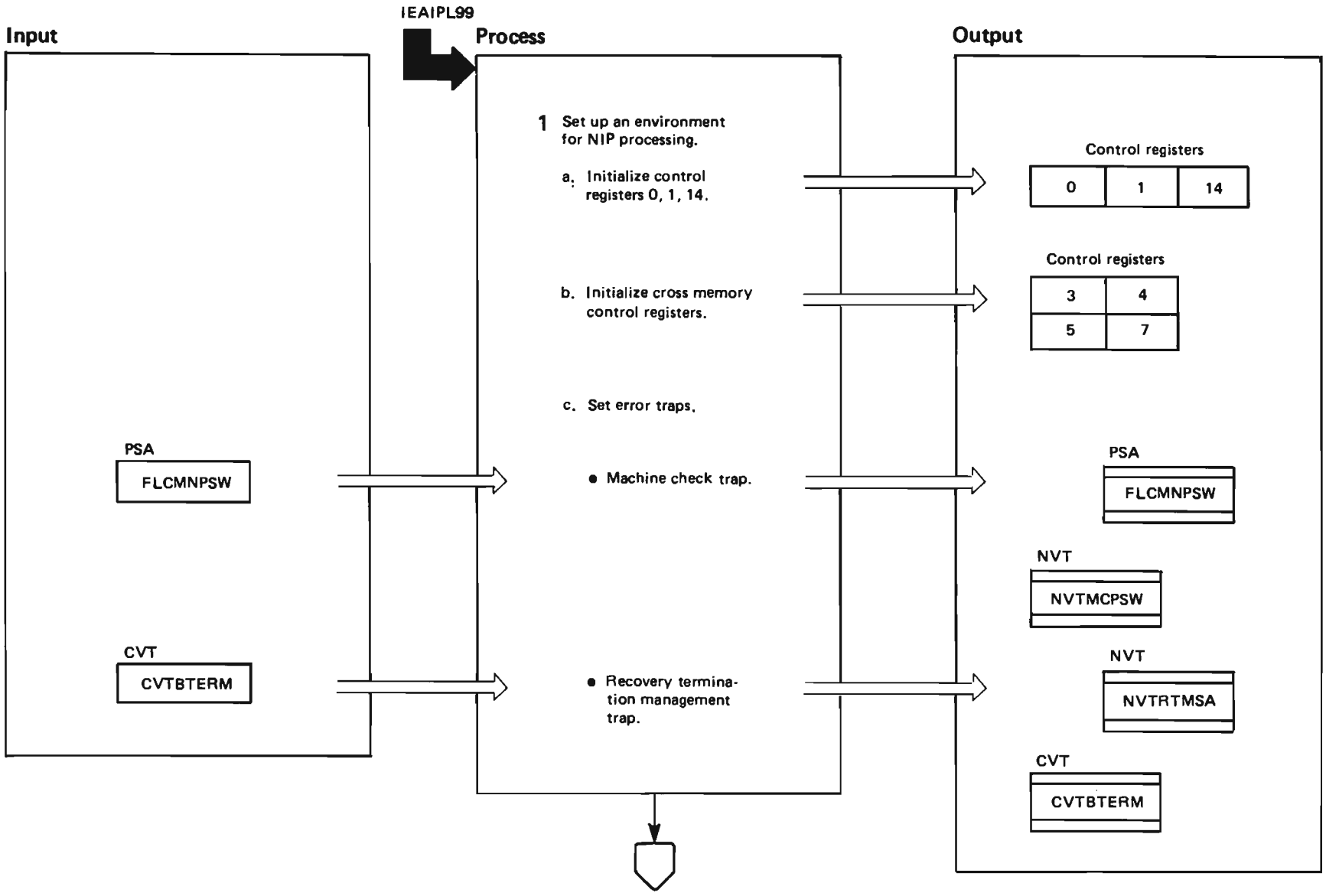


Diagram 9. IPL/NIP Interface Routine (IEAVNIP0) Part 2 of 16

Extended Description	Module	Label
IEAVNIP0 is the first nucleus initialization program (NIP) module to receive control. It allocates and initializes global system control blocks and the processor-related control blocks for the IPL processor. It then prepares an environment in the master scheduler address space that will allow the resource initialization modules (RIMs) to perform their functions. For a map of virtual storage at exit from IEAVNIP0,		
1 NIP sets up an environment for the second phase of the initialization process.	IEAVNIP0	NPOHKEEP
a. NIP initializes control registers 0, 1, and 14 with the values indicated in Figure 5-4.		
b. NIP issues the SETLOCK macro instruction to obtain the CPU lock. The processor is disabled for I/O and external interrupts until the input/output supervisor (IOS) storage manager is initialized in step 4. NIP then calls the CMSET service to initialize the cross memory functions in control registers 3, 4, 5, and 7. See Figure 5-4 for the contents of the control registers. Cross memory initialization must be done prior to any dispatcher entries or interrupt handler entries. For more information on the CMSET service, see *System Logic Library*.		
c. Before system recovery routines are available, NIP provides diagnostic support for hardware and software errors by trapping the errors. The trap routines in this module detect errors that occur during the execution of this module, stop the initialization process at the time the error occurs, and provide diagnostic information in the form of wait state codes and messages. NIP traps the following errors:		NPOTRAPS
• Machine checks: NIP sets a trap for machine checks and enables for them. NIP saves the sysgen machine check new PSW in NVTMCPSW and places in FLCMNPSW the PSW that will put the system in a disabled wait state with a wait state code of X'44'. The logical address of the IPL processor is placed in bits 32-47 of the wait PSW. Note that IEAVNP06 will restore the sysgen PSW.		
• Recovery termination manager (RTM) checks: NIP traps requests for abnormal termination that occur during its processing. It saves the entry point address of the ABTERM routine found in CVTBERM in NVTRTMSA and puts the address of the IEAVNIP0 table of RTM traps into CVTBTERM.		

Diagram 9. IPL/NIP Interface Routine (IEAVNIP0) Part 3 of 16

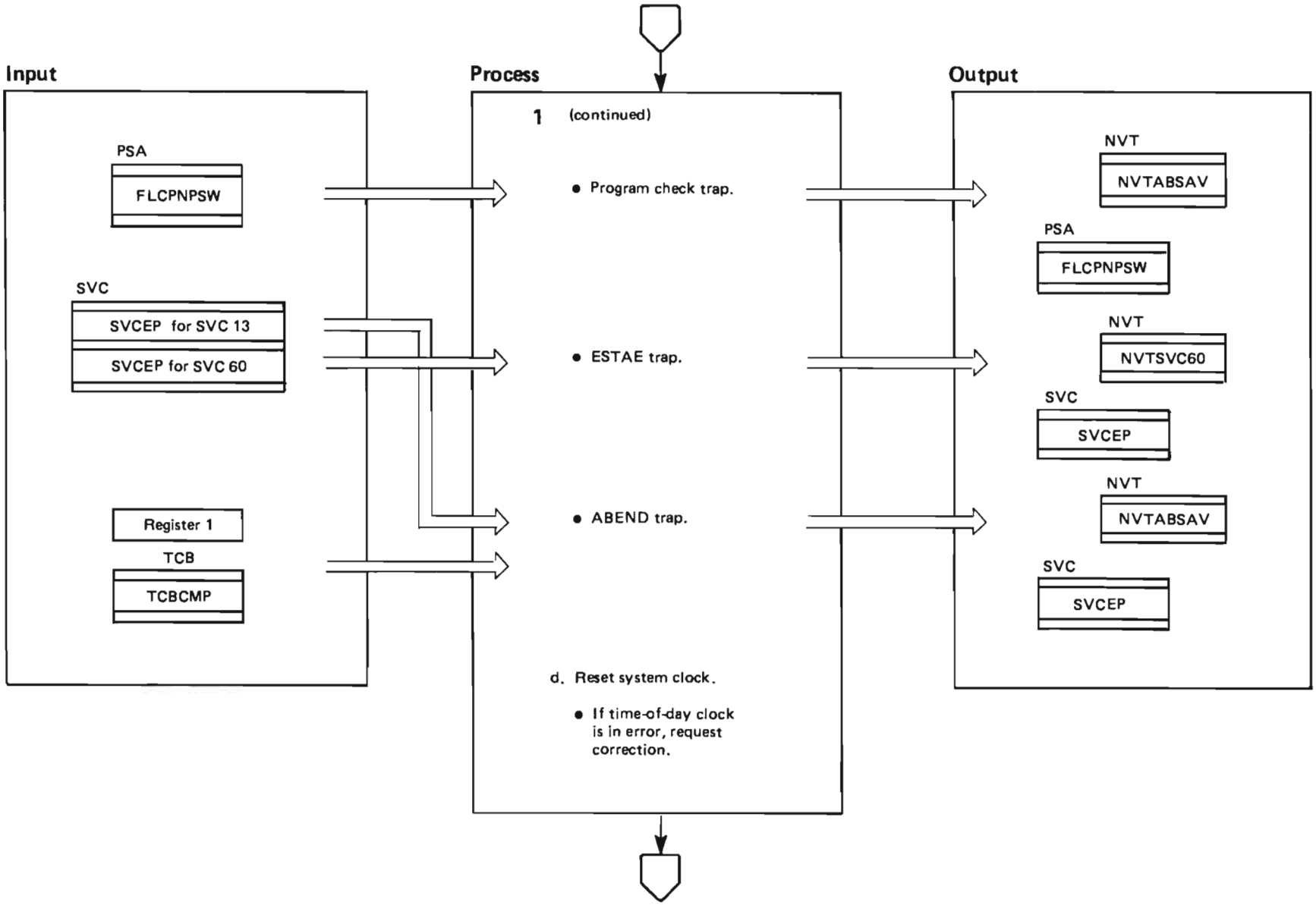


Diagram 9. IPL/NIP Interface Routine (IEAVNIP0) Part 4 of 16

Extended Description	Module	Label
1 (continued) • Program checks: NIP traps program checks. During normal operation, the program check FLIH tries to invoke SVC 13 (ABEND) to abend the offending task (during NIP, that task would be the master scheduler). Until SVC 13 is initialized, the sysgen program check new PSW has the wait bit on. Thus, if a program check occurs before IEAVNIP0 sets up this trap, NIP places the system in a wait state. NIP saves the sysgen program check new PSW and replaces it with a PSW that points to a BR 10 instruction. Register 10 contains the address of a subroutine that puts the system in a wait state. The subroutine stores the wait state code of X'46' in the address field (bits 56-63) of the current PSW. The current PSW can be displayed at the operator's console. • ESTAE: NIP traps requests for STAE or ESTAE (SVC 60) by saving the SVCTABLE entry for SVC 60 and placing in SVCEP an entry pointing to a NIP subroutine that clears register 15 to zero and returns to the caller. • ABEND: NIP traps request for ABEND (SVC 13) by moving the SVC entry in the SVC table to NVTABSAV and placing in SVCEP an entry pointing to a routine that handles ABEND requests. This routine stores the wait state code of X'30' into bits 56-63 of the PSW and a completion code, if any, into bits 36-47 of the PSW. If the TCB ABTERM bit flag (TCBABTRM) is set to 1, NIP obtains the completion code from TCBMCP. Otherwise, NIP obtains the completion code from register 1. For more information on IEAVNIP0 diagnostic trap routines, see Section 3: Diagnostic Aids.		
d. NIP issues the store clock (STCK) instruction to determine whether the time-of-day (TOD) clock is in error. If the clock is in error, NIP places the system into a restartable wait state with a wait state code X'04A'. In response to the wait state code, the operator must indicate a restart. Then NIP initializes the restart new PSW to point to a set clock (STCK) instruction. The STCK instruction sets the TOD clock to zero, and clears the error state. (If the clock remained in an error state, a set CPU timer instruction would cause a machine check and a disabled wait state.)		NPOICLKS SETTODCK
NIP sets the CPU timer and the clock comparator to maximum values so that no timer interrupts will occur when such interrupts are enabled.		SETTIME

Diagram 9. IPL/NIP Interface Routine (IEAVNIP0) Part 5 of 16

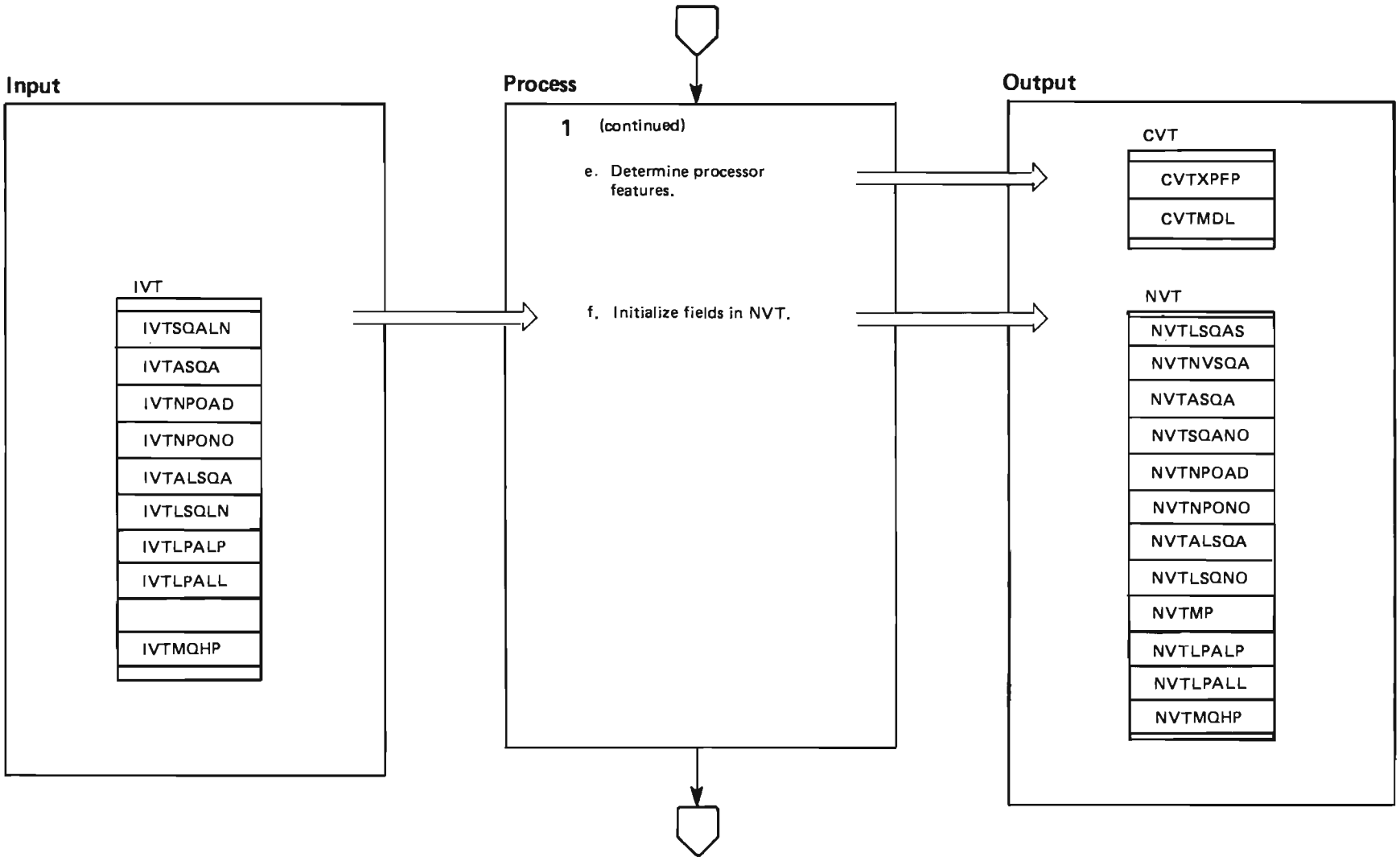


Diagram 9. IPL/NIP Interface Routine (IEAVNIP0) Part 6 of 16

Extended Description	Module	Label
1 (continued)		
e. NIP saves in the CVTMDL the processor model number that it obtains from the successful execution of the STIDP instruction. NIP must determine what extended precision floating point (EPFP) capabilities, if any, exist and place the appropriate indicator in the CVT. NIP executes the load rounded, long-to-short instruction (LRER). If a program interruption occurs, NIP assumes that the processor does not support EPFP. If no program interruption occurs, EPFP hardware is present. NIP sets the EPFP bit (CVTXPFP) on in the CVT.		
f. NIP initializes the nucleus vector table (NVT), which is the basic control block for communications and processing during NIP. NIP copies into fields in the NVT the addresses of required areas that IPL created and stored in the IPL vector table (IVT).		NPOKEEP

The fields and their values (addresses are virtual unless otherwise indicated) are:

Field initialized	*Initialization value*
NVTNVSQA	number of virtual segments of SQA
NVTASQA	low address of non-extended SQA
NVTSQANO	number of SQA pages
NVTNPOAD	address of IEAVNIP0
NVTNPONO	length of IEAVNIP0
NVTALSQA	low address of master scheduler LSQA
NVTLSQNO	number of LSQA pages to fix
NVTLSQAS	end address of master scheduler LSQA
NVTMP	'1' – indicating that this is a multiprocessing system
NVTLPALP	address of device support module list
NVTPALL	length of device support module list
NVTMQHP	address of message queue

The IVT is no longer needed; the storage it occupies is freed in step 8.

Diagram 9. IPL/NIP Interface Routine (IEAVNIP0) Part 7 of 16

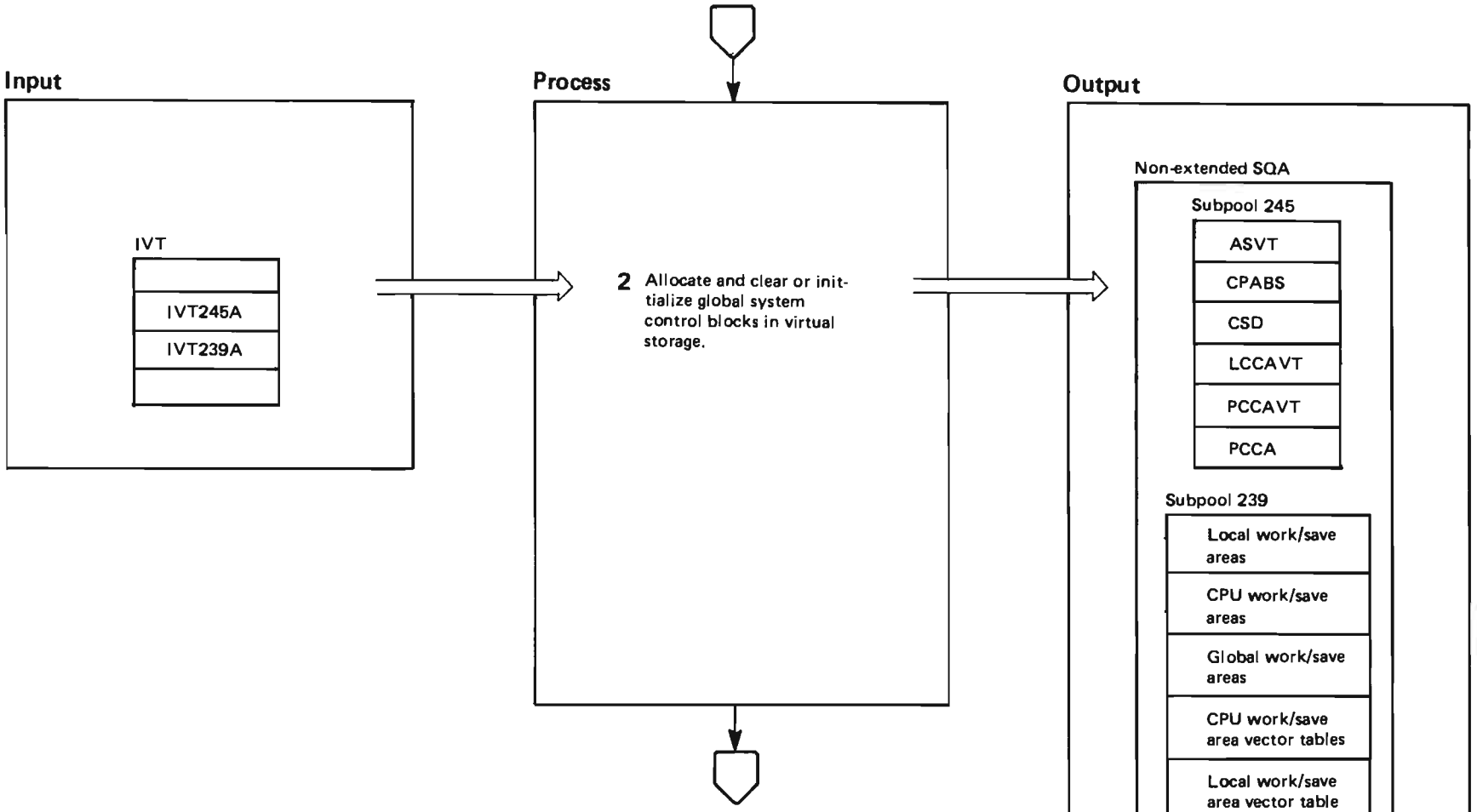


Diagram 9. IPL/NIP Interface Routine (IEAVNIP0) Part 8 of 16

Extended Description	Module	Label
2 The non-extended system queue area (SQA) contains control blocks and queues that relate to the entire system. IEAVNIP0 allocates and clears or initializes the global system control blocks in virtual storage and initializes pointers to these control blocks. For ease of reference, the pointers to these areas are grouped with the areas. NIP allocates, clears and initializes the following areas in subpool 245 below 16M: • Physical configuration communication area (PCCA) *Field initialized* — *Initialization value* PCCAPCCA — 'PCCA' PCCACPUA — Physical address of IPL processor PCCAPSAV — Address of reverse prefix PSA in SQA PCCAPSAR — Real address of reverse prefix PSA in SQA PCCACPID — IPL processor ID (in EBCDIC characters) PCCAISCE — Indication of which interrupt subclasses on the IPL processor are enabled PCCACAFM — Mask corresponding to IPL processor PCCACRG6 — Contents of control register 6 • PCCA vector table (PCCAVT) *Field initialized* — *Initialization value* CVTPCCAT — address of PCCAVT PCCAT00P — address of PCCA for IPL processor • Logical configuration communication area vector table (LCCAVT) CVTLCCAT contains the address of the LCCAVT.		NPOVSI
• Common system data (CSD) area *Field initialized* — *Initialization value* CSDCSD — 'CSD' CSDMP — '1' – indicating a multiprocessing system CSDCPUJS — mask of processors available for job scheduling CSDCPUAL — mask of processors currently online CSDCPUOL — '1' – the count of online processors CSDMASK — CSD table of processor masks CVTSAFF — affinity masks for online processors CVTCSD — address of CSD CVTCSDRL — real address of CSD NIP puts a mask corresponding to the IPL processor into field PCCACAFM of its PCCA. It also puts a mask and a count into the CSD area. The fields in the CSD will eventually contain a cumulative mask (CSDCPUAL) and count (CSDCPUOL) of all online processors. At this time, these fields reflect only the IPL processor so that the memory switch routine does not dispatch work to the non-IPL processor. • Cell pool anchor blocks (CPABs) Each CPAB contains a pre-allocated pool of cells. IEAVNIP0 initially allocates seven CPABs. *Field initialized* — *Initialization value* CPABCPID — CPID for this pool GDACPAB — address of first CPAB		

Diagram 9. IPL/NIP Interface Routine (IEAVNIP0) Part 9 of 16

No diagram
Extended Description continued on next page.

Diagram 9. IPL/NIP Interface Routine (IEAVNIP0) Part 10 of 16

Extended Description	Module	Label

- Address space vector table (ASVT)

Field initialized	*Initialization value*
ASVTASVT	'ASVT'
ASVTMAXU	Maximum users count in ASVT
ASVTAVAI	'1' – indicating that ASVTFRST is available
ASVTENTY	address of master scheduler's ASCB in first entry
CVTASVT	address of ASVT
CVTSHRVM	address of beginning of CSA

NIP allocates and then clears or initializes the following areas in subpool 239 below 16M:

- Logical configuration communication area (LCCA)

Field initialized	*Initialization value*
LCCALCCA	'LCCA'
LCCACPUA	logical address of the IPL processor
LCCAMCR1	value of control register 1
LCCACAFM	mask for the IPL processor
LCCATOOP	logical address of the IPL processor

- Functional recovery routine (FRR) stack area initialized by system routine, IEAVTRIN.

  PSACSTK contains the address of the FRR stacks.

- CPU and local work/save area vector tables

  ASXBSPSA points to the local work/save area vector table

  LCCACPUS points to the CPU work/save area vector table

- Global work/save areas

Field initialized	*Initialization value*
CVTWSAG	address of WSAG template
CVTSPSA	address of WSAG vector table

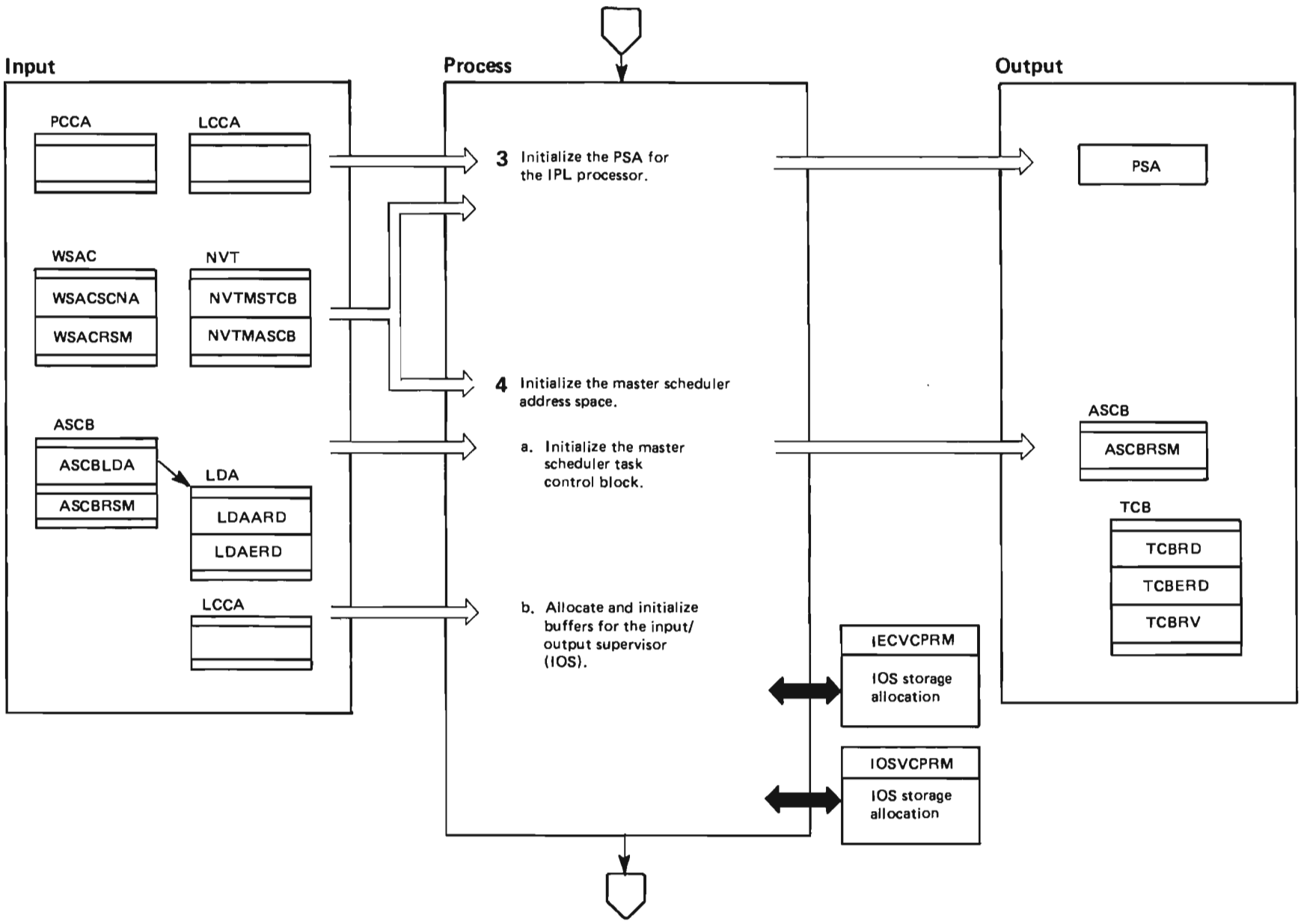
- Local work/save areas

Field initialized	*Initialization value*
CVTWSAL	address if WSAL template
ASXBSPSA	address of CPU work/save area vector table

- CPU work/save areas

Field initialized	*Initialization value*
CVTWSAC	address if CPU work/save area
LCCACPUS	address of WSAL vector table

Diagram 9. IPL/NIP Interface Routine (IEAVNIP0) Part 11 of 16



Extended Description	Module	Label
3 NIP initializes the prefix save area (PSA), beginning at location 0. It initializes the PSA by copying addresses from fields in the PCCA, LCCA, NVT, and WSAC and by setting pointers to control blocks for the IPL processor. The fields initialized and their values are:		NPOICPI

Field Initialized	*Initialization value*
PSAPSA	'PSA'
PSASTOR	value in control register1
PSACPUPA	physical processor address of IPL processor
PSACPUSA	static physical processor address of IPL processor
PSACPULA, PSALCPUA	logical processor address of IPL processor
PSAPCCAV	real address of PCCA
PSALCCAR	address of PCCA
PSALCCAV	real address of LCCA
PSATNEW, PSATOLD	address of master scheduler's TCB
PSAANEW, PSAAOLD	address of master scheduler's ASCB
PSACROSV	contents of control register 0
PSASCWA	address of supervisor control work/save area
PSARSMSA	address of RSM work save area
FLCPNPSW	sysgen program check PSW

NIP restores the sysgen program check PSW with the wait bit off. The ABEND routine is not able to handle program checks.

Extended Description	Module	Label
4 NIP initializes the master scheduler address space.		
a. NIP initializes the master scheduler's TCB with information found in the NVT, LDA, and ASCB:		NPOMSI

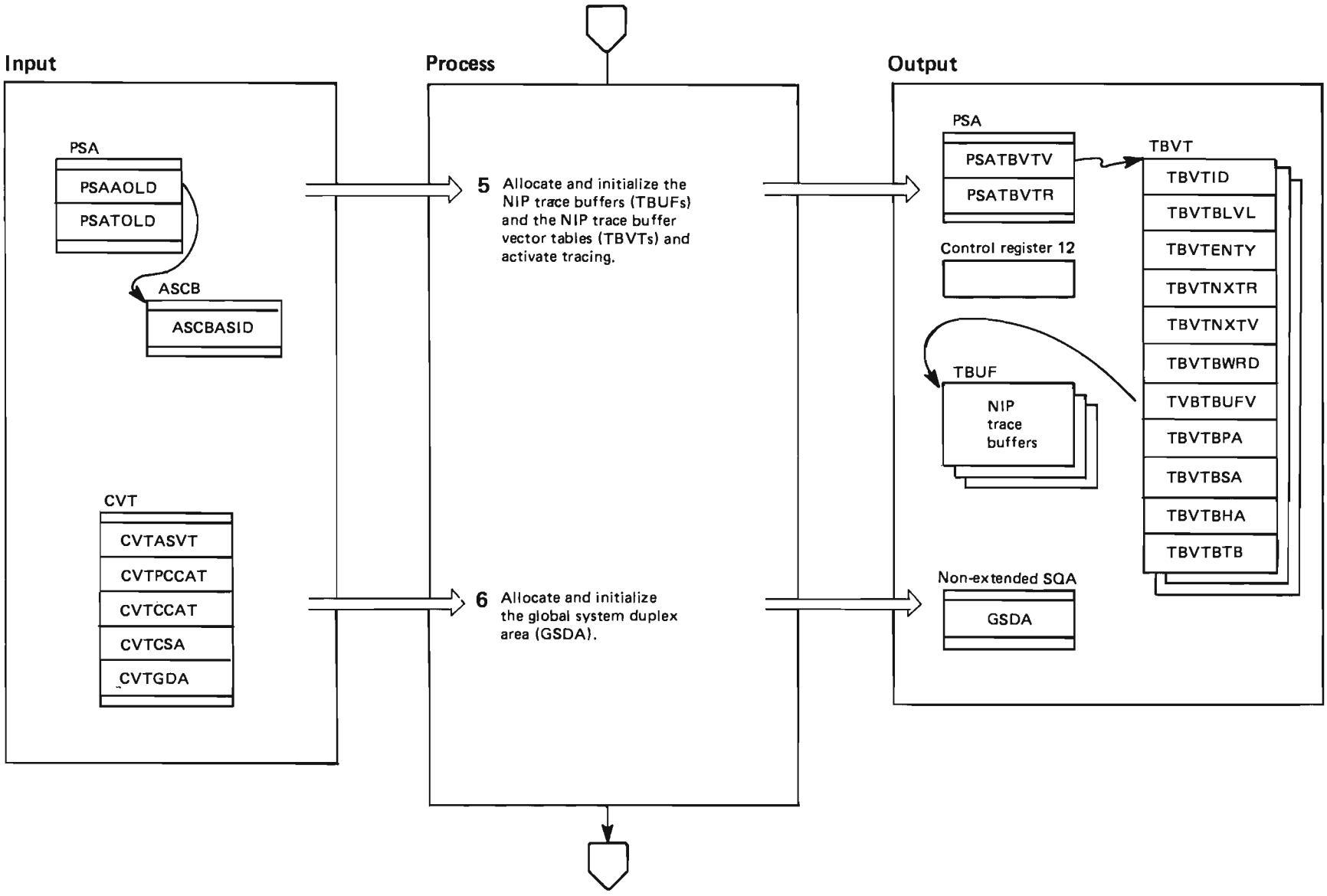
Field initialized	*Initialization value*
TCBRD	address of non-extended V=V region descriptor
TCBERD	address of extended V=V region descriptor
TCBERV	'0' – indicating non V=R

ASCBRSM contains the address of the master scheduler's RAB stored in the wait task's ASCB.

NIP frees the unused portions of extended and non-extended SQA subpools 245 and 239. — NPOFSQA

b. NIP calls the IOS storage allocation subroutines IECVCPRM and IOSVCPRM to allocate and initialize buffers for IOS. It then enables for I/O and external interrupts by releasing the processor lock that it obtained in Step 1. For more information about IECVCPRM and IOSVCPRM, refer to the input/output supervisor section of *System Logic Library*. — IECVCPRM, IOSVCPRM

Diagram 9. IPL/NIP Interface Routine (IEAVNIP0) Part 13 of 16



Extended Description	Module	Label
5 NIP allocates and initializes the NIP trace buffers and the NIP trace buffer vector tables (TBVTs). From subpool 245 in the extended system queue area (ESQA), NIP obtains 4K of storage for each of four initial NIP trace buffers and four NIP TBVTs. It initializes each of the NIP trace buffers with the identification 'TBUF'. NIP then implicitly chains the NIP trace buffers together by placing the NIP TBVTs into a circular queue. The following fields are initialized in each of the NIP TBVTs:	IEAVNIP0	NPOITRCE
Field initialized — *Initialization value* TBVTID — 'TBVT' TBVTBLVL — control block level number TBVTBUFV — address of 4K buffer associated with this TBVT TBVTCR12 — real address of associated buffer TBVTBLVL — control block level number TBVTENTY — real address of next available slot in trace buffer TBVTBWRD — address of previous NIP TBVT on queue TBVTNXTV — address of next NIP TBVT on queue TBVTNXTR — real address of next NIP TBVT on queue		
Finally, NIP activates trace by loading control register 12 with the real address of the first NIP trace buffer in the queue and other trace information. See Figure 5-4 for values in control register 12. NIP initializes the following fields:		NPOGTBVF
Field initialized — *Initialization value* TBVTBPA — PASID at time the buffer became current TBVTBSA — SASID at time the buffer became current TBVTBHA — HASID at time the buffer became current TBVTBTB — value of PSATOLD at time the buffer became current PSATBVTV — address of first PBVT PSATBVTR — real address of first TBVT		NPOITRCE
6 NIP allocates storage for the global system duplex area (GSDA), which contains copies of addresses used by recovery routines. It allocates the storage for the GSDA from the non-extended SQA (subpool 245), then initializes the GSDA, and turns on the validity bits as follows: *Field initialized* — *Initialization value* GSDAASVT — address of ASVT GSDAASVB — '1' – indicating valid address GSDAPCCT — address of PCCAVT GSDAPCCB — '1' – indicating valid address GSDALCCT — address of LCCAVT GSDALCCB — '1' – indicating valid address GSDACSD — address of CSD GSDACSDB — '1' – indicating valid address GSDAGDA — address of GDA GSDAGDAB — '1' – indicating valid address CVTGSDA — address of GSDA CVTGSDAB — '1' – indicating valid address		NPOIGSDA

Diagram 9. IPL/NIP Interface Routine (IEAVNIP0) Part 15 of 16

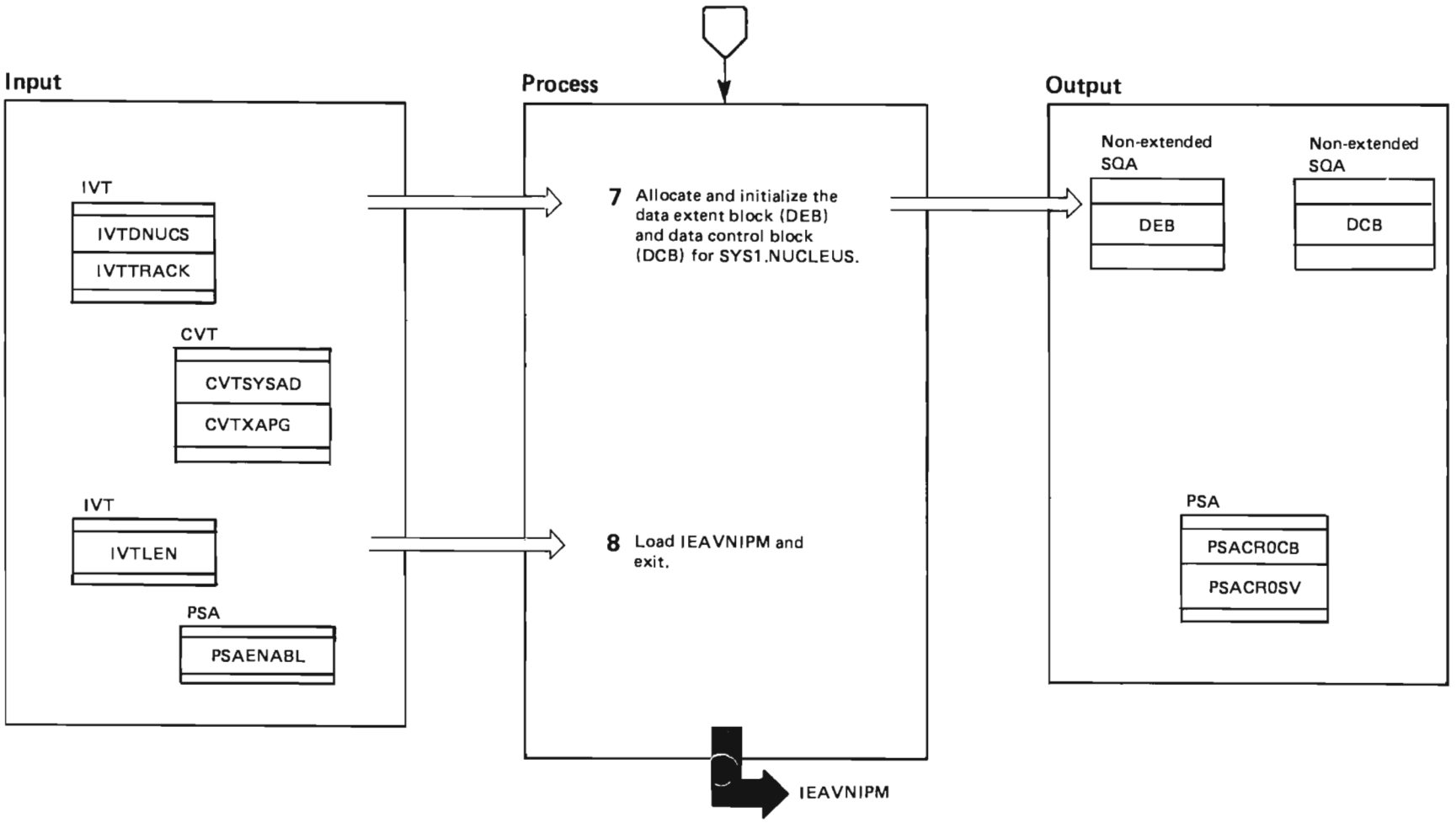


Diagram 9. IPL/NIP Interface Routine (IEAVNIP0) Part 16 of 16

Extended Description	Module	Label
7 NIP allocates a data control block (DCB) and a data extent block (DEB) for SYS1.NUCLEUS from non-extended SQA (subpool 245) and initializes them as follows:		NPOIDEB

Field initialized	*Initialization value*
DCBMFECP	'1' – indicating the DCB is for EXCP
DCBDEBA	address of associated DEB
DCBOFOPN	'1' – indicating a successful open
DCBOFUEX	'1' – indicating return from user exit to I/O support function that invoked the exit

Field initialized	*Initialization value*
DEBNMTRK	size of SYS1.NUCLEUS (in tracks)
DEBAPPAD	address of system appendage vector table
DEBAPFIN	authorized program facility (APF) flag for SYS1.NUCLEUS
DEBUCBA	address of UCB for SYS1.NUCLEUS dataset
DEBSTRCC, DEBSTRHH	start address of SYS1.NUCLEUS
DEBENDCC, DEBENDHH	end address of SYS1.NUCLEUS
DEBNMEXT	number of extents for SYS1.NUCLEUS
DEBDEBID	protect key and DEB block identity
DEBVMOD	file mask
DEBDCBB	address of DCB

Extended Description	Module	Label
8 NIP makes final preparations for giving control to IEAVNIPM and the resource initialization modules (RIMs).		NPOEXIT

- Because the IVT will no longer be used, NIP frees the storage it occupies.
- NIP sets the control register 0 control byte in the PSA (PSACR0CB) to the value in PSAENABL. NIP then issues the PROTPSA macro instruction to enable PSA protection.
- NIP issues a BLDL for IEAVNIPM. It checks the return code from the BLDL and proceeds as follows:

Return code	*NIP procedure*
0	Load IEAVNIPM and pass control to it
4—module not found	Load disabled wait state code '032'X
8—I/O error during BLDL	Load disabled wait state code '033'X
any other	Load disabled wait state code '059'X

Control Register	*Bits*	*Set to*	*Meaning*
0	0	0	Reserved
	1	1	SSM suppression control – enabled
	2	0	TOD clock synchronization – disabled
	3	0	Low address protect – disabled
	4	1	Extract authorized control – enabled
	5	1	Secondary space control – enabled
	6	1	Fetch protection override – enabled
	7	0	Reserved
	8-12	10110	4K page size, 1 megabyte segment size
	13-15	000	Reserved
	16	1	Malfunction alert – enabled
	17	1	Emergency signal – enabled
	18	1	External call – enabled
	19	0	TOD clock synch check – disabled
	20	1	Clock comparator – enabled
	21	1	CPU timer – enabled
	22	1	Service signal – enabled
	23-24	00	Reserved
	25	1	External interrupt key – enabled
	26-31	000000	Reserved
1	0	0	Reserved
	1-19		Real address of segment table
	20-25	000000	Reserved
	26-31		4K segment table size
3	0-15	0	Program key mask (PKM)
	16-31	1	Secondary address space identifier (SASID)
4	0-15	1	Authorization index (AX)
	16-31	1	Primary address space identifier (PASID)
5	0-31	0	Invalid linkage table origin (LTO)
6	0-6	1111111	Interrupts from all interruption subclasses 0-6 – enabled
	7	0	Interruption subclass 7 – disabled
	8-31	zeroes	Reserved
7	0-31		Same as control register 1
12	0	0	Branch tracing – disabled
	1-29		Real address of next available word in current trace buffer
	30	1	Address space tracing – enabled
	31	1	Explicit addressing – enabled
14	0-5	–	Unknown
	6	1	External damage reports – enabled
	7-11	–	Unknown
	12	1	Validate AFT pointer – enabled
	13-31	–	Real address of AFT

Figure 5-4. Contents of Certain Control Registers at Exit from IEAVNIP0

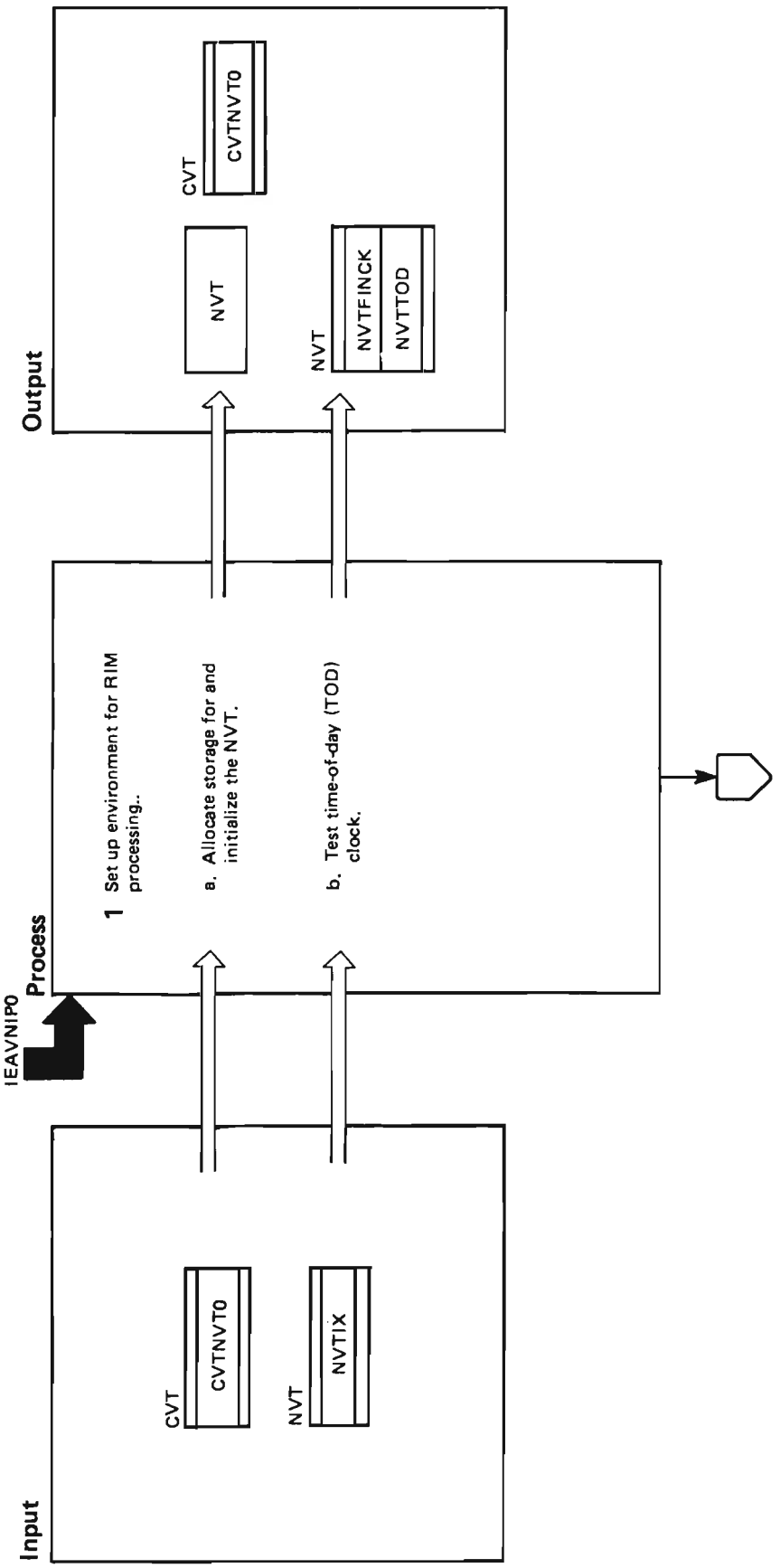


Diagram 10. NIP Control and Service Routine (IEAVNIPM) Part 1 of 44

Diagram 10. NIP Control and Service Routine (IEAVNIPM) Part 2 of 44

Extended Description	Module	Label

IEAVNIPM is the main control program for the loading and deleting of resource initialization modules (RIMs). It provides diagnostic support for software and hardware failures that might occur during NIP processing before system recovery services are available. It provides the NIP modules and RIMs with service routines that substitute for functions that are not yet available in the system.

1 The first function performed by NIP in IEAVNIPM is to set up an environment for the RIMs.

a. NIP obtains storage in the non-extended SQA (subpool 245) for the nucleus vector table (NVT). It copies into this area the NVT fields that IEAVNIP0 initialized. NIP initializes the following fields in the NVT with pointers to NIP services and other addresses:

Field initialized	*NIP Service Routine*
NVTMODMN	name of last module loaded
NVTMODEP	entry point of last module loaded
NVTDCBSN	address of SYS1.NUCLEUS DCB
NVTMOUNT	address of NIPMOUNT service routine
NVTOPIO	address of NIPOIO service routine
NVTPRMPT	address of NIPPROMPT service routine
NVTSENSE	address of NIPSENSE service routine
NVTOPEN	address of NIPOPEN service routine

Extended Description	Module	Label

Field initialized	*NIP Service Routine*
NVTSWAIT	address of NIPWAIT service routine
NVTTIME	address of NIPTIME routine
NVTUCBFN	address of NIPUCBFN routine
NVTWTO	address of NIPWTO service routine
NVTWTOR	address of NIPWTOR service routine
NVTWTOR2	address of NIPWTOR2 service routine
NVTNPM4	address of routine to find and read SYS1.PARMLIB members
NVTCPUAD	physical address of IPL processor

The NVT will reside in the non-extended SQA until IEAVNIPX releases the NVT storage.

b. NIP tests the time-of-day (TOD) clock on the IPL processor. If the TOD clock is accurate, NIP places the clock value in NVTTOD. Otherwise, NIP sets the flag field NVTFLNCK on to indicate that the TOD clock is not operating. In either case, processing continues and the reconfiguration RIM, IEAVNP00, will invoke IEAVRTOD to initialize the TOD clock.

Diagram 10. NIP Control and Service Routine (IEAVNIPM) Part 3 of 44

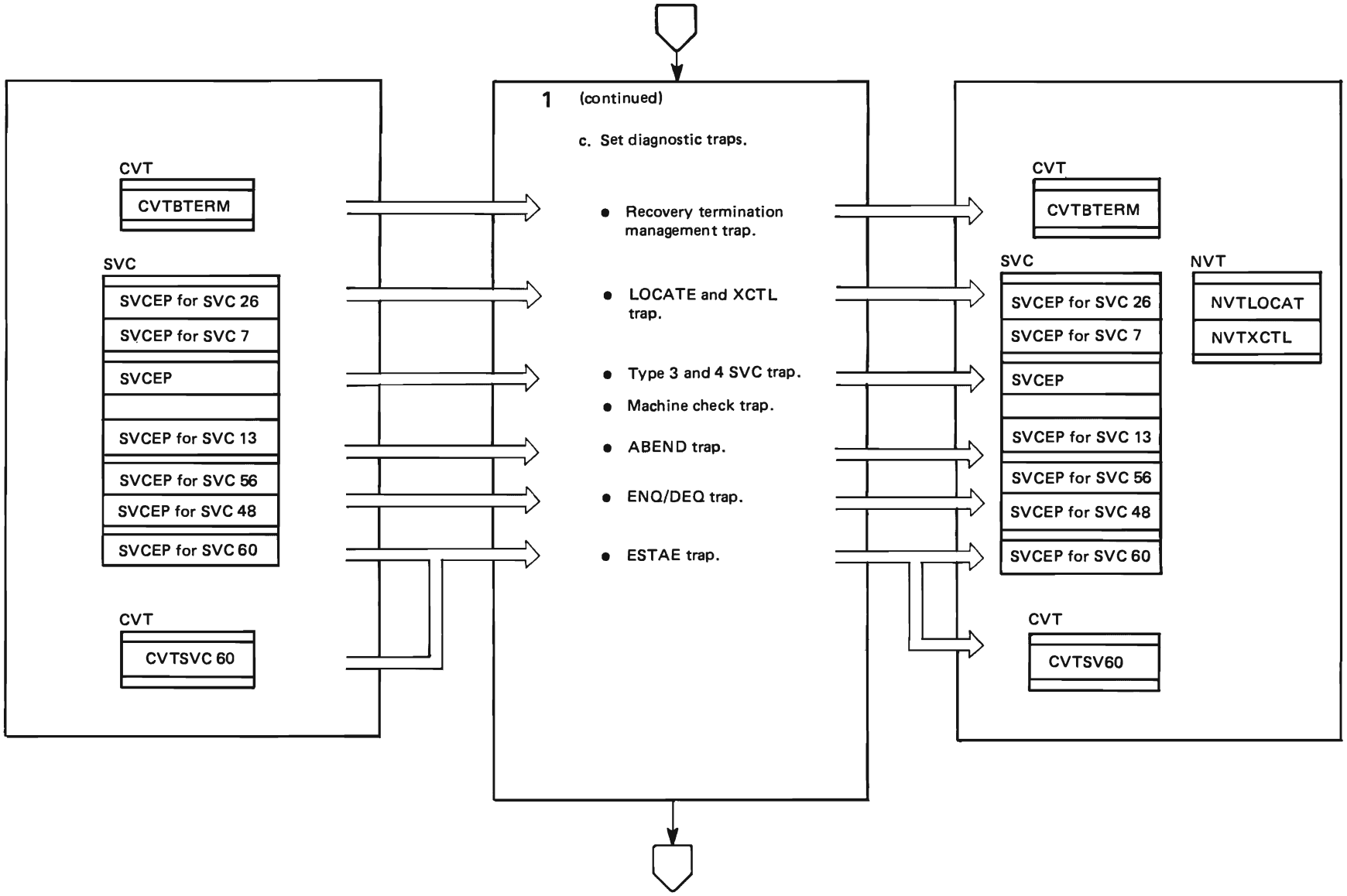


Diagram 10. NIP Control and Service Routine (IEAVNIPM) Part 4 of 44

Extended Description	Module	Label
1 (continued)		
c. Before system recovery routines are available, NIP provides diagnostic support for hardware and software errors by trapping the errors. NIP's trap routines detect errors that might occur during NIP, stop the initialization process at the time the error occurs, and provide diagnostic information in the form of wait state codes and messages. For more information on IEAVNIPM diagnostic trap routines, see Diagnostic Aids: Section 3. NIP sets the following error traps:		
• **Recovery termination management (RTM) trap:** NIP, in IEAVNIP0, saved the address of the ABTERM routine (found in CVTBTERM) in the NVT. NIP now must reset CVTBTERM to the address of the table of the RTM traps (RTMTRAP) within IEAVNIPM. The RTMTRAP contains addresses of trap routines. The module IEAVNIPX removes the RTM trap during NIP exit processing.		
• **LOCATE SVC and XCTL trap:** NIP saves the LOCATE (SVC 26) SVC table entry in field NVTLOCAT. NIP places the address of the NIP SVC trap in the LOCATE SVC table entry and changes the SVC type to 2 so that the contents supervisor RIM, IEAVNP05, will not alter the entry when it initializes the SVC table. Thus, NIP traps the LOCATE SVC throughout NIP processing, and calls IEAVNP12 to issue a LOCATE for the VSAM catalog. The system LOCATE cannot function until IEAVNIPX places the address of the routine in the SVC table. NIP saves the XCTL (SVC 7) SVC table entry in NVTXCTL and places the address of the NIP SVC trap in the XCTL SVC table entry. IEAVNP05 removes the traps after the LPA is constructed and places the proper addresses in the SVC table.		
• **Type 3 and type 4 SVC trap:** NIP places the address of the NIP SVC trap in the address field of each SVC table entry for a type 3 or type 4 SVC routine. The contents supervisor RIM, IEAVNP05, removes this trap after constructing the LPA and making the SVC routines residing there available.		
• **Machine check trap:** NIP handles machine checks by continuing to allow the system to enter a disabled wait state when the machine check new PSW is entered. The machine check new PSW was initialized by IEAVNIP0. The machine check RIM, IEAVNP06, removes this trap when it initializes the machine check handler.		
• **ABEND trap:** NIP, in IEAVNIP0, saved the contents of the ABEND SVC table entry in field NVTABSAV of the NVT. NIP places the address of the NIPABEND routine in the SVC table and changes the SVC type to 2 so that the contents supervisor RIM, IEAVNP05, will not alter the entry when it initializes type 3 and type 4 SVCs in the SVC table. IEAVNIPX removes this trap during exit processing.		
• **ENQ/DEQ trap:** NIP saves the contents of the ENQ (SVC 56) and the DEQ (SVC 48) SVC table entries. NIP places the address of the NIP ENQ trap routine in the SVC table entries for ENQ and DEQ. The NIP ENQ trap clears register 15 to zero and returns to caller. After the global resource serialization (GRS) RIM, IEAVNP23, has processed, NIP will reset the SVC trap.		
• **ESTAE:** NIP, in IEAVNIP0, saved the STAE/ESTAE (SVC 60) SVC table entry in NVTSVC60. NIP places the address of the NIP ESTAE trap routine in the ESTAE SVC table entry and in the ESTAE branch entry CVTSV60. The contents supervisor RIM, IEAVNP05, will reset the SVC trap and the RTM RIM, IEAVNPA6, will set the branch entry pointer. The contents supervisor RIM, IEAVNP05, removes this trap after the LPA has been constructed and the SVC routines residing there are available.		

Diagram 10. NIP Control and Service Routine (IEAVNIPM) Part 5 of 44

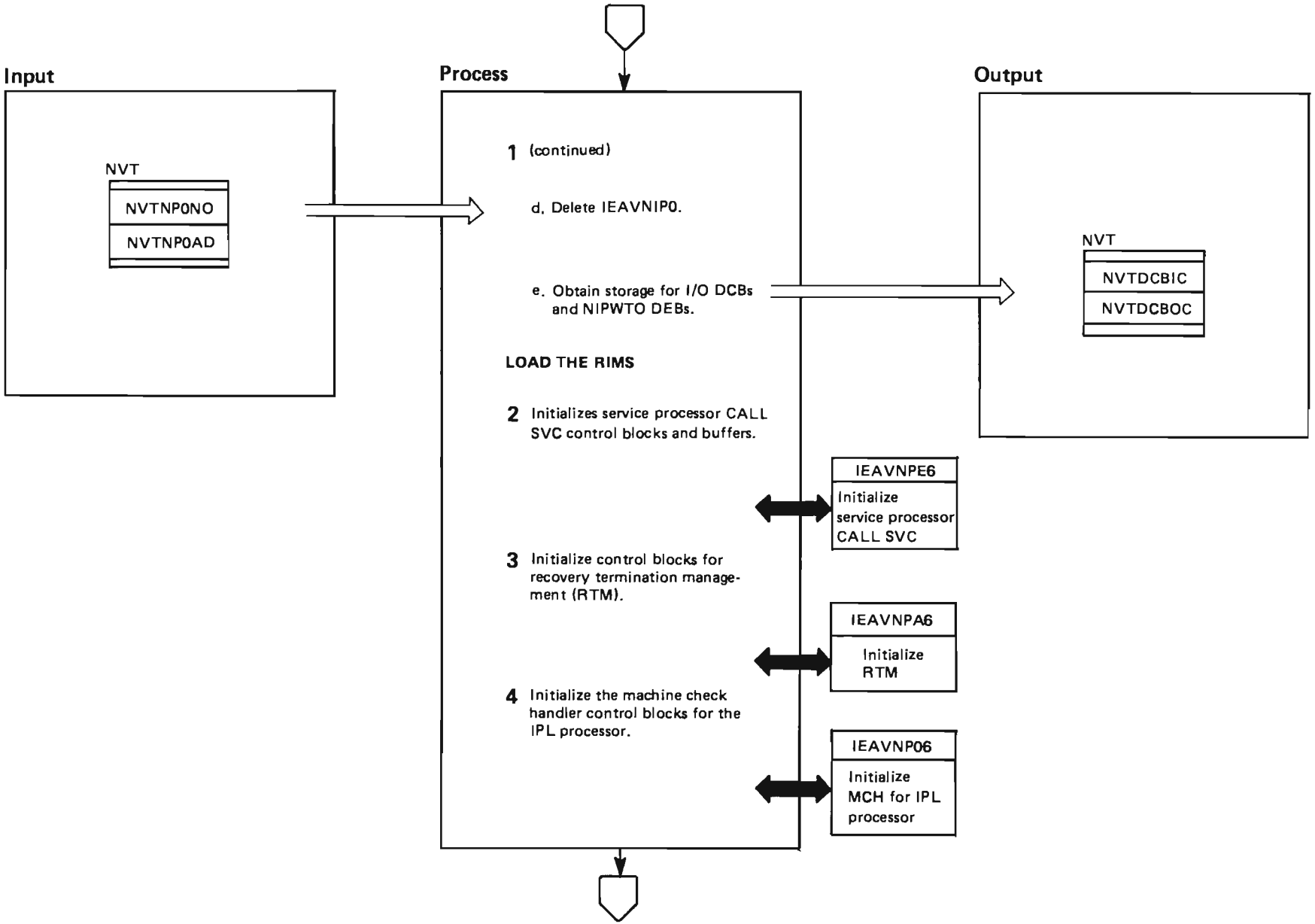


Diagram 10. NIP Control and Service Routine (IEAVNIPM) Part 6 of 44

Extended Description	Module	Label
1 (continued) d. NIP deletes IEAVNIP0 by invoking IARVFRMN to free the storage it occupied and return the frames that IEAVNIP0 occupied to the available frame queue. e. NIP obtains storage in the private area (subpool 252) for the input/output DCBs and DEBs for the NIPWTO services. Before passing control to the first RIM, NIP preloads the LOCATE module, IEAVNP12, so that NIP can use this service to locate the RIMs. NIP starts the load, branch and delete sequence that passes control to the resource initialization modules (RIMs) in the order specified in the suffix list. The suffix list, located within IEAVNIPM, contains entries of two characters which NIP appends to 'IEAVNP' to create full module names. The list ends with 'XX', which signals to NIP that all RIMs have been invoked.		
2 The first RIM that NIP calls is the service processor RIM, IEAVNPE6. The service processor RIM initializes the control blocks initialized by the service processor CALL SVC. The control blocks initialized are the service processor control block (MSFCB) and its buffer, the service processor attention block (MSFAB) and its buffer, and a service processor recovery buffer. See Diagram 9.	IEAVNPE6	
3 NIP calls the recovery termination management (RTM) RIM, IEAVNPA6, to initialize the RTM control blocks that support programs running in SRB mode. See Diagram 10.	IEAVNPA6	
4 NIP calls the machine check handler (MCH) RIM, IEAVNP06, to initialize the MCH control blocks for the processor. See Diagram 11. After the MCH RIM has processed, NIP restores the machine check entry in the RTM ABTERM routine so that normal ABTERM processing will begin.	IEAVNP06	

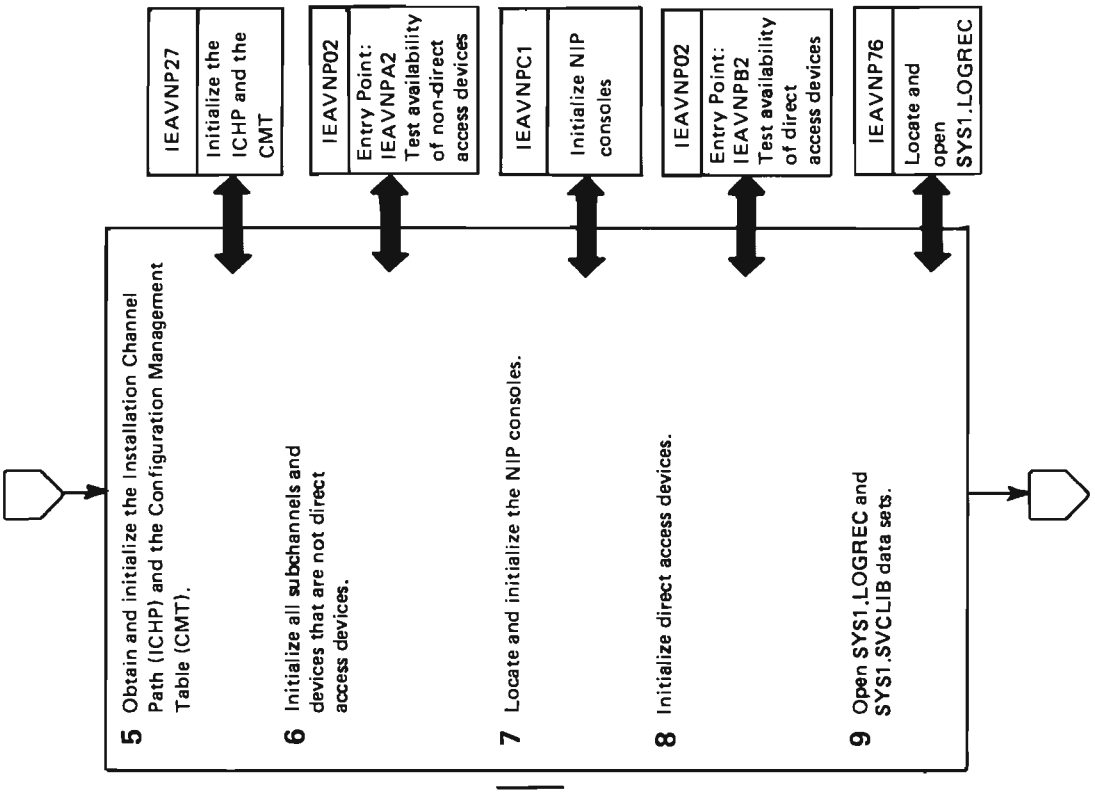


Diagram 10. NIP Control and Service Routine (IEAVNIPM) Part 7 of 44

Diagram 10. NIP Control and Service Routine (IEAVNIPM) Part 8 of 44

Extended Description	Module	Label
6 NIP calls the input/output service (IOS) RIM, at entry point, IEAVNPA2, to initialize all subchannels and all devices that are not direct access devices.	IEAVNP02	IEAVNPA2
7 NIP calls NIP RIM, IEAVNPC1 to locate and initialize the NIP console for the system. The NIP RIM then establishes initial operator communication. After IEAVNPC1 completes, NIP issues all of the messages on the IPL/NIP message queue. IPLWTOs and NIPWTOs that occur before a console is available are saved here. Then, NIP waits for the operator's response to the "SPECIFY SYSTEM PARAMETERS" message and queues each line of the response. IEAVNP03 will process the response.	IEAVNPC1	
8 NIP calls the IOS RIM, at entry point, IEAVNPB2, to initialize direct access devices.	IEAVNP02	IEAVNPB2
9 NIP calls IEAVNP76 to open SYS1.LOGREC and uses the NIPOPEN service routine to open SYS1.SVCLIB.	IEAVNP76 IEAVNIPM	

The following table shows the system data sets that are opened during NIP processing and the module that opens each:

Data Set	*Module*
SYS1.LOGREC	IEAVNP76
SYS1.SVCLIB	IEAVNIPM
System catalog	IEAVNP11
SYS1.PARMLIB	IEAVNP03
LNKLST concatenation, headed by SYS1.LINKLIB	IEAVNP03
LPALST concatenation, headed by SYS1.LPALIB	IEAVNP05
swap and page data sets	ILROPS00

Diagram 10. NIP Control and Service Routine (IEAVNIPM) Part 9 of 44

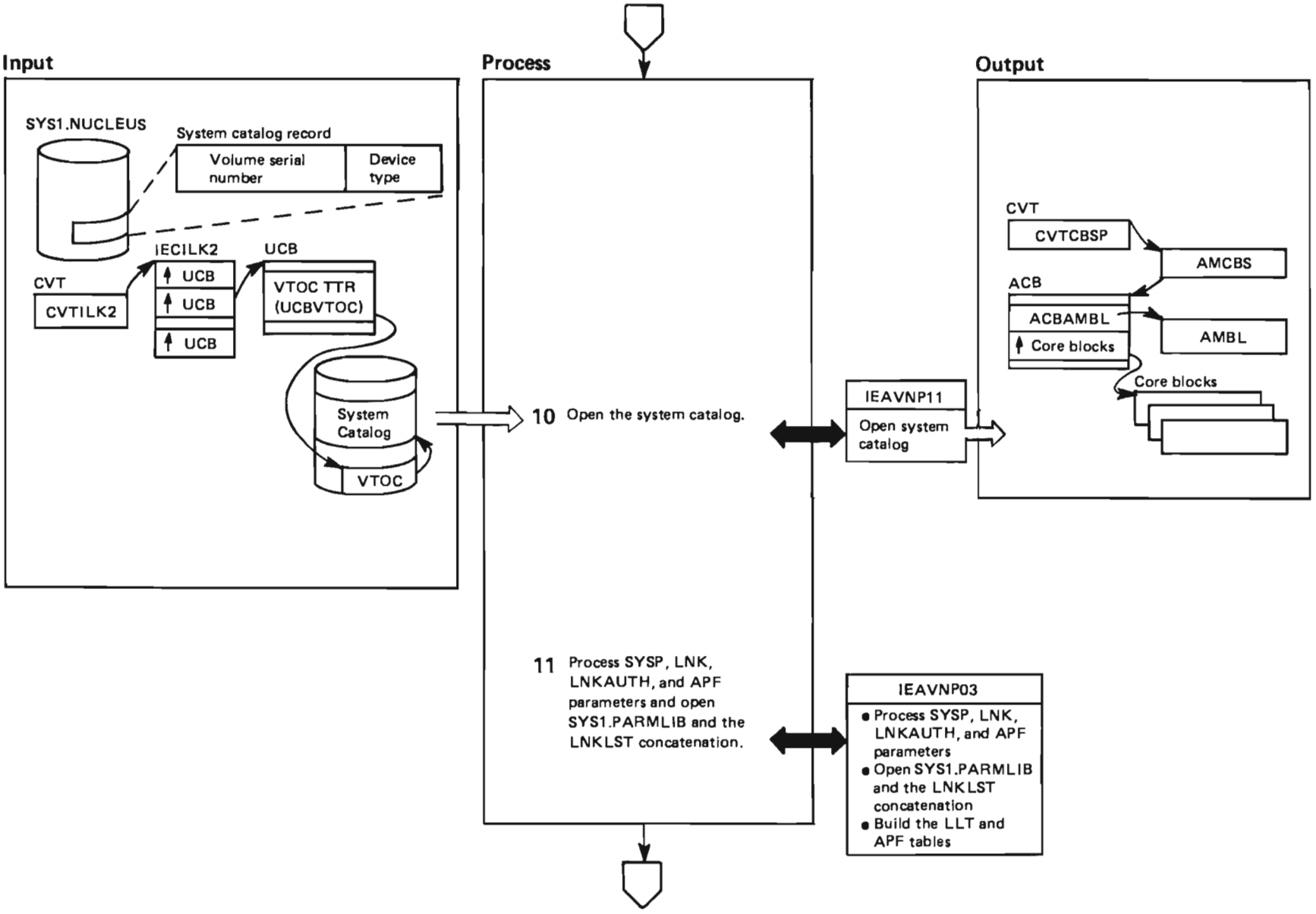


Diagram 10. NIP Control and Service Routine (IEAVNIPM) Part 10 of 44

Extended Description	Module	Label
10 IEAVNIPM calls the VSAM RIM, IEAVNIP11, to build the access method control block structure (AMCBS) and to open the master catalog. The VSAM RIM reads a SYSCATLG record from SYS1.NUCLEUS to determine the type of master catalog: VSAM or integrated catalog facility (ICF) catalog. If the master catalog is a VSAM catalog, IEAVNP11 reads the first ten records, which describe the catalog, and initializes the access method block (AMB) work areas. The VSAM open routine (IEAVNP1A) needs these work areas to build the control blocks required to access the catalog. If the master catalog is an ICF catalog, the VSAM RIM calls IEAVNP1B to read the VSAM volume data set (VVDS) and initialize the AMB work areas.	IEAVNP11	RDDIR
The VSAM RIM calls IEAVNP1A to open the master catalog. IEAVNP1A constructs a control block called the access method block list (AMBL), which contains pointers to the VSAM control block structure created at this time. Core blocks are built to describe the storage that has been allocated; the address of the first core block is placed in the access method control block (ACB). At the end of NIP processing, IEAVNP1VB frees the master catalog control blocks used during NIP. The close code in IEAVNP1B initiates the creation of the catalog address space. IEAVNP1A returns control to the VSAM RIM which returns to IEAVNIPM.	IEAVNP1A IEAVNP11	
11 NIP calls IEAVNP03 to check the syntax of the system parameters that are specified in SYS1.PARMLIB or by the operator. It places these parameters in PARMTAB. It processes the SYSP, LNK, LNKAUTH parameters, and calls IEAVNPA5 to process the APF parameter. It opens the system data sets SYS1.PARMLIB and the LNKLST concatenation, headed by SYS1.LINKLIB.	IEAVNP03	

Diagram 10. NIP Control and Service Routine (IEAVNIPM) Part 11 of 44

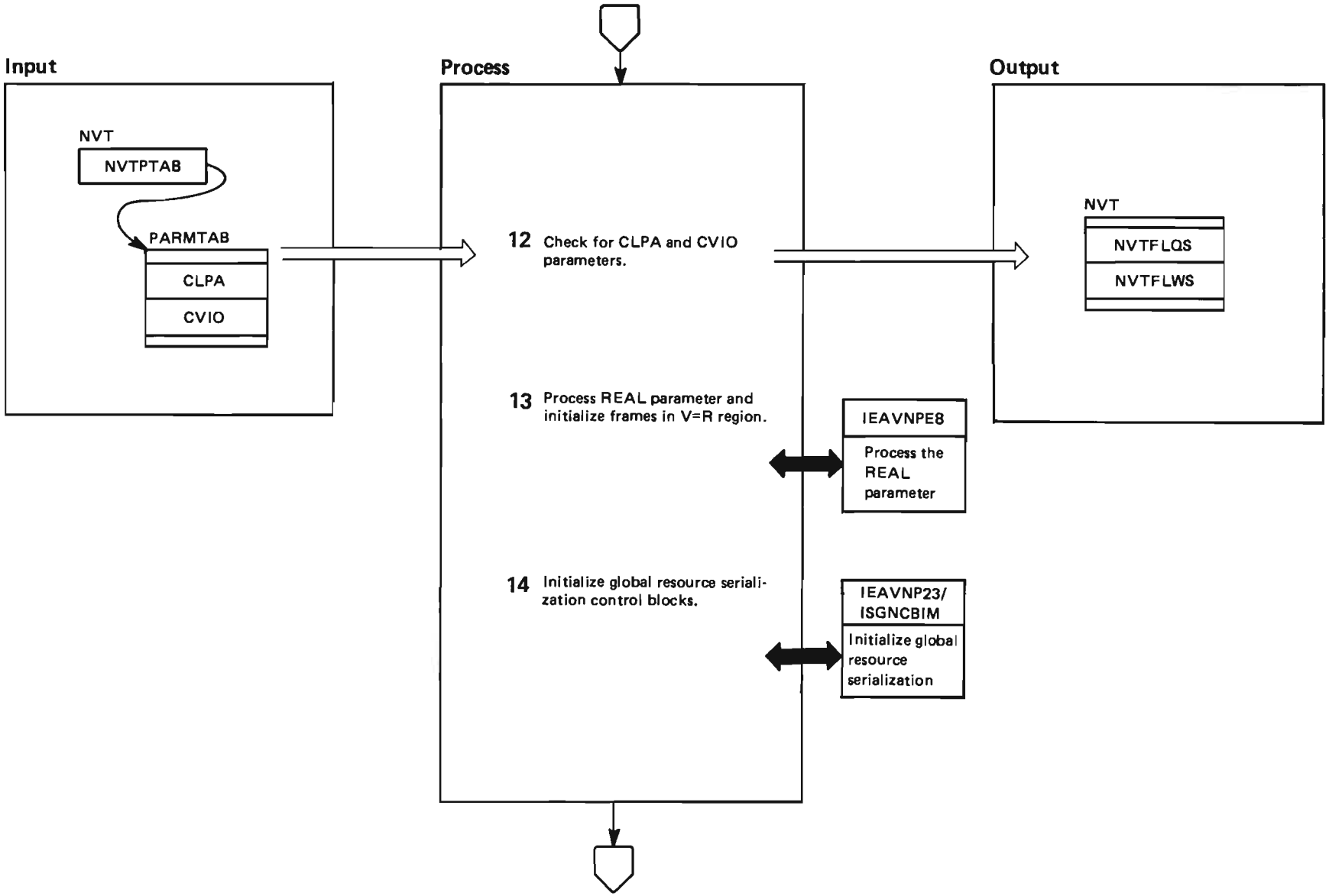


Diagram 10. NIP Control and Service Routine (IEAVNIPM) Part 12 of 44

Extended Description	Module	Label
12 NIP checks for CLPA and CVIO system parameters and sets flags in the NVT that ASM will reference later in system initialization. If the CLPA parameter is specified, a cold start is in process. If the CVIO parameter is specified, a quick start is in process. If neither parameter is specified, a warm start is in process.	IEAVNIPM	
13 NIP calls the real storage management (RSM) RIM, IEAVNPE8, to check the syntax of the REAL parameter, mark all of the page frame table entries (PFTEs) for frames in the V=R region, and mark the address increment map entries (AIMEs).	IEAVNPE8	
14 NIP calls the global resource serialization RIM, IEAVNP23/ISGNCBIM, to initialize the GRS control blocks and process the GRS system parameters GRS, GRSCNF, GRSRNL, and SYSNAME.	ISGNCBIM	
NIP restores the ENQ (SVC 56) and DEQ (SVC 48) entries in SVCTABLE.	IEAVNIPM	

Diagram 10. NIP Control and Service Routine (IEAVNIPM) Part 13 of 44

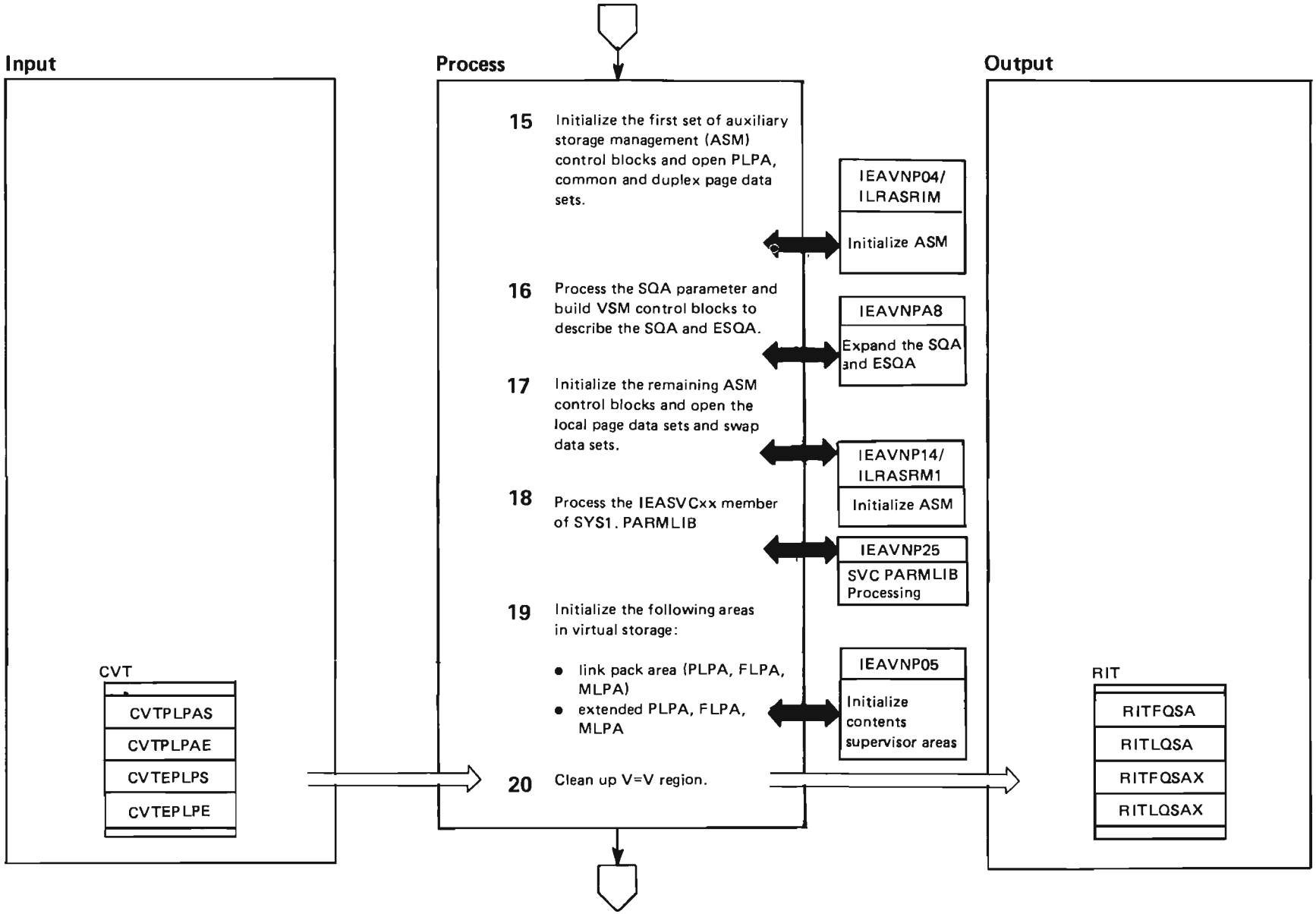


Diagram 10. NIP Control and Service Routine (IEAVNIPM) Part 14 of 44

Extended Description	Module	Label
15 NIP calls the first auxiliary storage management (ASM) RIM, IEAVNP04/ILRASRIM, to build and initialize the ASM control blocks and open the PLPA, common, and, if requested, duplex page data sets.	ILRASRIM	
16 NIP calls the virtual storage management (VSM) RIM, IEAVNPA8, to process the SQA parameter and build the VSM control blocks and page tables necessary to describe the expanded system queue area (SQA) and the extended SQA (ESQA) areas.	IEAVNPA8	
17 NIP calls the second auxiliary storage management (ASM) RIM, IEAVNP14/ILRASRM1, to build and initialize the remaining ASM control blocks. The RIM also opens the local page data sets and swap data sets.		
18 NIP calls the supervisor control's RIM, IEAVNP25, to process the IEASVCxx parmlib member, install nucleus-resident routines and to save the spcifications for LPA-resident user-SVCs until IEAVNP05 builds the LPA.	IEAVNP25	
19 NIP calls the contents supervisor RIM, IEAVNP05, to initialize the following areas in virtual storage: • Link pack areas: PLPA, FLPA, and MLPA • Extended link pack area: EPLPA, EFLPA, and EMLPA Modules loaded into the PLPA and EPLPA areas are obtained from the LPALST concatenation headed by SYS1.LPALIB. IEAVNP05 opens the LPALST. The contents supervisor RIM also initializes the authorized program facility (APF) table and the type 3 and 4 SVC routines in the SVC table. It then opens the SYS1.LPALIB data set.	IEAVNP05	
20 IEAVNIPM cleans up the V=V region.	IEAVNIPM	

Diagram 10. NIP Control and Service Routine (IEAVNIPM) Part 15 of 44

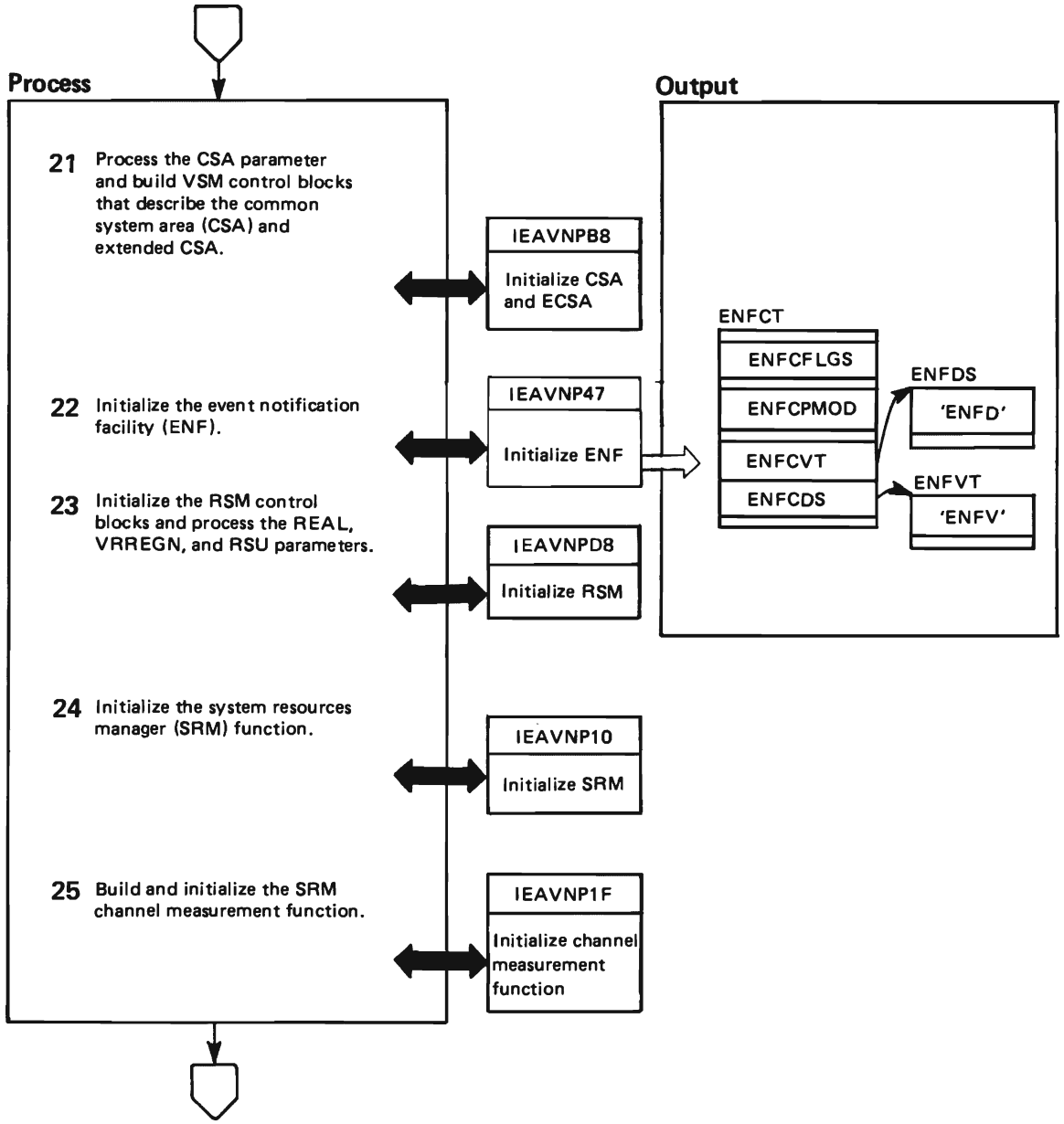


Diagram 10. NIP Control and Service Routine (IEAVNIPM) Part 16 of 44

Extended Description	Module	Label
21 NIP calls the VSM RIM, IEAVNPB8, to process the CSA parameter and build the VSM control blocks and page tables necessary to describe the common system area (CSA) and extended CSA.	IEAVNPB8	
22 NIP calls the event notification facility (ENF) RIM, IEAVNP47, to initialize the ENF function. After IEAVNP47 finishes executing, ENF is able to process synchronous requests for service and to accept asynchronous requests. However, it cannot process the asynchronous requests until IEEMB860 attaches the ENF wait routine (IEFENFWT), which completes ENF initialization.	IEAVNP47	
To initialize ENF, IEAVNP47: • Builds and initializes the ENF vector table (ENFVT). For each ENF event defined, ENFVT contains an entry relating the event to the listener. IEAVNP47 issues a GETMAIN to acquire enough storage from subpool 231 to hold entries for the maximum number of events that can be defined (specified in the ENFCMAX field of the event control table). IEAVNP47 sets each entry in ENFVT to zero, places a control block ID ('ENFV') in the first word, and stores ENFVT's address in the ENF control table. A GETMAIN failure causes a wait state code of X'064'. • Builds and initializes the ENF process table (ENFDS), which contains an event parameter list for each unprocessed asychronous ENF request. IEAVNP47 obtains enough storage from subpool 239 to hold the maximum number of unprocessed requests allowed (specified by the system in the ENFDSMAX field). IEAVNP47 sets each entry in the ENFDS to zero, places the control block ID ('ENFD') in the first word, and stores ENFD's address in the ENF control table (ENFCT). A GETMAIN failure causes a wait state code of X'064'. • Loads the address of the event notification processing routine (IEFENFNM) into the ENF control table (ENFCT). • Indicates ENF RIM initialization is complete by setting a flag in the ENFCFLGS field of ENFCT. The ENFCT is a CSET (IERENFDM) in IEANUC01 and needs no further initialization. • Returns to IEAVNIPM.		
23 NIP calls the real storage management (RSM) RIM, IEAVNPD8, to initialize the RSM control blocks and process the REAL, VRREGN, and RSU system parameters.	IEAVNPD8	
24 NIP calls the system resource management (SRM) RIM, IEAVNP10, to initialize the SRM function.	IEAVNP10	
25 NIP calls the SRM RIM, IEAVNP1F, to build and initialize the SRM channel measurement functions.	IEAVNP1F	

Diagram 10. NIP Control and Service Routine (IEAVNIPM) Part 17 of 44

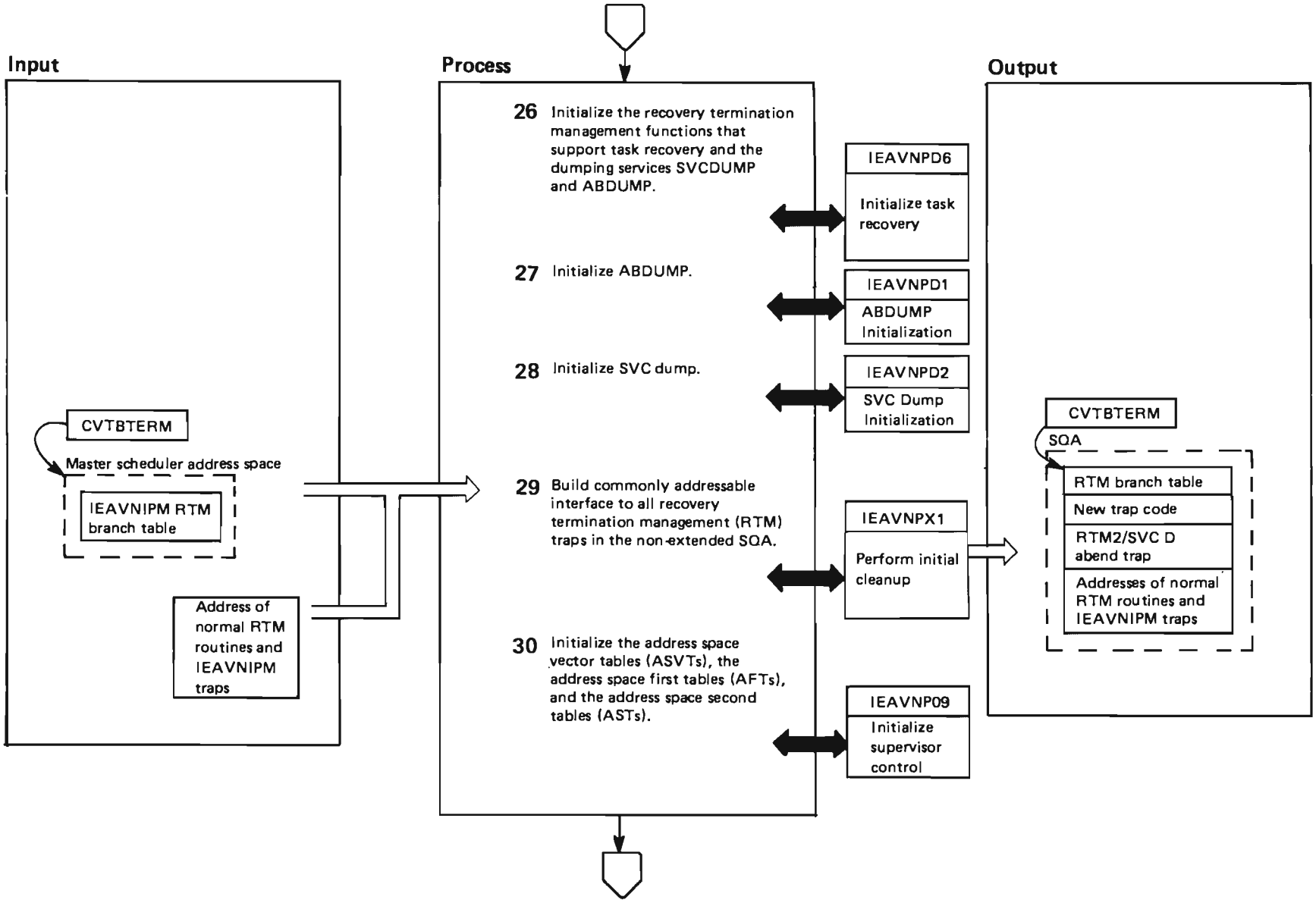


Diagram 10. NIP Control and Service Routine (IEAVNIPM) Part 18 of 44

Extended Description	Module	Label
26 NIP calls the recovery termination management (RTM) RIM, IEAVNPD6, to initialize the RTM functions that support task recovery (ESTAE/ESTAI).	IEAVNPD6	
27 NIP calls the ABDUMP RIM to initialize the ABDUMP control blocks and work areas. The ABDUMP RIM also initializes the default values for SYSABEND dumps, SYSUDUMP dumps, and SYSMDUMP dumps.	IEAVNPD1	
28 NIP calls the SVC dump RIM to initialize SVC dump and dump analysis and elimination (DAE) control blocks and work areas.	IEAVNPD2	
29 The NIP RIM, IEAVNPX1, builds in the non-extended SQA a commonly addressable interface to all recovery termination management (RTM) traps. This interface is necessary to handle abnormal conditions occurring in address spaces other than the master address space. The RIM includes code to trap these conditions, reestablish the master address space, and transfer control to the NIP trap routines. The RIM moves this code to the non-extended SQA in order to make it addressable regardless of which address space is in control. It resets the entries of the new RTM branch table to the proper points within the SQA code (some entries point to normal RTM routines) and resets CVTBTERM, the ABTERM routine entry point address, to point to this new RTM trap branch table.	IEAVNPX1	NPXRTRAP
The NIP RIM obtains space in non-extended CSA to copy the template PSA into, sets a pointer to the template PSA (CVTVPSA), and frees the non-extended SQA space that the PSA occupied.		NPXMPSA
30 After IEAVNPX1, NIP allows RIMs to use WTOs to issue messages to the NIP console. NIP calls the supervisor control RIM, IEAVNP09, to initialize the address space vector tables (ASVTs), the address space first tables (AFTs), and the address space second tables (ASTs).	IEAVNP09	

Diagram 10. NIP Control and Service Routine (IEAVNIPM) Part 19 of 44

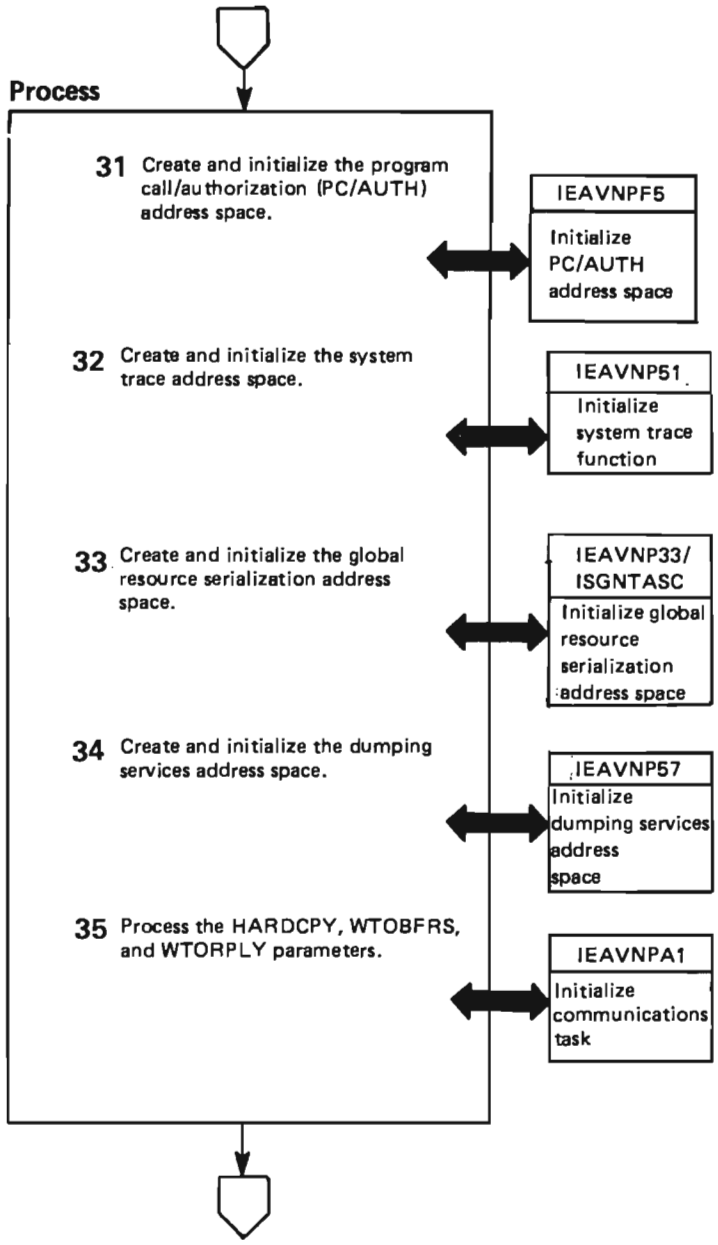


Diagram 10. NIP Control and Service Routine (IEAVNIPM) Part 20 of 44

Extended Description	Module	Label
31 NIP calls the program call/authorization RIM, IEAVNPF5, to create the PC/AUTH address space and initialize it.	IEAVNPF5	
32 NIP calls the system trace RIM, IEAVNP51, to create the system trace address space and initialize the trace function.	IEAVNP51	
33 NIP calls the global resource serialization RIM, IEAVNP33/ISGNTASC, to initiate the creation of the global resource serialization address space.	ISGNTASC	
34 NIP calls the dumping services RIM, IEAVNP57, to create and initialize the dumping services address space, DUMPSRV.	IEAVNP57	
35 NIP calls the communications task RIM, IEAVNPA1, to process the HARDCPY, WTOBFRS, and WTORPLY system parameters.	IEAVNPA1	

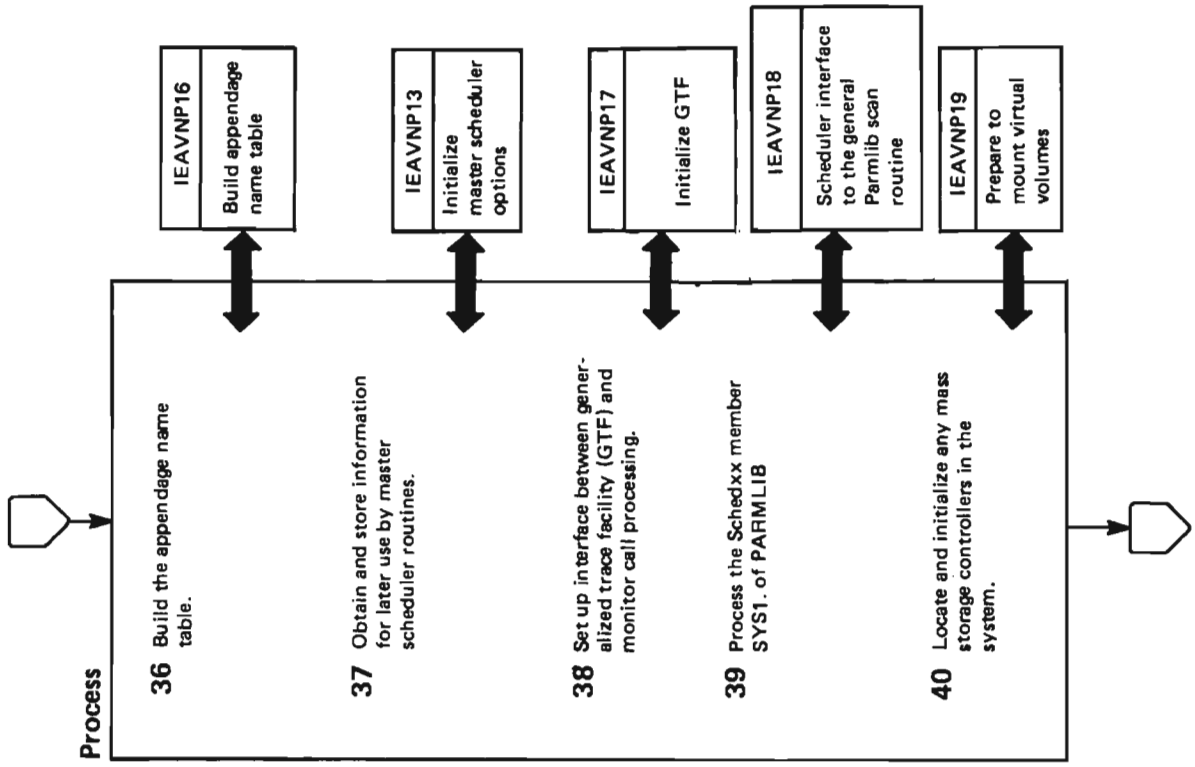


Diagram 10. NIP Control and Service Routine (IEAVNIPM) Part 21 of 44

Diagram 10. NIP Control and Service Routine (IEAVNIPM) Part 22 of 44

Extended Description	Module	Label
36 NIP calls the data management RIM, IEAVNP16, to build a table of user-written EXCP appendages that can be used by unauthorized programs.	IEAVNP16	
37 NIP calls the master scheduler RIM, IEAVNP13, to obtain and store information that will be needed later in the initialization process.		IEAVNP13
38 NIP calls the generalized trace facility (GTF) RIM, IEAVNP17, to set up an interface with monitor call processing.	IEAVNP17	
39 NIP calls the master scheduler interface to the General Parmlib Scan Routine (IEAVNP18). The interface initializes a parameter list, IEEZB819, and invokes the General Parmlib Scan Routine, IEEMB888.	IEAVNP18	
40 NIP calls the mass storage system communicator (MSSC) RIM, IEAVNP19, to locate and initialize any mass storage controllers in the system.	IEAVNP19	

Diagram 10. NIP Control and Service Routine (IEAVNIPM) Part 23 of 44

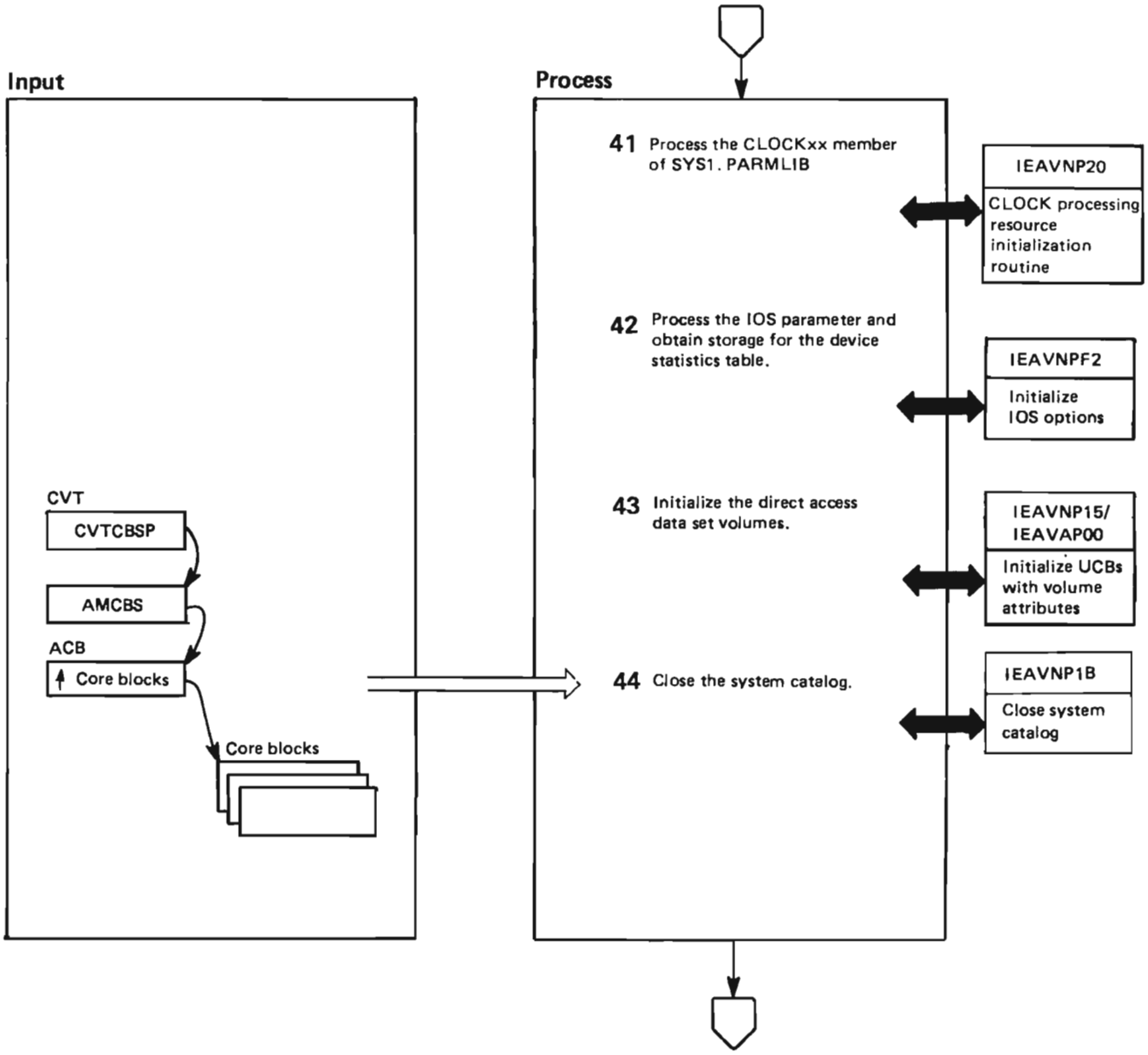


Diagram 10. NIP Control and Service Routine (IEAVNIPM) Part 24 of 44

Extended Description	Module	Label
41 NIP calls the supervisor control RIM, IEAVNP20, to process the CLOCKxx parmlib member. This may prompt the operator to set the TOD clock.	IEAVNP20	
42 NIP calls the IOS RIM, IEAVNPF2, to process the IOS system parameter and obtain storage for the device statistics table.	IEAVNPF2	
43 NIP calls the volume attribute RIM, IEAVNP15/ IEAVAP00, to initialize the unit control blocks (UCBs) with information about direct access data set volumes.	IEAVAP00	
44 IEAVNIPM calls the VSAM RIM, IEAVNP1B, to close the system catalog. The VSAM RIM deletes the control blocks that IEAVNP1A constructed and the address of the ACB is removed from the AMCBS, thus closing the system catalog. After system initialization is complete, the first reference to a cataloged data set will cause the system catalog to be opened.	IEAVNP1B	

Diagram 10. NIP Control and Service Routine (IEAVNIPM) Part 25 of 44

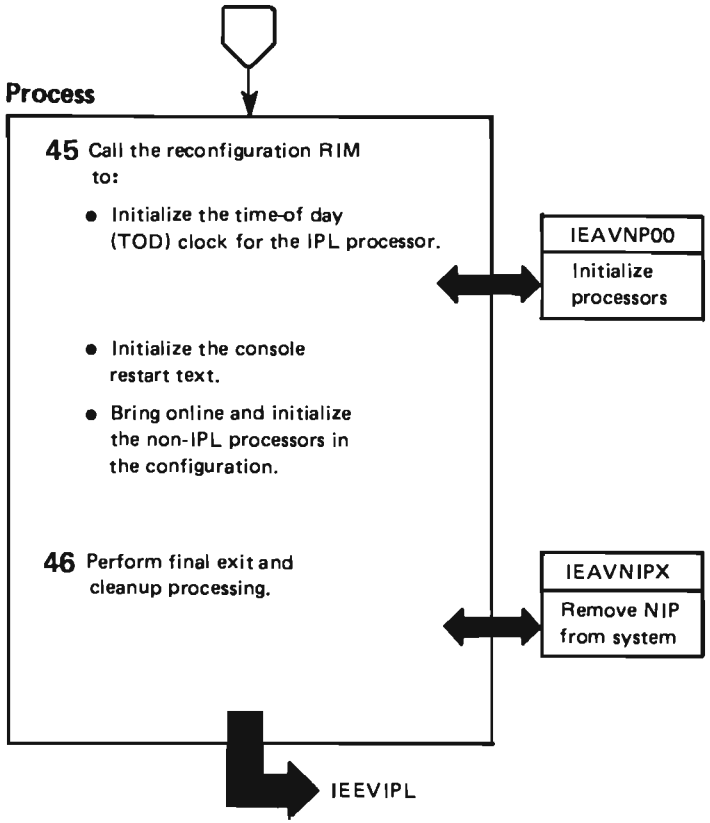


Diagram 10. NIP Control and Service Routine (IEAVNIPM) Part 26 of 44

Extended Description	Module
45 NIP calls the reconfiguration RIM, IEAVNP00 which initialized the IPL processor and brings the non-IPL processors online.	IEAVNP00
46 NIP calls IEAVNIPX to perform final exit and cleanup processing before passing control to master scheduler initialization.	IEAVNIPX

Diagram 10. NIP Control and Service Routine (IEAVNIPM) Part 27 of 44

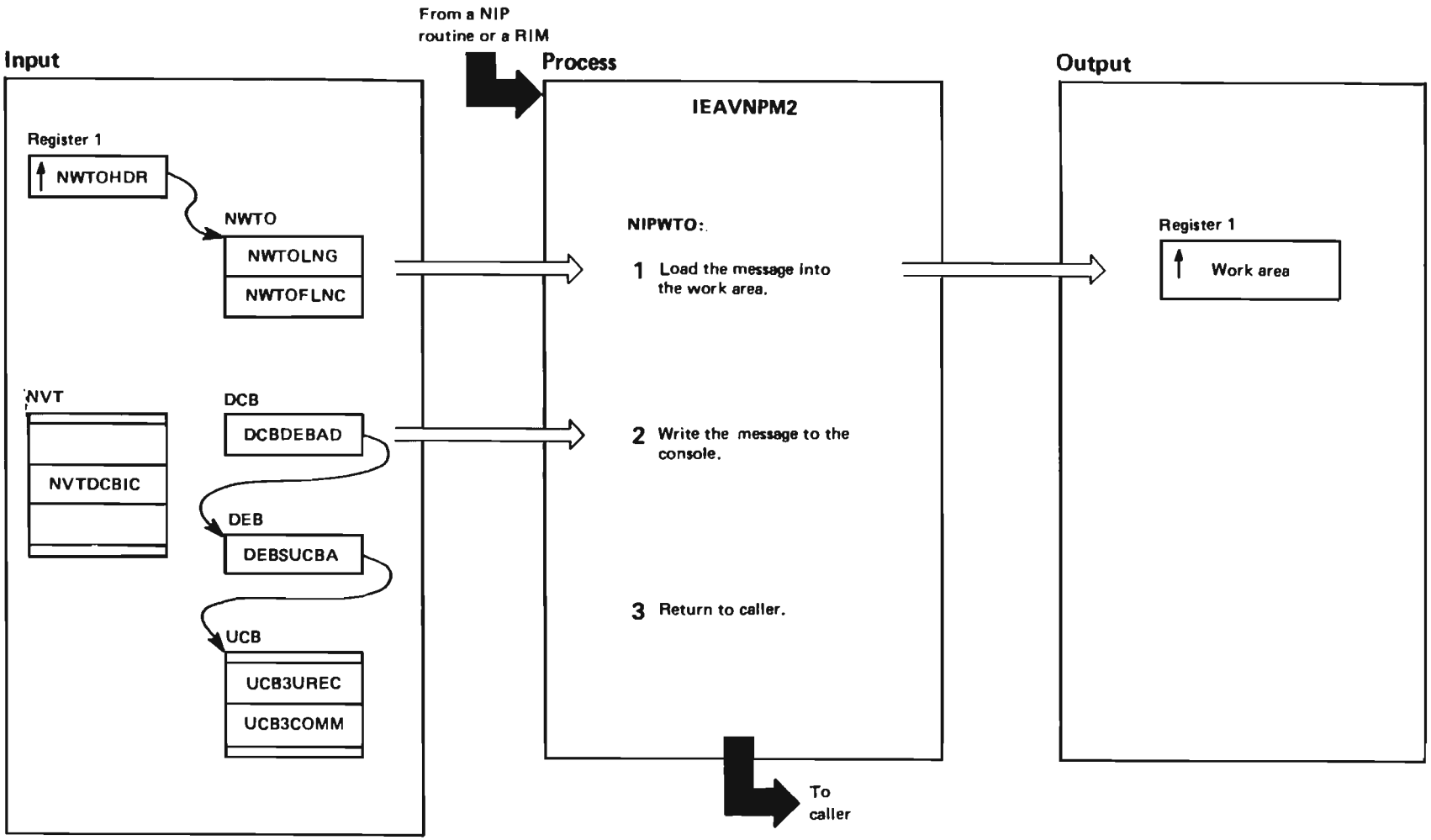


Diagram 10. NIP Control and Service Routine (IEAVNIPM) Part 28 of 44

Extended Description	Module	Label
IEAVNPM2 contains the NIPWTO, NIPWTOR, and the NIPWTOR2 service routines. These service routines execute I/O to any supported console device. They can be used by any NIP module.		
NIPWTO: NIPWTO writes to the NIP console, and, from NIPWTOR as a special case, issues a WTOR message to the NIP console. This routine requires, as input, a pointer in the parameter register to the NIP message header, immediately followed by the NIP message.	IEAVNPM2	NIPWTO
1 NIPWTO calculates the length of the message passed to it in the NIPWTO message header and loads the message into a temporary work area buffer. At this time, the message is compressed to eliminate extra blanks unless the caller has specifically requested "no compress". NIPWTO then substitutes the address of this work area for that of the NIPWTO message header in the parameter register.		
2 NIPWTO finds the UCB address from the input DEB. NIPWTO tests the NVT composite console flag, NVTFLIOC, to determine the source of the UCB address. If the flag is 1, the UCB address is found in the output DEB; otherwise, it is found in the input DEB. NIPWTO can determine the type of console device from the UCB. NIPWTO determines the appropriate channel program to write the message, or whether a display screen must be updated. If a display console is active, NIPWTO determines which type of console has been initialized and writes the message to the screen.		
3 NIPWTO returns to the caller.		

Diagram 10. NIP Control and Service Routine (IEAVNIPM) Part 29 of 44

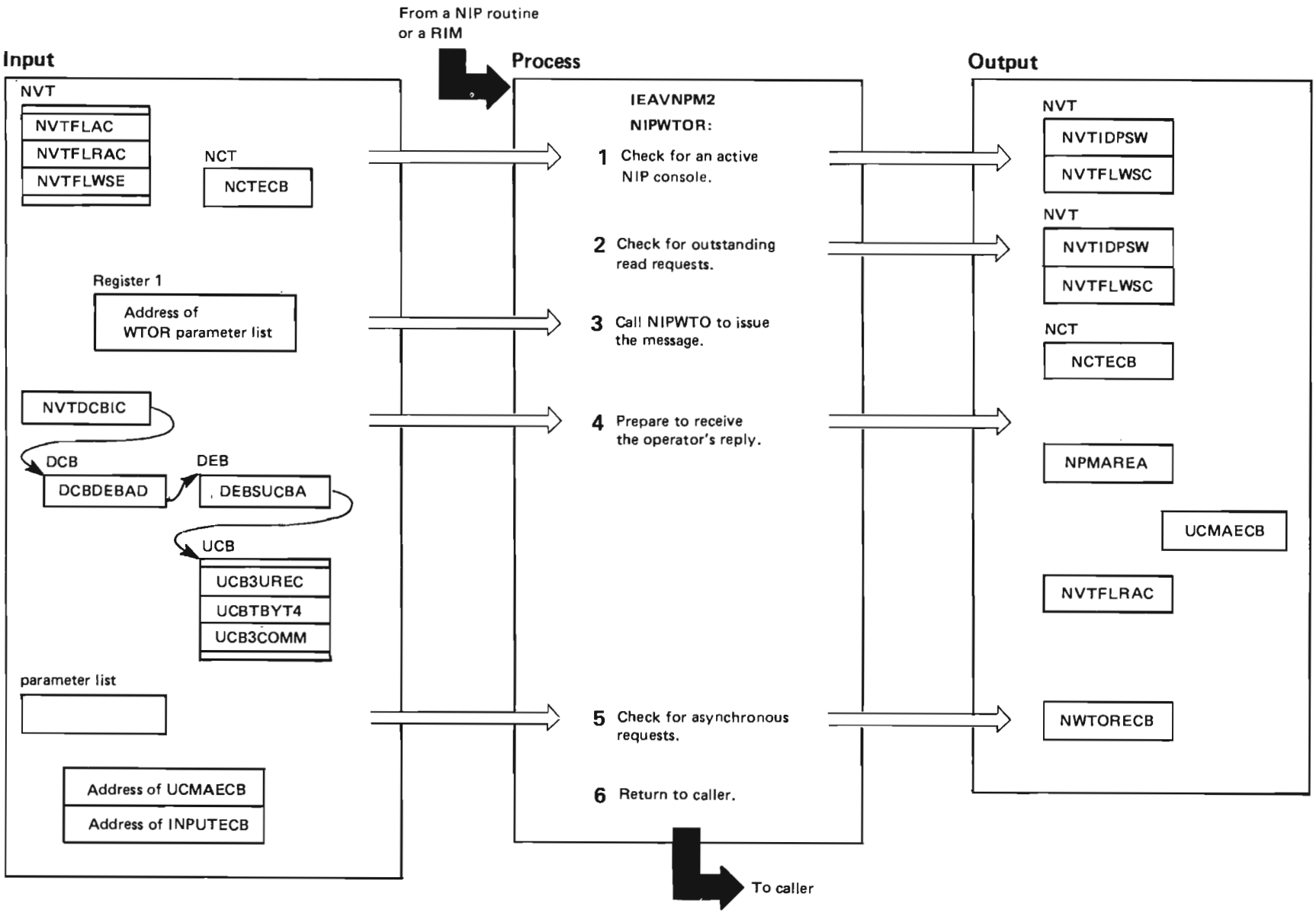


Diagram 10. NIP Control and Service Routine (IEAVNIPM) Part 30 of 44

Extended Description	Module	Label
NIPWTOR: NIPWTOR writes NIP messages to the master console, initiates the read I/O for the related operator's reply, and optionally, enters the NIPWTOR2 routine to await completion of the operator's reply. This routine requires, as input, a pointer in the parameter register to the NIP WTOR parameter list.		NIPWTOR
1 NIPWTOR tests to ensure that there is an active NIP console. If there is none, NIPWTOR enters a disabled wait state X'03F', which indicates a system error has been diagnosed.		
2 NIP then tests for outstanding read requests. If there is an outstanding request, NIPWTOR enters the X'03F' disabled wait state.		
3 NIPWTOR loads the message address from the WTOR parameter list. If the address is not zero, NIPWTO is called to issue the message. This message text is prefixed with a special character to indicate that action has been taken.		
4 NIPWTOR then prepares to receive the operator's reply. First NIPWTOR clears the reply area to blanks, then determines the device type, through the UCB, to set up the proper channel program for the read operation. Finally, NIPWTOR sets the field, NVTFLRAC, to indicate that a WTOR is outstanding.		
5 NIPWTOR checks the field, NWTORFLA, in the parameter list to determine if this is an asynchronous request. If it is, NIPWTOR calls NIPWTOR2 to wait for the operator's reply.		
6 NIPWTOR returns to the caller.		

Diagram 10. NIP Control and Service Routine (IEAVNIPM) Part 31 of 44

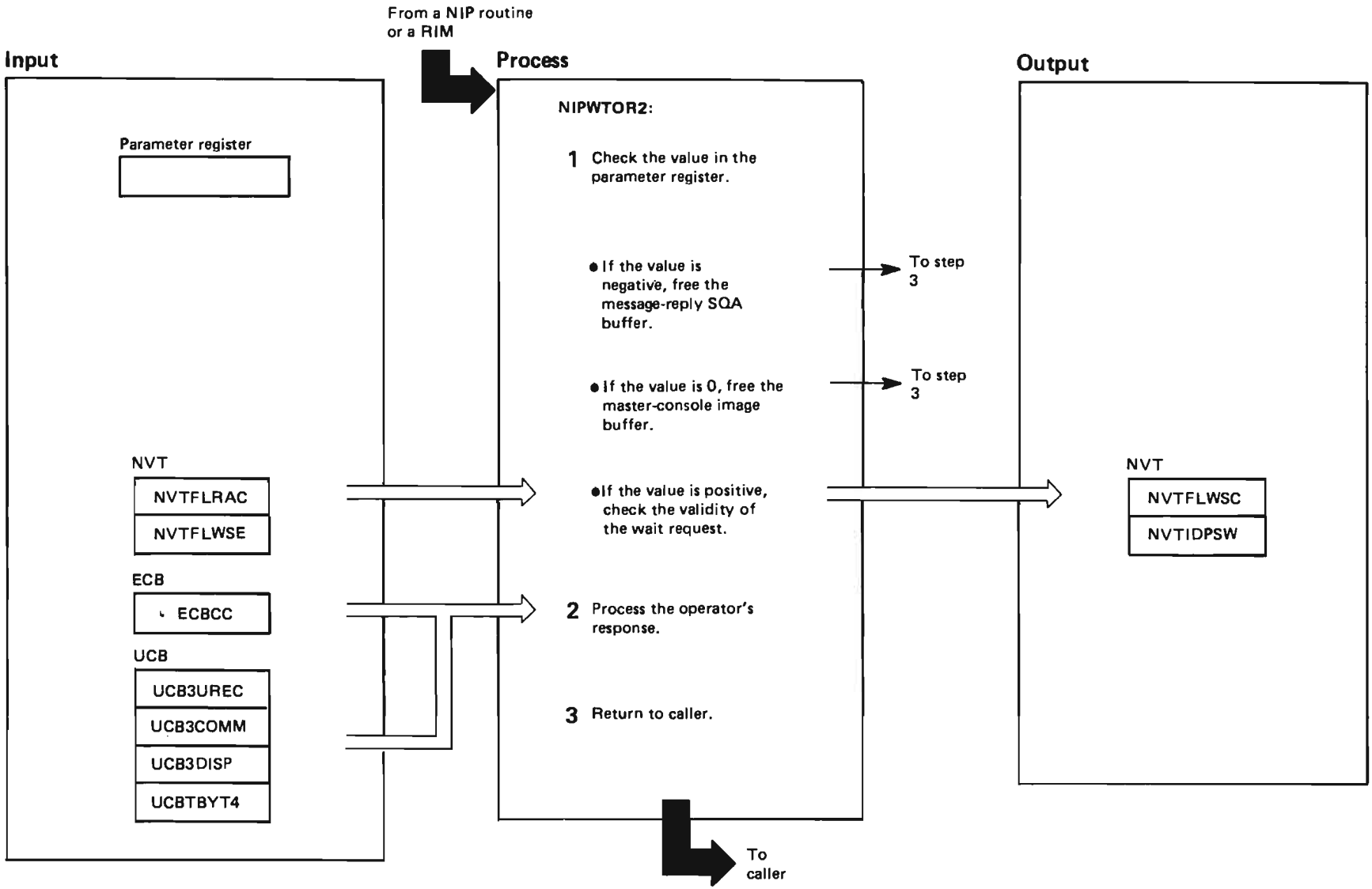


Diagram 10. NIP Control and Service Routine (IEAVNIPM) Part 32 of 44

Extended Description	Module	Label
NIPWTOR2: NIPWTOR2 performs one of the three unique services depending on the value found in the parameter register.		NIPWTOR2

- If the value is negative, NIPWTOR 2 frees a message-reply SQA buffer.
- If the value in the parameter register is zero, NIPWTOR2 frees the master-console-image buffer.
- If the value is positive, NIPWTOR2 waits for an operator's reply before it completes processing.

1 NIPWTOR2 checks the value in the parameter register.

- If the value is negative, it is the two's complement of the address of the SQA buffer to be freed. NIPWTOR2 issues a FREEMAIN macro instruction to free the storage occupied by the buffer.
- If the value is zero, NIPWTOR2 loads the address of the master-console-image buffer into register 1 and issues a FREEMAIN macro instruction to free the storage occupied by the buffer.
- If the value is positive, it is the address of the NIPWTOR parameter list associated with the operator's reply that NIPWTOR must complete. NIPWTOR checks the validity of the wait request. If there are no outstanding read requests, the wait is invalid, and NIPWTOR enters a X'03F' disabled wait state. If the wait returns with an unsuccessful return code in the ECB, NIPWTOR2 enters a X'021' disabled wait state, which indicates an I/O error on the system console.

2 NIPWTOR2 then checks for the device type and checks the operator's response for syntax validity. If the response is valid, NIPWTOR2 moves the response into NIP's hardcopy message buffer in SQA, and into the caller's reply buffer.

3 NIP returns to the caller.

Diagram 10. NIP Control and Service Routine (IEAVNIPM) Part 33 of 44

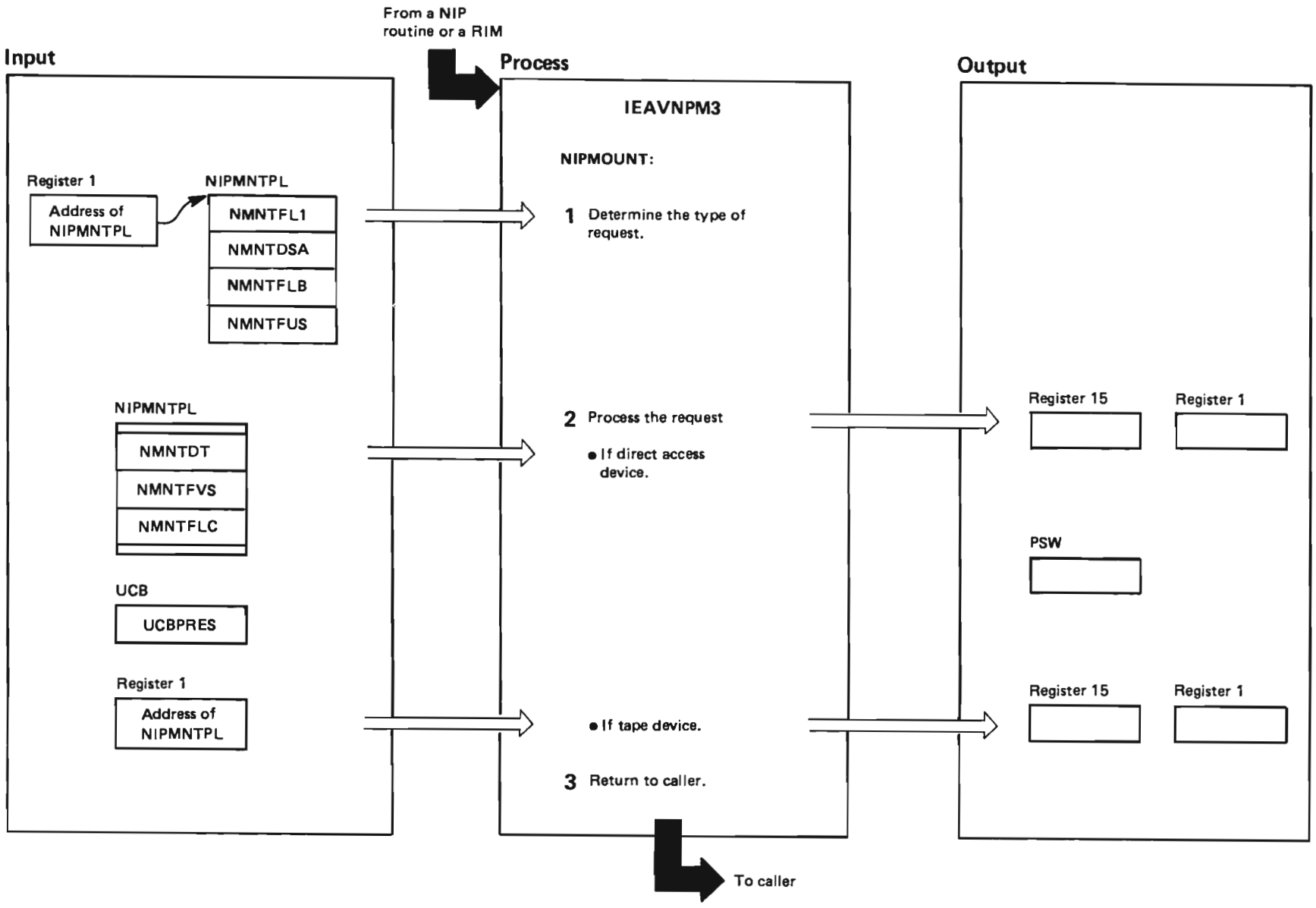
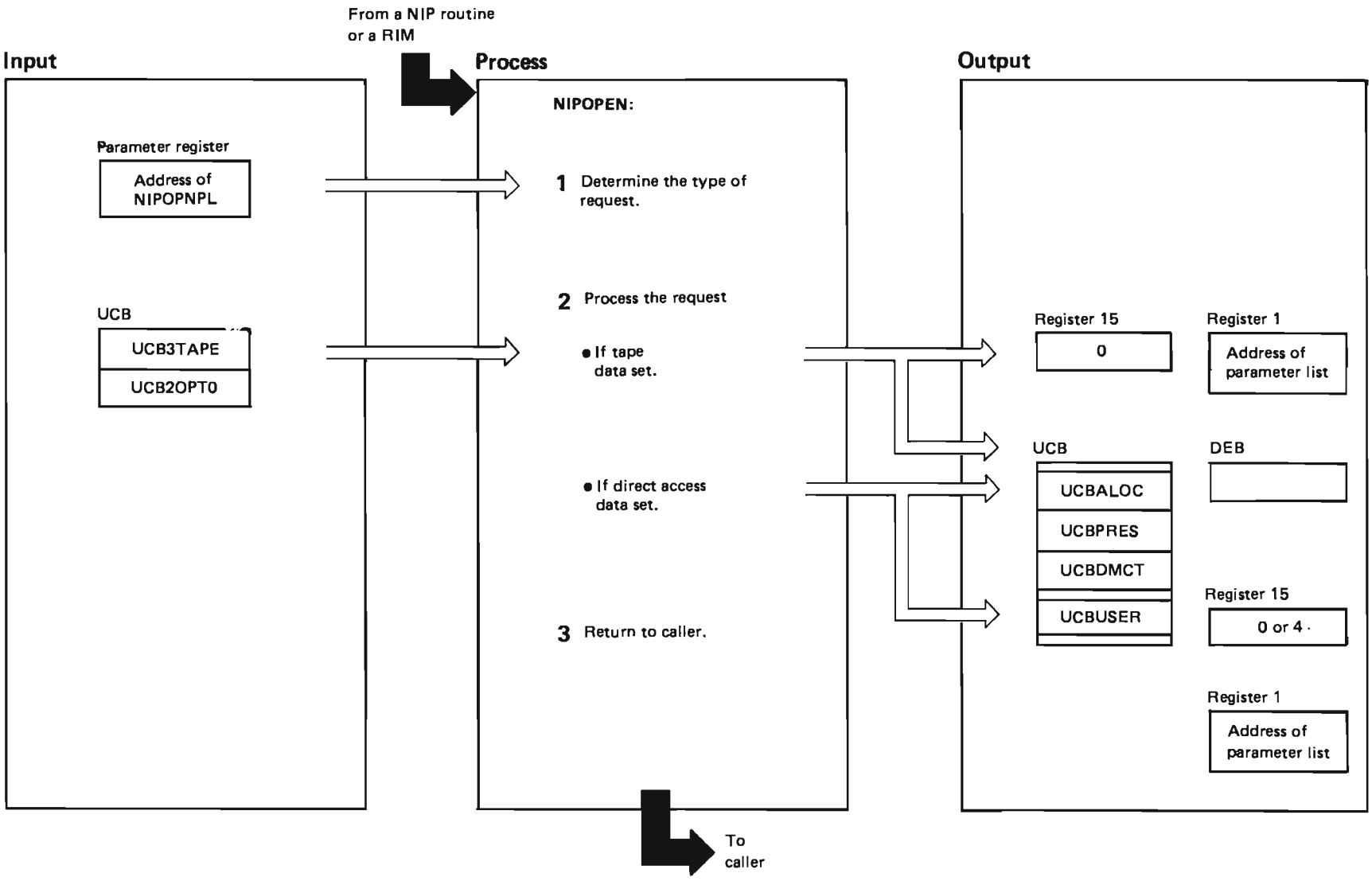


Diagram 10. NIP Control and Service Routine (IEAVNIPM) Part 34 of 44

Extended Description	Module	Label
IEAVNPM3 processes requests to mount direct access devices and tape devices, and opens direct access data sets and tape data sets. This service is used by any NIP module.		
NIPMOUNT:		
NIPMOUNT mounts direct access devices and magnetic tape volumes.	IEAVNPM3	NIPMOUNT
1 NIPMOUNT determines from the information passed to it in the parameter list whether the request is for a direct access device or a tape device.		
2 NIPMOUNT processes the requests.		
• NIPMOUNT processes the request to mount a direct access device as follows:		
If the device type is correct and the volume is accessible, the operator mounts the specified volume and NIPMOUNT sets a return code of zero in register 15 and the UCB address of the unit in register 1.		
If the requested volume is already mounted, but the device type is wrong for the specified UCB address, and the parameter list indicates to scan the UCBs only, NIPMOUNT sets a return code of four in register 15, and places zeroes in register 1. These registers indicate that the volume could not be found.		
If the requested volume is already mounted, but the device type is wrong for the specified UCB address, NIPMOUNT issues message IEA314I to inform the operator of the conflict.		
If the operator cannot remove the volume from the device because the UCB for the volume is marked as permanently resident, NIPMOUNT sets a return code of 4 in register 15 to cancel the request. If the request cannot be cancelled because the NMNTFLC flag is set in the parameter list, NIPMOUNT enters a disabled wait state X'039'.		
If the volume can be removed, NIPMOUNT issues message IEA312I to request that the operator vary the conflicting unit offline. NIPMOUNT then issues message IEA317A to request that the operator respecify the unit address on which the volume can be mounted, or to cancel the request. If the operator's response is not valid, NIPMOUNT issues message IEA318I to inform the operator that the response is unacceptable. It continues to reprompt the operator with message IEA317A until a valid response is received.		
If the operator's response is a valid address, NIPMOUNT issues message IEA315A requesting that the operator mount the specified unit. If the specified unit is mounted, NIPMOUNT must read the volume label. If NIPMOUNT encounters I/O errors reading the volume label, it issues message IEA311I to indicate that the device was unlabled, or message IEA307I to indicate that an error was encountered while reading the volume label. If an error messages is issued or the unit specified by the operator is not correct, NIPMOUNT issues message IEA316A to request that the operator demount the specified unit, and reprompts the operator with message IEA315A. Once the correct unit is mounted, NIPMOUNT sets a return code of zero in register 15 and the UCB address of the unit in register 1.		
• NIPMOUNT processes the request to mount a tape device as follows:		
If the specified unit is already allocated, NIPMOUNT sets a return code of four in register 15, and returns zeroes in register 1 to indicate that it cannot be mounted.		
If the unit is not currently used, NIPMOUNT issues message IEA315A requesting that the operator mount the specified unit. If an I/O error is detected, NIPMOUNT issues message IEA306I via the NIPSENSE service to inform the operator, then unloads the tape, and issues message IEA316A to demount the unit. NIPMOUNT then reprompts the operator with message IEA315A. Once a unit zero in register 15, and places the UCB address of the unit in register 1.		
3 NIPMOUNT returns to the caller.		

Diagram 10. NIP Control and Service Routine (IEAVNIPM) Part 35 of 44



Extended Description	Module	Label
NIPOPEN: This entry point is used by any NIP module that requires that a data set on a direct access device or a magnetic tape be opened. **1** NIPOPEN determines, from the information passed to it in the parameter list, whether the request is to open a data set on a direct access device or a tape device, and if the data set is permanently resident or not. The DEB that must be built to open a data set must reside in SQA for a permanently resident data set, and in the private area for a data set that is not permanently resident. **2** NIPOPEN processes the request. • If the request is for a data set on tape, NIPOPEN builds and initializes a DEB for the data set. If the data set is permanently resident, NIPOPEN initializes fields in the UCB to indicate that the tape is allocated and permanently resident. NIPOPEN sets return code of zero in register 15 and puts the parameter list address in register 1. • If the request is for a data set on a direct access device, NIPOPEN reads the format 1 DSCB that describes this data set. If the format 1 DSCB was found, but it indicates that the data set has additional extents, NIPOPEN reads the format 3 DSCB. If the format 1 DSCB could not be found, NIPOPEN issues message IEA319I indicating that the DSCB was not found. NIPOPEN sets a return code of 4 in register 15 and puts the two's complement address of the parameter list in register 1. If an I/O error occurred while reading either DSCB, NIPOPEN issues messages IEA211I and IEA306I to inform the operator. NIPOPEN enters a disabled wait state X'037' if the DSCB was not read and the request is conditional.		NIPOPEN
If the DSCB was read successfully, NIPOPEN builds and initializes a DEB to describe the data set. If the DEB is being expanded for a new concatenation, NIPOPEN checks for room in the DEB for more extents. If the DEB is full, NIPOPEN returns to its caller with return code of X'0C' in register 15 and puts the parameter list address in register 1. If the data set is permanently resident, NIPOPEN initializes fields in the UCB to indicate that the device is allocated and permanently resident. NIPOPEN sets the return code of zero in register 15 and puts the parameter list address in register 1. **3** NIPOPEN returns to the caller. Register 1 is unchanged if the data set was successfully opened.		

Diagram 10. NIP Control and Service Routine (IEAVNIPM) Part 37 of 44

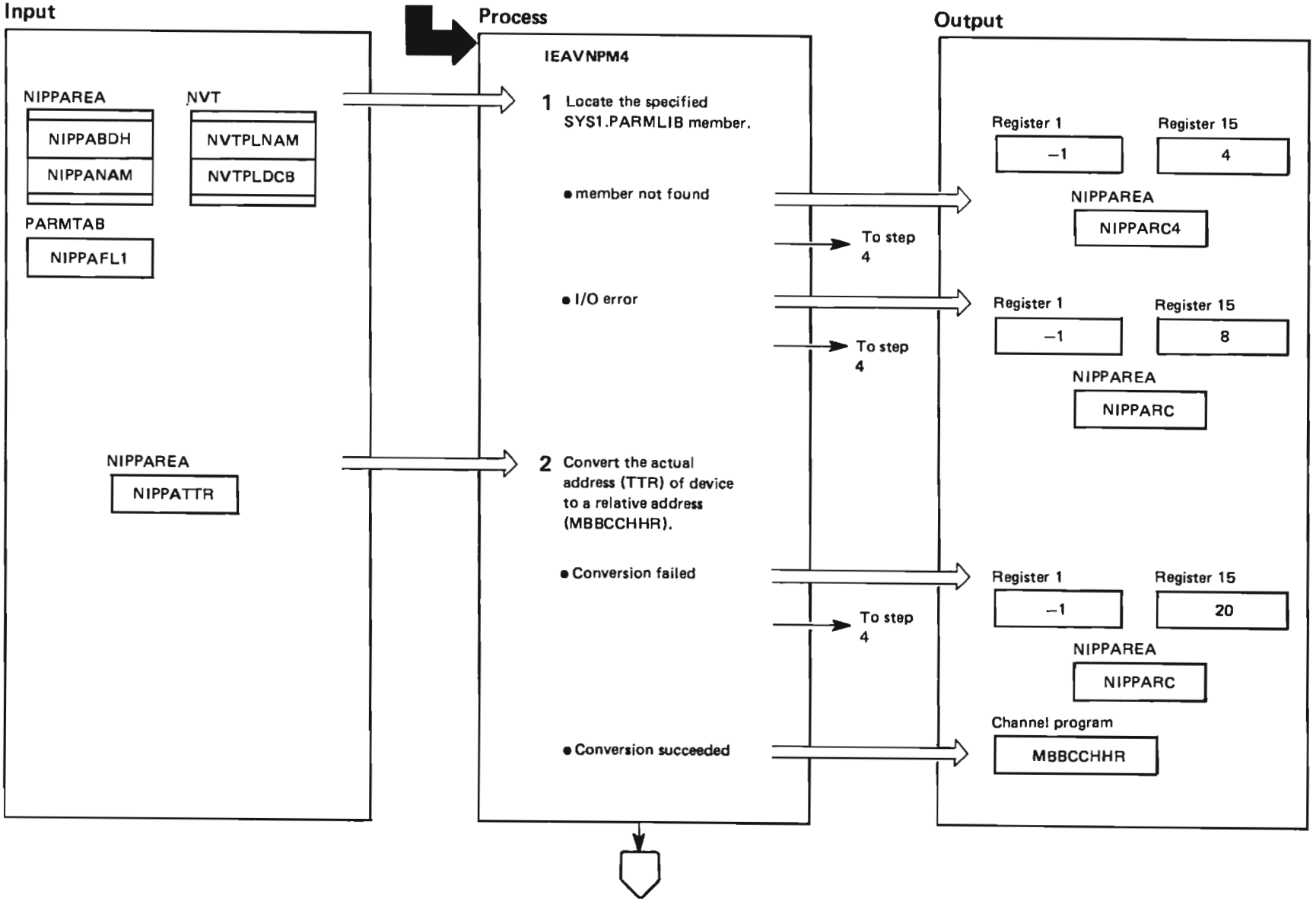


Diagram 10. NIP Control and Service Routine (IEAVNIPM) Part 38 of 44

Extended Description	Module	Label
IEAVNPM4 is called by any NIP module to read a member of SYS1.PARMLIB.		
1 IEAVNPM4 receives from the caller the name of the member of SYS1.PARMLIB to be read. IEAVNPM4 compares this member name to the name (saved in NVTPLNAM) of the last member processed or to zeroes if this is the first member. When a new member of SYS1.PARMLIB is specified, IEAVNPM4 issues the BLDL macro instruction to locate the TTR address of the member.	IEAVNPM4	
• If the BLDL could not locate the specified member and the suppress flag, NIPPAFL1, is set to 0, IEAVNPM4 issues messge IEA301I to inform the operator that it could not find the member. If NIPPAFL1 is set to 1, IEAVNPM4 will not issue the message. IEAVNPM4 sets a return code of 4 and continues with Step 4.		
• If the BLDL macro instruction encounters an I/O error, IEAVNPM4 issues message IEA300I to the operator. IEAVNPM4 sets a return code of 8 and continues with Step 4.		
2 IEAVNPM4 calls IECPCNVT to convert the actual (TTR) address returned by BLDL to a relative (MBBCCHHR) address.	IECPCNVT	
• If the conversion fails, IEAVNPM4 sets a return code of 20.	IEAVNPM4	
• If the conversion is successful, IEAVNPM4 puts the converted address into the channel program that will read the parmlib member's text records.		

Diagram 10. NIP Control and Service Routine (IEAVNIPM) Part 39 of 44

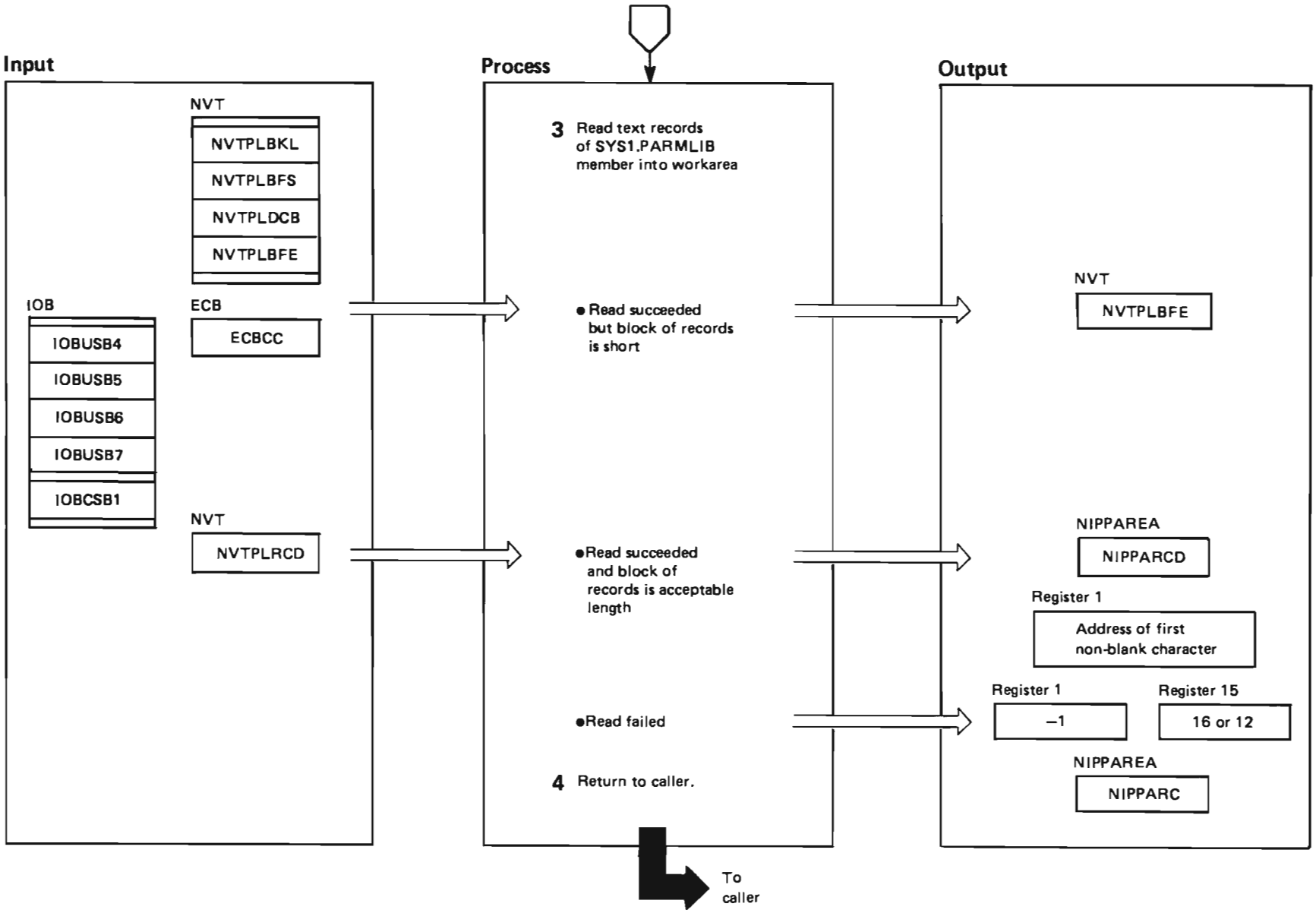


Diagram 10. NIP Control and Service Routine (IEAVNIPM) Part 40 of 44

Extended Description	Module	Label

3 IEAVNPM4 reads a block of records when a new parmlib member has been specified or when the buffer for the current member is empty.

IEAVNPM4 issues the EXCP macro instruction to read text records of a parmlib member into a buffer workarea it sets up.

- If the EXCP is successful, but a short block of records is read and that block is not a multiple of the logical record length of 80, IEAVNPM4 issues message IEA334A to inform the oeprator that the record length is invalid. IEAVNPM4 then waits for the operator's response. If the operator indicates that processing should continue, IEAVNPM4 rounds the invalid block size up to a multiple of 80 and adjusts the field NVTPLBFE to point to the end of the short block.

- If the EXCP is successful and the block is a multiple of the logical record length of 80, IEAVNPM4 moves the records from the buffer into the IEAVNPM4 parameter area and then scans the records for validity. When columns 1-71 are blank, but column 72 is non-blank, IEAVNPM4 automatically reads or scans a new record. If the record is valid, IEAVNPM4 sets column 72 to a blank, column 73 to a null (X'FF') character,. and register 1 to the address of the first non-blank character. When columns 1-72 are all blank, register 1 contains the address of column 72.

- If the EXCP fails, IEAVNPM4 determines what type of I/O error occurred. If the error is end-of-file, IEAVNPM4 sets a return code of 16. Otherwise, IEAVNPM4 sets a return code of 12 and issues message IEA306I via the NIPSENSE service to inform the operator of the error.

4 IEAVNPM4 returns to the caller with a return code in field NIPPARC of the IEAVNPM4 parameter area, and in register 15. If the return code is non-zero, IEAVNPM4 sets register 1 to –1.

Diagram 10. NIP Control and Service Routine (IEAVNIPM) Part 41 of 44

IEAVNPM5 - MODULE DESCRIPTION

DESCRIPTIVE NAME: NIP Console Attention Routine

FUNCTION:
This module receives control from IOS when IOS has received an attention. This module will post the NCT ECB whenever the NIP console issued the attention. Otherwise the attention is ignored.

ENTRY POINT: IEAVNPM5

PURPOSE: See function.

LINKAGE: Call from IOS.

CALLERS: IOS.

INPUT:
Register 1 will contain the address of the IOSB for the UCB which issued the attention.

OUTPUT: NIP ECB posted.

EXIT NORMAL: Return to caller via register 14.

OUTPUT: See output above.

EXTERNAL REFERENCES:

ROUTINES: None

CONTROL BLOCKS:
IOSB pointed to by Register 1.

IOSB - pointed to by Register 1.
CVT - Communications Vector Table
NVT - NIP Vector Table
NCT - NIP Console Table

Diagram 10. NIP Control and Service Routine (IEAVNIPM) Part 42 of 44

IEAVNPM5 - MODULE OPERATION

This module runs in SRB mode and therefore cannot issue any SVC's. By using the STACK procedure option, the compiler generates the entry and exit linkage with the specified overrides for the getmain and freemain macro. These macros will be locally coded to use the branch entry form of the macro. The post macro will also use branch entry.

Because this routine resides in the IEAVNIPM load module, it will run in 24-bit mode. However, IOS issues a BASSM to call this routine, so we must return in the mode of the caller by specifying AMODE(24,*) in the procedure options.

. If the attention came from the NIP console:

  - POST the NCT ECB.
  - Set the attention pending switch in the NCT on.

. Otherwise, do nothing.

Diagram 10. NIP Control and Service Routine (IEAVNIPM) Part 43 of 44

IEAVNPM5 - DIAGNOSTIC AIDS

ENTRY POINT NAME: IEAVNPM5

MESSAGES: None

ABEND CODES: None

WAIT STATE CODES: None

RETURN CODES: None

REGISTER CONTENTS ON ENTRY:

0 to 13 - Irrelevant
14 - Return address
15 - Entry point address

REGISTER CONTENTS ON EXIT: Irrelevant

Diagram 10. NIP Control and Service Routine (IEAVNIPM) Part 44 of 44

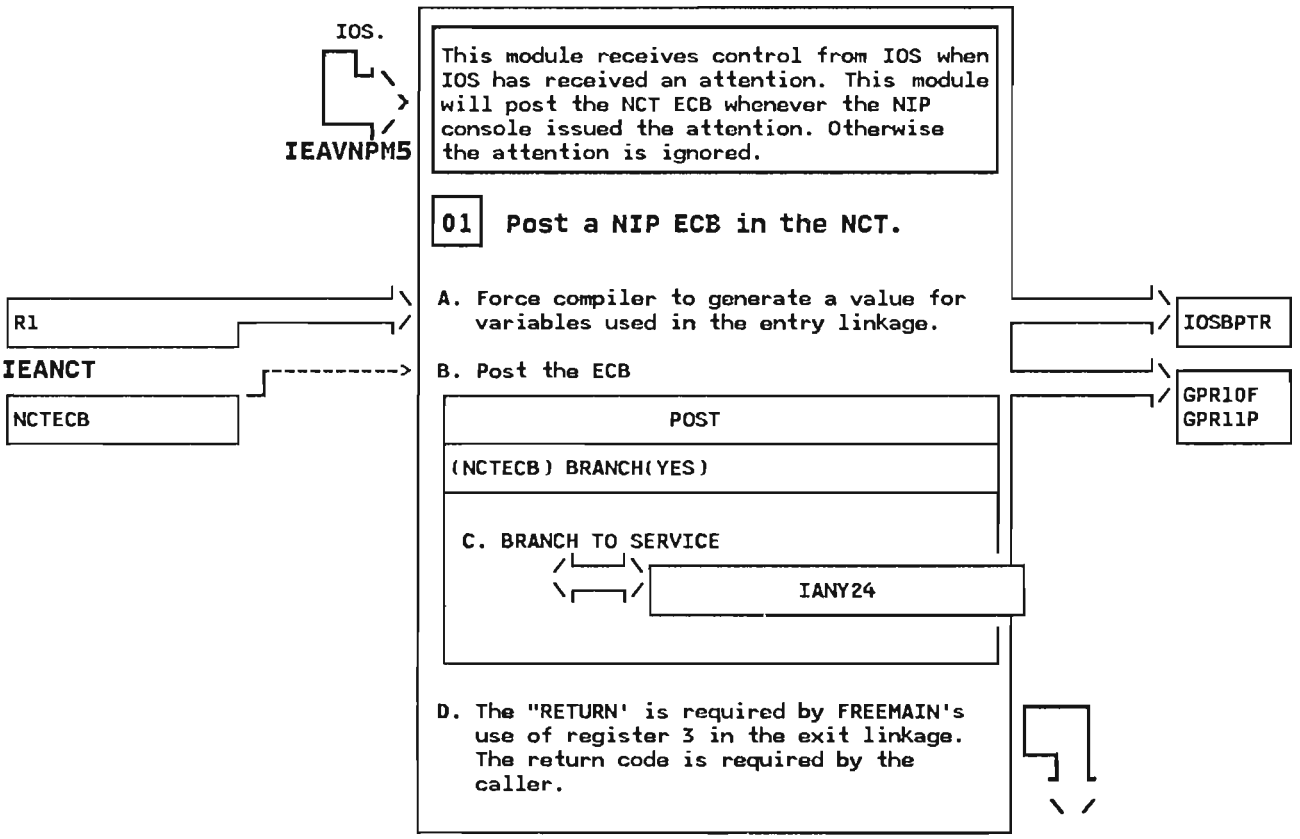


Diagram 11. Service Processor CALL SVC Initialization (IEAVNPE6) Part 1 of 2

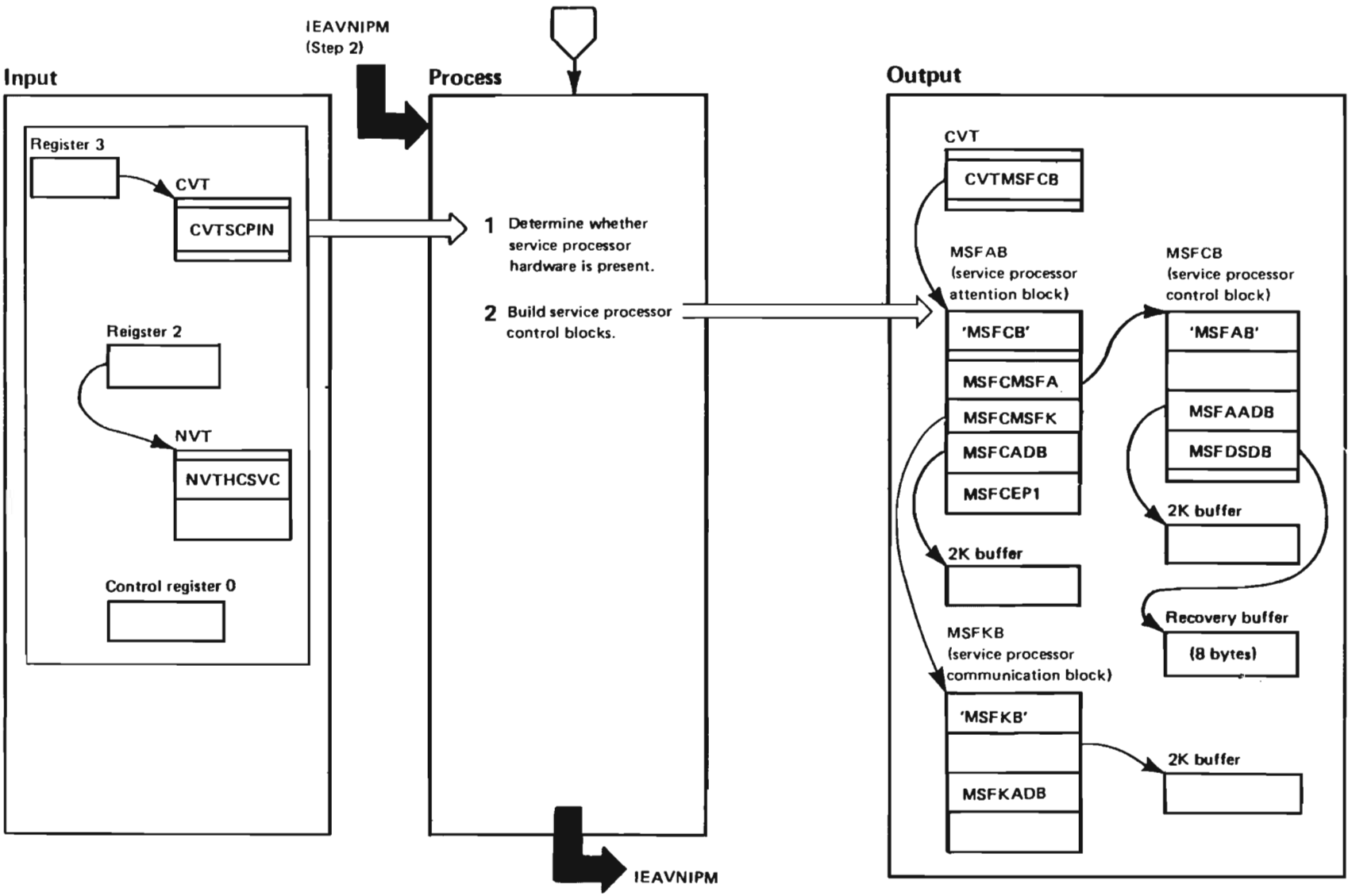


Diagram 11. Service Processor CALL SVC Initialization (IEAVNPE6) Part 2 of 2

Extended Description	Module	Label
IEAVNIPM calls the service processor RIM, IEAVNPE6, to build the control blocks and buffers required by the service processor CALL SVC, which MVS users use to communicate with the hardware's service processor. (See ***System Logic Library,*** for a description of service processor CALL SVC processing.)		
1 IEAVNPE6 first checks CVTSCPIN, which is set by NIP, to determine if service processor hardware is present. If service processor hardware is not present, IEAVNPE6 sets the pointer to the service processor CALL control blocks (the CVTMSFCB field) to zero indicating that the service processor CALL SVC is not supported, and returns to IEAVNIPM.	IEAVNPE6	
2 If service processor hardware is present, IEAVNPE6 builds the control blocks as follows:		
• Obtains storage for the control blocks and buffers shown in the output section of the diagram. Because the service processor buffers must be on 2K boundaries, IEAVNPE6 uses a GETMAIN macro instruction to acquire three 4K blocks from SQA and thus force page boundaries.		
• Chains the control blocks together.		
• Places the service processor CALL SVC branch entry point in the MSFCEP1 field.		
• Places the address of the MSFCB in the CVTMSFCB field.		
• Sets bit 22 of control register 0 to '1' to ensure that the processor is enabled for service processor interrupts. When type 2 extended SVC initialization is complete, the service processor CALL SVC can execute successfully.		
• Frees the unused part of the 12K of storage obtained at entry.		

Diagram 12. System Recovery Initialization (IEAVNPA6) Part 1 of 3

IEAVNPA6 - MODULE DESCRIPTION

DESCRIPTIVE NAME: RTM's early NIP initialization (RIM module)

FUNCTION:
This module is the resource initialization module (RIM) that is given control during nucleus initialization to allocate and to initialize permanent data areas used by the recovery termination manager (RTM). IEAVNPA6 aquires storage for the following data areas and initializes them:

- the RTM control table (RTCT)
- the RTM quick cell pool
- the recording buffers control block (RBCB)
- the recording buffers (RCBs)

All storage for these data areas is obtained from the system queue area (SQA), subpools 239 and 245.

ENTRY POINT: IEAVNPA6

PURPOSE: See function

LINKAGE: Load and BALR

CALLERS: IEAVNIPM

INPUT:
NIP Vector Table (NVT)
Communications Vector Table (CVT)

OUTPUT: Initialized control blocks

EXIT NORMAL: Returns to the caller

EXTERNAL REFERENCES:

ROUTINES:
GETMAIN - Allocate storage service
CPOOL - Build a cell pool for EEDs and SRBs

CONTROL BLOCKS:

Common name	macro id	usage	function
CVT	CVT	read, write	Establishes addressability to the RTCT.
PSA	IHAPSA	read	Establishes addressability to the CVT.
RBCB	RTMRBCB	create, read, write	Recording buffers control block
RCB	RTMRCB	create, write	Contains recording request entries.
RTCT	IHARTCT	create, write	Recovery termination control table
SRB	IHASRB	create	Service request block

Diagram 12. System Recovery Initialization (IEAVNPA6) Part 2 of 3

IEAVNPA6 - MODULE OPERATION

IEAVNPA6 receives control during nucleus initialization to allocate and to initialize the major control blocks used by RTM. IEAVNPA6 receives control from IEAVNIPM and performs the following processing:

- Obtains storage for the RTCT from subpool 245 (below the 16M line). Clears the RTCT to zero and inserts the control block id. SDUMPs are inhibited by setting the appropriate SVC dump option flag in the RTCT, and use of SDUMP 4K buffer (which has not yet been acquired) is inhibited by indicating, in the CVT, that the buffer is in use.

- Builds a cell pool for the EEDs and SRBs in subpool 239 (below the 16M line) (via the CPOOL macro).

- Obtains storage for the RBCB from subpool 239 (above the 16M line). Initializes the RBCB.

- Obtains storage for the RCBs from subpool 239 (above the 16M line). Initializes the RCBs.

Diagram 12. System Recovery Initialization (IEAVNPA6) Part 3 of 3

IEAVNPA6 - DIAGNOSTIC AIDS

ENTRY POINT NAME: IEAVNPA6

MESSAGES: None

ABEND CODES: None

WAIT STATE CODES: None

RETURN CODES:

EXIT NORMAL:

0 in register 15

REGISTER CONTENTS ON ENTRY:

```
Registers   0-1 - Irrelevant
Register      2 - Address of the NIP vector table (NVT)
Register      3 - Address of the CVT
Registers  4-12 - Irrelevant
Register     13 - Address of a 72 byte register save area
Register     14 - Return address
Register     15 - Entry Point address
```

REGISTER CONTENTS ON EXIT: Irrelevant

Diagram 13. Machine Check Handler Initialization (IEAVNP06) Part 1 of 4

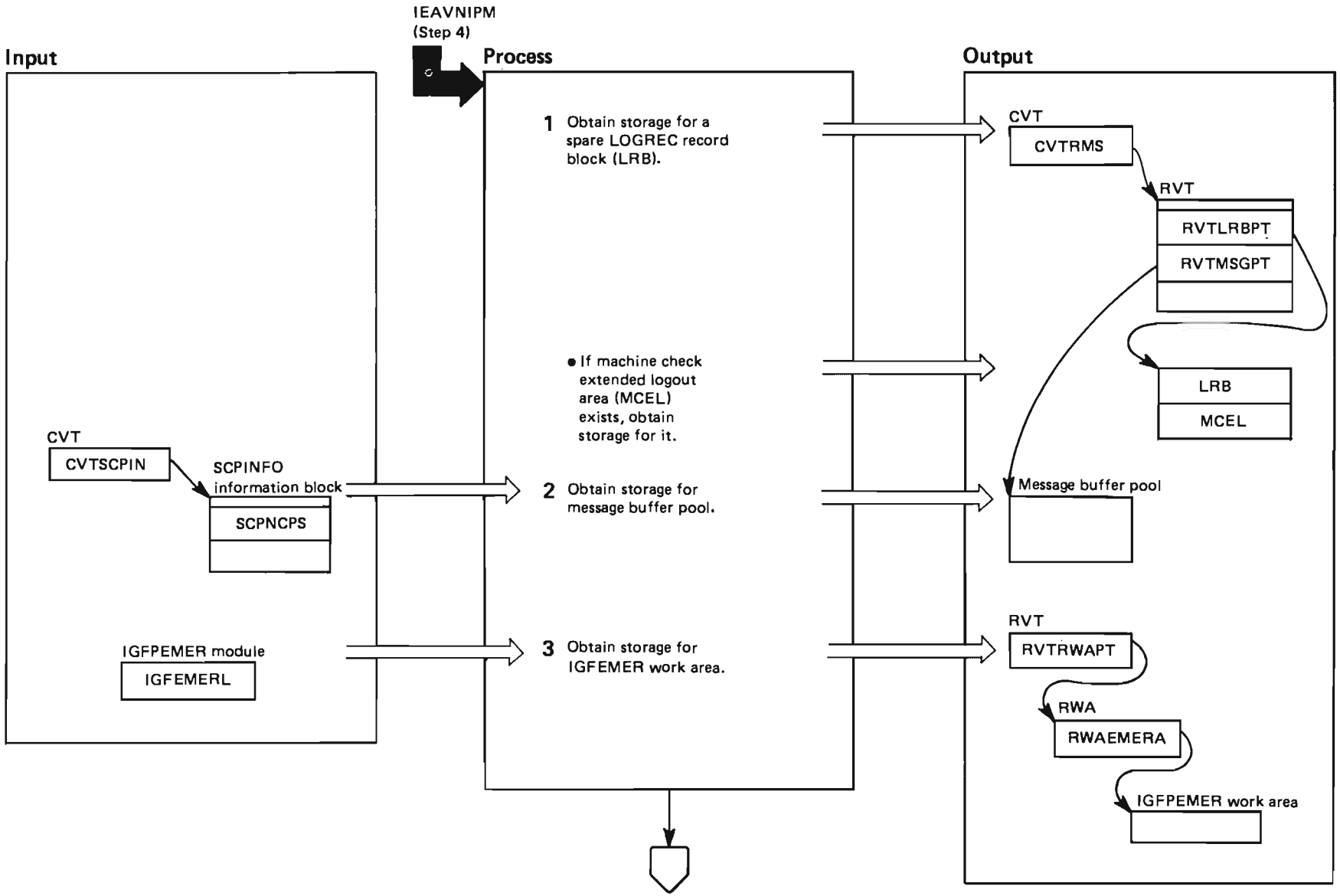


Diagram 13. Machine Check Handler Initialization (IEAVNP06) Part 2 of 4

Extended Description	Module	Label
IEAVNIPM calls the machine check handler (MCH) RIM, IEAVNP06, to initialize the machine check handler control blocks for the IPL processor. IEAVNP00 will perform this task for the non-IPL processor.		
1 The MCH RIM issues a GETMAIN macro instruction to obtain storage from subpool 239 for the spare LOGREC record block (LRB) and the machine check extended logout area (MCEL) which immediately follows the LRB, if there is a MCEL. The MCH RIM sets RVTLRBPT to the start address of the LRB. The spare LRB is used by IGFPMRTM during normal machine check processing.	IEAVNP06	IEAVNP06
If the GETMAIN macro instruction returns a non-zero return code, the MCH RIM uses the NIP macro IEAPMNIP TYPE=SWAIT to issue a wait state code of X'0E8' with a reason code of 83rr, where rr is the return code from the GETMAIN macro instruction.		ABEND
2 The MCH RIM issues a GETMAIN macro instruction to obtain storage from subpool 239 for the message buffer pool. There are 8 message buffers for each processor in the system. SCPNCPS gives the number of processors in this system. The last message buffer is marked in use so that alternate CPU recovery (ACR) routine IEAVTACR, is certain to have a buffer for the 'ACR COMPLETE' message, IEA858I.		IEAVNP06
If the GETMAIN macro returns a non-zero return code, the MCH RIM issues a wait state code of X'0E8' with a reason code of 83rr, where rr is the return code from the GETMAIN macro instruction.		ABEND
3 The MCH RIM issues a GETMAIN macro instruction to obtain storage from subpool 239 for the work area that will be used by IGFPEMER during abnormal system termination. IGFEMERL is an external field located in IGFPEMER; it contains the size of the automatic data area for IGFPEMER during abnormal system termination. The MCH RIM initializes RWAEMERA with the start address of the workarea.		IEAVNP06
If the GETMAIN macro returns a non-zero return code, the MCH RIM issues a wait state code of X'0E8' with a reason code of 83rr, where rr is the return code from the GETMAIN macro instruction.		ABEND

Diagram 13. Machine Check Handler Initialization (IEAVNP06) Part 3 of 4

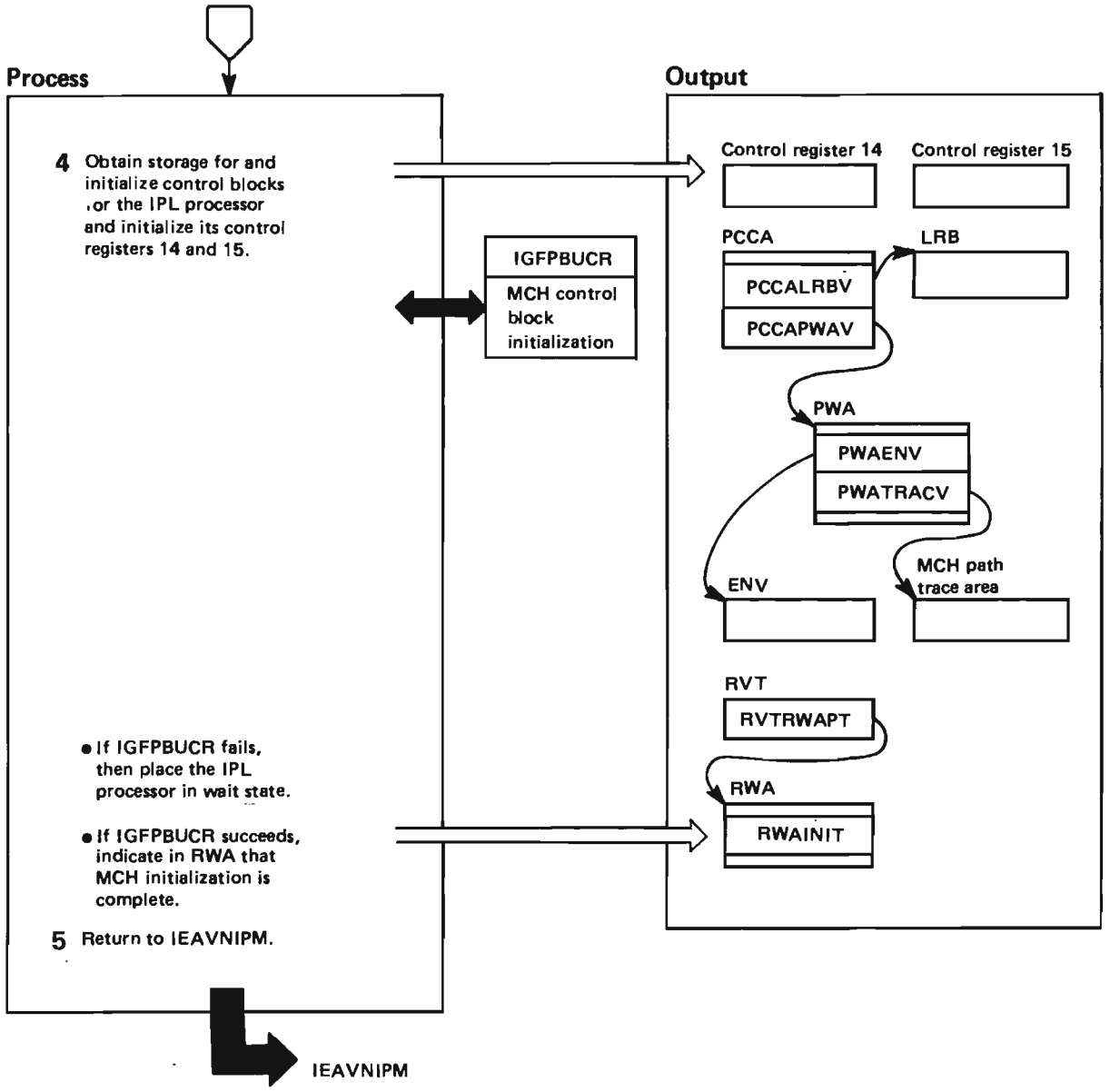


Diagram 13. Machine Check Handler Initialization (IEAVNP06) Part 4 of 4

Extended Description	Module	Label
4 The MCH RIM calls IGFPBUCR to initialize control registers 14 and 15 and the following CPU-related control blocks: the LRB, PWA, ENV, and MCH path trace area.	IGFPBUCR	
• If IGFPBUCR was not successful, the MCH RIM places the IPL processor into a wait state with a code of X'0E8'. Bytes 4 through 7 of the loaded PSW will be in the form 'xxrr00E8' where xx is the reason code and rr is the return code.	IEAVNP06	ABEND
Return Code — *Meaning* 81 — IGFPBUCR returned a non-zero return code (rr) when it was called to allocate the MCH control blocks. 82 — IGFPBUCR returned a non-zero return code (rr) when it was called to initialize the control registers.		
• If IGFPBUCR was successful, the MCH RIM indicates that MCH initialization is complete for the IPL processor by setting the bit RWAINIT to 1.		IEAVNP06
For more information on IGFPBUCR, see *System Logic Library*.		
5 The MCH RIM returns to IEAVNIPM via the BSM instruction.		

Diagram 14. Reconfiguration Installed Resources RIM (IEAVNP27) Part 1 of 20

IEAVNP27 - MODULE DESCRIPTION

DESCRIPTIVE NAME: Reconfiguration Installed Resources RIM.

FUNCTION:
Obtain and initialize the Installation Channel Path Table (ICHPT) and the Configuration Management Table (CMT).

ENTRY POINT: IEAVNP27.

PURPOSE: See FUNCTION.

LINKAGE: CALL IEAVNP27.

CALLERS: IEAVNIPM - NIP Control/Service Routines.

INPUT: See entry registers.

OUTPUT: None

EXIT NORMAL: Return to caller.

OUTPUT:
1. CVTICHPT points to the ICHPT.
2. The ICHPT is initialized.
3. CSDCMT points to the CMT.
4. The CMT is initialized.

EXIT ERROR: Disabled wait state.

OUTPUT: See Wait State Codes for diagnostic data.

EXTERNAL REFERENCES:

ROUTINES: IEEVORGI - Obtain Resource Group Information.

CONTROL BLOCKS:

Common name/use	Macro name	Description
BITMAPCB PCR D	IEEMBITM	Resource Bit Map Arrays
CHPI PCR D	IEEMCHPI	Read Channel Path Information Command Information
CMT PCRM	IEEMCMT	Configuration Management Table
CSD M	IHACSD	Common System Data Area
CVT RM	CVT	Communication Vector Table
ICHPT C M	IHAICHPT	Installation Channel Path Table
NVT RM	IHANVT	NIP Vector Table
ORGIPARM PCRMD	IEEMORGI	Obtain Resource Group Information Parameter List
SCCB CRMD	IHASCCB	Service Call Control Block
SCCBSCPI R	IHASCCB	SCP Information
SPCS PCRMD	IEEMSPCS	Service Processor Call Parameter list

C=created, R=referenced, M=modified, D=deleted,
P=private to the reconfiguration component

Diagram 14. Reconfiguration Installed Resources RIM (IEAVNP27) Part 2 of 20

IEAVNP27 - MODULE OPERATION

Obtain channel path information from the service processor
command: Read Channel Path Information.

If the service processor architecture is supported,
obtain side information for each side from
the Obtain Resource Group Information routine (IEEVORGI).

If an MSSF is installed,
obtain I/O side information by issuing the MSSF command:
I/O Side Information.

Obtain and initialize the Configuration Management Table (CMT)
with side, real storage element, and channel path information.

Obtain and initialize the Installation Channel Path Table
Table (ICHPT) with the channel path information.

Initialize pointers to the CMT and the ICHPT.

RECOVERY OPERATION: None, since IEAVNP27 executes before RTM is initialized.

Diagram 14. Reconfiguration Installed Resources RIM (IEAVNP27) Part 3 of 20

IEAVNP27 - DIAGNOSTIC AIDS

ENTRY POINT NAME: IEAVNP27.

MESSAGES: None

ABEND CODES: None

WAIT STATE CODES:

```
X'xxxxx020' - Reconfiguration Initialization failed.

X'00010020' - IEAVNP27 failed to obtain channel path
              information.
              Diagnostic information is stored in
              IEAVNP27 diagnostic save area pointed to by
              CSDCMT.
  Offset  0 - Failure type descriptor: attempt to use SVC 122
              Extended SVC Routing Code 6, 'SPCS' = X'E2D7C3E2'.
  Offset  4 - Return code from SVC 122 ESR 6.
  Offset  8 - Address of the NVT.
  Offset 12 - Address of the CVT.
  Offset 16 - Unpredictable.
  Offset 18 - Service processor response code.
  Offset 20 - Service call command word.
  Offset 24 - Address of the SCCB.
  Offset 28 - Unpredictable.
  Offset 40 - Data register.
  Offset 44 - First code register.
  Offset 48 - Second code register.
  Offset 52 - Address of standard save area.
  Offset 56 - Return address.
  Offset 60 - Address of entry point.

X'00020020' - IEAVNP27 failed to obtain I/O side
              information.
              Diagnostic information is stored in
              IEAVNP27 diagnostic save area pointed to by
              CSDCMT.
  Offset  0 - Failure type descriptor: attempt to use SVC 122
              Extended SVC Routing Code 6, 'SPCS' = X'E2D7C3E2'.
  Offset  4 - Return code from SVC 122 ESR 6.
  Offset  8 - Address of the NVT.
  Offset 12 - Address of the CVT.
  Offset 16 - Unpredictable.
  Offset 18 - Service processor response code.
  Offset 20 - Service call command word.
  Offset 24 - Address of the SCCB.
  Offset 28 - Unpredictable.
  Offset 40 - Data register.
  Offset 44 - First code register.
  Offset 48 - Second code register.
  Offset 52 - Address of standard save area.
  Offset 56 - Return address.
  Offset 60 - Address of entry point.

X'00030020' - IEAVNP27 failed to obtain side
              information.
              Diagnostic information is stored in
              IEAVNP27 diagnostic save area pointed to by
              CSDCMT.
  Offset  0 - Failure type descriptor: call IEEVORGI,
              'ORGI' = X'D6D9C7C9'
  Offset  4 - Return code from IEEVORGI.
  Offset  8 - Address of the NVT.
  Offset 12 - Address of the CVT.
```

Diagram 14. Reconfiguration Installed Resources RIM (IEAVNP27) Part 4 of 20

IEAVNP27 - DIAGNOSTIC AIDS (Continued)

```
   Offset 16 - Unpredictable.
   Offset 40 - Data register.
   Offset 44 - First code register.
   Offset 48 - Second code register.
   Offset 52 - Address of standard save area.
   Offset 56 - Return address.
   Offset 60 - Address of entry point.

 X'00040020' - IEAVNP27 failed to obtain
               the length of the side information block.
   Offset  0 - Failure type descriptor: call IEEVORGI,
               'ORGI' = X'D6D9C7C9'
   Offset  4 - Return code from IEEVORGI.
   Offset  8 - Address of the NVT.
   Offset 12 - Address of the CVT.
   Offset 16 - Unpredictable.
   Offset 40 - Data register.
   Offset 44 - First code register.
   Offset 48 - Second code register.
   Offset 52 - Address of standard save area.
   Offset 56 - Return address.
   Offset 60 - Address of entry point.
```

RETURN CODES: None

REGISTER CONTENTS ON ENTRY:

```
 GPR0-1:   Unpredictable
 GPR2:     Address of NVT
 GPR3:     Address of CVT
 GPR4-12:  Unpredictable
 GPR13:    Address of standard save area
 GPR14:    Return address
 GPR15:    Address of entry point
```

REGISTER CONTENTS ON EXIT:

```
 EXIT NORMAL:

   GPR0-15: Unchanged

 EXIT ERROR:

   GPR0-15: Unpredictable
```

Diagram 14. Reconfiguration Installed Resources RIM (IEAVNP27) Part 5 of 20

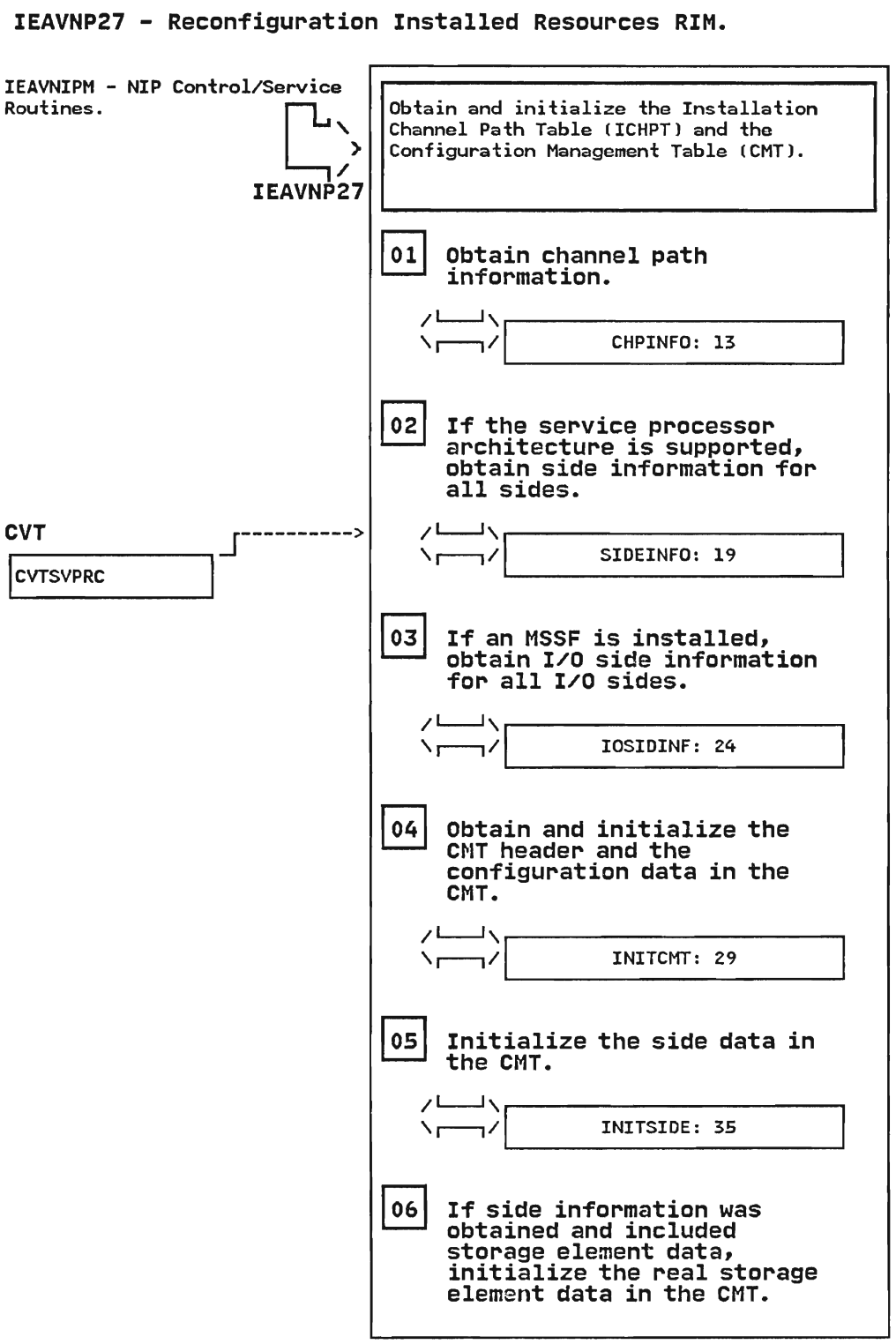


Diagram 14. Reconfiguration Installed Resources RIM (IEAVNP27) Part 6 of 20

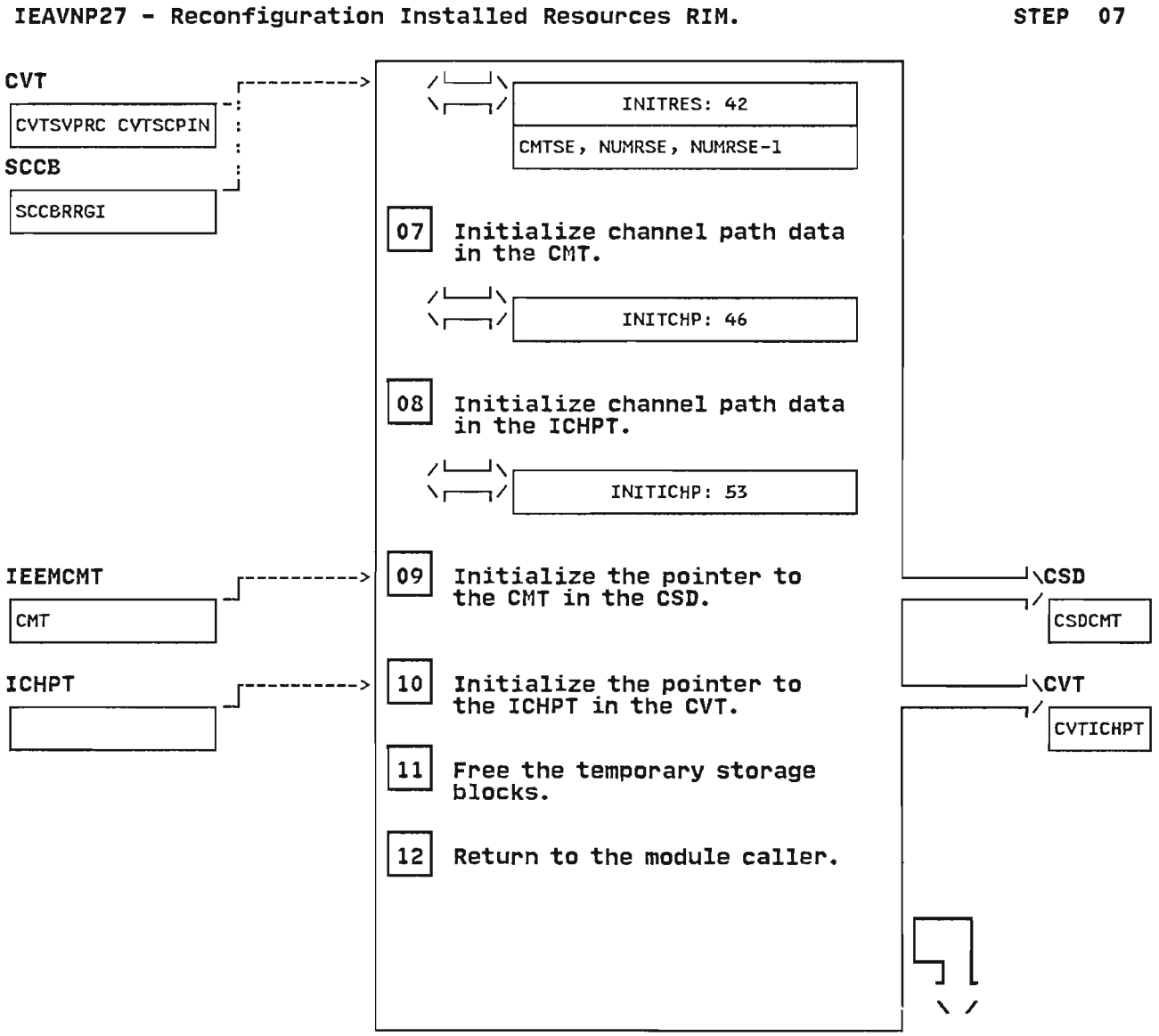


Diagram 14. Reconfiguration Installed Resources RIM (IEAVNP27) Part 7 of 20

IEAVNP27 - Reconfiguration Installed Resources RIM. STEP 13

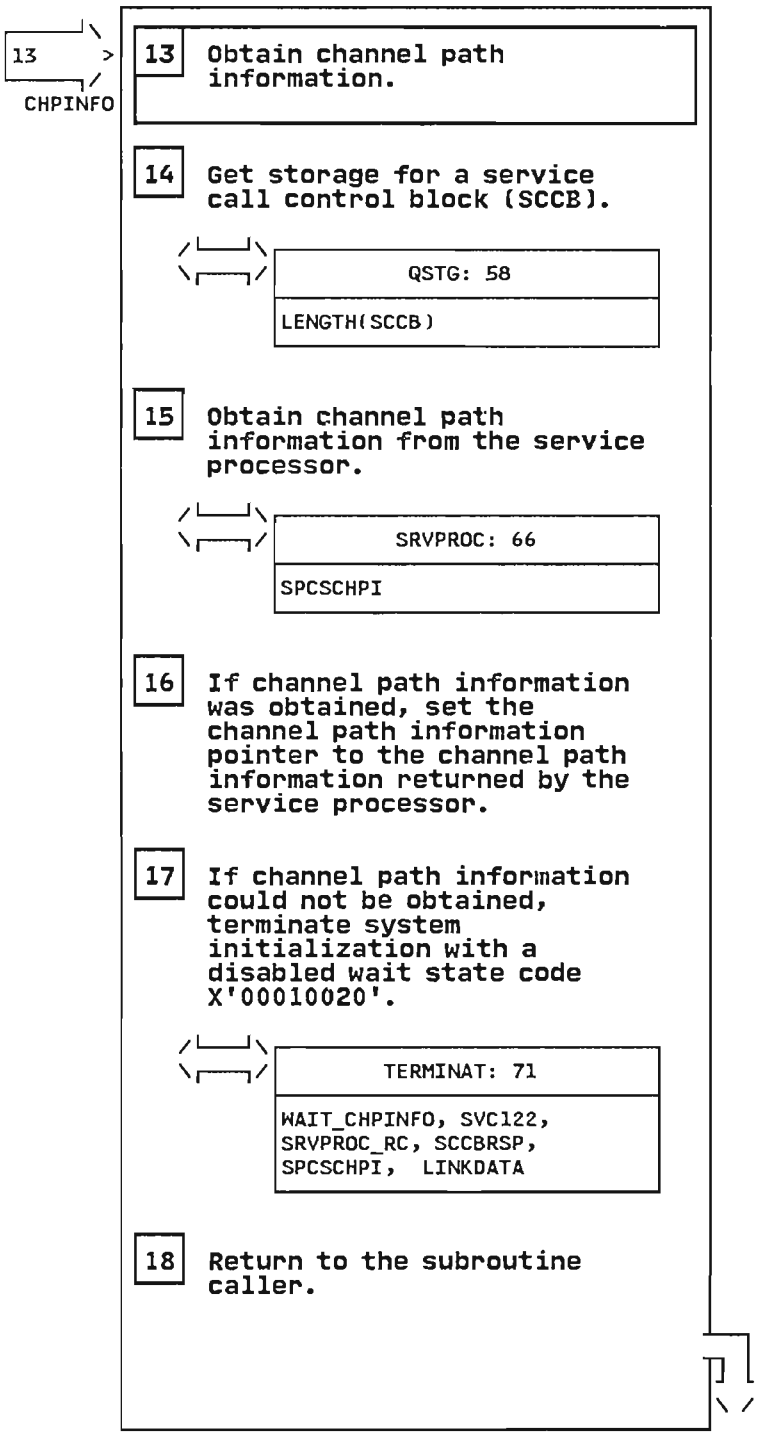


Diagram 14. Reconfiguration Installed Resources RIM (IEAVNP27) Part 8 of 20

IEAVNP27 - Reconfiguration Installed Resources RIM. STEP 19

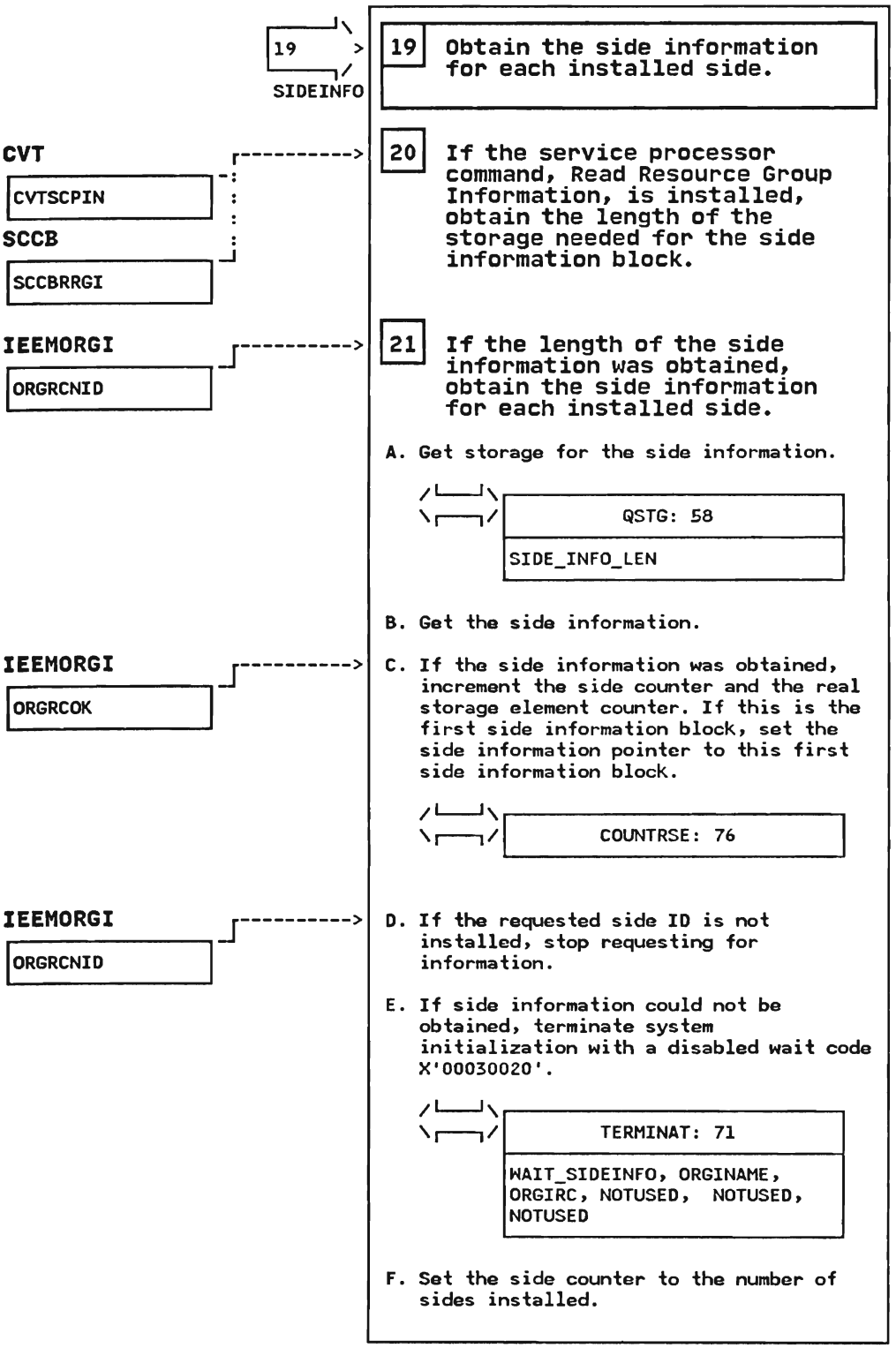


Diagram 14. Reconfiguration Installed Resources RIM (IEAVNP27) Part 9 of 20

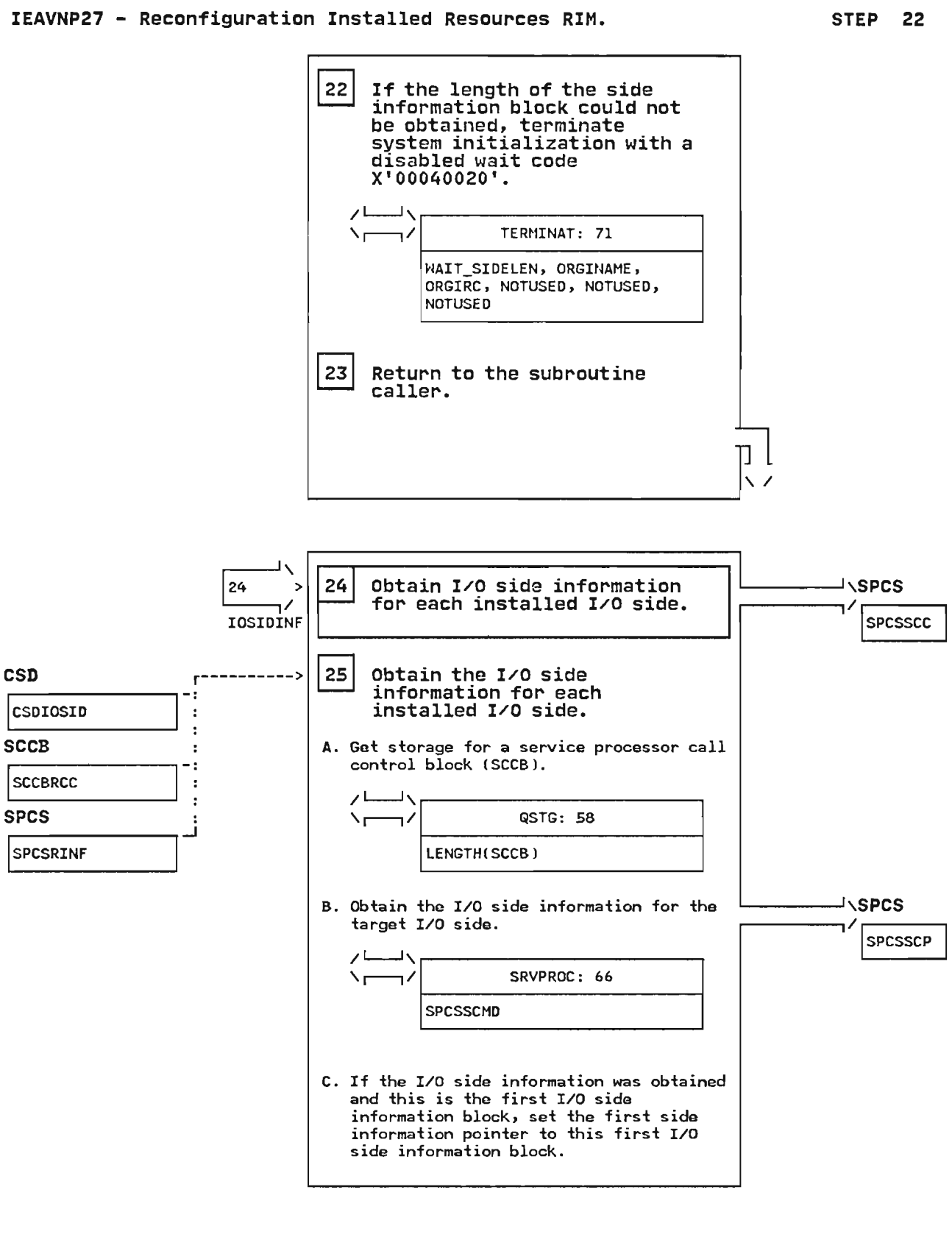


Diagram 14. Reconfiguration Installed Resources RIM (IEAVNP27) Part 10 of 20

IEAVNP27 - Reconfiguration Installed Resources RIM. STEP 25D

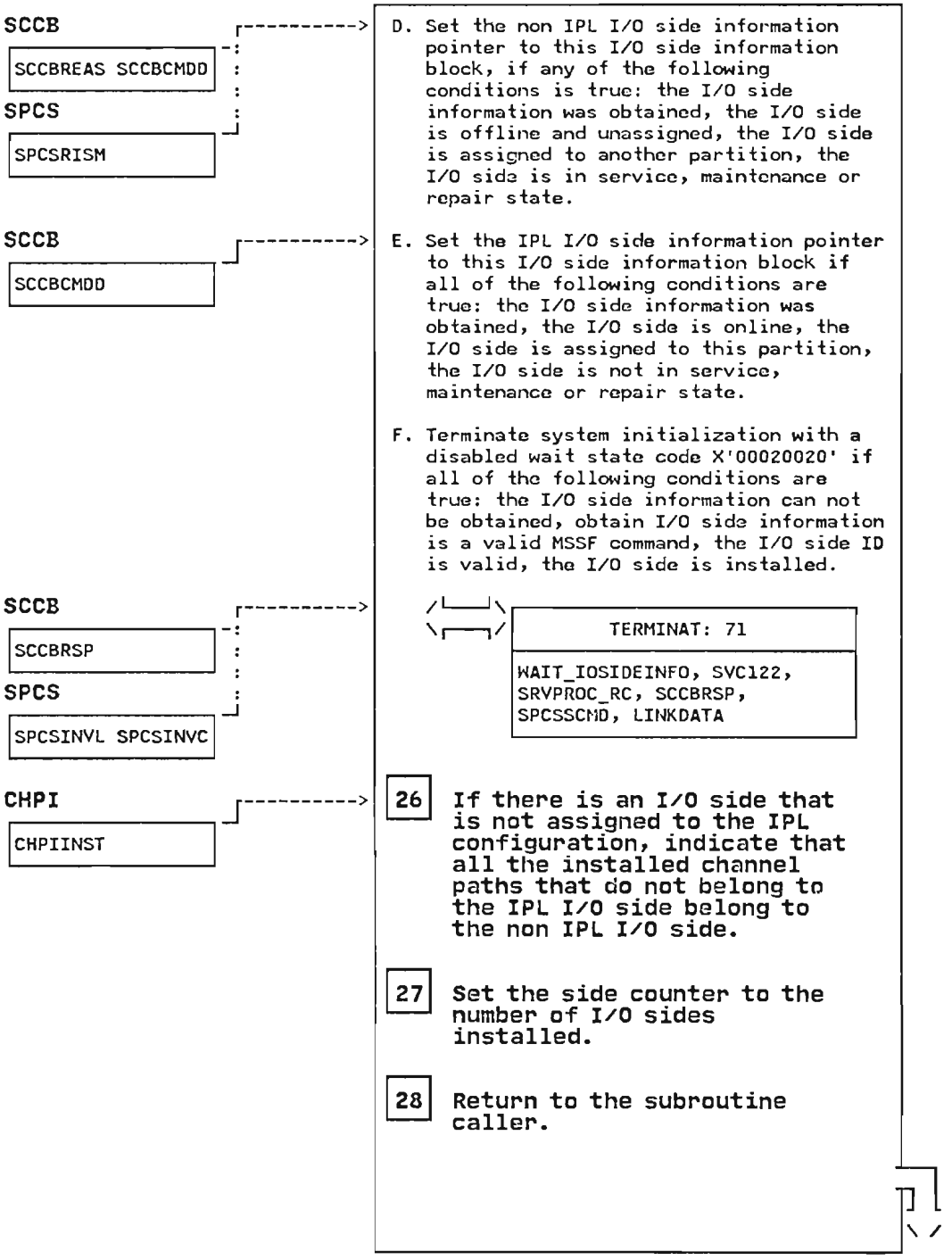


Diagram 14. Reconfiguration Installed Resources RIM (IEAVNP27) Part 11 of 20

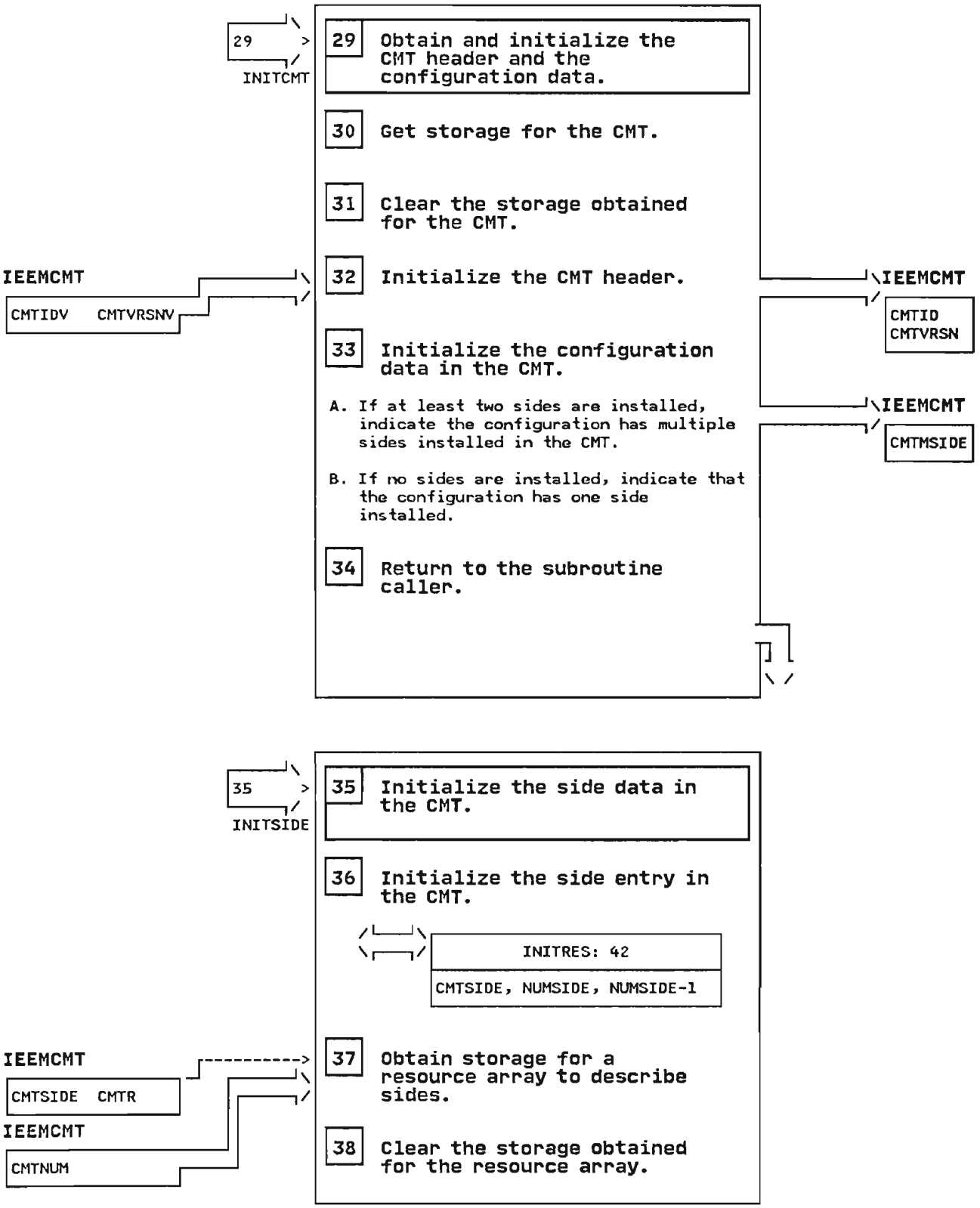


Diagram 14. Reconfiguration Installed Resources RIM (IEAVNP27) Part 12 of 20

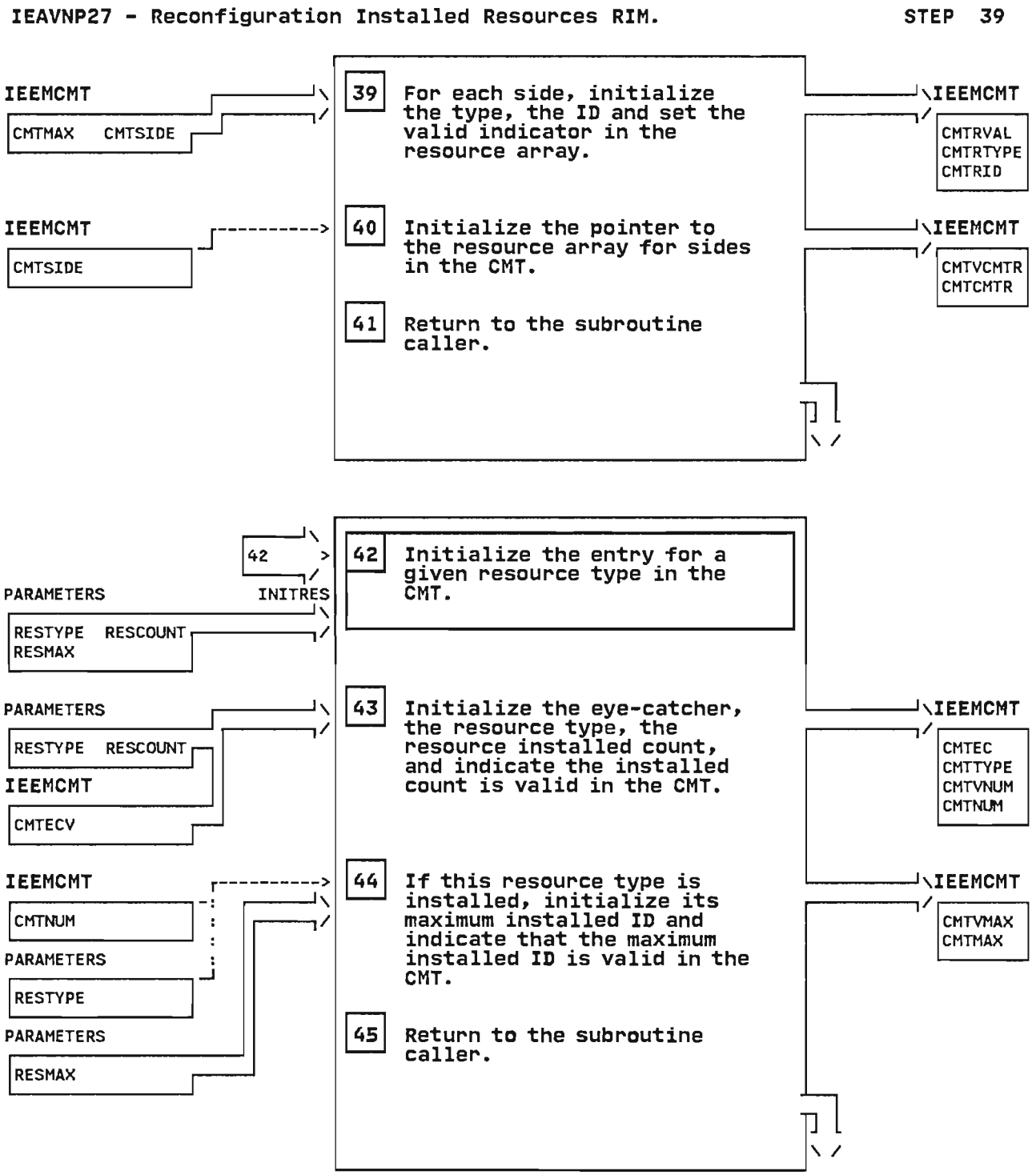


Diagram 14. Reconfiguration Installed Resources RIM (IEAVNP27) Part 13 of 20

IEAVNP27 - Reconfiguration Installed Resources RIM. STEP 46

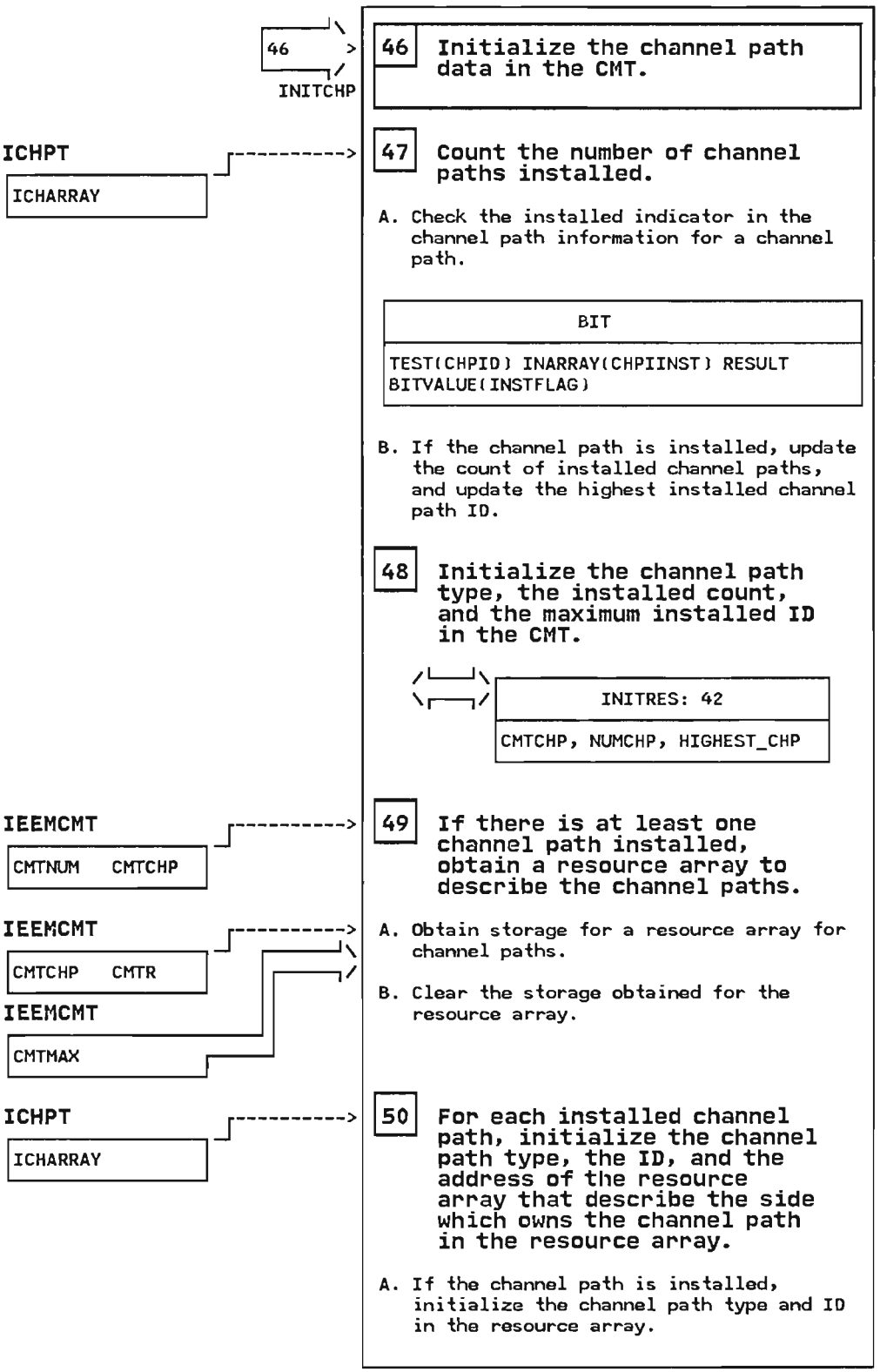


Diagram 14. Reconfiguration Installed Resources RIM (IEAVNP27) Part 14 of 20

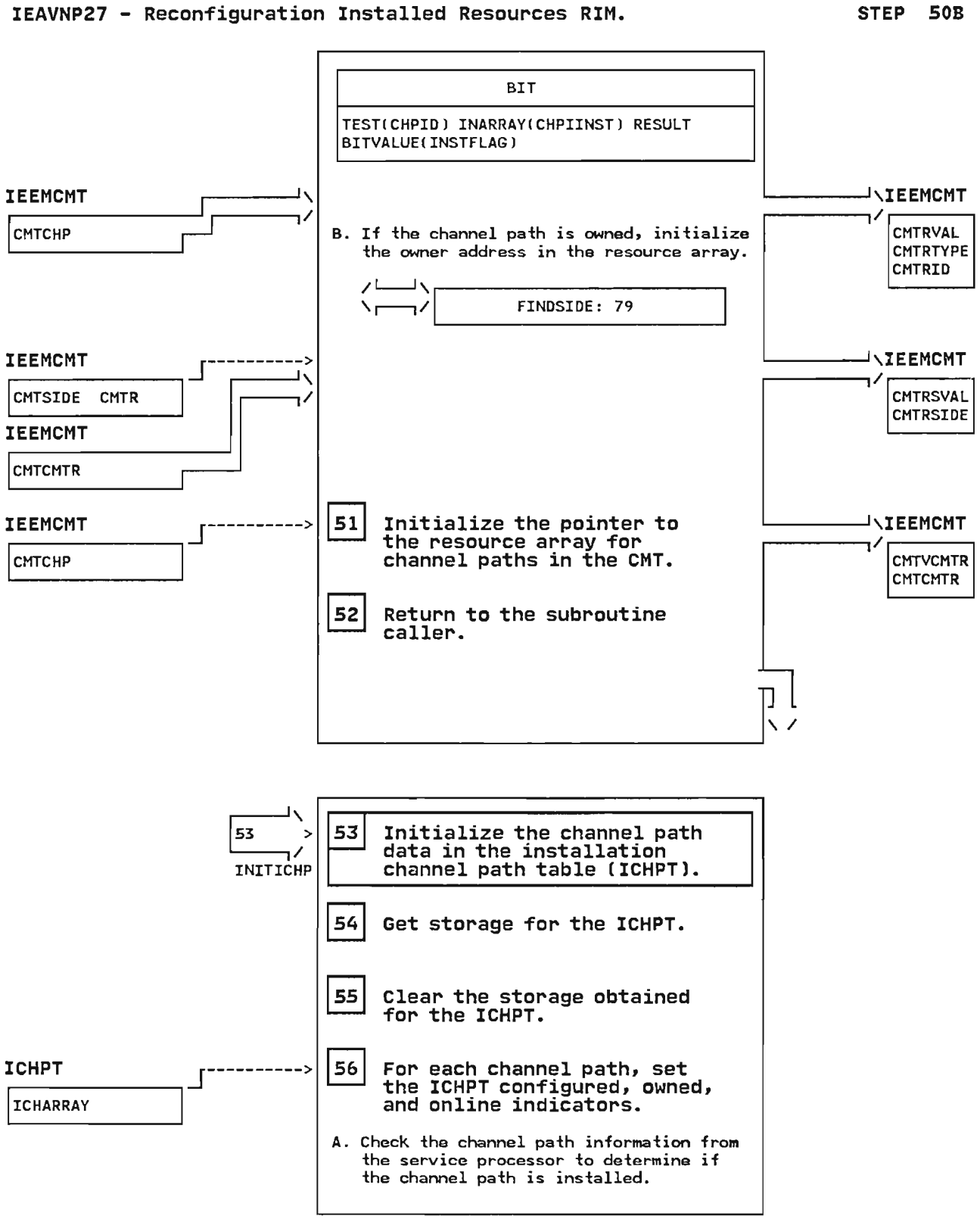


Diagram 14. Reconfiguration Installed Resources RIM (IEAVNP27) Part 15 of 20

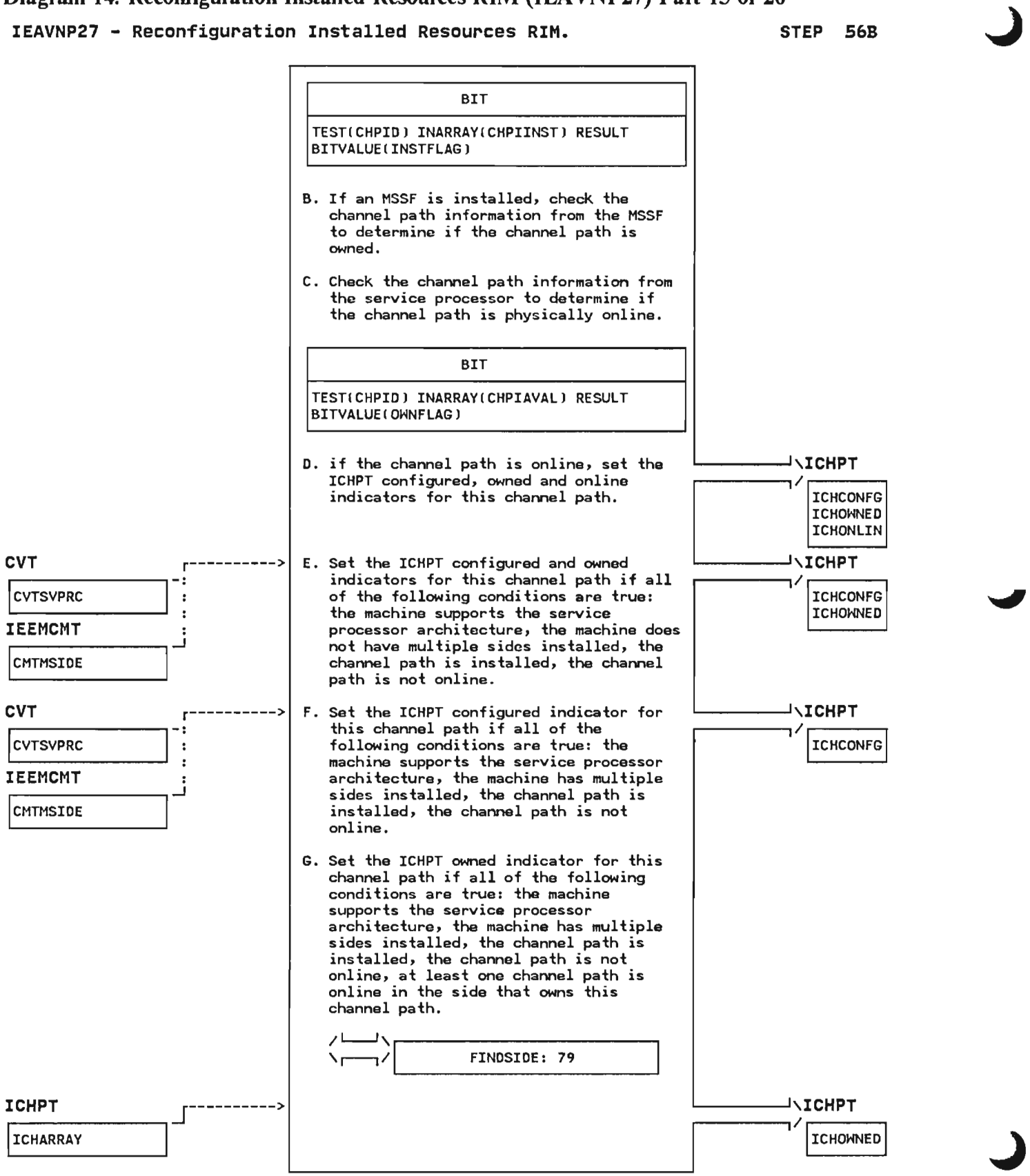


Diagram 14. Reconfiguration Installed Resources RIM (IEAVNP27) Part 16 of 20

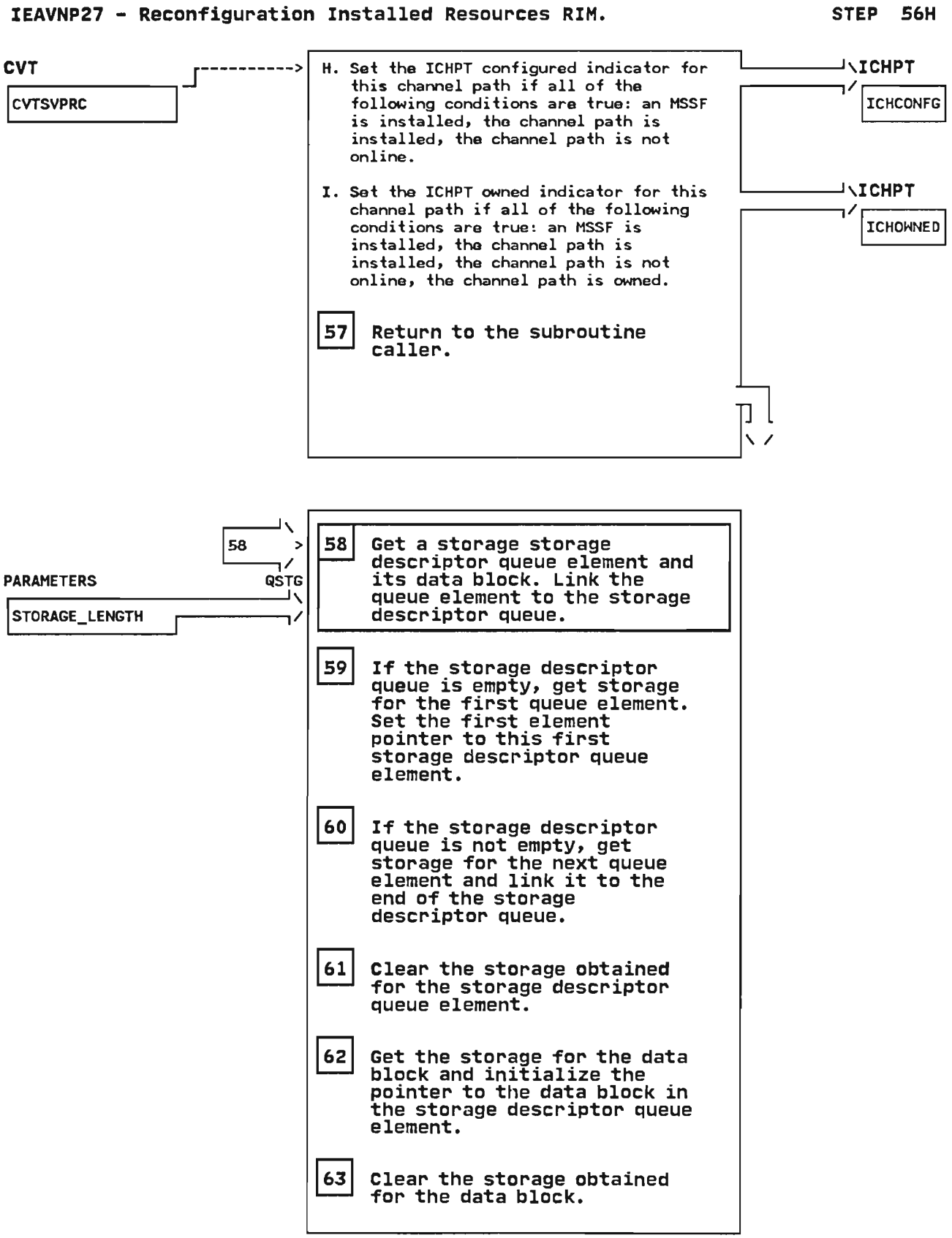


Diagram 14. Reconfiguration Installed Resources RIM (IEAVNP27) Part 17 of 20

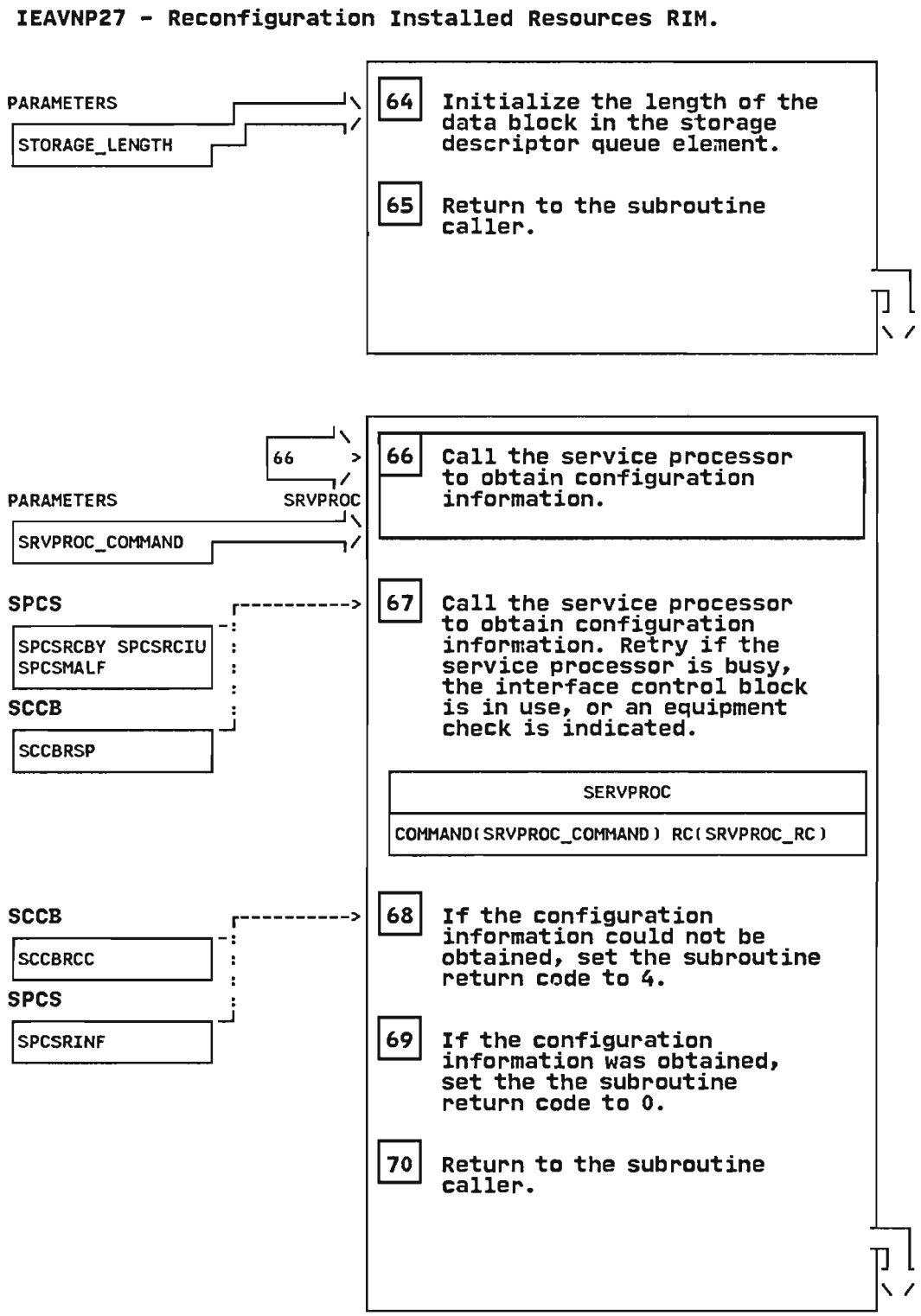


Diagram 14. Reconfiguration Installed Resources RIM (IEAVNP27) Part 18 of 20

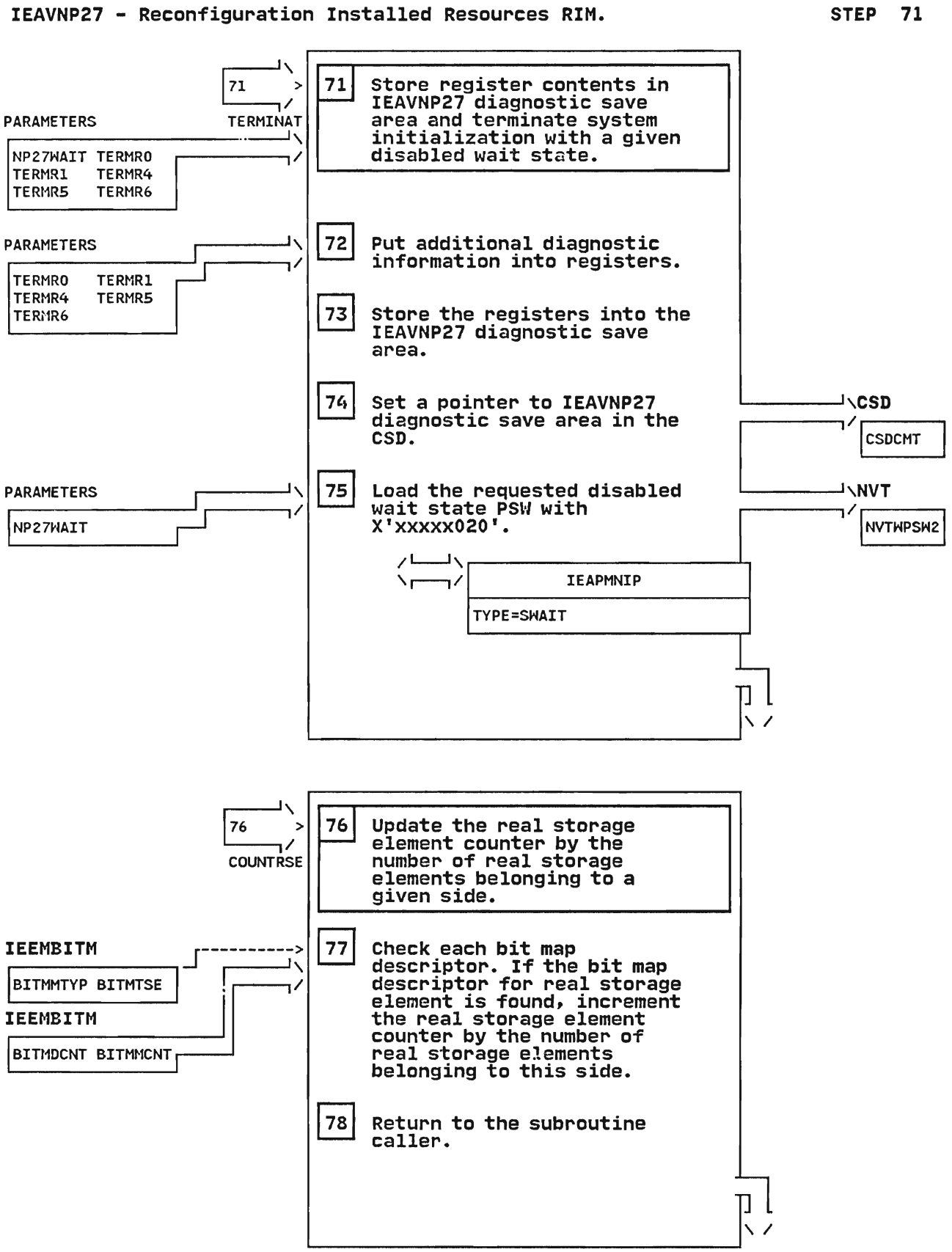


Diagram 14. Reconfiguration Installed Resources RIM (IEAVNP27) Part 19 of 20

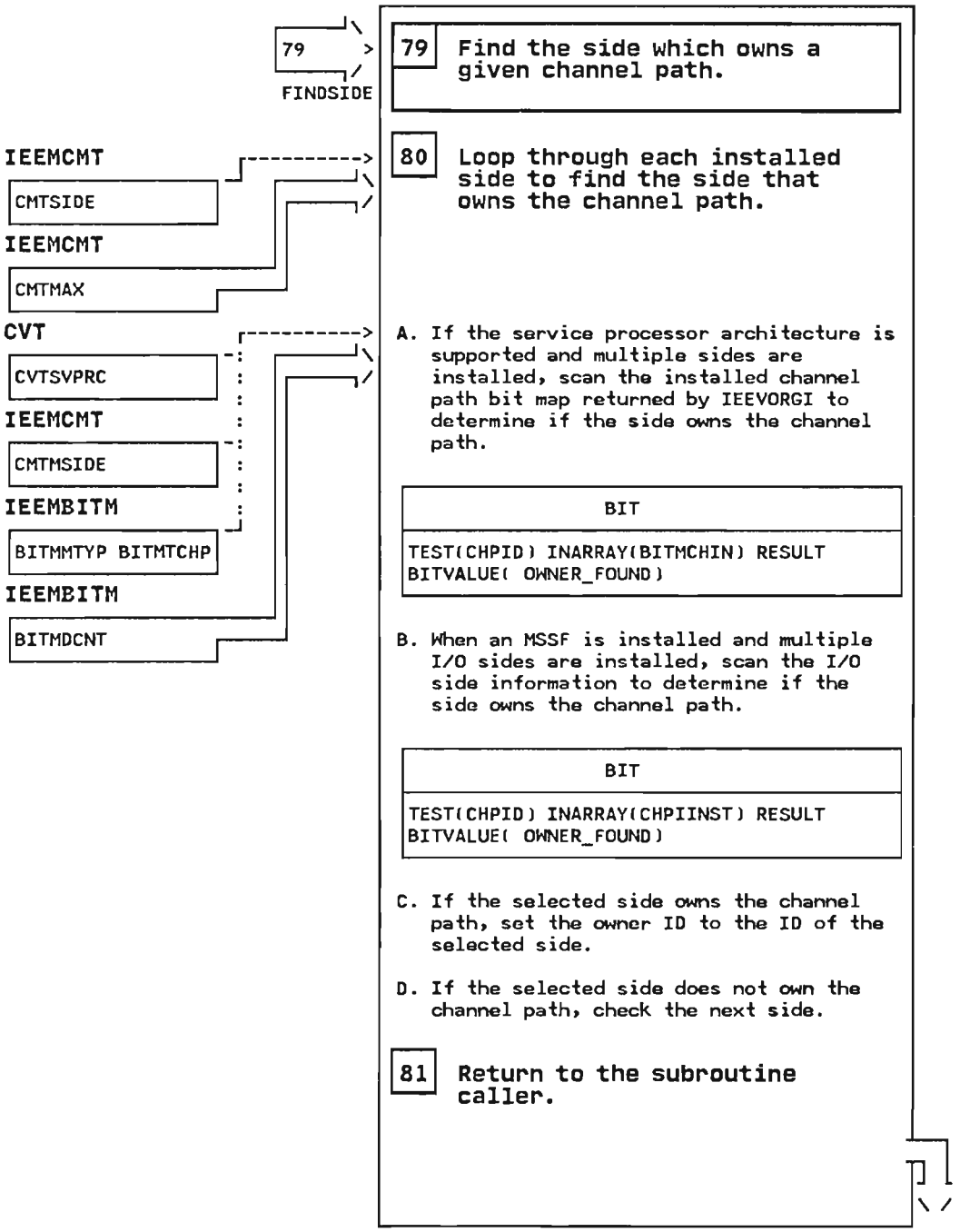


Diagram 14. Reconfiguration Installed Resources RIM (IEAVNP27) Part 20 of 20

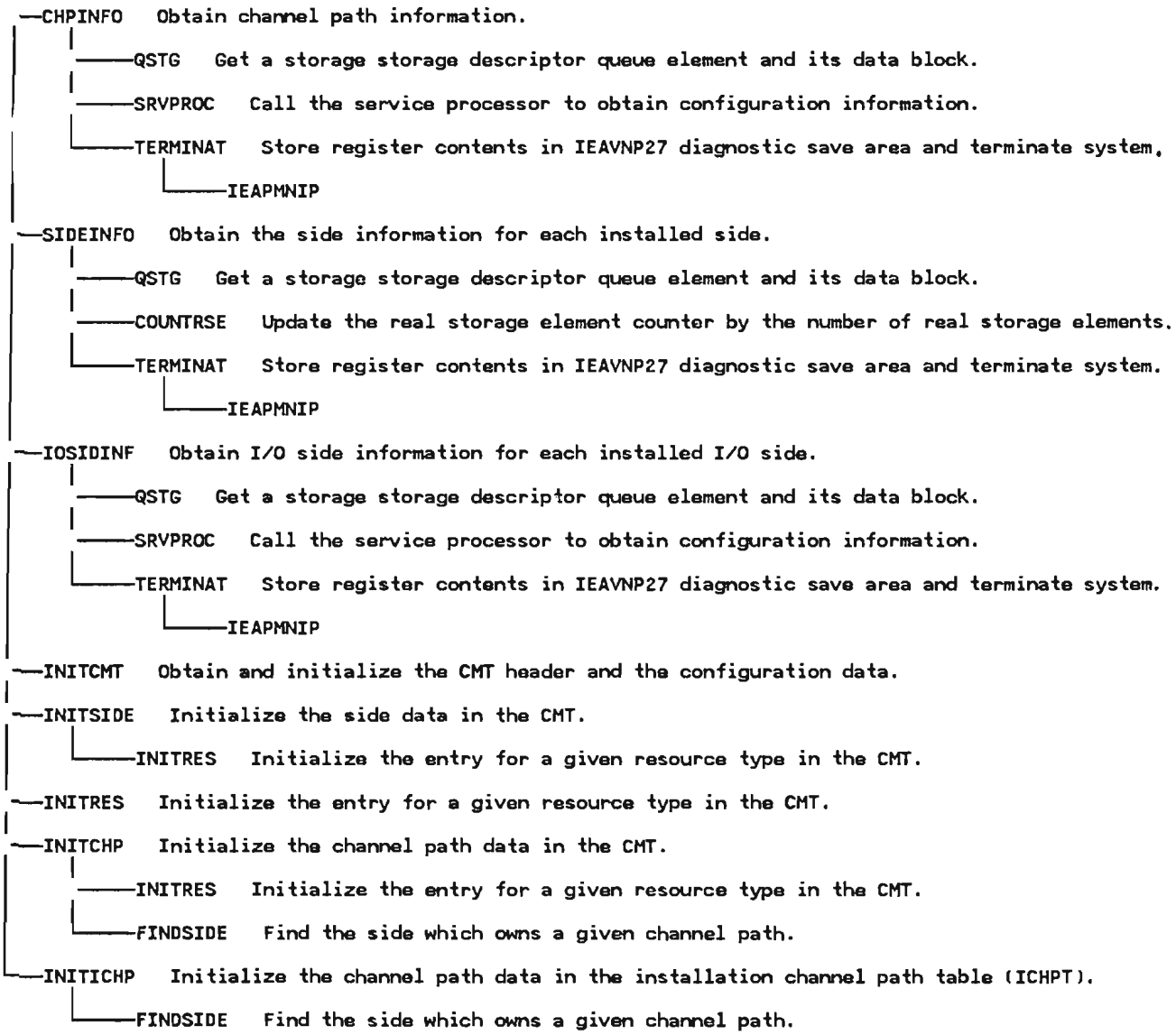


Diagram 15. Non-Direct Access Device Initialization (IEAVNPA2) Part 1 of 4

Diagram 15. Non-Direct Access Device Initialization (IEAVNPA2) Part 2 of 4

Extended Description	Module	Label
IEAVNIPM calls the input/output supervisor (IOS) RIM, IEAVNPA2, to initialize all subchannels, build the installed channel path table (ICHPT), and test the availability of all devices that are not direct access devices. It also initializes their unit control blocks (UCBs). When IEAVNPA2 finishes processing, NIP will be able to issue the message SPECIFY SYSTEM PARAMETERS to the operator. There is a UCB for each device attached to the system. It describes the characteristics of the device to the I/O supervisor and the job scheduler. Entries in the device class queue (DCQ) point to the UCBs; thus, components can locate the UCBs for all devices in a device class without having to scan through the entire UCB chain.		
1 To initialize subchannels, the IOS RIM performs the following steps for all subchannels, beginning with subchannel number 0 and ending when the RIM has tested all subchannels: • Issues a store subchannel (STSCH) command. The output of the store subchannel command is the hardware subchannel information block (SCHIB), which includes the path mask control word (PMCW). • Issues the IOSLOOK macro instruction to find the address of the UCB associated with the subchannel. – If IOSLOOK does not return the address of a UCB, the IOS RIM leaves the subchannel disabled. – If IOSLOOK returns the address of a UCB, the IOS RIM initializes the UCB and the IOS SCHIB for the subchannel to the following values:	IEAVNP02	

Field initialized	*Initialization value*
UCBCHPID	'8' – Channel path IDs
UCBENABL	'1' – allowing interrupts to occur
UCBISC	'5' – indicating priority within a range of 0 to 7 with 0 being high
UCBLPM	logical path mask
UCBNOCON	'0' – indicating UCB connected
UCBPIM	path installed mask
UCBSID	subchannel number
SCHIP	UCB address
SCHLM	'0' – address limit mode
SCHFLG1	'5' – indicating priority within a range of 0 to 7 with 0 being high
SCHE	'1' – indicating the subchannel is enabled

• Issues a modify subchannel command to enable the subchannel and update the hardware SCHIB with the values in the IOS SCHIB.

Diagram 15. Non-Direct Access Device Initialization (IEAVNPA2) Part 3 of 4

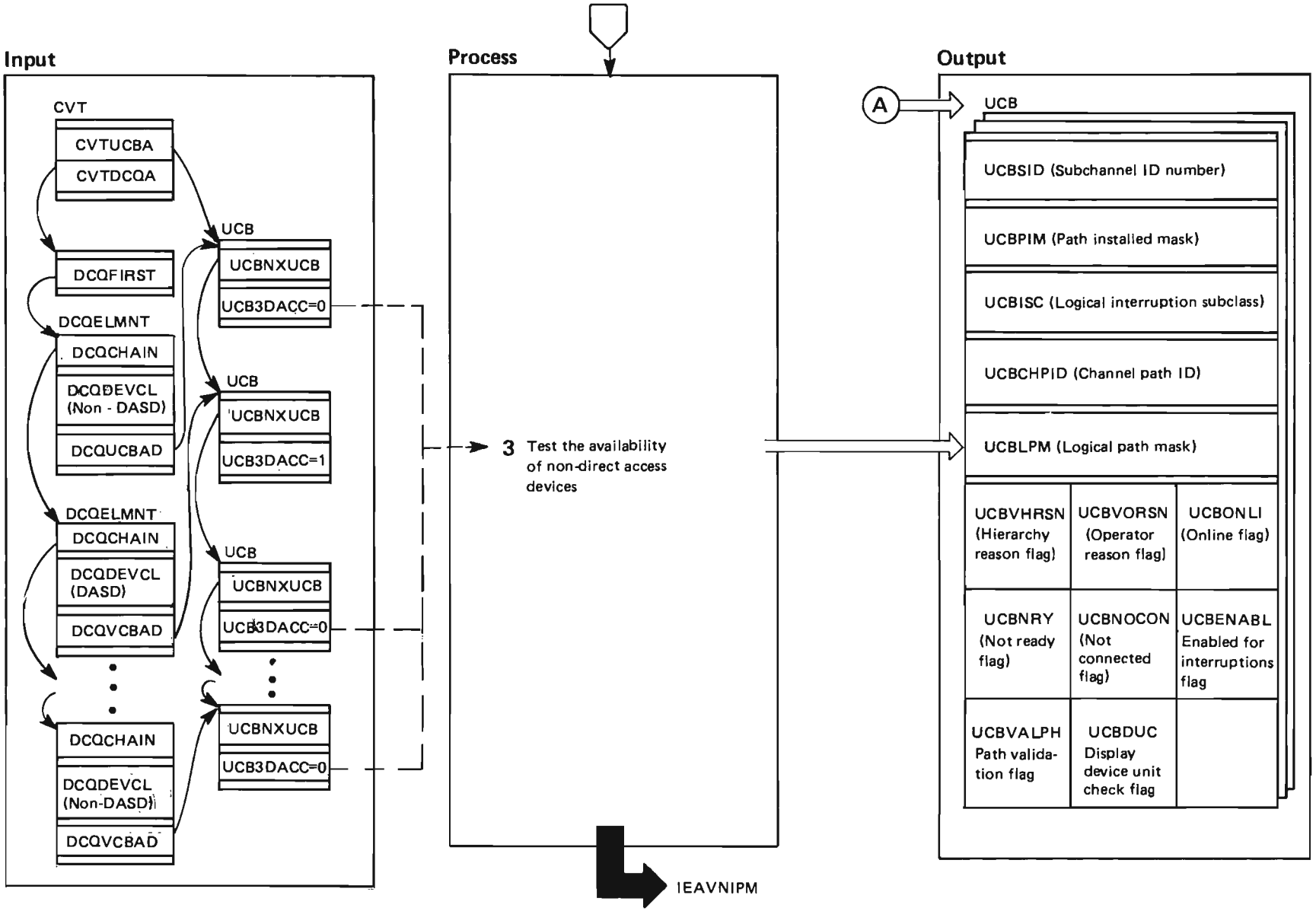


Diagram 15. Non-Direct Access Device Initialization (IEAVNPA2) Part 4 of 4

Extended Description	Module	Label
3 The IOS RIM uses the DCQ to select the UCBs for each non-direct access device that was initialized at SYSGEN time. It calls IOSVNPTH to test the availability of each device. IOSVNPTH issues a STARTIO macro instruction with an I/O NOP CCW command along each installed online path to determine if the device is physically available on that path. IOSVNPTH times all I/O operations. If an operation does not complete within 1.5 seconds, the RIM issues SVC 16 to purge the operation and resets the logical path mask to indicate that the channel path is offline.	IOSVNPTH	NPTHIO
If any path to the device verifies successfully, IOSVNPTH sets the following UCB fields: *Field initialized* — *Initialization value* UCBONLI — '1' – device is online UCBLPM — indicates what paths are available to the device UCBVALPH — '0' – indicates a path has been verified for this device If there are no valid paths to the device, IOSVNPTH marks the device offline, sets UCBLPM to the initial state (the same value as the path installed mask) and sets fields UCBVHRSN and UCBVORSN to indicate why the device is offline.		NPTHSTIO
If after the above processing, the device is to be brought online and a device initialization exit exists for the device, then IOSVNPTH issues a sense-id command. The device initialization exit then initlalizes any feature fields unique to this device. If the exit fails, the device is marked offline. After the IOS RIM has verified all paths to non-direct access devices, it returns control to IEAVNIPM.		DEVINIT

Diagram 16. NIP Console Initialization (IEAVNPC1) Part 1 of 5

IEAVNPC1 - MODULE DESCRIPTION

DESCRIPTIVE NAME: NIP Console Initialization Routine

FUNCTION:
The IEAVNPC1 load module is responsible for selecting the NIP console and identifying it to IEAVNIPM for initial operator communications.

ENTRY POINT: IEAVNPC1

PURPOSE: See function above.

LINKAGE: Call from IEAVNIPM.

CALLERS: IEAVNIPM.

INPUT:
IEANCTxx - Member of SYS1.NUCLEUS containing descriptions of candidate NIP consoles.

OUTPUT: NIP console identified.

EXIT NORMAL: Return to caller via register 14.

OUTPUT: See output above.

EXIT ERROR:
Abnormal return to the NIP system wait state routine in IEAVNIPM, via a branch. The wait state routine is located using the NIP Vector Table (NVT).

The following abnormal conditions may occur during processing by IEAVNPC1 routine and will result in a system wait state:

- If no active NIP console is found, a '07'X wait state code is placed in the NIP Vector Table (NVTFLWSC), and IEAVNPC1 exits to the IEAVNIPM wait routine (NIPSWAIT via NVTSWAIT) to place the system in a disabled wait state.

- The IEANCT member may not be found. This will cause the LOAD to fail.

EXTERNAL REFERENCES:

ROUTINES: NIP Control/Service Routine (IEAVNIPM).

CONTROL BLOCKS:

COMMON NAME	MAPPING	DESCRIPTION
CVT	CVT	Communications Vector Table
DCB	IHADCB	Data Control Block
	IHADCBDF	
DEB	IEZDEB	Data Extent Block
NCT	IEANCT	NIP Console Table
NVT	IHANVT	NIP Vector Table
UCB	IEFUCBOB	Unit Control Block

Diagram 16. NIP Console Initialization (IEAVNPC1) Part 2 of 5

IEAVNPC1 - MODULE OPERATION

- Load the NIP Console Table (NCT) from SYS1.NUCLEUS.
- Verify the consoles in the order that they reside in the NIP Console Table searching for a valid console. A console is valid if:

  (A) the device is online
  (B) the device is connected
  (C) the device is path installed
  (D) the device is ready
  (E) the device is not unit checked

- When a valid console has been found:
  - Set the appropriate bits to indicate that the NIP console is active. Save the address of the console.
- If no valid console is found, a '007'X wait state code is placed in the NIP Vector Table (NVTFLWSC), and IEAVNPC1 exits to the IEAVNIPM WAIT routine to place the system in a disabled wait state.

Diagram 16. NIP Console Initialization (IEAVNPC1) Part 3 of 5

IEAVNPC1 - DIAGNOSTIC AIDS

ENTRY POINT NAME: IEAVNPC1

MESSAGES: None

ABEND CODES: None

WAIT STATE CODES:

Hexadecimal disabled wait state codes issued via NIPSWAIT:
'007'X - No valid NIP console was found

RETURN CODES: None

REGISTER CONTENTS ON ENTRY:

0 - Irrelevant
1 - Irrelevant
2 - NIP Vector Table address
3 - CVT address
4 to 13 - Irrelevant
14 - Return address
15 - Entry point address

REGISTER CONTENTS ON EXIT: Irrelevant

Diagram 16. NIP Console Initialization (IEAVNPC1) Part 4 of 5

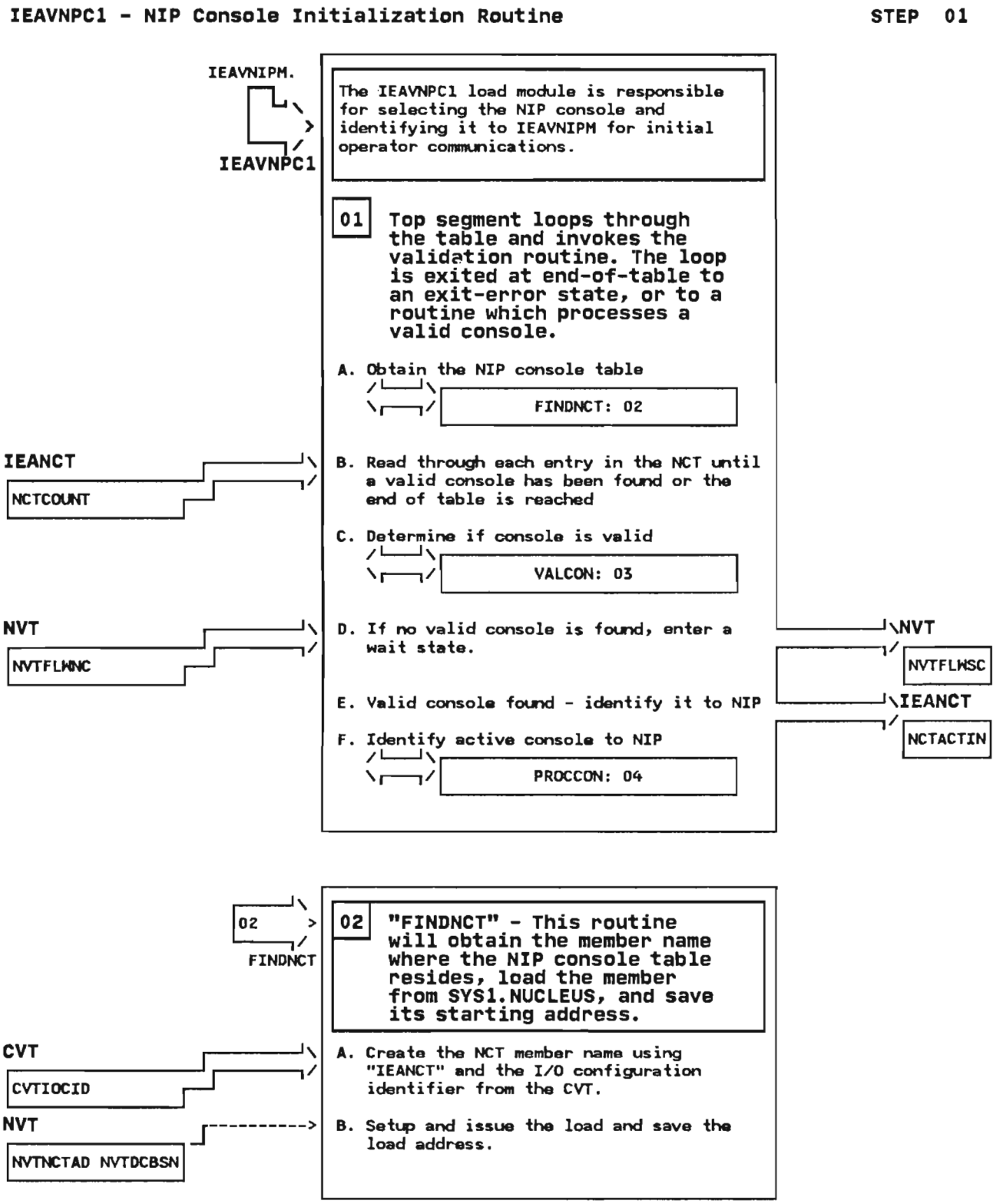


Diagram 16. NIP Console Initialization (IEAVNPC1) Part 5 of 5

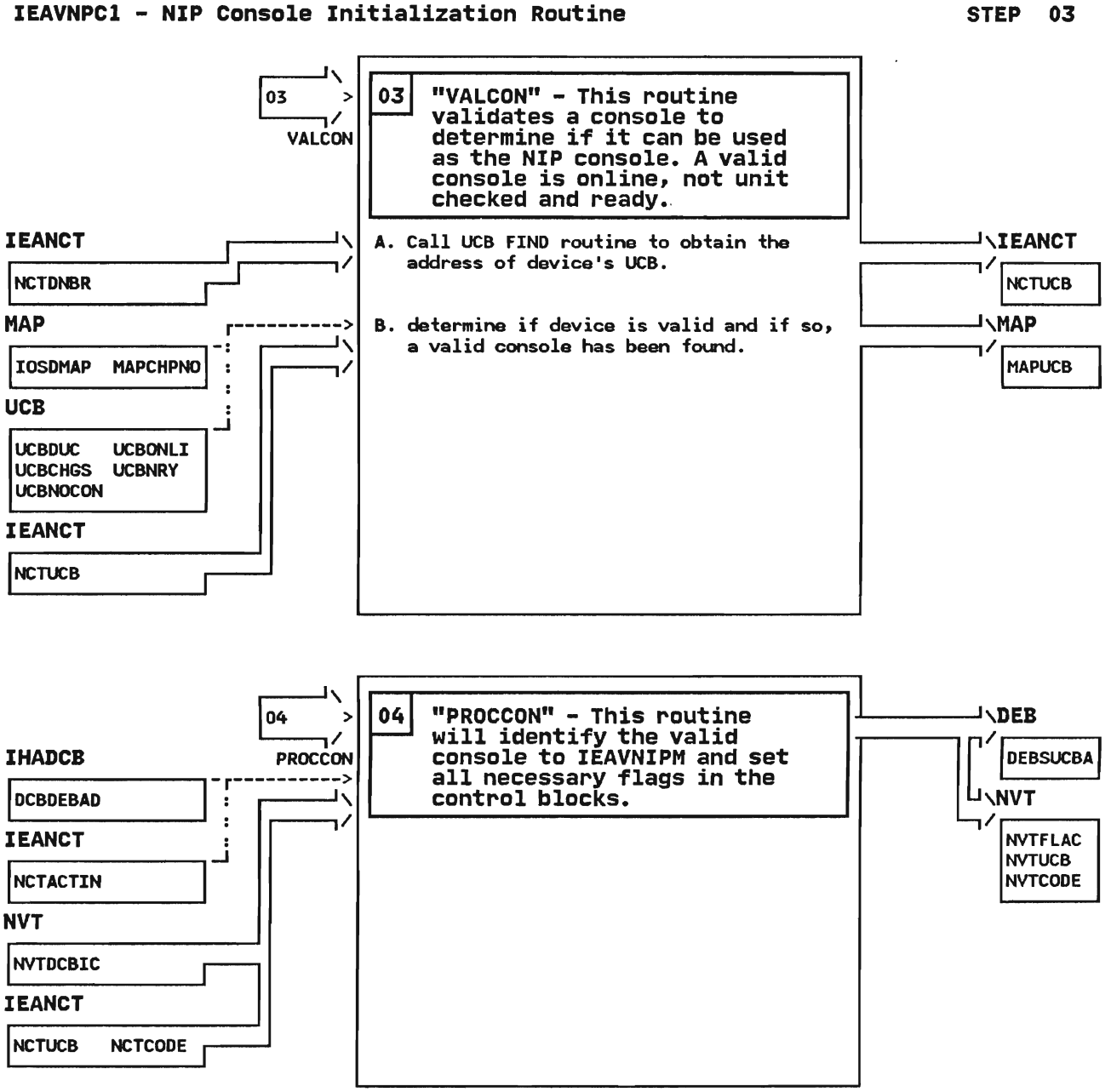


Diagram 17. Direct Access Device Initialization (IEAVNPB6) Part 1 of 2

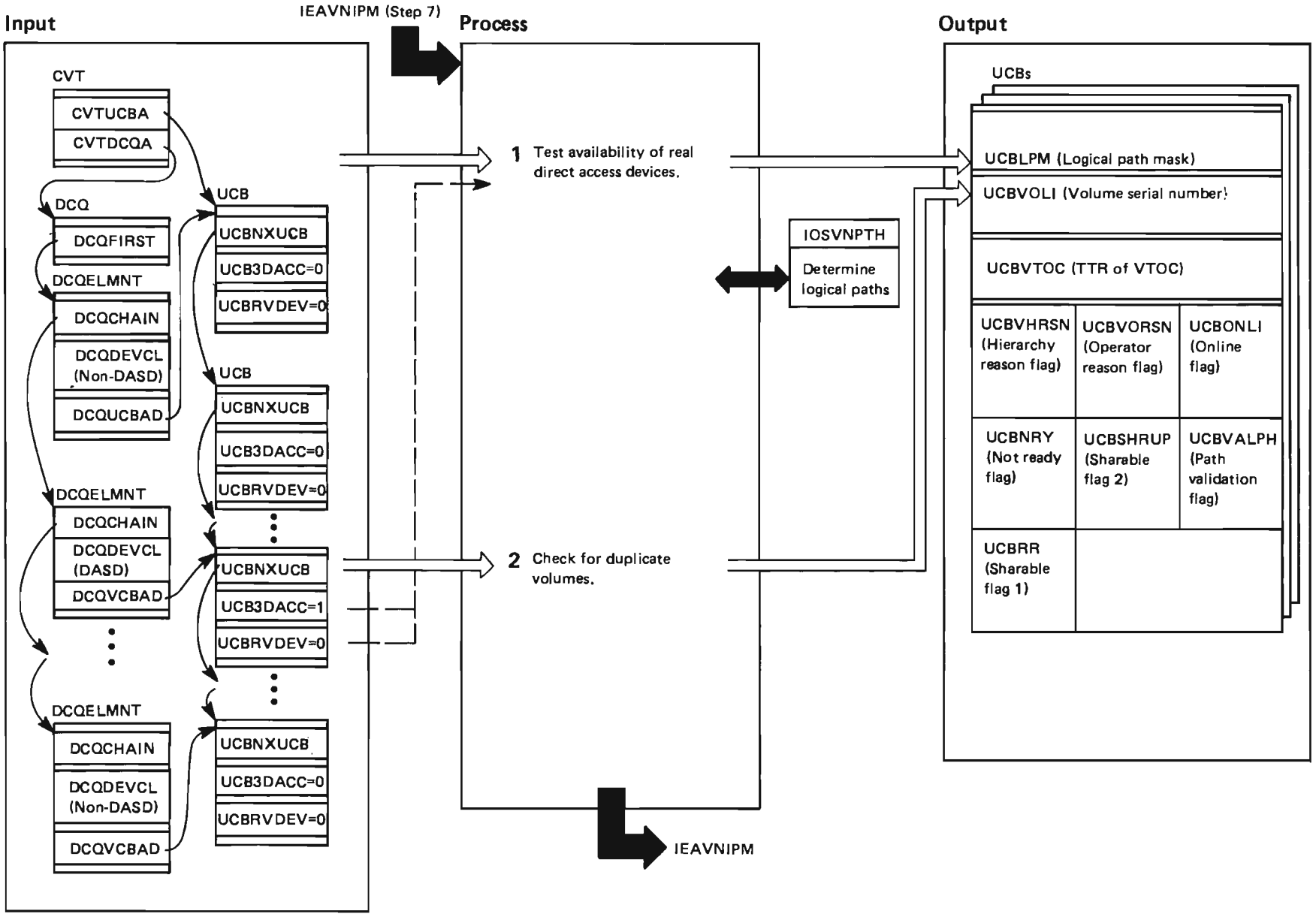


Diagram 17. Direct Access Device Initialization (IEAVNPB6) Part 2 of 2

Extended Description	Module	Label
IEAVNIPM calls the input/output supervisor (IOS) RIM, at entry point IEAVNPB2, to test the sharability and availability of all real direct access devices. It also initializes their unit control blocks (UCBs).		
1 The IOS RIM uses the device class queue (DCQ) to select the UCBs for direct access devices and calls the service routine IOSVNPTH to test the availability of each of these devices. IOSVNPTH issues a STARTIO macro instruction along each installed path to determine if the device is physically available on that path.	IEAVNP02 IOSVNPTH	NPTHIO
When a device is marked sharable at sysgen (UCBRR=1), IOSVNPTH verifies its sharability using a RELEASE CCW command. If the device doesn't have this capability, it rejects the RELEASE command. IOSVNPTH sets the flags UCBSHRUP and UCBRR to indicate that the device is not sharable.		
If the device was designated as not sharable at sysgen. IOSVNPTH issues an I/O NOP CCW command.		
IOSVNPTH times all I/O operations. If an operation does not complete within 1.5 seconds, IOSVNPTH prompts the operator to indicate whether the processing should continue or whether the system should wait for the operation to complete. (The device may be reserved by another system). If the operator indicates that processing should continue, IOSVNPTH purges the operation, marks the channel path offline for the device and processes the next path for the device. Otherwise, the system waits until the operation completes.		NPTHSTIO
If any path to the device verifies successfully, IOSVNPTH sets the following UCB flags: *Field initialized* — *Initialization value* UCBONLI — '1'—device is online UCBLPM — indicates what paths are available to the device UCBVALPH — '0'—indicates a path has been verified for this device		
If there are no valid paths to the device, IOSVNPTH marks the device offline, sets the logical path mask (LPM) to the initial state (the same value as the paths installed mask), and sets fields UCBVHRSN and UCBVORSN to indicate why the device is offline.		
If, after the above processing, the device is to be brought online and a device initialization exit exists for the device, then IOSVNPTH issues a sense-id command. The device initialization exit then initializes any feature fields unique to this device. If the exit fails, the device is marked offline.		DEVINI
2 The IOS RIM checks for duplicate volume serial numbers for online direct access devices. To obtain the volume serial number for a device, the IOS RIM issues an I/O operation. The operation consists of SEARCH, TIC, and READ commands. The IOS RIM places the volume serial number and the VTOC TTR for each direct access device, except SYSRES, into the UCB that corresponds to the device. If it finds a duplicate serial number, it issues message IEA212A to request that the operator demount one of the volumes. After the operator responds with the device number, the IOS RIM places zeroes in the volume serial number and VTOC fields of the UCB for the device that had the volume to be demounted, marks the device offline and not ready, and sets the operator reason flag in the UCB (UCBVORSN=1).	IEAVNP02	NP2READ NP2LDUP
The IOS RIM returns control to IEAVNIPM.		

Diagram 18. System Parameter Analysis and Data Set Opening (IEAVNP03) Part 1 of 4

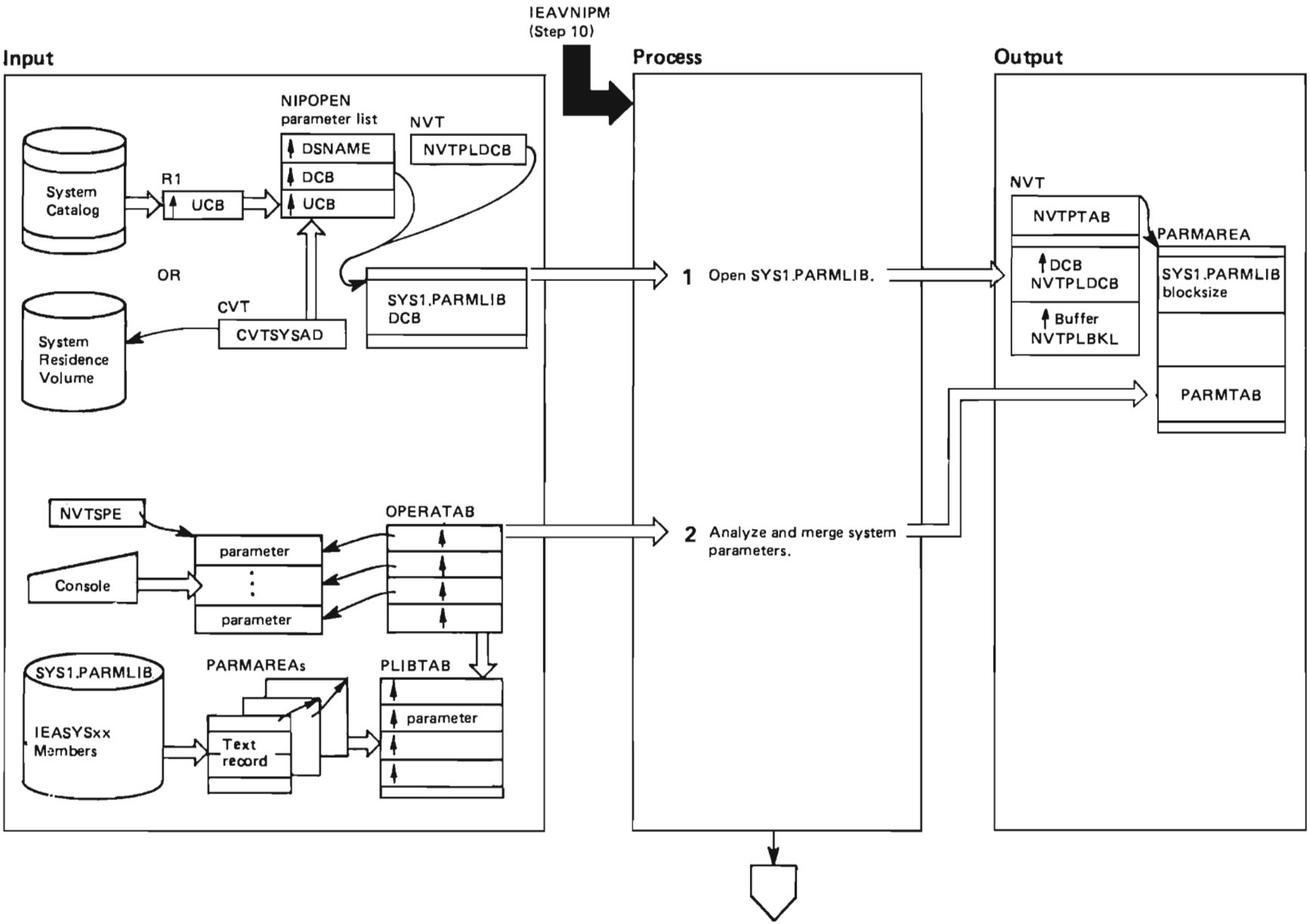


Diagram 18. System Parameter Analysis and Data Set Opening (IEAVNP03) Part 2 of 4

Extended Description	Module	Label
The NIP RIM, IEAVNP03 checks the syntax of the system parameters that were specified in SYS1.PARMLIB (including installation-specified parameters) or by the operator and places these parameters in the PARMTAB. It processes the SYSP, LNK, and LNKAUTH parameters and calls IEAVNPA5 to process the APF parameter. IEAVNP03 opens the system data sets, SYS1.PARMLIB and data sets specified as part of the LNKLST concatenation.		
Except for opening the two system data sets, this routine is the code of NIP PROMPT service, NIPPRMPT. The NIP PROMPT service is called when the operator responds to a message requesting that system parameters be respecified. At this time, IEAVNP03 checks the syntax of the new parameter and puts it in PARMTAB.		
1 NIP assumes either that the system catalog defines the location of SYS1.PARMLIB or, if SYS1.PARMLIB is not cataloged, that the data set resides on the system residence volume. First, NIP issues a LOCATE SVC to read the system catalog entry for SYS1.PARMLIB. If the LOCATE is successful, NIP extracts the volume serial number for SYS1.PARMLIB from its catalog entry and passes control to the NIPMOUNT service routine to ensure that the volume is mounted. NIPMOUNT returns the related UCB address in register 1. If the LOCATE is not successful (SYS1.PARMLIB is not cataloged), NIP assumes that SYS1.PARMLIB is on the system residence volume; NIP obtains the related UCB address from field CVTSYSAD.	IEAVNP03 IEAVNP12 IEAVNPM3 NIPMOUNT	NP3PMLIB NP3PMLIB
NIP places the appropriate UCB address in the NIPOPEN parameter list.		
NIP issues the GETMAIN macro instruction to obtain storage from the private area (subpool 252) for the SYS1.PARMLIB DCB and sets NVTPLDCB to point to the DCB.	IEAVNPM2	NIPWTOR
After NIP opens SYS1.PARMLIB, it returns the block size in PARMAREA. If the blocksize is not a multiple of LRECL 80, NIP issues error message IEA333A and continues processing the operator's request by rounding the blocksize up to a multiple of 80. NIP issues the GETMAIN macro instruction to obtain the storage from the private area for a buffer the length of the block. NVTPLBFS points to this buffer.	IEAVNPM3	NIPOPEN
Along with the UCB address, NIP places the address of the SYS1.PARMLIB DCB in the NIPOPEN parameter list. The NIPOPEN service routine then opens the SYS1.PARMLIB data set by building a DEB for the data set in the private area (subpool 252).	IEAVNPM3	NIPOPEN
2 From field NVTSPE, NIP obtains the address of the operator's reply to the SPECIFY SYSTEM PARAMETERS message. NIP examines the reply:	IEAVNP03	NP3OPSP
• If the operator specified system parameters, NIP scans the parameters for validity and places the addresses of the specified parameter values into preassigned fields in OPERTAB.		NP3SCAN
• NIP reads a block of text records from IEASYS00 into a buffer and returns a single record at a time. It scans the parameters in each text record for validity, then places the addresses of the parameter values into their respective fields in PLIBTAB. The format of PLIBTAB is identical to that of OPERTAB.	IEAVNPM4 IEAVNP03	NP3SYSP NP3SCAN
• If the operator specified SYSP, indicating that parameters come from requested IEASYSxx members of SYS1.PARMLIB, NIP first processes the IEASYS00 member, then each IEASYSxx member specified by SYSP. NIP reads text records from each IEASYSxx member, scans each text record, and places the addresses of the specified parameter values into PLIBTAB. Parameter values from the last IEASYSxx member read override previous IEASYSxx values for those parameters.		NP3SYSP NP3SCAN
NIP merges parameters from OPERTAB into PLIBTAB; as it merges the parameters, NIP overrides PLIBTAB entries with any valid operator-specified values for those entries. (The exceptions to this are: the PAGE parameter, which allows the operator specification to be merged with the SYS1.PARMLIB specification; and parameters for which the installation specified OPI=NO.)		NP3PLMRG
Finally, NIP initializes PARMTAB in PARMAREA: for each parameter that has a nonzero entry in PLIBTAB, NIP moves the input data into PARMTAB. For active keyword parameters that have no input data, NIP places an address value of X'0A' in PARMTAB. For a description of PARMTAB, refer to the "Data Areas" section of this publication and to *Data Areas.*		NP3PTAB

Diagram 18. System Parameter Analysis and Data Set Opening (IEAVNP03) Part 3 of 4

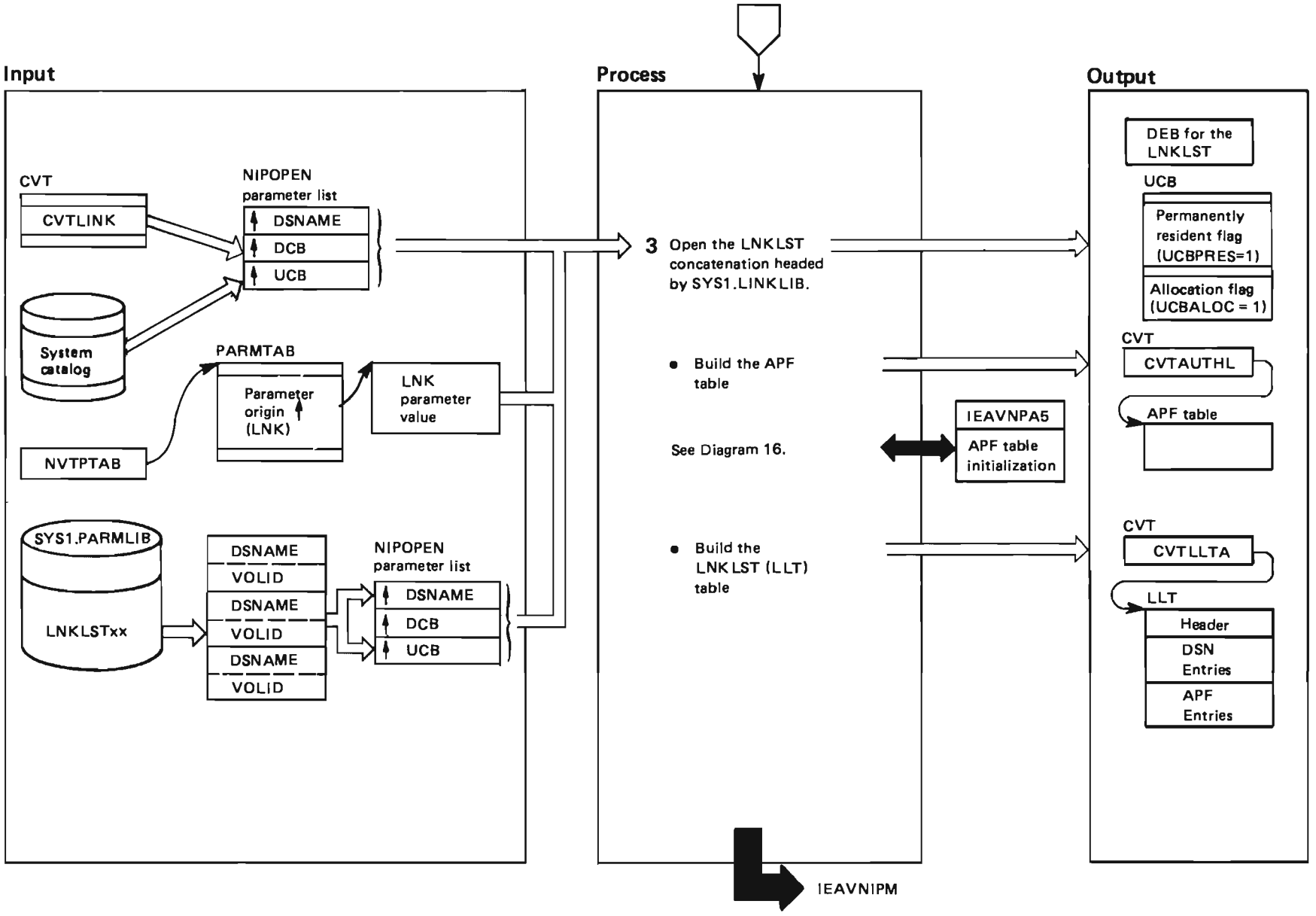


Diagram 18. System Parameter Analysis and Data Set Opening (IEAVNP03) Part 4 of 4

Extended Description	Module	Label
3 NIP assumes that the system catalog defines the location of the SYS1.LINKLIB data set. NIP issues a LOCATE macro instruction to read the system catalog entry for SYS1.LINKLIB. If the LOCATE is successful, NIP extracts the volume serial number for SYS1.LINKLIB from its catalog entry and calls the NIPMOUNT service routine to ensure that the required volume is mounted. NIPMOUNT returns the related UCB address in register 1 and NIP places the address in the NIPOPEN parameter list.	IEAVNP03	NP3LKLIB
	IEAVNPM3	NIPMOUNT
Next, NIP retrieves the address of the SYS1.LINKLIB DCB from field CVTLINK and places it in the NIPOPEN parameter list. NIP then calls the NIPOPEN service routine, which opens the SYS1.LINKLIB data set by building a DEB in the SQA for the data set. NIPOPEN marks the UCBSTAT for the SYS1.LINKLIB volume permanently resident and allocated, and increases the use count (UCBUSER). NIPOPEN indicates in the DEB that the data set is an authorized program library. After NIP opens the SYS1.LINKLIB data set, NIP concatenates installation-specified data sets with SYS1.LINKLIB to create the system link library (the LNKLST concatenation). The installation specifies these data sets by entering their data set names in LNKLSTxx member(s) of SYS1.PARMLIB and by specifying the LNK system parameter. (If the installation does not specify LNK, NIP attempts to find the LNKLST00 member.)	IEAVNP03	NP3LKLIB
	IEAVNPM3	NIPOPEN
	IEAVNP03	NP3LCAT

Extended Description	Module	Label
NIP obtains an 8K-byte workarea to build a table of data set names from the input LNKLSTxx member(s). NIP reads each record from the requested LNKLSTxx member(s), scans the record, and enters the data set names in the workarea.		
Next, NIP processes each entry in the workarea, as follows. First, NIP issues a LOCATE macro instruction for the data set names in the entries. If the locate is successful, NIP initializes the corresponding entry with the volume ID returned from the catalog entry. If LOCATE is not successful, NIP flags the entry unavailable. After LOCATE, NIP passes each available entry to the NIPMOUNT service routine to mount the volume. NIPMOUNT returns the UCB address of the mounted volume. NIP places the UCB address along with the DCB address from field CVTLINK into the NIPOPEN parameter list. Finally, NIP passes control to the NIPOPEN service routine to concatenate the current data set with the previously defined portion of the link library. If the data set is successfully opened, NIPOPEN sets the corresponding UCBSTAT to permanently resident and allocated, and increases the use count (UCBUSER).	IEAVNPM3 IEAVNP03 IEAVNPM3	NIPMOUNT NP3LCAT NIPOPEN
If too many data sets or extents were specified, IEAVNP03 truncates the LNKLST, issues message IEA328E and stores the LNKLSTxx suffix in the LNKLST lookaside (LLA) control block for LLA to reissue IEA328E when the system is initialized.	IEAVNP03	NP3LCAT
IEAVNP03 calls IEAVNPA5 to build the APF table in the SQA. IEAVNP03 processes the LNKAUTH parameter and builds the LNKLST table (LLT) in the SQA. The LLT contains the LNKLST data set names and flags that indicate which LNKLST data sets are APF authorized. If at least one LNKLST data set is not APF authorized, IEAVNP03 turns off DEBAPFIN to indicate that the concatenation is not authorized as a whole.		

Diagram 19. Authorized Program Facility Table Initialization (IEAVNPA5) Part 1 of 2

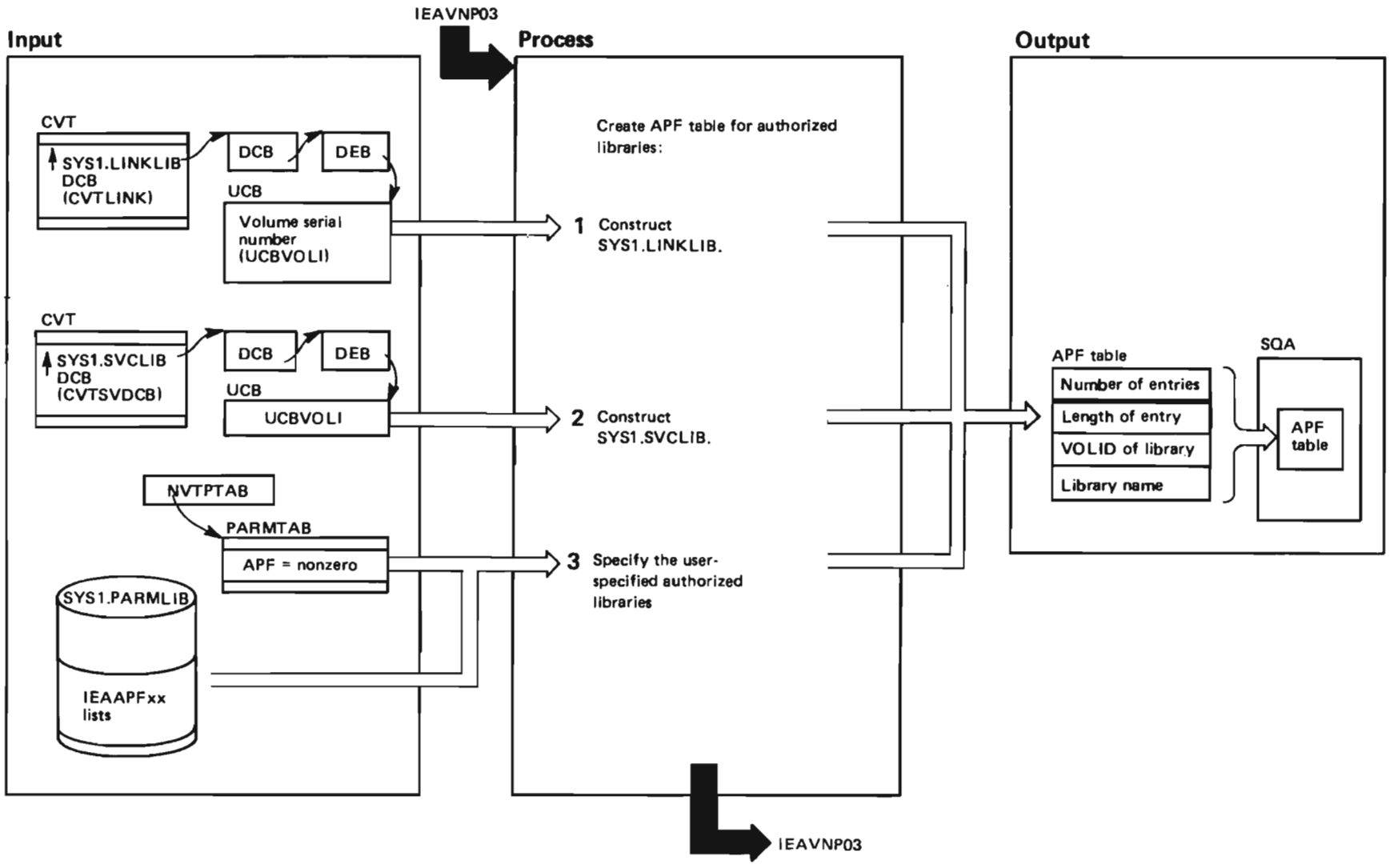


Diagram 19. Authorized Program Facility Table Initialization (IEAVNPA5) Part 2 of 2

Extended Description	Module	Label
The authorized program facility (APF) permits the installation to identify system or user programs that are allowed to use restricted system functions. The APF table contains an entry for each library of authorized programs. The APF table entry contains the length of the entry, the volume ID of the volume on which the library can be found, and the library name. The NIP RIM IEAVNP03 calls IEAVNPA5 to build the APF table in SQA as follows:	IEAVNP03	
	IEAVNPA5	NPA5APF

1 IEAVNPA5 always constructs an entry for SYS1.LINKLIB. IEAVNPA5 obtains the volume ID from the UCB for SYS1.LINKLIB; and the SYS1.LINKLIB DCB address from field CVTLINK. It follows a chain of pointers to the UCB. Then, IEAVNPA5 constructs an APF table entry for SYS1.LINKLIB.

2 IEAVNPA5 always constructs an entry for SYS1.SVCLIB. IEAVNPA5 obtains the volume ID from the UCB for SYS1.SVCLIB; IEAVNPA5 obtains the SYS1.SVCLIB DCB address from field CVTSVDCB and follows a chain of pointers to the UCB. Then, IEAVNPA5 constructs an APF table entry for SYS1.SVCLIB.

3 The installation can specify authorized libraries by listing them in a member or members of SYS1.PARMLIB named IEAAPFxx and using the APF system parameter to indicate the suffix for IEAAPFxx. After it initializes the SYS1.LINKLIB and SYS1.SVCLIB entries in the APF table, IEAVNPA5 checks the APF parameter entry in PARMTAB to determine whether the installation specified authorized libraries. If the parameter entry in PARMTAB is not zero, it contains the suffix or suffixes for IEAAPFxx. IEAVNPA5 builds an APF table entry for each library listed in the IEAAPFxx member of SYS1.PARMLIB.

Diagram 20. REAL Parameter Initialization (IEAVNPE8) Part 1 of 2

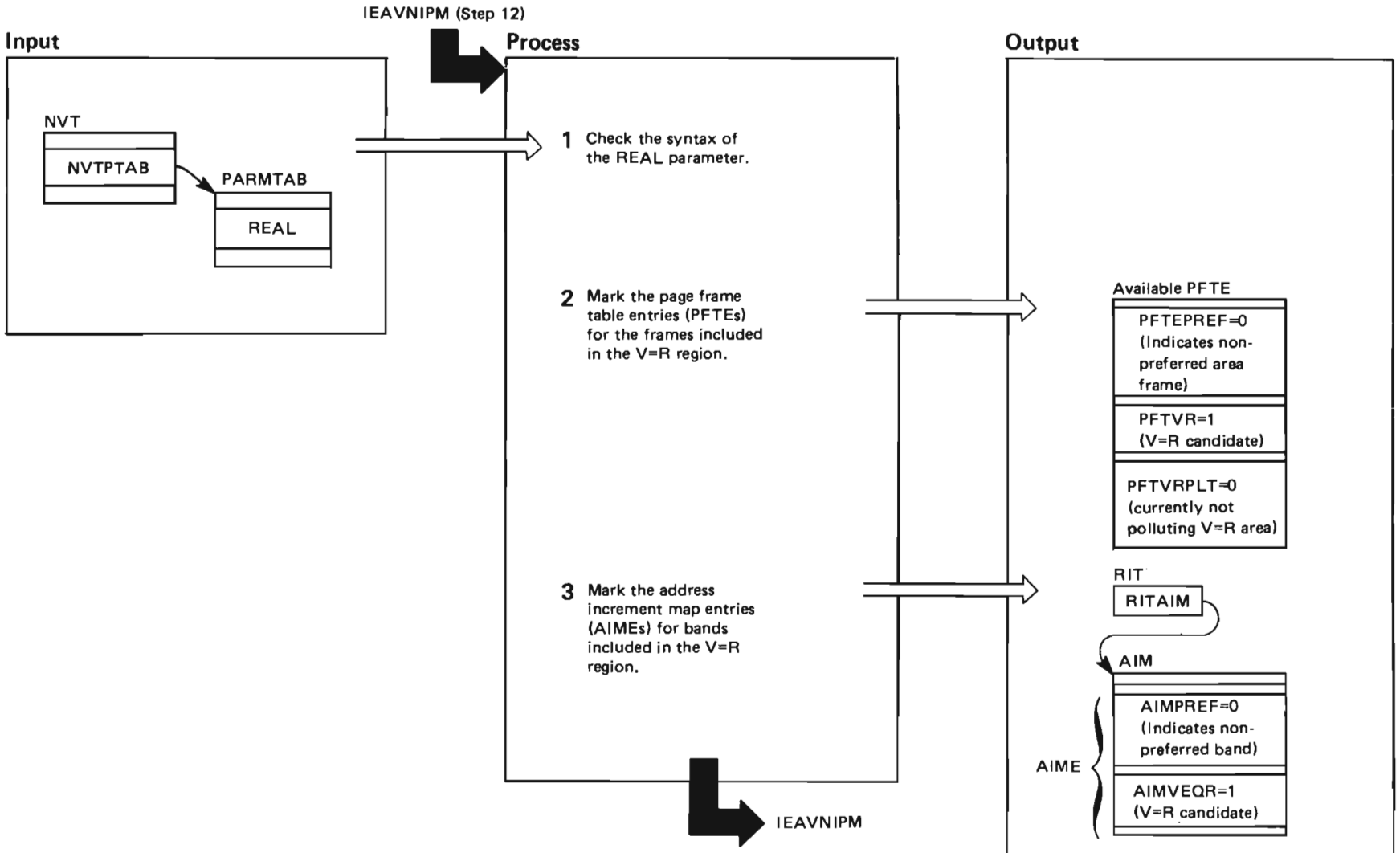


Diagram 20. REAL Parameter Initialization (IEAVNPE8) Part 2 of 2

Extended Description	Module	Label
IEAVNIPM calls the real storage manager (RSM) RIM, IEAVNPE8, to process the REAL parameter. This parameter specifies the amount of storage to be reserved for programs running V=R (that is, programs in which the real address and the virtual addresses must be the same). The RSM RIM marks all the page frame table entries (PFTEs) for frames in the resulting V=R region, and marks the appropriate address increment map entries (AIMEs). After the VSM RIM, IEAVNP88, initializes the CSA, the RSM RIM, IEAVNPD8, will make necessary adjustments to the control blocks.that describes the V=R region. **1** The RSM RIM checks the syntax of the REAL parameter specified by the operator. If the parameter is not valid, the RIM issues message number IAR006A to request that the operator respecify a new value for the REAL parameter. **2** The RSM RIM marks each frame in the V=R area that is set up by the REAL parameter as non-preferred, V=R candidate frames. If any of the frames in the V=R area are already in use for the non-extended system queue area (SQA), the local system queue area (LSQA), or part of a double frame pair, the RSM RIM marks the frames as frames that are polluting the V=R area. **3** The RSM RIM marks the AIME for each band in the V=R region that contains: • All V=R candidate frames (AIMVEQR=0) • Not all preferred frames (AIMPREF=0)	IEAVNPE8	

Diagram 21. Global Resource Serialization Control Block Initialization (IEAVNP23/ISGNCBIM) Part 1 of 4

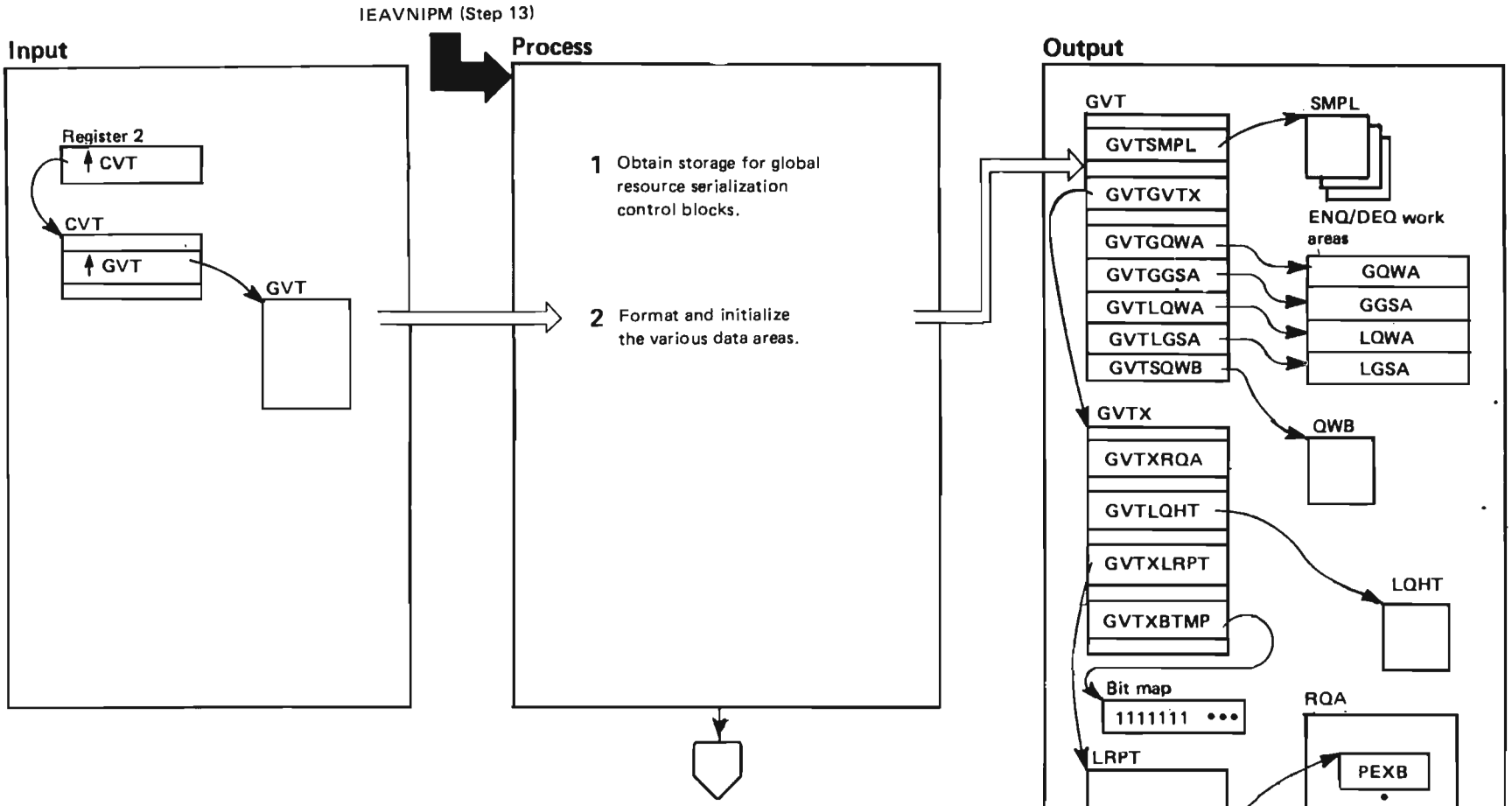


Diagram 21. Global Resource Serialization Control Block Initialization (IEAVNP23/ISGNCBIM) Part 2 of 4

Extended Description	Module	Label

IEAVNIPM calls the global resource serialization RIM, ISGNCBIM, to obtain storage for and initialize the global resource serialization control blocks before the creation of the global resource serialization address space. ISGNCBIM is the first global resource serialization RIM to receive control. (ISGNTASC is the other RIM.) IEAVNIPM invokes ISGNCBIM in supervisor state and in key 0.

Note that IEAVNIPM knows ISGNCBIM as IEAVNP23. ISGNCBIM is an entry point in load module IEAVNP23.

1 ISGNCBIM obtains storage for the following control blocks and data areas:

- Global resource serialization vector table extension (GVTX)
- Local queue hash table (LQHT)
- Local resource pool table (LRPT)
- Resource queue area (RQA)
- Queue work block (QWB)
- Storage manager parameter list (SMPL)
- ENQ/DEQ work areas, which are the global queue work area (GQWA), local queue work area (LQWA), global group summary area (GGSA), and local group summary area (LGSA)

ISGNCBIM issues the GETMAIN macro instruction to obtain the storage from SQA, subpool 245. ISGNCBIM obtains approximately 12K of storage.

Extended Description	Module	Label

2 ISGNCBIM formats and initializes the GVTX, LRPT, and LQHT. ISGNCBIM places the address of the LRPT in the GVTX (GVTXLRPT) and the address of the LQHT in the GVTX (GVTXLQHT). ISGNCBIM initializes a bit map to all ones and store its address in the GVTX (GVTXBTMP). This bit map is used to identify the pages in the RQA that are allocated to the global resource serialization storage manager and the ones that are not. ISGNCBIM sets the bit map to all ones because, at this time, all pages in the RQA are allocated and fixed. ISGNCBIM initializes the GVTX with the address of the RQA (GVTXRQA). ISGNCBIM also initializes the GVT with the address of the QWB and with the addresses of the ENQ/DEQ work areas: GQWA, LQWA, GGSA, and LGSA (GVTGQWA, GVTLQWA, GVTGGSA, and GVTLQSA). ISGNCBIM then initializes the pool extent blocks (PEXBs). A PEXB maps a portion of the RQA and contains cells for one of the following control blocks:

- QWB – queue work block
- QXB – queue extension block
- QEL – queue element
- QCB – queue control block (either a small, medium, or large QCB)

ISGNCBIM initializes the storage manager parameter lists (SMPLs). Each SMPL represents a request to the global resource serialization storage manager and identifies the storage associated with a particular control block or data area. A SMPL exists for the following control blocks and data areas: QWB, QXB, QEL, and QCB (either small, medium, or large). Note that, because there are only a limited number of control blocks available at this time for processing enqueue or dequeue requests, concurrent ENQs/DEQs will be limited.

Diagram 21. Global Resource Serialization Control Block Initialization (IEAVNP23/ISGNCBIM) Part 3 of 4

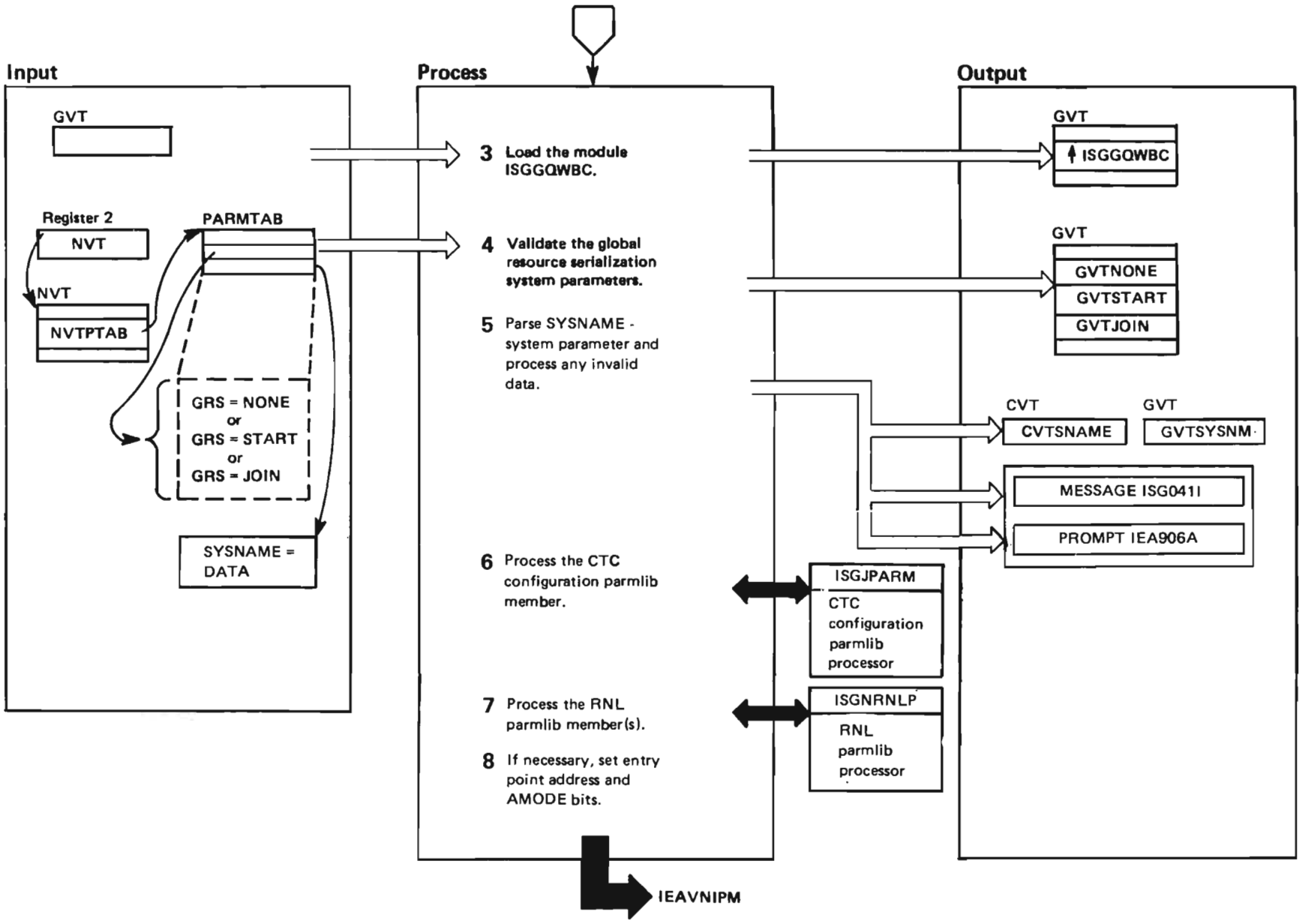


Diagram 21. Global Resource Serialization Control Block Initialization (IEAVNP23/ISGNCBIM) Part 4 of 4

Extended Description	Module	Label
3 ISGNCBIM issues a BLDL macro instruction to obtain the size of a module ISGGQWBC. ISGGQWBC is the global resource serialization QWB copy routine, which resides in SYS1.LINKLIB. Using the size of the ISGGQWBC module, ISGNCBIM issues a GETMAIN macro instruction to obtain storage from SQA (subpool 245). ISGNCBIM then issues the LOAD macro instruction to load ISGGQWBC directly into this acquired storage. ISGNCBIM saves the address of ISGGQWBC in the GVT (GVTGQWBC) and also its size so that the storage it occupies can be released later.	ISGNCBIM	
4 ISGNCBIM ensures that the GRS system parameter is either NONE, JOIN, or START. If the parameter is valid, ISGNCBIM sets the appropriate flag in the GVT: GRS=NONE GVTNONE='1' GRS=START GVTSTART='1' GRS=JOIN GVTJOIN='1' Otherwise, ISGNCBIM asks the operator (using the NIPWTOR service routine) to respecify the GRS parameter.	ISGNCBIM IEAPMNIP	
5 ISGNCBIM verifies the data specified for the SYSNAME= system parameter. If the parameter is valid, ISGNCBIM stores the SYSNAME in the CVT CVTSNAME) and the GVT (GVTSYSNM). Otherwise, ISGNCBIM issues message ISG041I and prompts the operator with message IEA906A. When control is returned, ISGNCBIM verifies the new data specified for the SYSNAME= system parameter. When the SYSNAME= system parameter is valid, processing continues at step 6.	ISGNCBIM IEAPMNIP	
6 ISGNCBIM calls the global resource serialization CTC configuration parmlib processor (ISGJPARM) to parse the GRSCNF system parameter and the GRSCNFxx SYS1.PARMLIB member.	ISGJPARM	
7 ISGNCBIM calls the global resource serialization resource name list parmlib processor (ISGNRNLP) to parse the GRSRNL system parameter and to parse the GRSRNLxx SYS1.PARMLIB member(s).	ISGNRNLP	
8 If the GRS system parameter is either START or JOIN, ISGNCBIM notifies the NUCLKUP service routine to set the entry point addresses and the addressing mode bits of the SYSTEM inclusion, SYSTEMS exclusion, and the RESERVE conversion exits in the GVT.	ISGNCBIM	
If an error was encountered by ISGNRNLP (GVTRNLER = 1), ISGNCBIM deactivates global resource serialization.		
If an error was encountered by ISGJPARM (GVTCNFER = 1), ISGNCBIM deactivates global resource serialization only if ISGNRNLP also encountered an error (that is, only if GVTRNLER also is 1). If only GVTCNFER = 1, ISGNCBIM activates global resource serialization to allow ISGBTC to display GRSCNFxx error messages. Then ISGNASIM deactivates global resource serialization.		
If the GRS system parameter is NONE (GVTNONE='1'), ISGNCBIM indicates that global resource serialization is inactive (sets GVTGRSNA to 1) and that the GRS option processing is complete (sets GVTGRSPC to 1).		
Recovery Processing		
None.		

Diagram 22. Global Resource Serialization CTC Configuration Parmlib Processor (ISGJPARM) Part 1 of 4

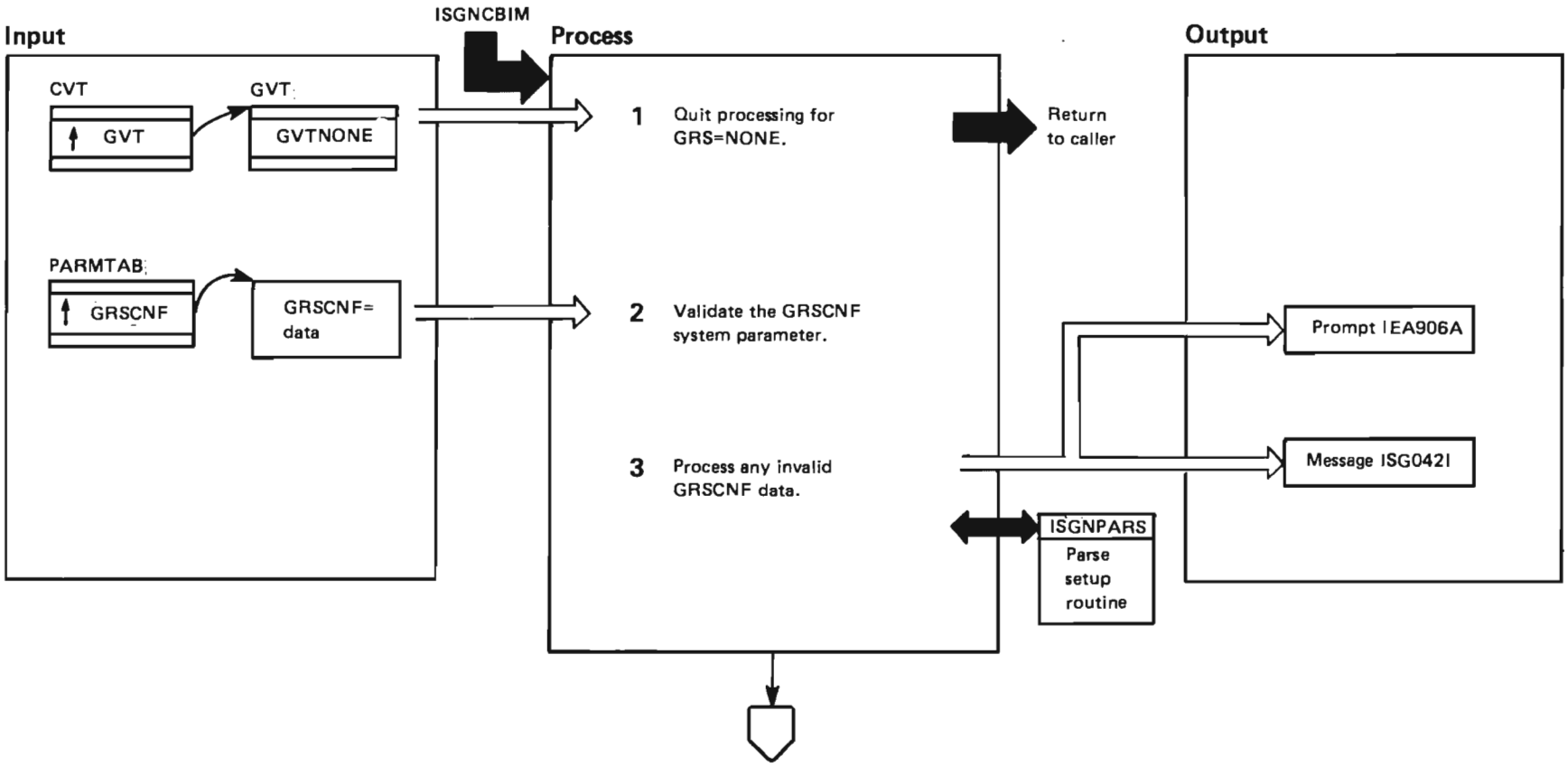


Diagram 22. Global Resource Serialization CTC Configuration Parmlib Processor (ISGJPARM) Part 2 of 4

Extended Description

ISGJPARM parses the GRSCNF system parameter used to initialize global resource serialization and the SYS1.PARMLIB members, GRSCNFxx. ISGJPARM resides in the IEAVNP23 load module.

1 If GRS=NONE was specified (GVTNONE='1'), then this system is not part of a global resource serialization complex. ISGJPARM returns to the caller.

2 ISGJPARM verifies the data specified for the GRSCNF= system parameter. The xx must consist of two alphameric characters and be syntactically correct.

3 If the GRSCNF= system parameter is invalid, ISGJPARM issues message ISG042I and prompts the operator with message IEA906A. When control is returned, processing continues at step 2.

If the GRSCNF= system parameter is valid, processing continues at step 4.

Module **Label**

Diagram 22. Global Resource Serialization CTC Configuration Parmlib Processor (ISGJPARM) Part 3 of 4

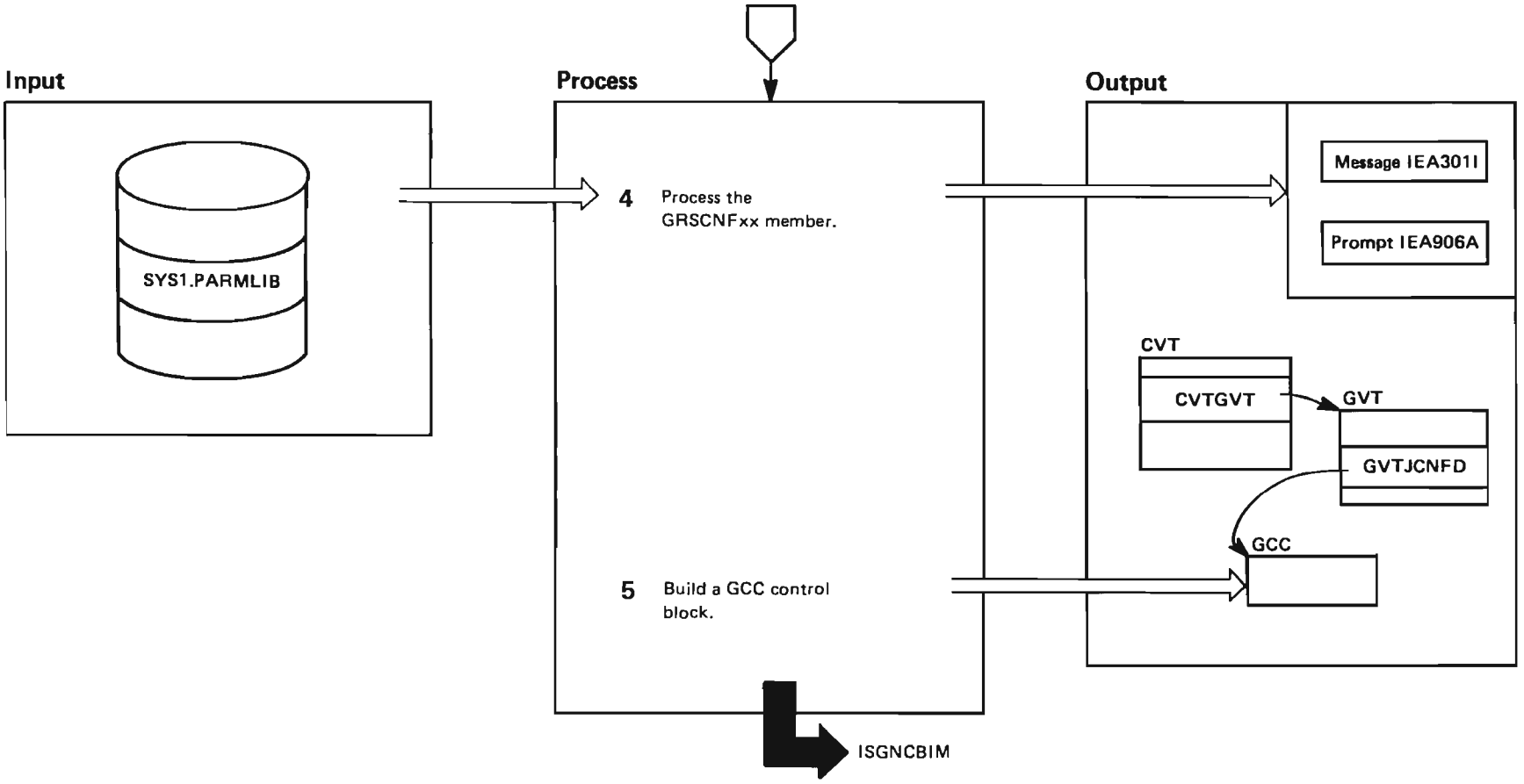


Diagram 22. Global Resource Serialization CTC Configuration Parmlib Processor (ISGJPARM) Part 4 of 4

Extended Description	Module	Label
4 ISGJPARM calls the global resource serialization initialization parse setup module (ISGNPARS) which sets up to parse the GRSCNFxx SYS1.PARMLIB member. (IEEMB887 parses the GRSCNFxx SYS1.PARMLIB member.)		
If the first record of the SYS1.PARMLIB member was not read successfully by ISGNPARS, RDRTNEOR (a subroutine in ISGJPARM) receives control to process the error as follows:	IEAPMNIP	
• If the GRSCNFxx member does not exist, ISGJPARM issues message IEA301I and prompts the operator for a new GRSCNFxx member with message IEA906A. If the operator specified an end-of-block (EOB) and a default member does not exist, ISGJPARM goes to step 5 to build a global resource serialization CTC-driver control card table (GCC) which indicates the error.	IEAPMNIP	RDRTNEOR
Note: If the read processing is successful, IEEMB887 calls exit routines in ISGJPARM according to the data found in the GRSCNFxx SYS1.PARMLIB member.	ISGJPARM	PROCSTAT MTSYSFD RSMLFD CTCFD RESTFD
5 If no errors have occurred, ISGJPARM builds a GCC control block. This GCC control block contains all the information specified on the GRSDEF control statement that matches the system. ISGJPARM stores the UCB address of the CTC adapter for the system in the GCC control block.	IOSLOOK	
If errors exist, ISGJPARM builds a GCC control block to contain the appropriate error flag.	GETMAIN	
ISGJPARM then returns to the caller.		

Recovery Processing

None.

Diagram 23. Global Resource Serialization Resource Name List Parmlib Processor (ISGNRNLP) Part 1 of 6

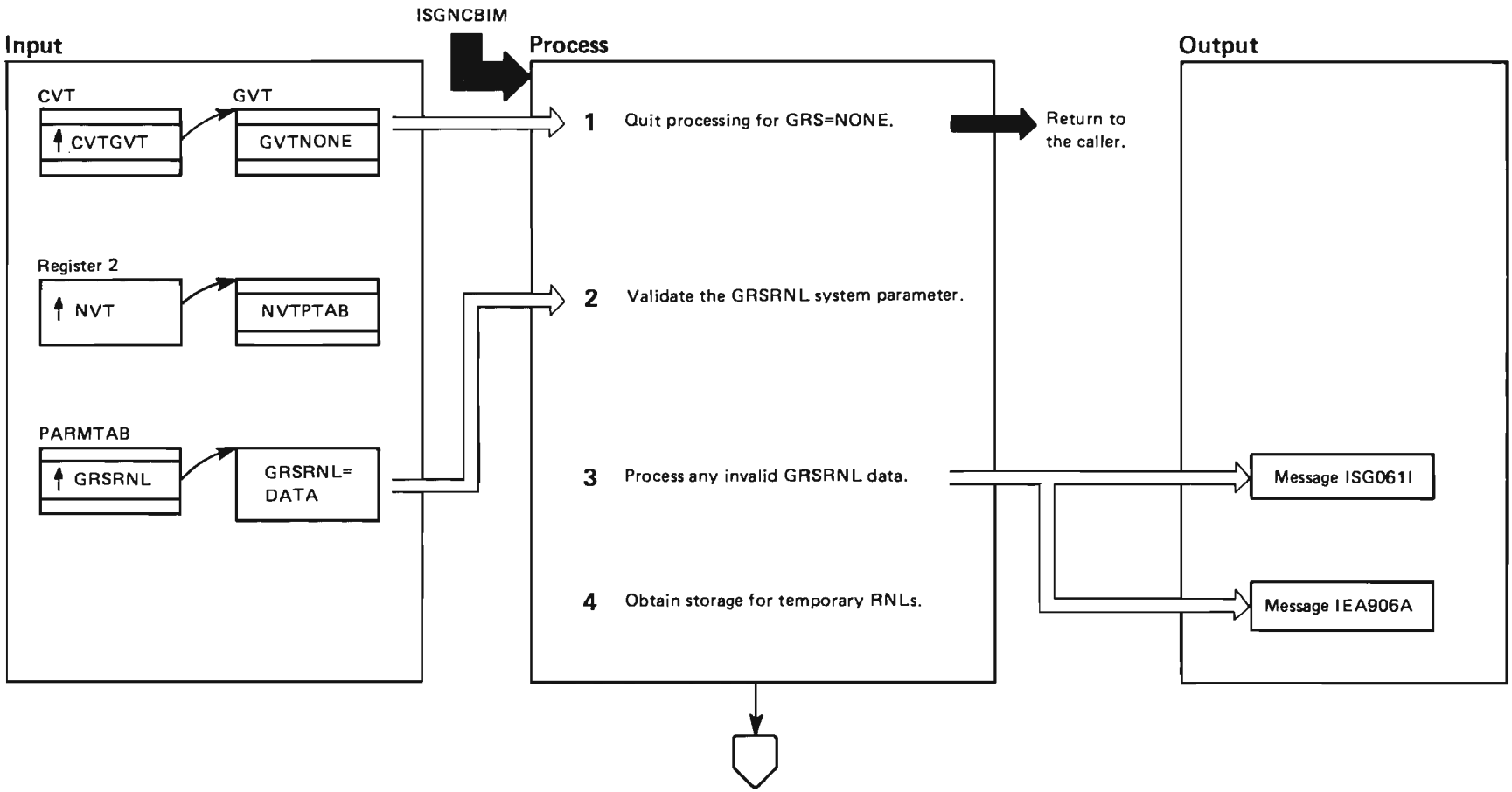


Diagram 23. Global Resource Serialization Resource Name List Parmlib Processor (ISGNRNLP) Part 2 of 6

Extended Description	Module	Label
ISGNRNLP validates the GRSRNL=(xx,yy,. . .) system parameter, parses the GRSRNL member(s) of SYS1.PARMLIB, and builds the resource name lists (RNLs) in SQA storage. ISGNRNLP resides in the IEAVNP23 load module.		
1 If the GRS=NONE was specified (GVTNONE='1'), then this system is not to be part of a global resource serialization complex. ISGNRNLP returns to the caller.	ISGNRNLP	
2 ISGNRNLP verifies the data specified for the GRSRNL= system parameter. The (xx,yy,. . .) must consist of two alphameric characters and be syntactically correct.	ISGNRNLP	VALDTEXX
3 If the GRSRNL= system parameter is invalid, ISGNRNLP issues message ISG061I and prompts the operator with message IEA906A to re-enter the system parameter correctly. When control is returned, processing continues at step 2. (*Note:* ISGNRNLP keeps prompting the operator until the system parameter is valid.)	IEAPMNIP	
If the GRSRNL= system parameter is valid, processing continues at step 4.	ISGNRNLP	GETSTOR
4 ISGNRNLP obtains (via a GETMAIN macro instruction) a buffer in subpool 127 for each resource name list (RNL).		

Diagram 23. Global Resource Serialization Resource Name List Parmlib Processor (ISGNRNLP) Part 3 of 6

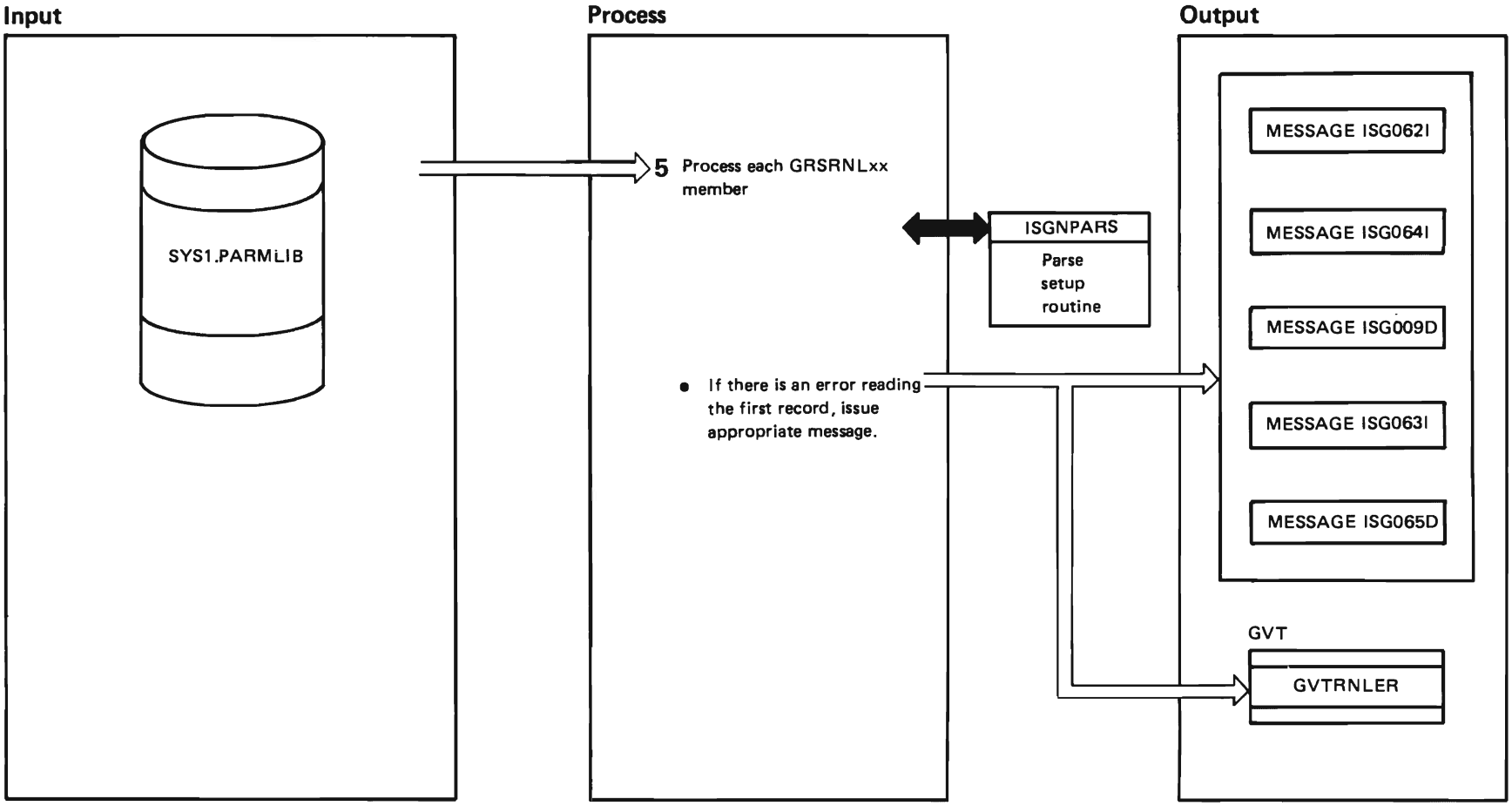


Diagram 23. Global Resource Serialization Resource Name List Parmlib Processor (ISGNRNLP) Part 4 of 6

Extended Description	Module	Label
5 For each GRSRNL=(xx,yy, . . .) system parameter specified, ISGNRNLP calls the global resource serialization parse setup module (ISGNPARS), which uses the NIP service routine, or IEAVNPM4 IEAPSIO, to read the first GRSRNLxx SYS1.PARMLIB member record. (The suffix, 'xx', represents one xx,yy,. . . pair specified for the GRSRNL= system parameter.) If the read was successful, ISGNPARS invokes the generalized parser (IEEMB887) to parse the specified GRSRNLxx SYS1.PARMLIB member.	ISGNPARS	
If the first record of the member was not read successfully, ISGNPARS calls a subroutine (RDRTNEOR) in ISGNRNLP to process the error as follows:	ISGNRNLP	RDRTNEOR
• If the GRSRNLxx member does not exist, RDRTNEOR does the following (message IEA301I has already been sent to the operator):		
– ISGNRNLP prompts the operator with message ISG065D to either reload the system or reply 'U' or 'NONE'. ISGNRNLP validates the reply.		
– If the operator responded with an invalid reply, ISGNRNLP issues the message ISG064I (indicating an invalid reply) and repeats the previous prompt until the reply is valid.		
– If 'NONE' was specified, ISGNRNLP sets GVTRNLER='1'. Processing continues with step 9.		
– If 'U' was specified, ISGNRNLP processes the next GRSRNLxx member.	IEAPMNIP	RDRTNEOR
• If an I/O error occurred that prevented the system from reading the GRSRNLxx SYS1.PARMLIB member, ISGNRNLP issues message ISG062I and does the following:		
– ISGNRNLP prompts the operator with message ISG009D to either reload the system or reply 'NONE'. ISGNRNLP validates the reply.	IEAPMNIP	
– If the operator responded with an invalid reply, ISGNRNLP issues message ISG064I (indicating an invalid reply) and repeats the previous prompt until the reply is valid.	IEAPMNIP	
– If 'NONE' was specified, ISGNRNLP sets GVTRNLER='1'. Processing continues with step 9.		
Note: If the processing is successful, IEEMB887 calls exit routines in ISGNRNLP to build the SYSTEM inclusion RNL, SYSTEMS exclusion RNL, and RESERVE conversion RNL based on the contents of each GRSRNLxx SYS1.PARMLIB member. ISGNRNLP builds these RNLs temporarily in subpool 127. (ISGNRNLP excludes duplicate entries in the same RNL.) If the RNLDEF statement contains LINKLIB=YES, ISGNRNLP copies the RNLs from SYS1.LINKLIB, rather than SYS1.PARMLIB.	ISGNRNLP	BLDLIST CONFND EXCLFND GENFND INCLFND LNKLBFND QNAMFNDE QNAMFNDX RNAMFNDE RNAMFNDX SPECFND

Diagram 23. Global Resource Serialization Resource Name List Parmlib Processor (ISGNRNLP) Part 5 of 6

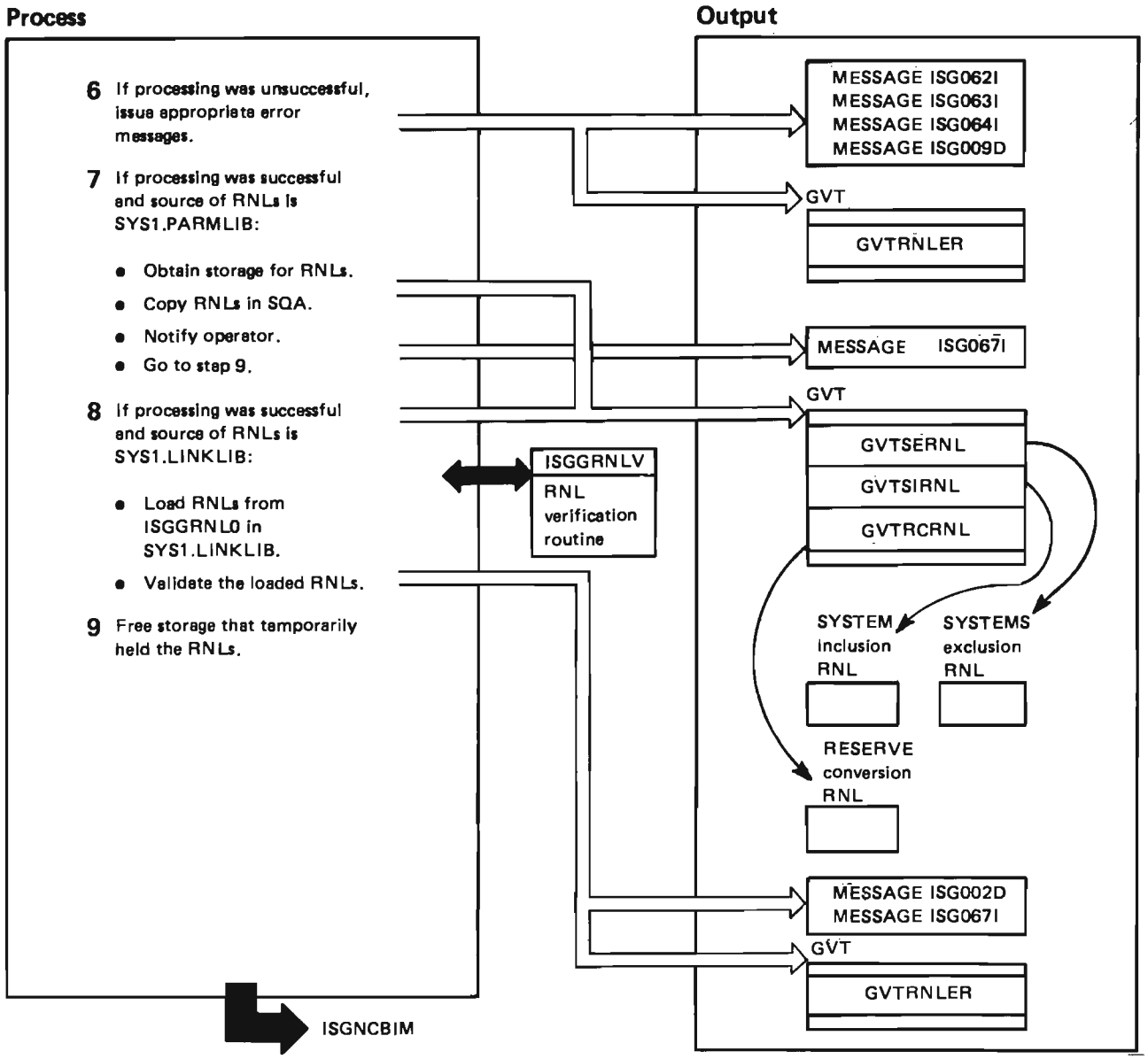


Diagram 23. Global Resource Serialization Resource Name List Parmlib Processor (ISGNRNLP) Part 6 of 6

Extended Description	Module	Label
6 ISGNRNLP checks the return code from ISGNPARS to determine if the processing of the GRSRNLxx member was successful. If GRSRNLxx was not processed successfully, ISGNRNLP processes the error as follows:	ISGNRNLP	ERRMSSG
• If a syntax error was found, ISGNRNLP issues message ISG063I.	IEAPMNIP	
• If an I/O error occurred which prevented the system from reading the GRSRNLxx SYS1.PARMLIB member, ISGNRNLP issues message ISG062I.	IEAPMNIP	
• After issuing the appropriate error message, ISGNRNLP prompts the operator to reload the system or reply 'NONE' with message ISG009D. ISGNRNLP validates the reply.	IEAPMNIP	
• If the operator responded with an invalid reply, ISGNRNLP issues the invalid reply message ISG064I and repeats the previous prompt until the reply is valid.	IEAPMNIP	
• If 'NONE' was specified, ISGNRNLP sets GVTRNLER='1'. Processing continues with step 7.		
If GRSRNLxx was processed successfully, the RNLs are to be built from SYS1.PARMLIB (LINKLIB = NO), and more GRSRNLxx SYS1.PARMLIB members were specified on the GRSRNL parameter, ISGNRNLP goes to step 5.	ISGNRNLP	
7 If all GRSRNLxx SYS1.PARMLIB members have been processed successfully and the RNLs are to be built from SYS1.PARMLIB, ISGNRNLP continues:		
• Obtains storage (via the GETMAIN macro instruction) for the RNLs in subpool 245.	ISGNRNLP	GETSTOR
• Copies the RNLs from SQA subpool 127 to subpool 245.	ISGNRNLP	COPYRNLS
• Informs the operator with message ISG067I that the source of the RNLs is SYS1.PARMLIB. Processing continues with step 9.	ISGNRNLP	FREESTOR
8 If all the GRSRNLxx SYS1.PARMLIB members are processed successfully and the RNLs are to be loaded from SYS1.LINKLIB (LINKLIB= YES), ISGNRNLP processes as follows:		
• ISGNRNLP obtains storage (via GETMAIN macro instruction with the LOC = (BELOW,ANY) operand) for ISGGRNL0 from subpool 245. ISGNRNLP then loads the SYSTEM inclusion, the SYSTEMS exclusion, and the RESERVE conversion RNLs (ISGGRNL0) into this storage.		
• ISGNRNLP calls the global resource serialization resource name list verification routine (ISGGRNLV) to validate the RNLs.	ISGGRNLV	
• If the RNLs are valid, ISGNRNLP informs the operator with message ISG067I that the source of the RNLs is SYS1.LINKLIB.		
• If any of the RNLs are invalid, ISGNRNLP informs the operator of the error and prompts the operator with message ISG002D to reload the system or to reply 'NONE'. If the operator's reply is invalid, ISGNRNLP repeats the message and the prompt until a valid reply is received. If 'NONE' was specified, ISGNRNLP sets GVTRNLER to 1.		
9 ISGNRNLP frees (via the FREEMAIN macro instruction) all of subpool 127 storage that was used to temporarily store the RNLs.		

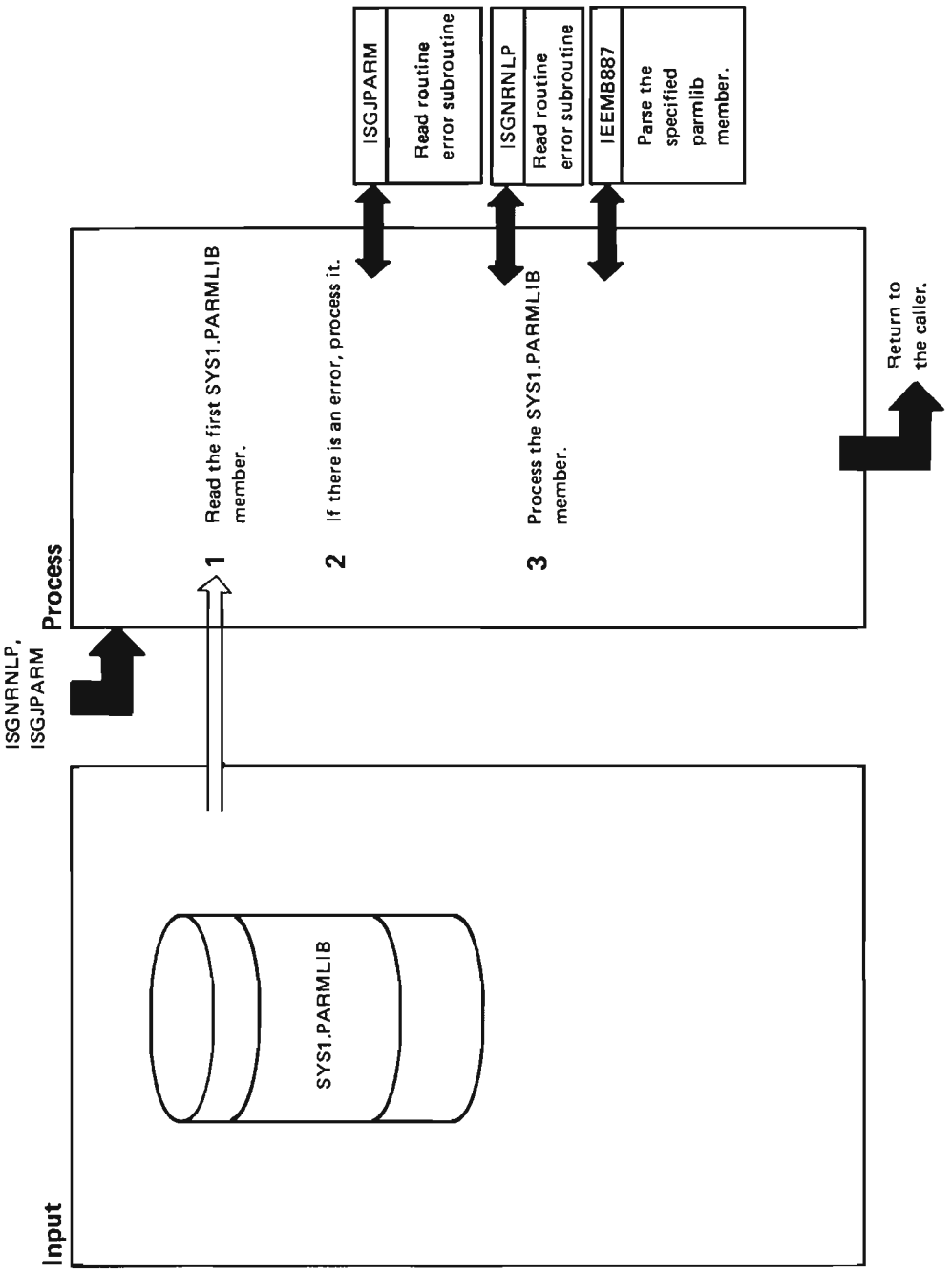


Diagram 24. Global Resource Serialization Parse Setup Routine (ISGNPARS) Part 1 of 2

Diagram 24. Global Resource Serialization Parse Setup Routine (ISGNPARS) Part 2 of 2

Extended Description	Module	Label
ISGNPARS initializes the parameter list for the generalized parser (IEEMB887) and invokes IEEMB887 to process the specified SYS1.PARMLIB member. ISGNPARS resides in load module IEAVNP23.		
1 ISGNPARS uses the NIP service routine (IEAVNPM4) to prepare the first record of the specified SYS1.PARMLIB member to be read.	IEAPMNIP	
2 If the specified SYS1.PARMLIB member is not found or there was an error encountered from reading the records of the SYS1.PARMLIB member, ISGNPARS notifies the read routine error routine (RDRTNEOR) to correct the error. (RDRTNEOR is a subroutine contained in the caller's routine.)	ISGJPARM ISGNRNLP	
If the error was corrected, ISGNPARS returns to step 1. If the error was not corrected, ISGNPARS returns to the caller.		
3 ISGNPARS initializes the scan parameter list (SCL) and then links to the generalized parser (IEEMB887) to process the specified SYS1.PARMLIB member. (See the Command Processing section of the *System Logic Library* for a description of IEEMB887.)		

Recovery Processing

None.

Diagram 25. Auxiliary Storage Management Initialization, Part 1 (IEAVNP04/ILRASRIM) Part 1 of 4

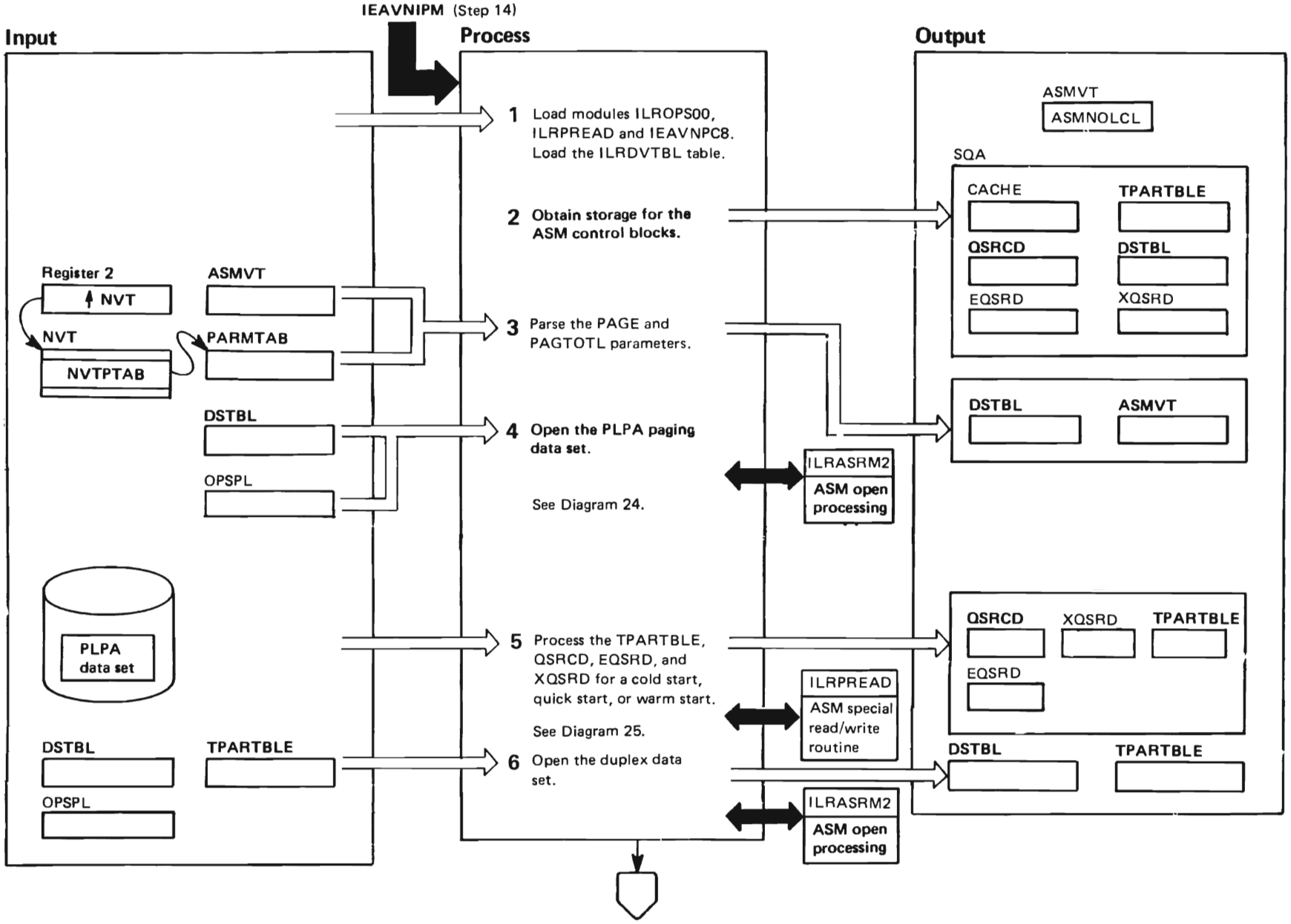


Diagram 25. Auxiliary Storage Management Initialization, Part 1 (IEAVNP04/ILRASRIM) Part 2 of 4

Extended Description	Module	Label
ILRASRIM builds and initializes ASM control blocks and opens the PLPA, common, and (if requested) duplex page data sets. IEAVNIPM enters ILRASRIM during NIP processing. ILRASRIM's normal exit is to IEAVNIPM.		
ILRASRIM initializes ASM for cold, warm and quick starts. Unless otherwise specified, the processing is the same all three types or starts. Both ILRASRIM and ILRASRM1 call ILRASRM2. ILRASRM2 contains service routines that parse data set names, call data set open modules, and fill in the TPARTBLE control block.		
Note that IEAVNIPM knows ILRASRIM as IEAVNP04. ILRASRIM is the entry point in the load module IEAVNP04.	ILRASRIM	RIMINIT
1 Load ILROPS00, ILRPREAD, and IEAVNPC8. In addition, load the ILRDVTBL table found in module, ILRASTBL. ILROPS00, ILRPREAD and ILRDVTBL are required to open, read, and write to the page and swap data sets.		
Set ASMNOLCL so that no local page data sets are used until ILRTMI00 executes. ILRTMI00 resets ASMNOLCL.		
2 Issue the GETMAIN macro instruction to obtain storage in ESQA (subpool 245) for the following:		
• the quick start record (QSRCD)		
• the extended quick start record (EQRSD)		
• the work area for the quick start record extension (XQSRD)		
• the temporary page activity reference table (TPARTBLE)		
• the cache array table (CACHE)		
• a data set work area (DSTBL)		
3 Use the input parameters in PARMTAB to locate and parse the input data. Scan and save the data for later use. Save the page data set names in the DSTBL. Use the IEAPMNIP service routine to prompt the operator to re-specify the PAGTOTL or PAGE parameters if they are invalid. Process the page data set name strings. Verify that the necessary data sets (PLPA, common, and at least one local page data set) are specified. If the necessary ones are not specified, end the IPL and issue message IEA935W to indicate that insufficient paging resources are specified.	ILRASRIM IEAPMNIP ILRIMMSG	

Extended Description	Module	Label
4 Invoke ILRASMR2 to open the first data set (the data set for PLPA) named in the DSTBL. If this is a cold or quick start, set the ILROPSPL reset flag to indicate that if the page data set resides on a cached auxiliary storage system, the ILRPREAD must reset the cache. Reset of the cached auxiliary storage subsystem that has a page data set causes a TPARTBLE flag (TPARTRAP) to be set to invalidate future warm starts until the system is sufficiently initialized to allow a warm start. If an error occurs during ILROPS00 processing, use ILRIMMSG to issue an appropriate error message.	ILRASMR2 ILRIMMSG	OPEN
5 If this is a cold start, initialize the TPARTBLE, QSRCD, and EQSRD. If this is a warm or quick start, read the first records of the PLPA into storage. (The first records of the PLPA data set contain the TPARTBLE, QSRCD, and EQSRD from the previous IPL.) If this read fails, issue message IEA939D to tell the operator to re-IPL or force a cold start. After the operator replies to IEA939D, issue message IEA929E to indicate a forced cold start. If the read of PLPA is successful, check the TPARTRAP flag. If this is a warm start and the flag is on, a quick start is forced. The TPARTRAP flag indicates incomplete initialization of page data sets during a previous IPL where a page data set was a cached auxiliary storage subsystem whose cache was reset. The TPARTRAP flag is set to indicate loss of cache data. The TPARTBLE is written to the PLPA data set.	ILRASRIM ILRPREAD ILRIMMSG	
6 The duplex data set is an optional page data set. The third entry in the DSTBL is reserved for the duplex data set name. If this is a cold start, store the duplex data set name. If this is a cold start, store the duplex data set parameter data in the DSTBL. If this is a warm or quick start and a duplex data set name is already in the TPARTBLE, move the name to the DSTBL. Open the duplex data set named in the DSTBL. If an error occurs, issue an appropriate message. If this is a quick or warm start and only some of the slots on the PLPA were present on the duplex data set (not on the PLPA data set), issue the warning message IEA927I and continue processing at step 7.	ILRASRIM ILRASRM2 ILRIMMSG	OPENDPLX OPEN

Diagram 25. Auxiliary Storage Management Initialization, Part 1 (IEAVNP04/ILRASRIM) Part 3 of 4

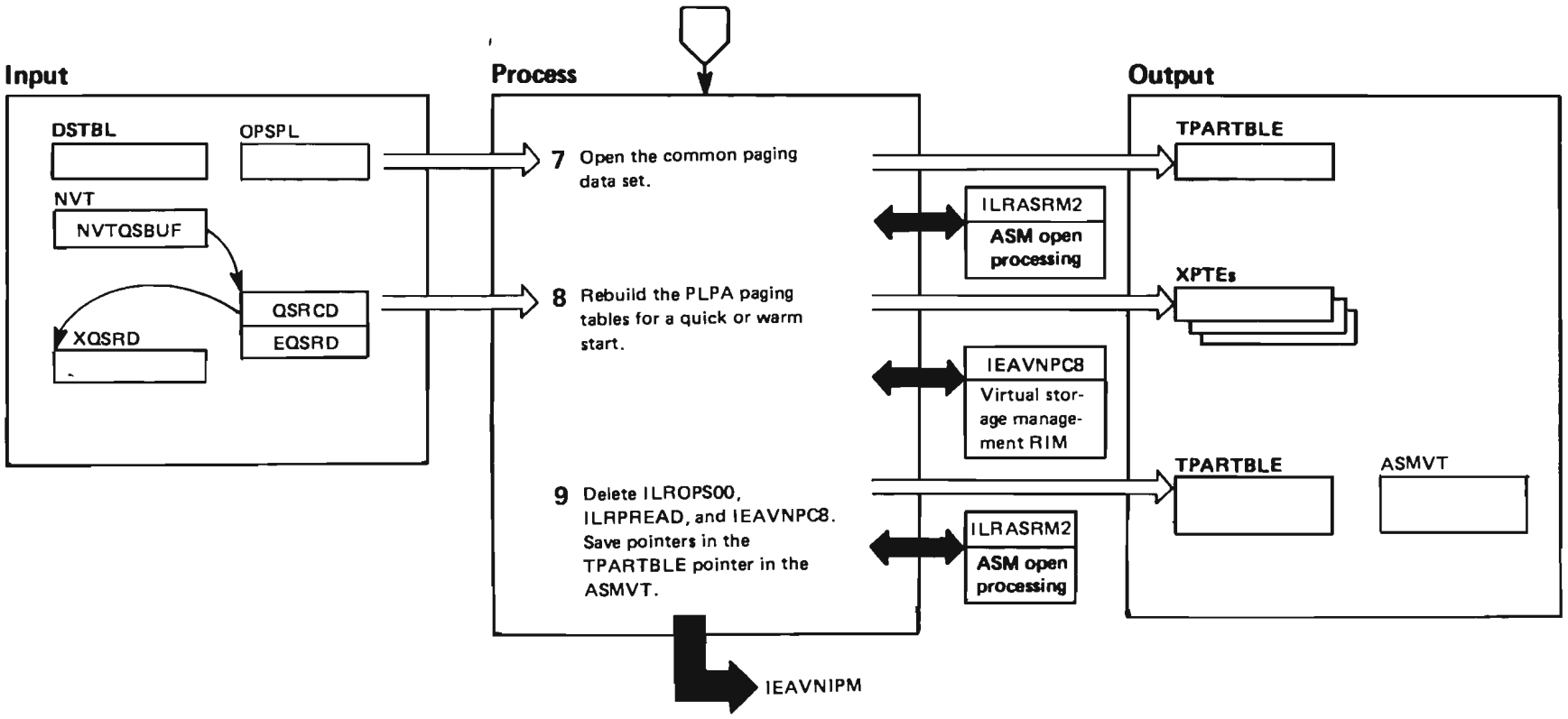


Diagram 25. Auxiliary Storage Management Initialization, Part 1 (IEAVNP04/ILRASRIM) Part 4 of 4

Extended Description	Module	Label
7 Invokes ILRASRM2 to open the second data set named in the DSTBL, the common data set. If this is a cold or quick start, sets the ILROPSPL reset flag to indicate that if the page data set resides on a cached auxiliary storage subsystem, ILRPREAD must reset the cache. Reset of the cached auxiliary storage subsystem that has a page data set causes a TPARTBLE flag (TPARTRAP) to be set to validate future warm starts until the system is sufficiently initialized to allow a warm start. If an error occurs, issues an appropriate message depending on the nature of the error resulting from the open processing. If no error occurs, continues with step 8.	ILRASRIM ILRASRM2 ILRIMMSG	OPENCOMM OPEN
8 If this is a cold start, continues with step 9. Otherwise, restores the PLPA and EPLPA to their states in the previous IPL. Reads the quick start record (QSRCD) and extended quick start record (EQSRD) from the PLPA page data set. The QSRCD and EQSRD were built during the previous cold start. (See the extended descriptions for IEAVNP05 and ILRQSRIT.)	ILRASRIM	RBLDPLPA
Fills in the CVTPLPAS, CVTPLPAE, and CVTVVMDI from the QSRCD. Fills in NVTONUCS, NVTONUCE, CVTEPLPS, and CVTEPLPE from the QESRD.		
Calls the virtual storage management (VSM) module, IEAVNPC8 (an entry point in IEAVNP08), to initialize the page tables (PGTs) and external page tables (XPTs) for the existing PLPA and EPLPA. IEAVNPC8 receives from the ASM RIM the starting addresses and lengths of the PLPA and EPLPA. It calculates the number of segments that must be initialized and builds a PGT and an XPT for each segment containing PLPA or EPLPA pages. As IEAVNPC8 initializes each PGT and XPT, it initializes the corresponding SGT entry with the real address of the page table and turns off the segment-invalid bit.	IEAVNPC8	
Upon return from IEAVNPC8, ILRASRIM uses the QSRCD and EQSRD maps of LSIDs to read the XQSRDs on the PLPA data set into storage. Each XQSRD contains maps of LSIDs for the PLPA and EPLPA pages on the PLPA data set. These LSIDs are used to restore the appropriate XPTLSIDs and turn on the valid bits in each XPT entry. The first reference to PLPA or EPLPA page causes a page-in from the slot indicated in the XPTE for that page. The PLPA and EPLPA are thus restored. If errors occur in this rebuilding process, issues an appropriate error message and forces a cold start.	ILRASRIM ILRIMMSG	
9 Prepares for the end of the first ASM RIM. Deletes modules ILROPS00, ILRPREAD, and IEAVNPC8. Saves these pointers and values in the TPARTBLE for use by the second ASM RIM (IEAVNP14/ILRASRM1). Saves the pointer to the TPARTBLE in the ASMVT.		
Note: Errors in ILRASRIM processing are handled by issuing messages that request the operator to specify or to allow alternative actions to take place. In some cases, messages are a notification of a problem and the action ASM takes to correct the problem. If a major error occurs during ILRASRIM processing and there are no alternative actions possible, then the system is put into a wait state with codes X'03C', X'060', X'061', or X'063'. In this case, the operator must attempt to correct the problem (for example a bad pack or data set) and then re-IPL.		
Recovery Processing		
None.		

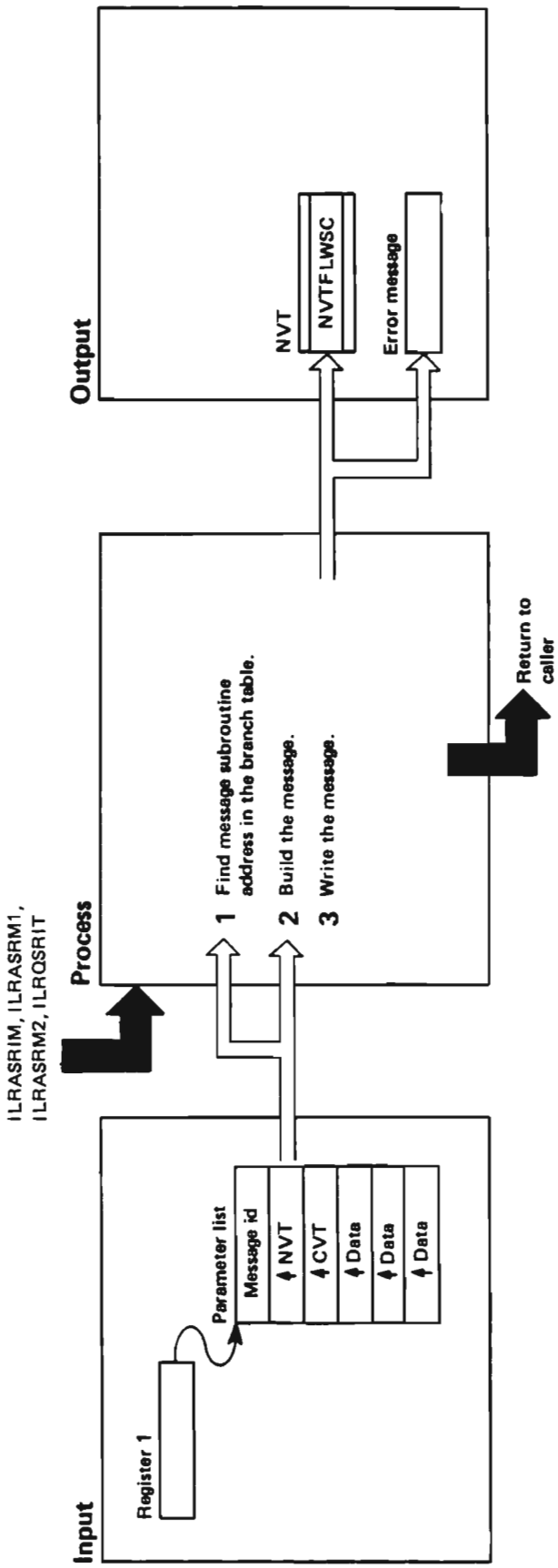


Diagram 26. Auxiliary Storage Management Message Module (ILRIMMSG) Part 1 of 2

Diagram 26. Auxiliary Storage Management Message Module (ILRIMMSG) Part 2 of 2

Extended Description	Module	Label
ILRIMMSG contains and writes all ASM RIM messages to the operator.		
1 Loop through an internal branch table within ILRIMMSG looking for a match to the input message id. When a match is found, invoke the internal subroutine that prepares the message.	ILRIMMSG	
2 Each of the internal subroutines builds the appropriate message content for the input message id. Build the message one line at a time.		
3 Use the IEAPMNIP NIP service routine to write out the messages. If a "W" type message is to be written, write the message first and then put the system into the proper wait state, setting the wait state code in the NVT (NVTFLWSC). If the message is not type "W", write the entire message and then return to the caller. If the message that is being written needs a reply, use IEAPMNIP with the type=WTOR option. Check for a reply and repeat issuing the message until a valid reply is given.	IEAPMNIP	
If an "I", "E", or "W" type message is to be written, write only the last line of the message.		
For the "W" type message, use the IEAPMNIP NIP service routine with the type=SWAIT option to write the message.		
Recovery Processing:		
None.		

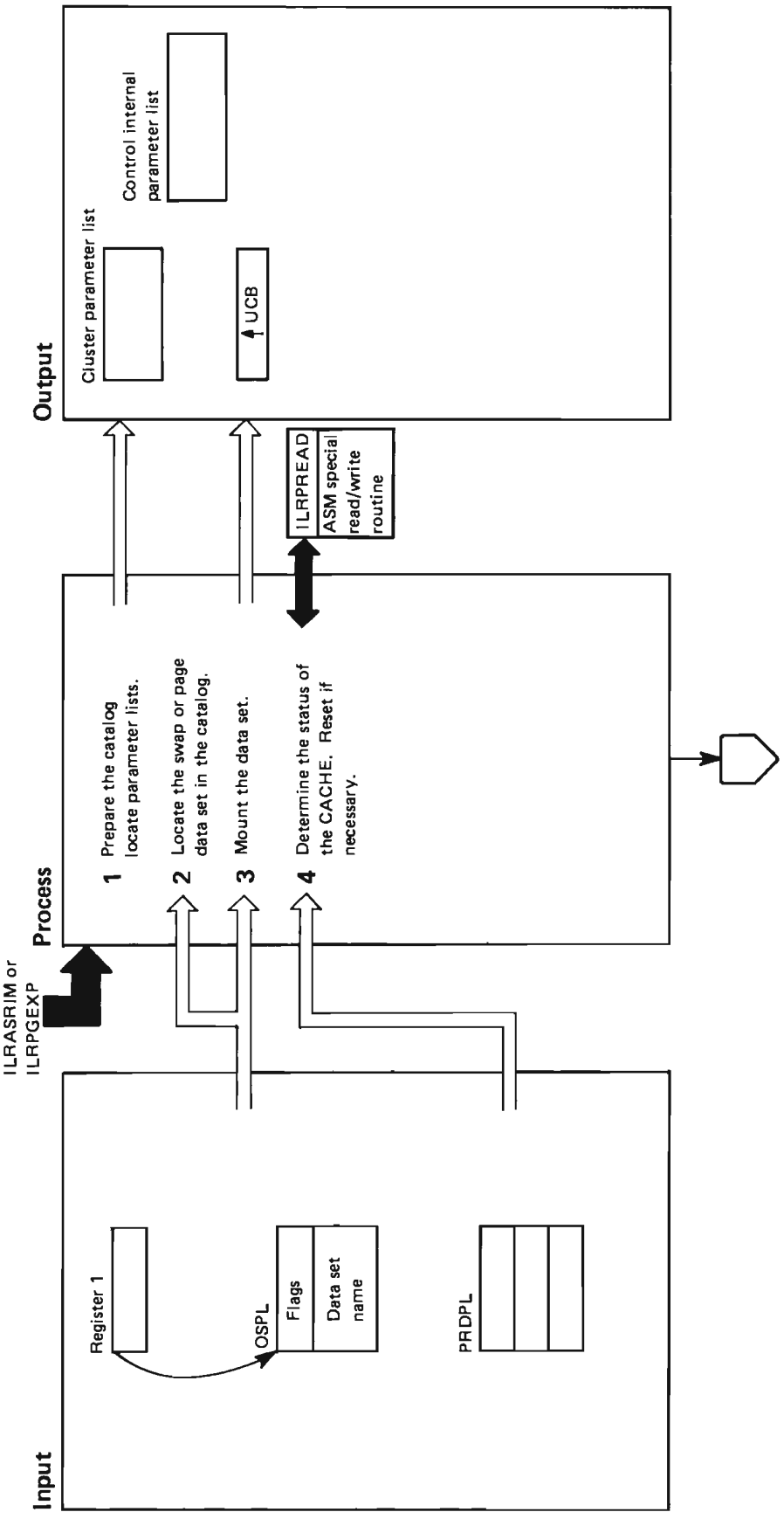


Diagram 27. Auxiliary Storage Management Open Processing (ILROPS00) Part 1 of 4

Diagram 27. Auxiliary Storage Management Open Processing (ILROPS00) Part 2 of 4

Extended Description	Module	Label
ILROPS00 opens a page or swap data set on behalf of the ASM RIM (ILRASRM2's open service routine, called by ILRASRIM or ILRASRM1) during system initialization or on behalf of ASM's page expansion processing (ILRPGEXP) after system initialization.		
1 ILROPS00 initializes the catalog locate parameter lists and catalog field parameter lists in order to perform a cluster locate and data control interval location, respectively, to obtain VSAM catalog information. These locates will return information such as data set type, volume serial number, volume attributes, device types, the address of the catalog ACB, and statistical data from the VSAM catalog.	ILROPS00	LOCPREP
2 For a page data set, ILROPS00 performs the cluster locate. If the cluster locate is successful, ILROPS00 issues a locate for the page data set's volume serial number, device type, attributes, and AMDSB.	ILROPS00	LOCPAGE
If either the cluster locate fails or the second locate fails, ILROPS00 returns to the caller with an error return code of 12.		
For a swap data set, ILROPS00 performs the cluster locate. If the cluster locate is successful, ILROPS00 issues a locate for the swap data set's volume serial number, device type, data set attributes, and AMDSB.	ILROPS00	LOCSWAP
If the cluster locate fails or the second locate fails, ILROPS00 returns to the caller with an error return code of 12.		
3 ILROPS00 verifies that the page or swap data set is mounted and online. If ILROPS00 is being used during system initialization (that is, the caller is ILRASRIM), ILROPS00 uses the NIPMOUNT service routine to verify the mount status. If ILROPS00 is being used after system initialization (that is, the caller is ILRPGEXP), ILROPS00 uses dynamic allocation. If this mount verification is unsuccessful, ILROPS00 returns to the caller with an error return code of 8.	ILROPS00	VMTVER
4 If the data set is on a cached auxiliary storage subsystem, ILROPS00 calls ILRPREAD to find the cache element address. Then ILROPS00 performs one of the following:		
• If the cache element (CACE) indicates that the cache is bad and was in use, ILROPS00 cannot use the cache and the data set. Processing continues with step 8.		
• If the cache is not in use and (1) a PAGEADD request is being processed, (2) this is a cold start, or (3) this is a quick start, ILROPS00 calls ILRPREAD to reset the cache.	ILRPREAD	
• If the cache is not in use or this is a warm start, ILROPS00 calls ILRPREAD to test the cache. If the cache is bad and not in use (for warm starts only), ILROPS00 calls ILRPREAD a second time to reset the cache.		

Diagram 27. Auxiliary Storage Management Open Processing (ILROPS00) Part 3 of 4

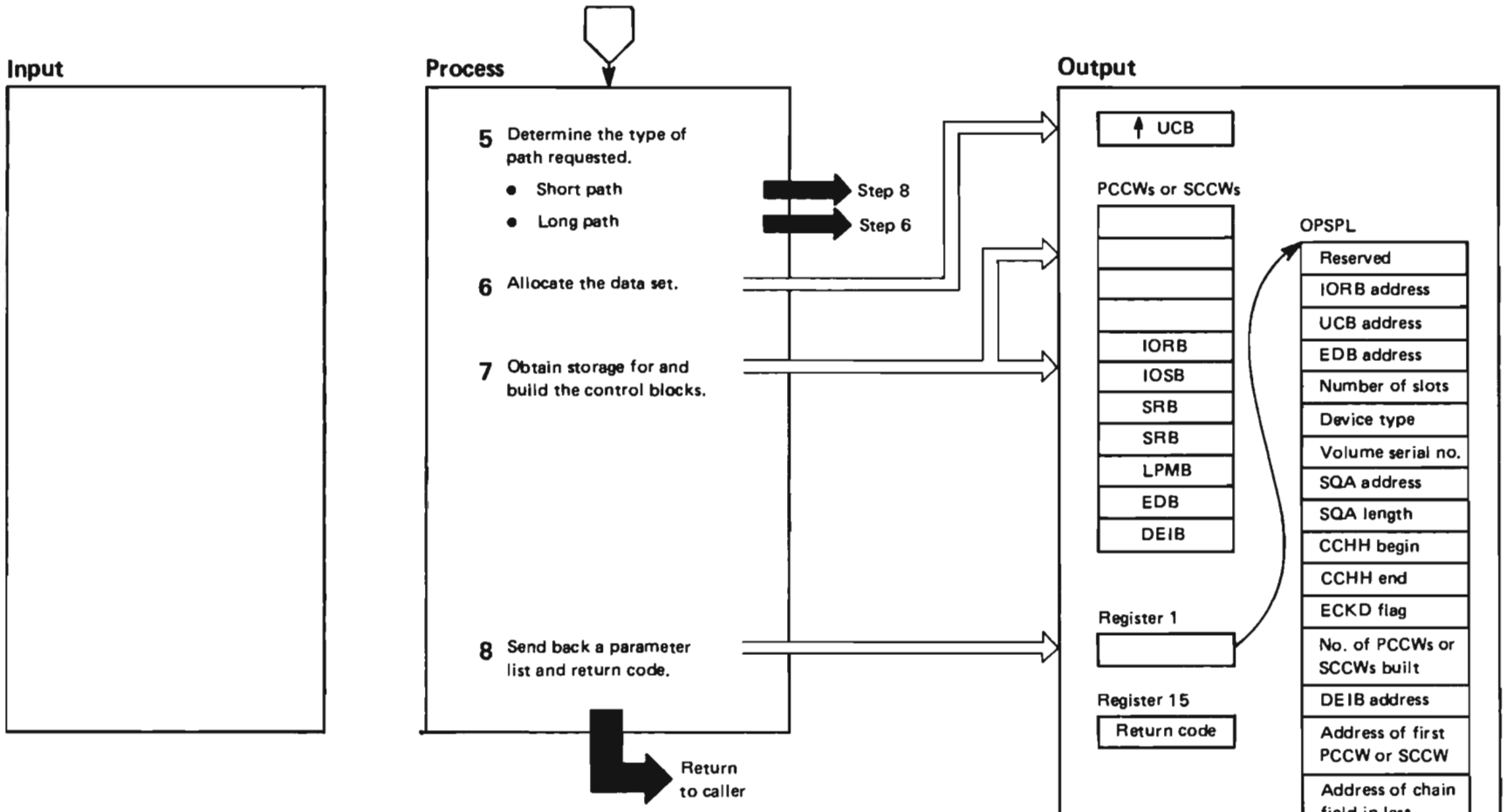


Diagram 27. Auxiliary Storage Management Open Processing (ILROPS00) Part 4 of 4

Extended Description	Module	Label
5 Two paths are possible through ILROPS00: a short path or a long path. The short path, requested in the flag settings of the input parameter list, is used during system initialization by ILRASRM1 (only for a warm start). With the short path, ILROPS00 returns to the caller (see step 8) at this point having successfully opened a page or swap data set. The short path is used to verify the availability of "needed local" page data sets before control blocks are built. If a "needed local" page data set is not available, the warm start fails. The long path, requested in the flag settings of the input parameter list, is by both ILRASRIM, ILRASRM1, and ILRPGEXP to open a page or a swap data set and build the required ASM control blocks. Continues processing with step 6.	ILROPS00	GETCORE
6 ILROPS00 allocates the data set. If ILROPS00 is being used during system initialization (that is, that caller is ILRASRIM or ILRASRM1), ILROPS00 calls the dynamic allocation NIP service routine. If ILROPS00 is being used after system initialization (that is, the caller is ILRPGEXP), ILROPS00 sets UCBPRES to 1. The dynamic allocation call (in step 3) sets the other UCB bits for allocation.		
7 If the UCB (and its associated control blocks) for the device on which the data set resides indicates that extended count, key, and data (ECKD) is allowed, ILROPS00 turns on the ECKD flag and puts the CCHH beginning and ending address in the output parameter list. ILROPS00 calculates the amount of extended SQA and nucleus buffer area required to build the I/O control blocks. ILROPS00 issues the GETMAIN macro instruction to obtain storage from subpool 245. If the GETMAIN is unsuccessful, ILROPS00 sets a return code of 16 (for SQA space not available). If the GETMAIN is successful, ILROPS00 builds PCCWs if processing a page data set, or SCCWs if processing a swap data set, and builds the other I/O control blocks (IORB, EDB, IOSB, SRBs, LPMB) in SQA.	ILROPS00	GETCORE
8 If processing in steps 1 thru 7 is successful, ILROPS00 returns to the caller with a return code of zero and an initialized parameter list containing the open information for the page or swap data set. If processing in steps 1 thru 7 was unsuccessful, ILROPS00 returns to the caller with a nonzero return code.	ILROPS00	
Recovery Processing None.		

Diagram 28. Auxiliary Storage Management Special Read/Write (ILRPREAD) Part 1 of 6

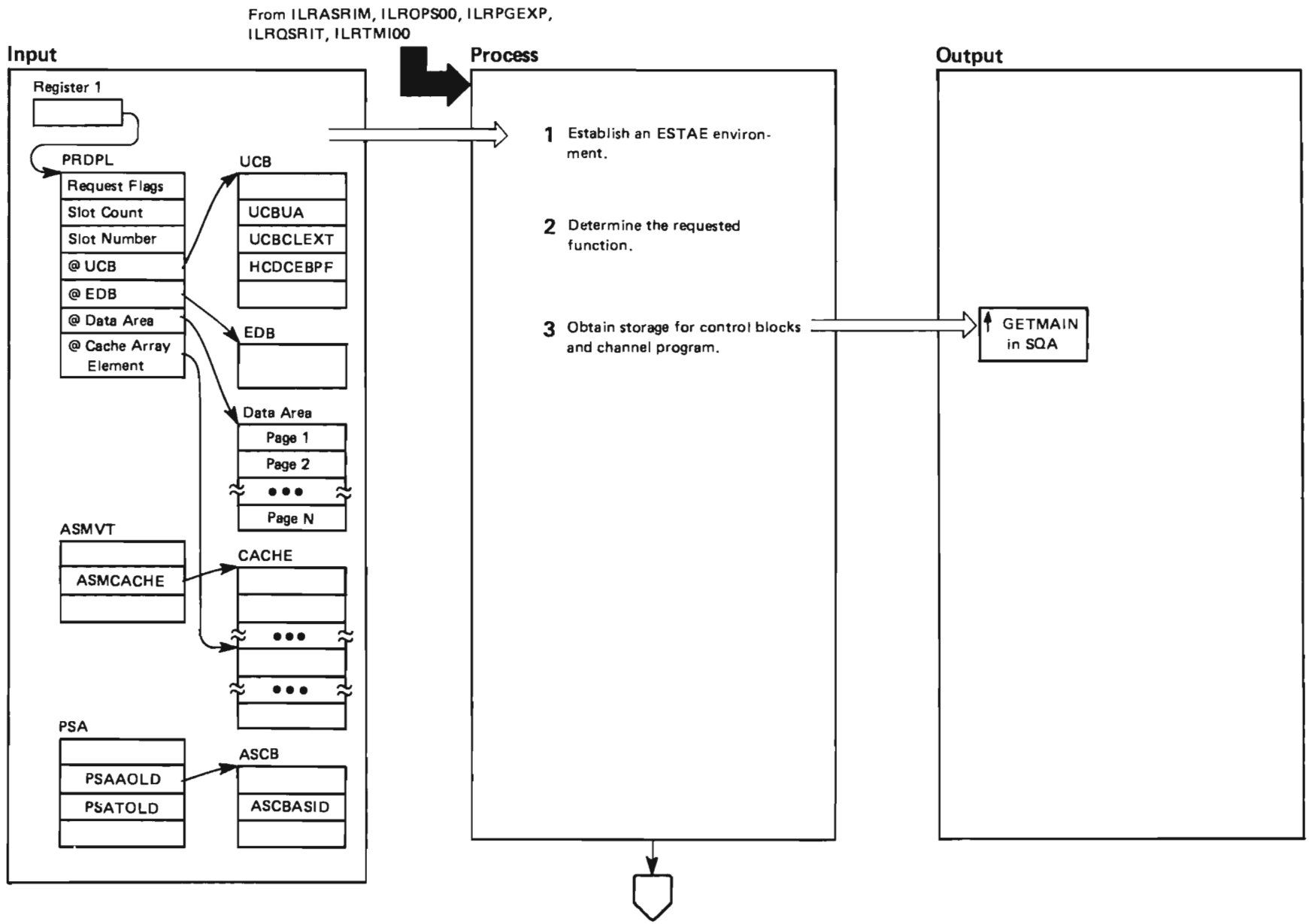


Diagram 28. Auxiliary Storage Management Special Read/Write (ILRPREAD) Part 2 of 6

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

ILRPREAD is an I/O driver that reads and writes slots containing control blocks and other information that ASM needs either during system initialization or when page and swap data sets are added after initialization. ASM initialization reads and writes ILRTPARB (the TPARTBLE), the quick start record (QSRCD), the extended quick start record (EQSRD) and the quick start record extensions (XQSRDs) on the PLPA data set and record 0 time stamps on the common and duplex data sets. When page and swap data sets are added after initialization, ASM uses ILRPREAD to read and write the TPARTBLE. ASM needs this information for quick and warm starts. In order for ASM to use page and swap data sets defined on a cached auxiliary storage subsystem, ILRPREAD also obtains the address of the cache array element and tests or resets the cache as specified on input. If reset is required after the cache has been tested or reset is requested, ILRPREAD reinitializes the directories and control information used by the cache.

1 Establishes an ESTAE environment for abnormal termination. Note that the ESTAE routine gets control only when page or swap data sets are added; it does not get control for abnormal termination during system initialization.

2 ILRPREAD determines the requested function by examining the request flag. The request flags also indicate which parameters contain valid information and the length of the channel program required to satisfy the request. The request flags and the valid parameters are:

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

2 *(continued)*

Significant Input Parameters						
	Function					
Parameter	Read Slots	Write Slots	Test Cache	Reset Cache	Obtain Cache Array Element Address	Obtain slot for XQSRD on PLPA
Request Flag	'80'X	'40'X	'20'X	'10'X	'08'X	'04'X
Slot Count	Yes	Yes	No	No	No	No
Slot Number	Yes	Yes	No	No	No	No
@ UCB	Yes	Yes	Yes	Yes	Yes	No
@ EDB	Yes	Yes	No	No	No	No
@ Data Area	Yes	Yes	No	No	No	Yes
@ Cache Array Element	No	No	Yes	Yes	No	No

3 ILRPREAD obtains storage for an IOSB, an SRB, a save area for both control blocks, and for the channel program.

Diagram 28. Auxiliary Storage Management Special Read/Write (ILRPREAD) Part 3 of 6

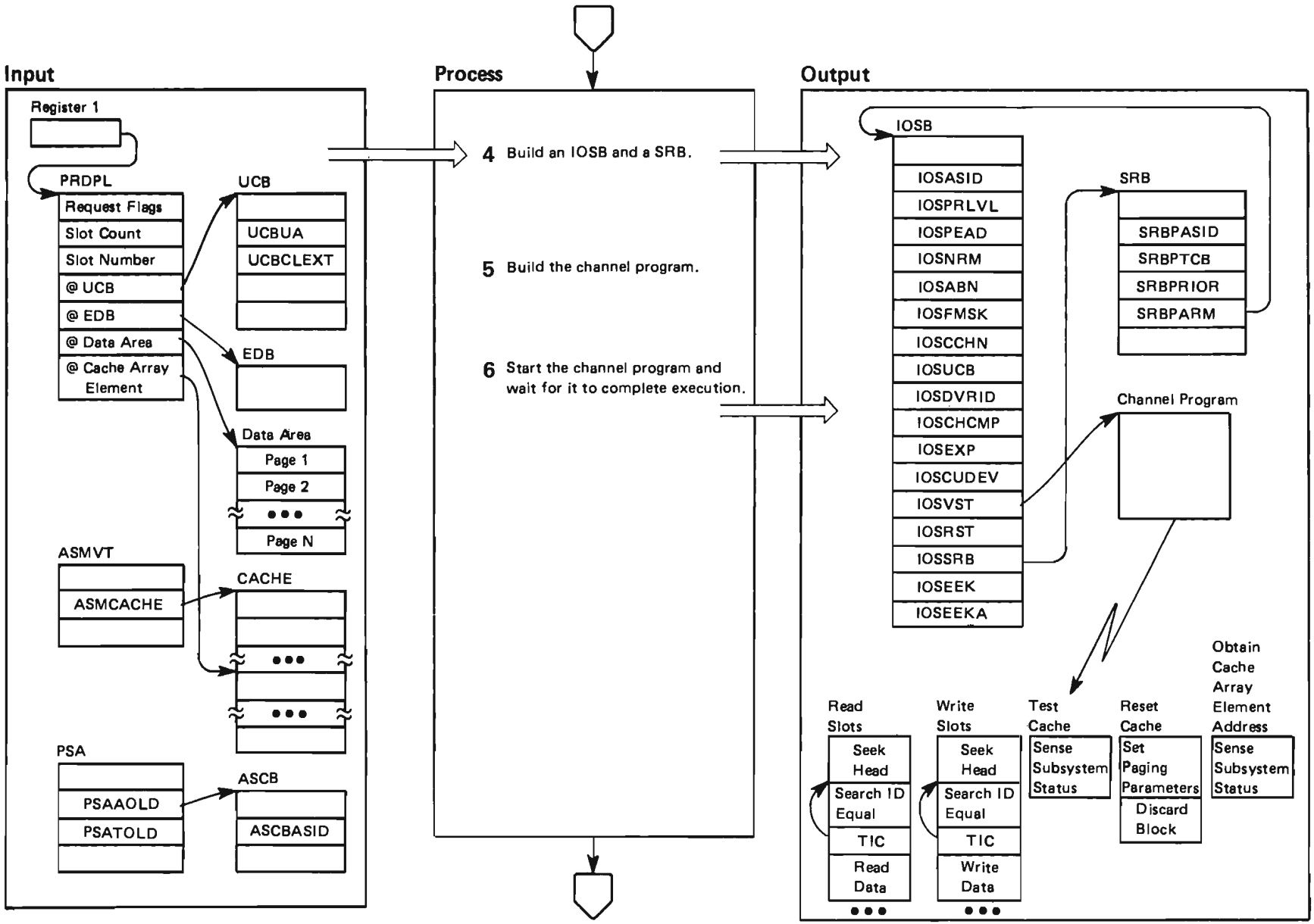


Diagram 28. Auxiliary Storage Management Special Read/Write (ILRPREAD) Part 4 of 6

Extended Description	Module	Label
4 The address of normal end appendage, abnormal end appendage, and I/O termination are placed in the IOSB. These routines are secondary entry points in ILRPREAD. The address of the TCB and the ASID are also placed in the IOSB. Because the TCB and the ASCB are identified in the IOSB, RTM processing cleans up any ILRPREAD resources and outstanding I/O if errors occur in IOS or the ILRPREAD appendages. The IOSB also indicates whether the channel program requires an IOS-provided prefix and a specific device exposure. The SRB address is also placed in the IOSB, and the TCB address is placed in the SRB.		
5 Build the channel program that performs the function identified in step 2:		
• To read slots, the channel program consists of one or more sets of the following CCW sequence: SEEK HEAD SEARCH ID EQUAL TIC READ DATA IOS supplies a prefix for the channel program. That is, IOS prefixes the channel program with CCWs.		
• To write slots, the channel program consists of one or more of the following CCW sequence: SEEK HEAD SEARCH ID EQUAL TIC WRITE DATA IOS provides a prefix for the channel program. That is, IOS prefixes the channel program with CCWs:		
• To test a cache, the channel program consists of the following CCW sequence: SENSE SUBSYSTEM STATUS NO-OPERATION IOS does not supply a prefix for the channel program.		

Extended Description	Module	Label
5 *(continued)*		
• To reset a cache, the channel program consists of the following CCW sequence: SET PAGING PARAMETERS DISCARD BLOCK IOS does not supply a prefix for the channel program.		
• To obtain the address of a cache array element, the channel program consists of the following CCW sequence: SENSE SUBSYSTEM STATUS NO-OPERATION IOS does not supply a prefix for the channel program.		
6 Start the channel program and wait for it to complete. ILRPREAD uses the STARTIO macro instruction to pass the address of the SRB to IOS. Note the SRB addresses the IOSB which in turn, points to the channel program.		

Diagram 28. Auxiliary Storage Management Special Read/Write (ILRPREAD) Part 5 of 6

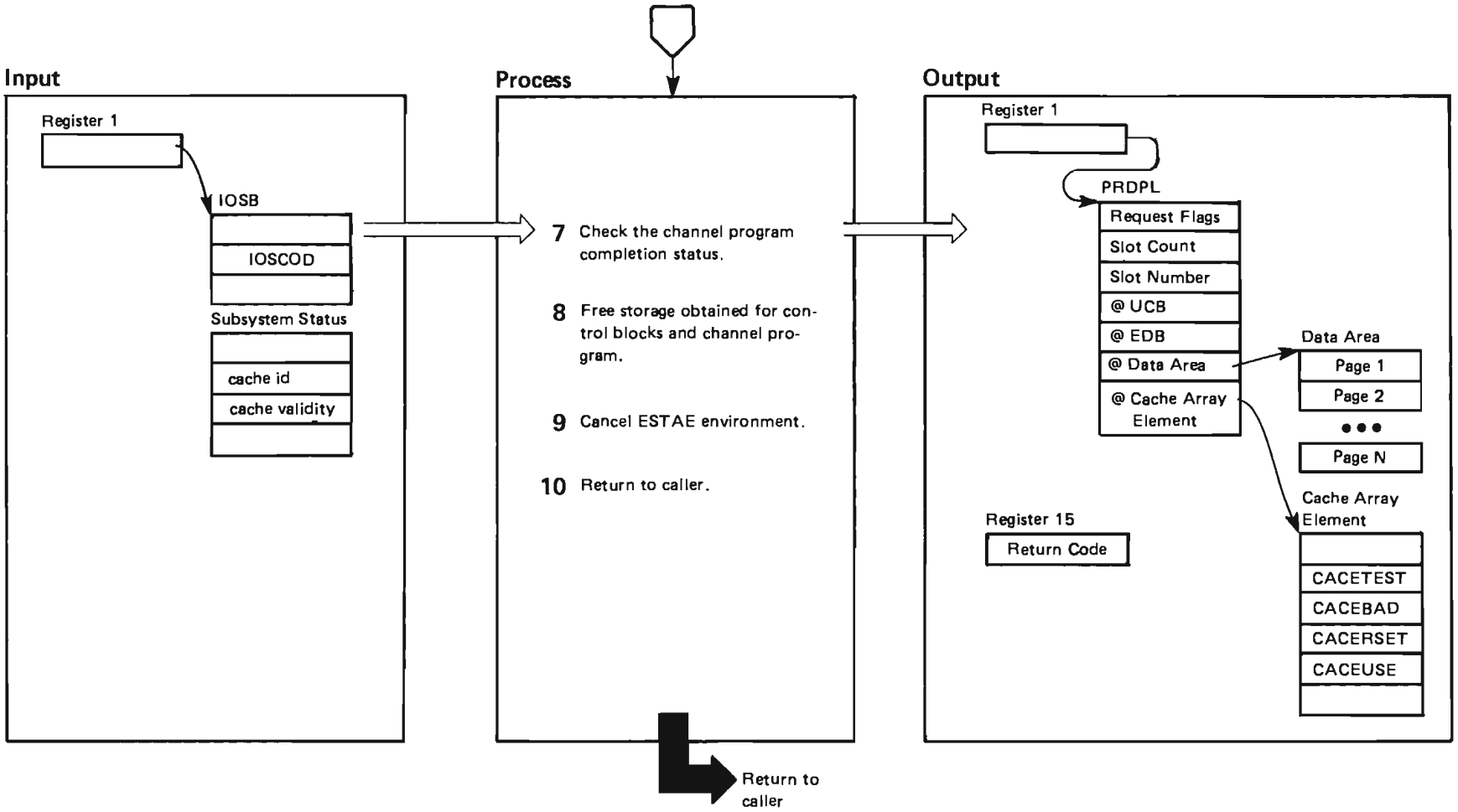


Diagram 28. Auxiliary Storage Management Special Read/Write (ILRPREAD) Part 6 of 6

Extended Description	Module	Label

7 Check the channel program completion status. When the channel program completes successfully, the completion codes are as follows:

Function	*Action*
Read slots	Return code = 0
Write slots	Return code = 0
Test cache	CACETEST = 1 If cache is valid, CACEBAD = 0 and the return code = 0 If cache is invalid, CACEBAD = 1 and the return code = 8 If cache is valid but not in use, CACEUSE = 1
Reset cache	CACETEST = 1 CACERSET = 1 CACEBAD = 0 CACEUSE = 1 Return code = 0
Obtain cache array entry address	Calculate cache array entry address Return code = 0

When the channel program completes unsuccessfully, the completion codes are as follows:

Function	*Action*
Read slots	Return code = 8
Write slots	Return code = 8
Test cache	CACETEST = 1 CACEBAD = 1 Return code = 8
Reset cache	CACETEST = 1 CACEBAD = 1 Return code = 8
Obtain cache array entry address	Return code = 8

Extended Description	Module	Label

8 ILRPREAD frees the storage it previously acquired to execute the channel program.

9 ILRPREAD cancels the ESTAE environment.

10 ILRPREAD returns to its caller the address of the parameter list in register 1 and the return code in register 15.

Recovery Processing

The ESTAE routine frees the storage ILRPREAD acquired to execute the channel program, sets a return code = 0, and requests percolation. — ILRPREAD — ESTAEXIT

Normal End Appendage

Returns to the caller. — ILRPREAD — PREADNRM

Abnormal End Appendage

Returns to the caller. — ILRPREAD — PREADABN

I/O Termination Routine

Notifies ILRPREAD (Step 7) that the IOS interface has completed and releases the local lock IOS obtained to build the channel programs (Step 5). — ILRPREAD — PREADTRM

Diagram 29. SQA Parameter Initialization (IEAVNPA8) Part 1 of 6

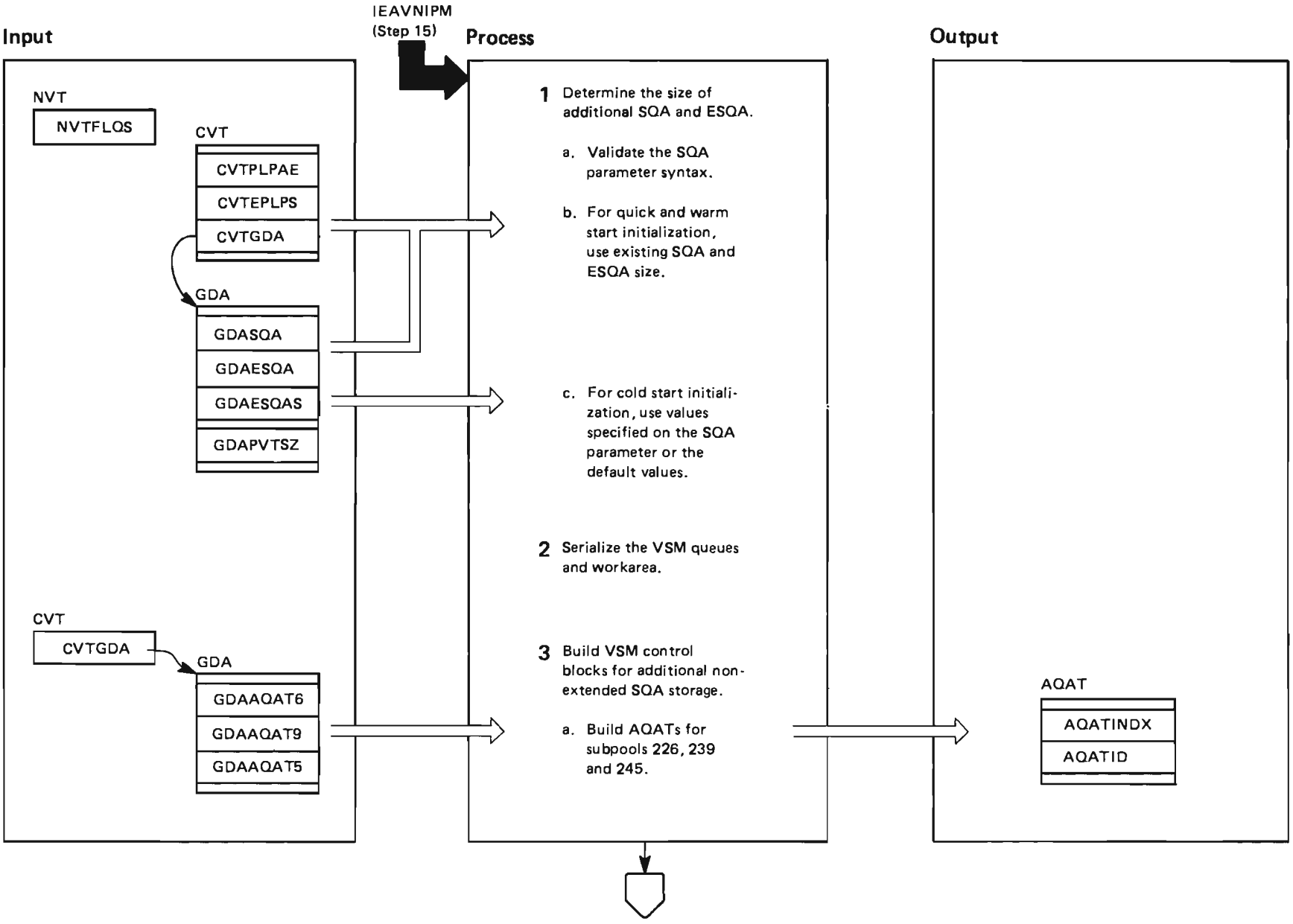


Diagram 29. SQA Parameter Initialization (IEAVNPA8) Part 2 of 6

Extended Description	Module	Label
IEAVNIPM calls the virtual storage management (VSM) RIM, IEAVNPA8, to process the SQA system parameter and build the VSM control blocks and page tables necessary to describe the additional system queue area (SQA) and extended SQA (ESQA). The form of the parameter is SQA=(a, b), where 'a' and 'b' are values that are multiplied by 64K to determine the amount of storage to be added to SQA and ESQA respectively. The VSM RIM adds the requested amount of SQA storage to the storage that IEAIPL04 allocated. When this RIM gets control, there is 192K of non-extended SQA storage and 192K + 8Mb of extended SQA storage. (IEAIPL04 allocated the 8Mb for common area page tables, therefore this area is not available for allocation in response to GETMAINs.)		
1 The VSM RIM checks NVTFLQs to determine if the initialization in progress is a quick start and validates the SQA parameter.	IEAVNPA8	IEAVNPA8
a. The VSM RIM validates the SQA parameter syntax and parses the parameter to obtain the 'a' and 'b' values.		
b. If a quick or warm start is in process, the size of the SQA is determined by the start address of the initial SQA (GDASQA) and the end address of the PLPA (CVTPLPAE). The size of the ESQA is determined by the end address of the ESQA (GDAESQA + GDAESQAS) and the start address of the extended PLPA (CVTEPLPS). If the operator specified an alternate nucleus or specified an 'a' or 'b' value different from the values that existed from a previous cold start, the VSM RIM issues message IEA908I to inform the operator that it is ignoring the keyword. The message also states the size values the RIM is using for the SQA and ESQA.		
c. If a cold start is in process, and the syntax is invalid, the RIM issues message IEA321I. This message informs the operator that the syntax is invalid and prompts the operator for new values. If the operator does not specify 'a' and 'b' values, or if the parameter is omitted, the VSM RIM uses the default values SQA=(1, 0). If the operator specified a size value that caused the SQA or ESQA to overlap into NIP's private area, the RIM issues message IEA909I to inform the operator that the value is too large. The RIM also prompts the operator for new values.		VALIDATE
2 The VSM RIM issues a SETLOCK macro instruction to obtain the LOCAL and VSMFIX locks. These locks serialize access to the VSM queues and the VSM workarea (VSWK).		IEAVNPA8
3 The VSM RIM initializes VSM control blocks for the new non-extended SQA storage.		
a. The RIM issues a GETMAIN macro instruction to obtain storage in the SQA for the AQATs that describe subpools 226, 239, and 245. It zeros the AQATs, sets the AQAT index table entries (AQATINDX) to point to the appropriate AQAT, and initializes the control block ID fields (AQATID).		BUILDAQT

Diagram 29. SQA Parameter Initialization (IEAVNPA8) Part 3 of 6

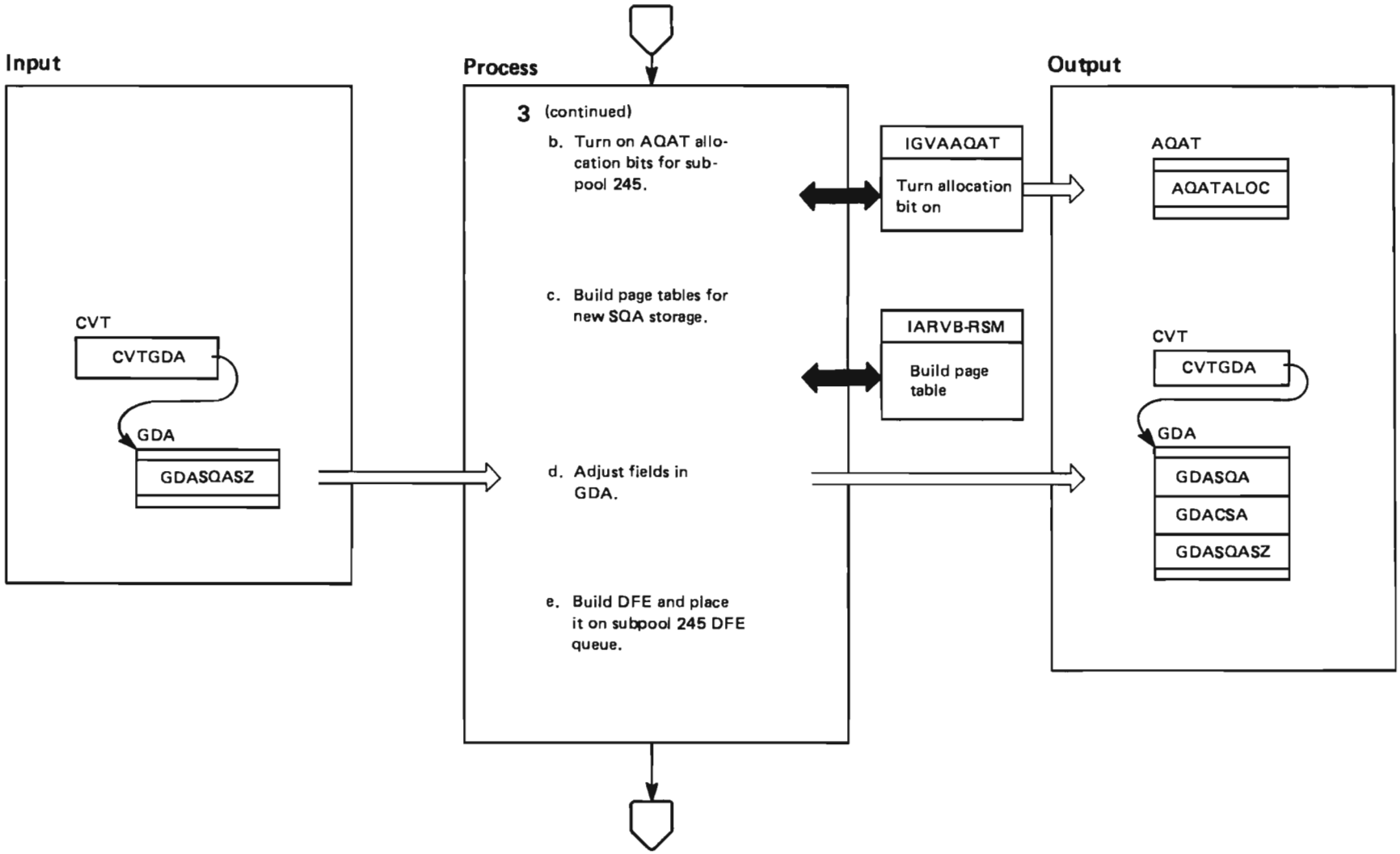


Diagram 29. SQA Parameter Initialization (IEAVNPA8) Part 4 of 6

Extended Description	Module	Label
3 (continued)		
b. The RIM calls IGVAAQAT to turn on the AQAT allocation bits for subpool 245. These bits indicate that the storage is part of subpool 245 and that it is available for use. Note that the allocation bits for subpools 226 and 239 are not turned on in this module. When a system component issues a GETMAIN macro instruction for storage in these subpools, VSM will obtain storage for the requested subpool from subpool 245 and turn on the corresponding AQAT bits.	IGVAAQAT	
c. For each new segment of SQA storage, the VSM RIM calls RSM module IARVB to build a page table. If there is not enough real storage available for a page table, the VSM RIM issues message IEA907W to inform the operator that it cannot initialize the page tables for the SQA. It then places the system in a wait state with a wait state code of X'03D'.	IARVB IEAVNPA8	IARVBCSG
d. The VSM RIM adjusts fields in the GDA to reflect the change in the SQA.		
e. The RIM issues a branch entry FREEMAIN macro instruction, which causes a double free element (DFE) to be built and enqueued on the non-extended subpool 245 queue.		

Diagram 29. SQA Parameter Initialization (IEAVNPA8) Part 5 of 6

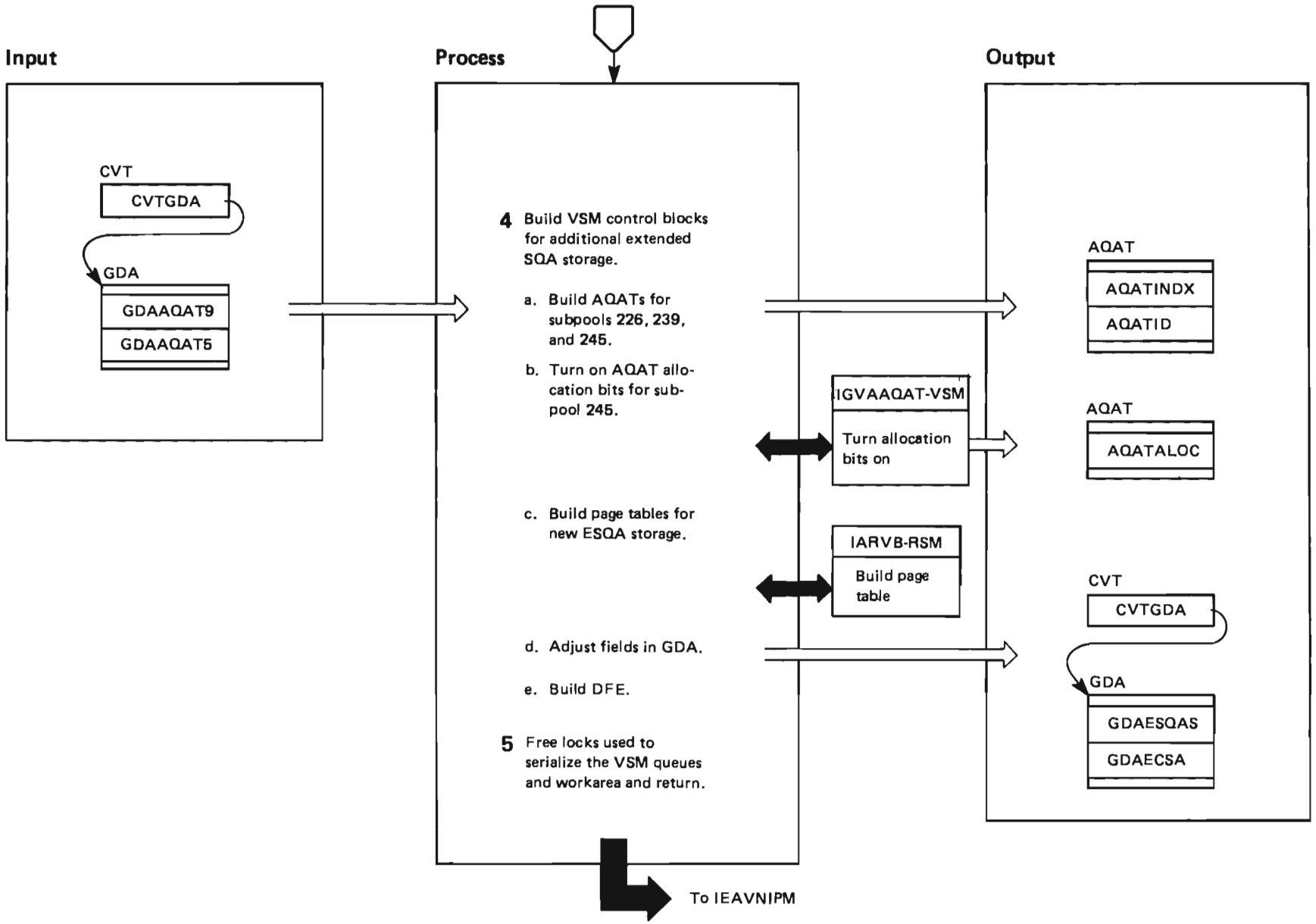


Diagram 29. SQA Parameter Initialization (IEAVNPA8) Part 6 of 6

Extended Description	Module	Label
4 The VSM RIM builds control blocks for the additional ESQA storage. To avoid using up the entire amount of initial SQA to build AQATs to describe the ESQA, the RIM repeats the following steps for each 8-megabyte block of requested ESQA storage.		
a. The RIM issues a GETMAIN macro instruction to obtain storage in the ESQA for the AQATs that describe subpools 226, 239, and 245. It zeroes the AQATs, sets the AQAT index table entries (AQATINDX) to point to the appropriate AQAT, and initializes the control block ID fields (AQATID).		BUILDAQT
b. The RIM calls IGVAAQAT to turn on the AQAT allocation bits for subpool 245. These bits indicate that the storage is part of extended subpool 245 and that it is available for use. Note that the allocation bits for subpool 239 is not turned on in this module. When a system component issues a GETMAIN macro instruction for storage in this subpool, VSM will obtain storage for the requested subpool from subpool 245 and turn on the corresponding AQAT bit.	IGVTAAQA	
c. For each new segment of ESQA storage, the VSM RIM calls RSM module IARVB to build a page table. If there is not enough real storage available for a page table, the VSM RIM issues message IEA907W to inform the operator that it cannot initialize the page tables for the ESQA. It then places the system in a wait state with a wait state code of X'03D'.	IARVB IEAVNPA8	IARVBCSG
d. The VSM RIM adjusts fields in the GDA to reflect the change in the ESQA areas.	IEAVNPA8	IEAVNPA8
e. The RIM issues a branch entry FREEMAIN macro instruction, which causes a double free element (DFE) to be built and enqueued on the extended subpool 245 queue.		
5 The VSM RIM issues a SETLOCK macro instruction to free the LOCAL and VSMFIX locks and returns control to IEAVNIPM.		

Diagram 30. Auxiliary Storage Management Initialization, Part 2 (IEAVNP14/ILRASRM1) Part 1 of 6

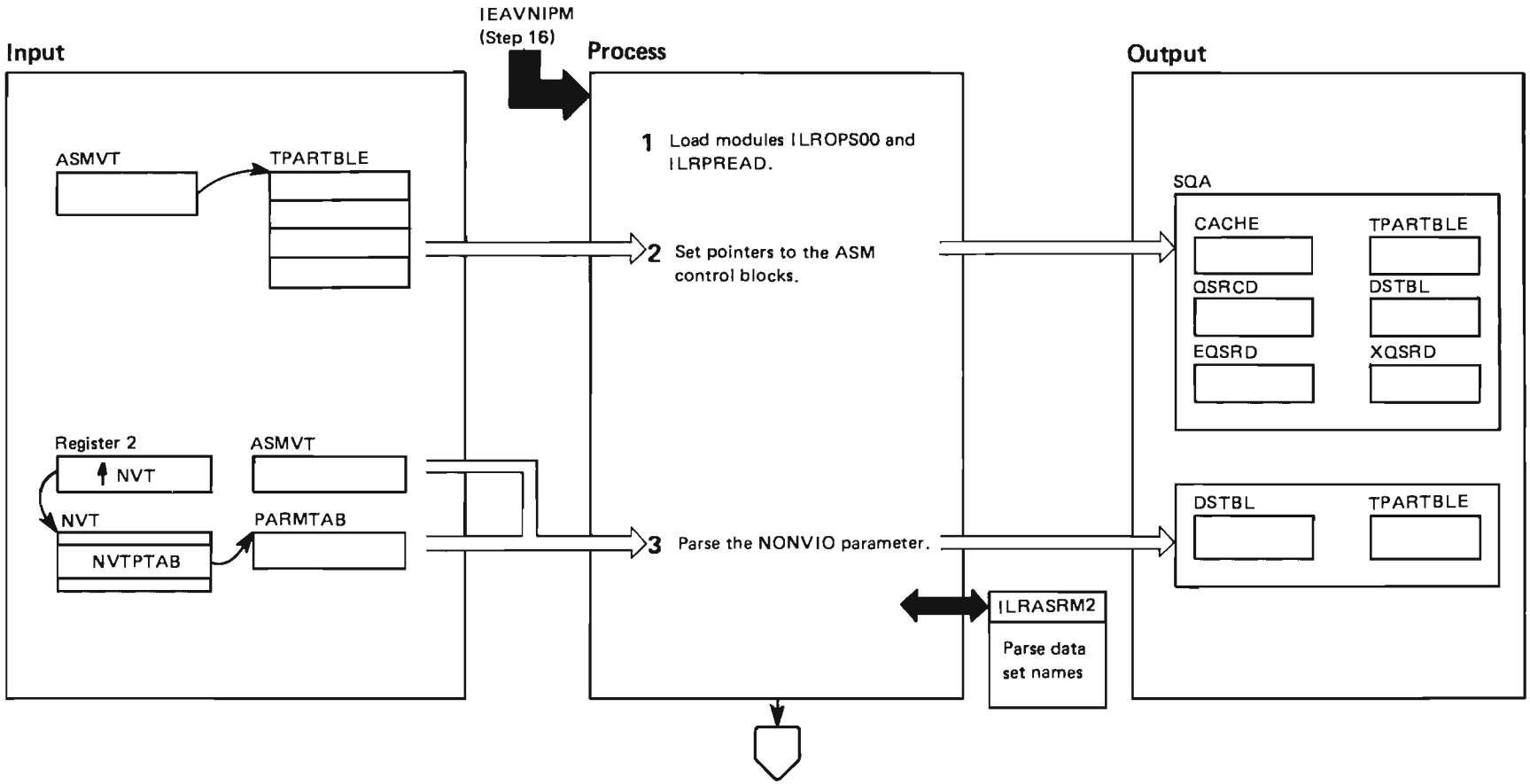


Diagram 30. Auxiliary Storage Management Initialization, Part 2 (IEAVNP14/ILRASRM1) Part 2 of 6

Extended Description	Module	Label
ILRASRM1 builds and initializes ASM control blocks and opens the page data sets and swap data sets. IEAVNIPM enters ILRASRM1 during NIP processing. ILRASRM1's normal exit is to IEAVNIPM. IEAVNIPM calls ILRASRM1 for cold, warm, and quick starts. Unless otherwise specified, the processing is the same for all three types or starts. Note that IEAVNIPM knows ILRASRM2 as IEAVNIP14. ILRASRM1 is the entry point in the load module IEAVNP14.	ILRASRM1	RIMINIT
1 ILRASRM1 loads ILROPS00 and ILRPREAD. ILROPS00 and ILRPREAD are required to open, read, and write to the page and swap data sets. **2** ILRASRM1 obtains addressability to the TPARTBLE; and restores the following control block pointers in the TPARTBLE: • the quick start record (QSRCD) • the extended quick start record (EQRSD) • the work area for the quick start record extension (XQSRD) • the device table (DVTAB) • the cache array table (CACHE) • a data set work area (DSTBL)		
3 ILRASRM1 uses the parameters in PARMTAB to locate and parse the NONVIO data set name string and mark the specified data sets as NONVIO in the DSTBL or the TPARTBLE. ILRASRM1 calls the parse routine in ILRASRM2 to parse the NONVIO data set name list.	ILRASRM1 ILRASRM2	NOVIOCHK PARSE

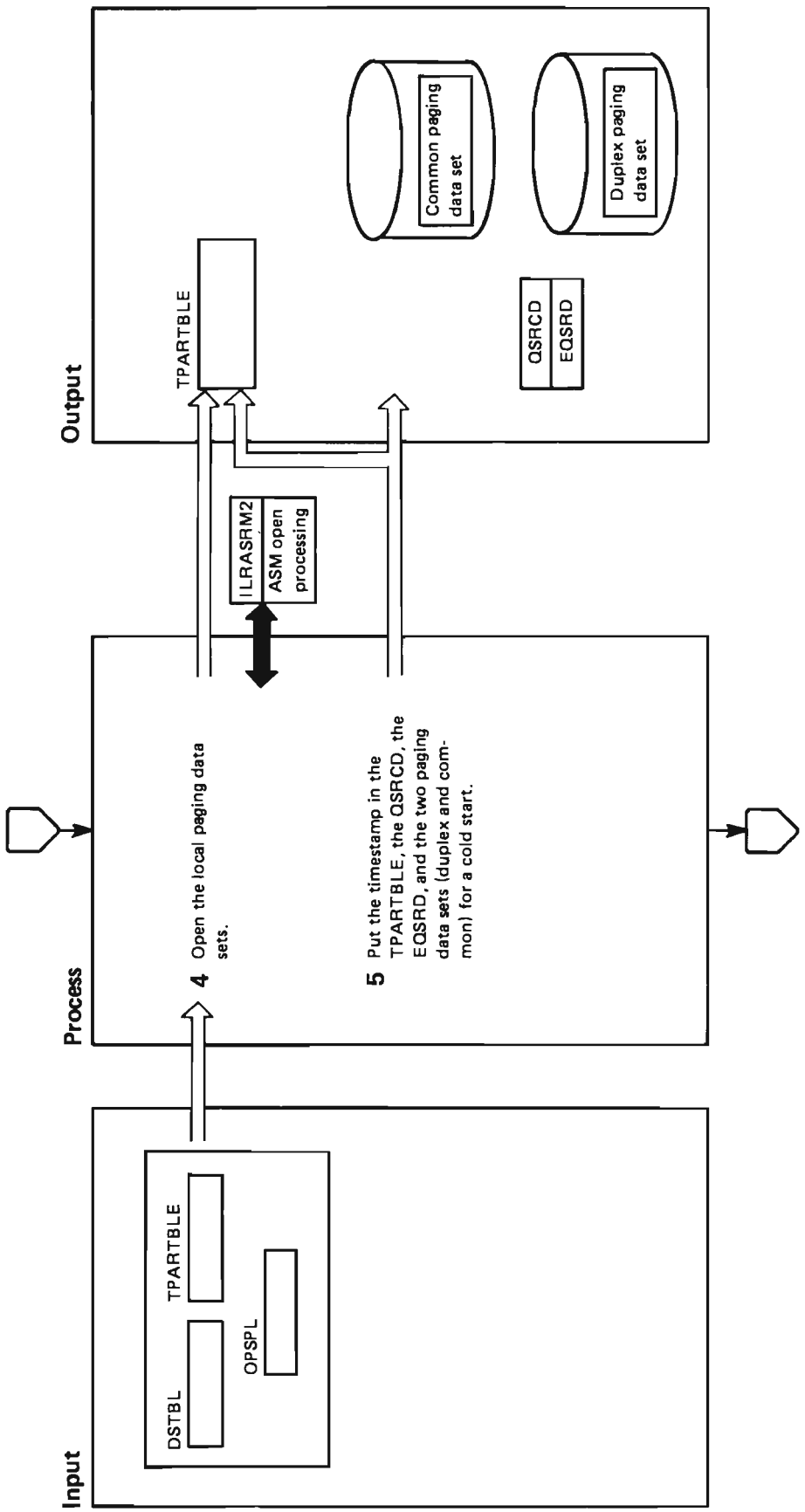


Diagram 30. Auxiliary Storage Management Initialization, Part 2 (IEAVNP14/ILRASRM1) Part 3 of 6

Diagram 30. Auxiliary Storage Management Initialization, Part 2 (IEAVNP14/ILRASRM1) Part 4 of 6

Extended Description	Module	Label
4 On a warm start, there are local page data sets containing VIO pages that are identified in the TPARTBLE. These VIO pages are usable. Open the local page data sets containing these VIO pages. If any of these local data sets are not usable, the warm start will fail. The IPL will continue as a quick start provided that the operator does not choose to re-IPL. Ignore the NONVIO local data sets during this processing. At this point in the IPL, a NONVIO local data set is not considered to be a required local data set for warm start even if it contains VIO pages. After the local page data sets identified in the TPARTBLE are processed, open the local page data sets listed in the DSTBL. If this is a cold or quick start, set the ILROPSPL reset flag to indicate that if the page data set resides on a cached auxiliary storage subsystem, ILRPREAD must reset the cache. Reset of the cached auxiliary storage subsystem that has a page data set causes a TPARTBLE flag (TPARTRAP) to be set to invalidate future warm starts until the system is sufficiently initialized to allow a warm start.	ILRASRM1	NEEDLOCL
	ILRASRM2	OPEN
	ILRASRM1	OPENLOCL
	ILRASRM2	OPEN
If errors occur for the open processing, issue an appropriate message depending on the nature of the error. Otherwise, continue with step 5.	ILRIMMSG	
5 Verify on a quick or warm start that the correct level of QSRCD, EQSRD, and TPARTBLE are being used. Also, verify that the data sets containing PLPA (that is, the PLPA data set, the common data set, and the duplex data set) are at the correct level. Continue with step 6. If this is a cold start, put a timestamp in the QSRCD and TPARTBLE, and PLPA and common page data sets. Continue with step 6.	ILRASRM1	TIMESTMP
	ILRPREAD	

Diagram 30. Auxiliary Storage Management Initialization, Part 2 (IEAVNP14/ILRASRM1) Part 5 of 6

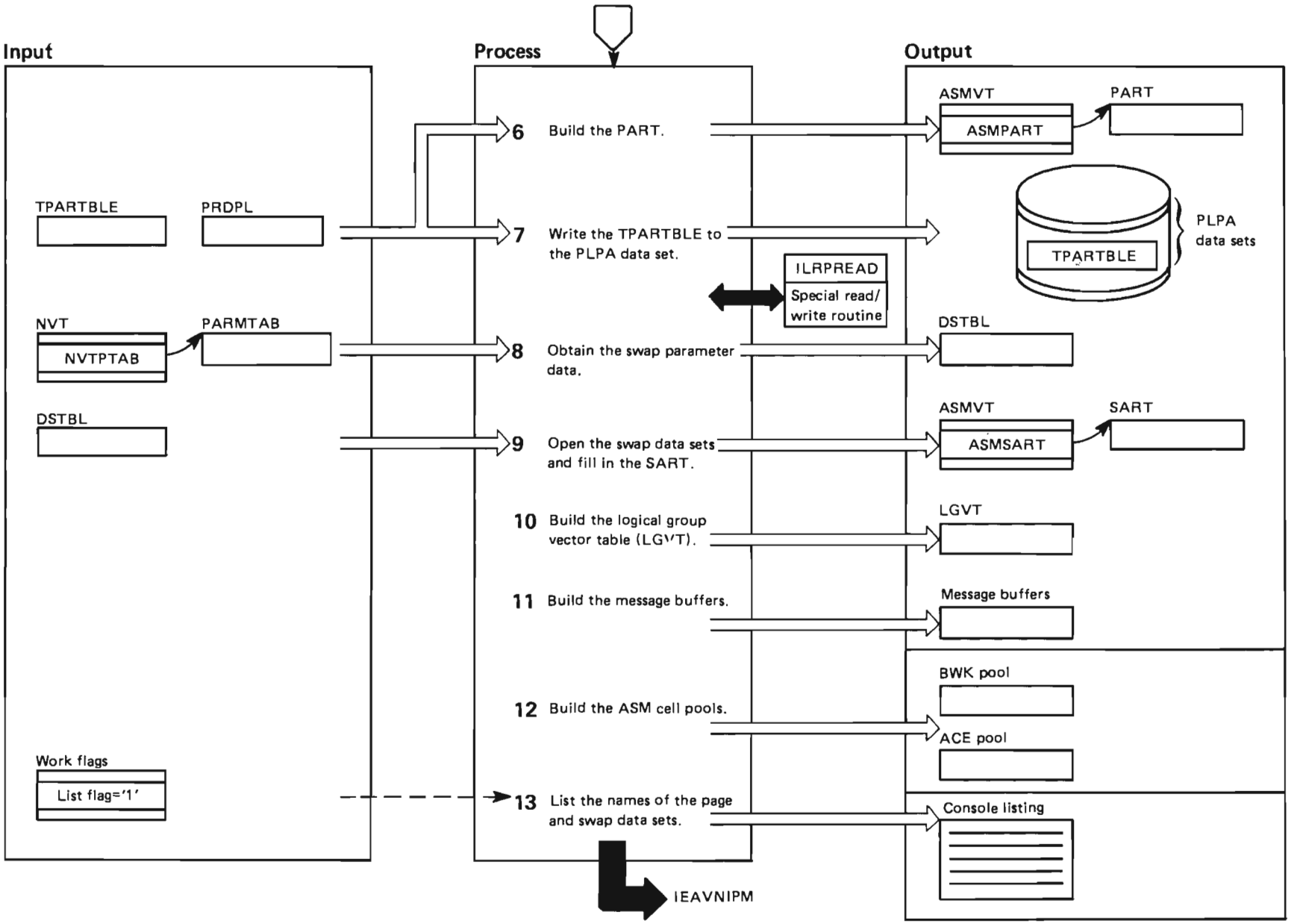


Diagram 30. Auxiliary Storage Management Initialization, Part 2 (IEAVNP14/ILRASRM1) Part 6 of 6

Extended Description	Module	Label
6 Use the TPARTBLE, now complete with information about the paging data sets opened for this IPL, and build the paging activity reference table (PART).	ILRASRM1	BLDPART
7 If this is a quick or warm start, issue the FREEMAIN macro instruction to free the QSRCD, EQSRD, and XQSRD. Do not free the TPARTBLE, but write the TPARTBLE to the PLPA data set. If this is a cold start, the NIP module, IEAVNP05 will call ILRQSRIT later in the IPL. ILRQSRIT will fill in the QSRCD, EQSRD, and XQSRDs, write the QSRCD, EQSRD, and XQSRDs, and TPARTBLE to the PLPA data set, and then free the QSRCD, EQSRD, and XQSRD storage. The TPARTBLE is freed during master scheduler initialization by ILRTMI00.	 ILRPREAD ILRQSRIT	QSRTPARB
8 Use the parameter data in PARMTAB to locate the swap parameter data. Initialize the swap data set names in the DSTBL.	ILRASRM1 ILRASRM2	INITSART PARSE
9 Using the DSTBL, invoke ILROPS00 to open the swap data sets. Put information about the opened data sets in the swap activity reference table (SART).	 ILRASRM2	OPENSWAP OPEN
10 Clear the area used for the DSTBL and ILRPREAD buffer areas and reuse it for the logical group vector table (LGVT). ASM uses the LGVT to locate VIO data. After the LGVT is initialized, delete modules ILROPS00, ILRPREAD, and ILRDVTBL, using the DELETE macro instruction.	ILRASRM1	LGVTSRB
11 Build the message buffers used by the ASM message module, ILRMSG00. ILRMSG00 gets control during normal system operation (that is, after an IPL).		MSGBUF
12 Obtain storage for ASM cell pools. ASM cell pools are pools of control blocks used by ASM. Having pools of storage available removes the overhead of performing GETMAINs during its processing. The cell pools are for ASM control elements (ACEs) and work areas (BWKs).		BLDCELLS
13 If the PAGE=(. . . , L) parameter was used to request a listing of the paging and swap data sets or if any data set changes occurred during ILRASRIM or ILRASRM1 processing, list the names of the page and swap data sets. **Note:** Errors in ILRASRM1 processing are handled by issuing messages that request the operator to specify or to allow alternative actions to take place. In some cases, messages are a notification of a problem and the action ASM takes to correct the problem. If a major error occurs during ILRASRM1 processing and there are no alternative actions possible, then the system is put into a wait state with codes X'03C', X'061', or X'063'. In this case, the operator must attempt to correct the problem (for example a bad pack or data set) and then re-IPL.	ILRIMMSG	LISTMSG
Recovery Processing: None.		

Diagram 31. SVC PARMLIB Processing (IEAVNP25) Part 1 of 23

IEAVNP25 - MODULE DESCRIPTION

DESCRIPTIVE NAME: SVC PARMLIB Processing RIM

FUNCTION:
The SVC Table RIM is called by IEAVNIPM to Process SVCPARM Statements from IEASVCxx members in SYS1.PARMLIB. For PARMLIB processing, IEAVNP25 uses the General Parmlib Scan Routine (IEEMB888) to read PARMLIB based on the "SVC=" specification. IEEMB888 builds a logical record to pass to the Statement Processor routine (IEAVNS25 entry point in IEAVNP25). IEAVNS25 uses the Generalized Parser (IEEMB887) to parse the logical record. IEAVNP25 builds an entry in the SVC Table Work Area (SVCTWA) for each valid SVCPARM statement. If there is an invalid SVC= specification, the operator will be prompted via NIPPROMPT and PARMLIB processing will be re-initiated. @D1C Mainline processing continues after all SVC PARM statements have been read and parsed. IEAVNP25 scans the SVC Table Work Area (SVCTWA). For nucleus-resident SVCs (Type 1, 2 & 6), the SVC Table is updated using the SVC Dynamic Update Routine (IEAVESTU). For LPA-resident SVCs (Type 3 & 4), IEAVNPS5 will update the SVC Table based on data saved in the SVCTWA.

ENTRY POINT: IEAVNP25

PURPOSE:
   This is the main entry point to the module.
   See OPERATION Section.

LINKAGE: BASSM R14,R15

CALLERS: IEAVNIPM

INPUT:
   - NIP Parameter Address Table referenced by NVTPTAB
   - NIP Parameter Area Header (IEAPPNIP)

OUTPUT:
   - NUCLEUS-RESIDENT - SVC Table updated.
   - LPA-RESIDENT - SVC Table Workarea (SVCTWA) contains entry point names and attributes. IEAVNPS5 will use SVCTWA to update SVC Table after LPA is initialized.

EXIT NORMAL: Return to IEAVNIPM

ENTRY POINT: IEAVNS25

PURPOSE:
   This entry point is responsible for verifying and processing individual SVCPARM statements read from SYS1.PARMLIB members by the General Parmlib Scan Routine (IEEMB888). This routine manages the interface to the Generalized Parser (IEEMB887)

LINKAGE: BASSM R14,R15

CALLERS: IEEMB888

INPUT:

   - Statement Processor List (IEEZB821)
   - A logical SVCPARM statement pointed to by the Statement Processor List
   - The SVC Table referenced by SCVTSVCT
   - The SVC Table Work Area pointed to by NVTSVCN

   - Scan Parameter List (SCL)
   - Character string for Keyword processed by Parser pointed to by Scan Parameter List.
   - Workarea (with 'footprints')

Diagram 31. SVC PARMLIB Processing (IEAVNP25) Part 2 of 23

IEAVNP25 - MODULE DESCRIPTION (Continued)

- The SVC Table Work Area

OUTPUT:
- SVC Table Work Area containing entry point names and attributes of user SVCs

EXIT NORMAL: Return to IEEMB888

EXIT ERROR: Return to IEEMB888

OUTPUT: - Workarea 'footprints' set

ENTRY POINT: SVCSEM

PURPOSE: Parser Exit Routine for processing <SVCNUM> SVCPARM token.

LINKAGE: None

CALLERS: None

INPUT: None

OUTPUT: None

ENTRY POINT: REPSEM

PURPOSE: Parser Exit Routines for processing <REPLACE> SVCPARM token

LINKAGE: None

CALLERS: None

INPUT: None

OUTPUT: None

ENTRY POINT: EPNSEM

PURPOSE: Parser Exit Routines for processing <EPNAME> SVCPARM token

LINKAGE: None

CALLERS: None

INPUT: None

OUTPUT: None

ENTRY POINT: TYPESEM

PURPOSE: Parser Exit Routines for processing <TYPE> SVCPARM token

LINKAGE: None

CALLERS: None

INPUT: None

OUTPUT: None

ENTRY POINT: APFSEM

PURPOSE: Parser Exit Routines for processing <APF> SVCPARM token

LINKAGE: None

CALLERS: None

Diagram 31. SVC PARMLIB Processing (IEAVNP25) Part 3 of 23

IEAVNP25 - MODULE DESCRIPTION (Continued)

INPUT: None

OUTPUT: None

ENTRY POINT: LOCKSSEM

PURPOSE: Parser Exit Routine for processing <LOCKS> SVCPARM token

LINKAGE: None

CALLERS: None

INPUT: None

OUTPUT: None

ENTRY POINT: LOCK1SEM

PURPOSE: Parser Exit Routine for processing a single lock token

LINKAGE: None

CALLERS: None

INPUT: None

OUTPUT: None

ENTRY POINT: NPSEM

PURPOSE: Parser Exit Routines for processing <NPRMPT> SVCPARM token

LINKAGE: None

CALLERS: None

INPUT: None

OUTPUT: None

ENTRY POINT: LPAREN

PURPOSE:
Parser Exit Routine to setup detection of unbalanced
parenthesis syntax error

LINKAGE: None

CALLERS: None

INPUT: None

OUTPUT: None

ENTRY POINT: UNBCHEK

PURPOSE:
Parser Exit Routine for detecting unbalanced parenthesis
syntax error

LINKAGE: None

CALLERS: None

INPUT: None

OUTPUT: None

Diagram 31. SVC PARMLIB Processing (IEAVNP25) Part 4 of 23

IEAVNP25 - MODULE DESCRIPTION (Continued)

EXTERNAL REFERENCES:

```
ROUTINES:
   IEAVESTU - Dynamic SVC Update Service Routine
   IEEMB888 - General Parmlib Scan Routine
   IEEMB887 - Generalized Parser

DATA AREAS:
   FOR ALL ROUTINES:
     SVC Table Work Area (IHASVCT)                          -(c,w)

   Passed by IEAVNP25 to IEEMB888:
     General Parmlib Scanner Parameter List (IEEZB819) -(c,w)

   Passed by IEEMB888 to IEAVNS25:
     Statement Processor List (IEEZB821)                -(r)

   Passed by IEAVNS25 to IEEMB887
     Scan Parameter List (IEEZB815)                     -(c,w)

   Passed by IEEMB887 to each Semantic Exit Routine:
     Scan Parameter List (IEEZB815)                     -(r,w)

CONTROL BLOCKS:
   CVT  (Communication Vector Table)         - (r)
   NVT  (NIP Vector Table)                   - (r,w)
   IEAPPNIP (NIP Parameter Header Block)     - (r,w)
   SCVT (Secondary CVT)                      - (r)
```

TABLES:

```
Parse Table (IEEPARSE)
SVCTABLE                                  - (r,w)
SVC Update Recording Table                - (w)
```

SERIALIZATION: NIP RIM

Diagram 31. SVC PARMLIB Processing (IEAVNP25) Part 5 of 23

IEAVNP25 - MODULE OPERATION

IEAVNP25 - SVC PARMLIB processing RIM

Obtain the SVC specification SVC=(xx,yy..,L) parameter from 'IEAPPNIP' If there is no SVC specification, return to NIPM.

Obtain storage for the SVC Table work area, (SVCTWA) and store its address into NVTSVCN in the NIP Vector Table. Load the General Parmlib Scan Routine (IEEMB888) and the Generalized Parser (IEEMB887).

Initialize the input areas (mapped by IEEZB819) for the General Parmlib Scan Routine (IEEMB888):

- General Parmlib Scanner Parameter List defines the SYS1.PARMLIB member(s) to be processed. Also if the list option has been requested in response to "SPECIFY SYSTEM PARAMETERS", set a bit to indicate SVCPARM statements are to be logged.
- Statement Type Table defines IEAVNS25 as the Statement Processor routine for SVCPARM statements

Call IEEMB888 to process SYS1.PARMLIB members. If the return code from IEEMB888 indicates a bad SVC= specification, issue an appropriate message (See MESSAGES section). For this error situation, prompt the operator to respecify the member suffix value(s) or to enter EOB. If the operator enters EOB, then free the SVCTWA, clear NVTSVCN, delete IEEMB888 and IEEMB887, and return to NIPM. The IBM supplied default SVC Table is used. If the operator specifies values for SVC=, then reinitialize the General Parmlib Scanner Parameter List and call IEEMB888.

When all PARMLIB processing is complete and all SVCPARM statements are valid, IEAVNP25 will scan through the SVC Table Work Area. For Type 1, 2 and 6 (nucleus-resident) SVCs, the SVC Dynamic Update Routine (IEAVESTU) will be invoked to update the SVC Table Entry. Type 3 or 4 (LPA-resident) SVCs requests will be counted.

If the count is zero indicating no Type 3 or Type 4 SVCs, then free the SVCTWA and clear NVTSVCN, otherwise leave the SVCTWA for IEAVNPS5 to process.

IEEMB888 - General Parmlib Scan Routine

IEEMB888 will open each PARMLIB member and read records. IEEMB888 will handle any I/O errors or invalid statement types. It will call the Statement Processor routine (IEAVNS25) with a processing type indicator.
For "STMT" type, a logical record is built and passed via the Statement Processor List (IEEZB821).
For "EOP" type, the Statement Processor routine is invoked for final processing and error checking.

IEAVNS25 - SVCPARM Statement Processor Routine

If the call is for End-of-Parmlib (EOP) processing, return to IEEMB888 with a return code of zero.

If the call is to process a logical SVCPARM statement, initialize the Scan Parameter List (IEEZB815) and call the Generalized Parser (IEEMB887). If the parser suffers an error, a message will be issued to the operator.

Diagram 31. SVC PARMLIB Processing (IEAVNP25) Part 6 of 23

IEAVNP25 - MODULE OPERATION (Continued)

The Scan Parameter List and the Parse Table (built via the IEEPARSE macro), controls the processing of IEEMB887. As the parser processes the SVCPARM statement, exit routines will be called as specified on the ROUT options of the IEEPARSE macro. Footprints will be set as each keyword is parsed. When a match is found, routines will check for:

--Required keywords.
--Duplicate keywords.
--Invalid specifications.
--Inconsistent options.
--A general syntax error.

In addition to validity checking these same exit routines will perform necessary intermediate processing to:

- Convert the specified SVC number to binary and build a default entry point name.
- Convert SVC attribute specifications (TYPE, LOCKS, etc.) into an appropriate bit string.
- Obtain a specified EPNAME.

Syntactically correct duplicate statements are ignored, with an appropriate informational message issued. Invalid statements and statements with inconsistent specifications result in an appropriate message being issued and in the operator being prompted to respecify SVC= parameters. In all cases, processing of the IEASVCxx member continues.

For Type 1,2, and 6 (nucleus-resident) SVCs, the nucleus lookup routine will be invoked via NUCLKUP to find the entry point address. If the entry point is not found, a message will be issued.

For all correct SVCPARM statements, the entry point name, SVC attributes, and IEASVCxx suffix will be stored into the SVC Table Work Area.

For all invalid Statements, a footprint will be set to control End-of-Parmlib processing.

RECOVERY OPERATION: None. At NIP time recovery services are unavailable.

Diagram 31. SVC PARMLIB Processing (IEAVNP25) Part 7 of 23

IEAVNP25 - DIAGNOSTIC AIDS

ENTRY POINT NAMES: IEAVNP25
IEAVNS25
SVCSEM
REPSEM
EPNSEM
TYPESEM
APFSEM
LOCKSSEM
LOCK1SEM
NPSEM
LPAREN
UNBCHEK

MESSAGES:

THIS MODULE HAS RESERVED THE MESSAGES IEA821 - IEA835

THE FOLLOWING INSERTS WILL BE REPLACED WITH:
'&&&&&' WITH A PASSED PARAMETER
'*****' WITH THE CURRENT RECORD NUMBER BEING WORKED ON
' % ' WITH THE SVC NUMBER SPECIFIED
'¢¢ ' WITH THE PARMLIB MEMBER CURRENTLY OPEN
'@ ' WITH THE SVC TYPE SPECIFIED

IEA821I SYNTAX ERROR IN "SVC=" PARAMETER
IEA823I IEASVC¢¢: SVC % : SYNTAX ERROR=&&&&&&&&.
IEA824I IEASVC¢¢: SVC % : DUPLICATE &&&&&&&& KEYWORD.
IEA825I IEASVC¢¢: SVC % : NO VALID &&&&&&&&&&&&&& SPECIFICATION.
IEA826I IEASVC¢¢: SVC % : TYPE @ ROUTINE &&&&&&&& NOT FOUND.
IEA828I IEASVC¢¢: PARSE ERROR, STATEMENT *****.
IEA830I IEASVC¢¢: DUPLICATE UPDATES TO SVC % IGNORED.
IEA832I IEASVC¢¢: SVC % : &&&&&&&& IS NOT A VALID &&&&&&&&.
IEA833I IEASVC¢¢: STATEMENT ****: % IS NOT A VALID SVCNUM.

ABEND CODES: None

WAIT STATE CODES: None

RETURN CODES:

ENTRY POINT IEAVNP25:

EXIT NORMAL:

0

ENTRY POINT IEAVNS25:

EXIT NORMAL:

0 = SVCPARM Statement is valid

EXIT ERROR:

20 = SVCPARM Statement is invalid

ENTRY POINT SVCSEM: None

ENTRY POINT REPSEM: None

ENTRY POINT EPNSEM: None

Diagram 31. SVC PARMLIB Processing (IEAVNP25) Part 8 of 23

IEAVNP25 - DIAGNOSTIC AIDS (Continued)

ENTRY POINT TYPESEM: None

ENTRY POINT APFSEM: None

ENTRY POINT LOCKSSEM: None

ENTRY POINT LOCK1SEM: None

ENTRY POINT NPSEM: None

ENTRY POINT LPAREN: None

ENTRY POINT UNBCHEK: None

REGISTER CONTENTS ON ENTRY:

ENTRY POINT IEAVNP25:

```
  2   NVT
  3   CVT
 13   Address of 18 word save area
 14   Return Address
 15   Entry Point Address
```

ENTRY POINT IEAVNS25:

```
  1   Address of a word which contains the address
      of the Statement Processor List (IEEZB821)
 13   Address of 18 word save area
 14   Return Address
 15   Entry Point Address
```

ENTRY POINT SVCSEM: Irrelevant

ENTRY POINT REPSEM: Irrelevant

ENTRY POINT EPNSEM: Irrelevant

ENTRY POINT TYPESEM: Irrelevant

ENTRY POINT APFSEM: Irrelevant

ENTRY POINT LOCKSSEM: Irrelevant

ENTRY POINT LOCK1SEM: Irrelevant

ENTRY POINT NPSEM: Irrelevant

ENTRY POINT LPAREN: Irrelevant

ENTRY POINT UNBCHEK: Irrelevant

REGISTER CONTENTS ON EXIT:

ENTRY POINT IEAVNP25:

EXIT NORMAL:

```
 0-13 Unchanged
 14   Return Address
 15   Zero
```

ENTRY POINT IEAVNS25:

EXIT NORMAL:

Diagram 31. SVC PARMLIB Processing (IEAVNP25) Part 9 of 23

IEAVNP25 - DIAGNOSTIC AIDS (Continued)

```
      0-13 Unchanged
     14    Return Address
     15    Zero

   EXIT ERROR:

         0-13 Unchanged
        14    Return Address
        15    Zero

     THE FOLLOWING ENTRY POINTS ARE INVOKED AS EXIT ROUTINES
     BY THE GENERALIZED PARSER ('IEEMB887')
     EACH EXIT ROUTINE WILL GET CONTROL AS FOLLOWS:

ENTRY POINT SVCSEM: Irrelevant

ENTRY POINT REPSEM: Irrelevant

ENTRY POINT EPNSEM: Irrelevant

ENTRY POINT TYPESEM: Irrelevant

ENTRY POINT APFSEM: Irrelevant

ENTRY POINT LOCKSSEM: Irrelevant

ENTRY POINT LOCK1SEM: Irrelevant

ENTRY POINT NPSEM: Irrelevant

ENTRY POINT LPAREN: Irrelevant

ENTRY POINT UNBCHEK: Irrelevant
```

Diagram 31. SVC PARMLIB Processing (IEAVNP25) Part 10 of 23

IEAVNP25 - SVC PARMLIB Processing RIM **STEP 01**

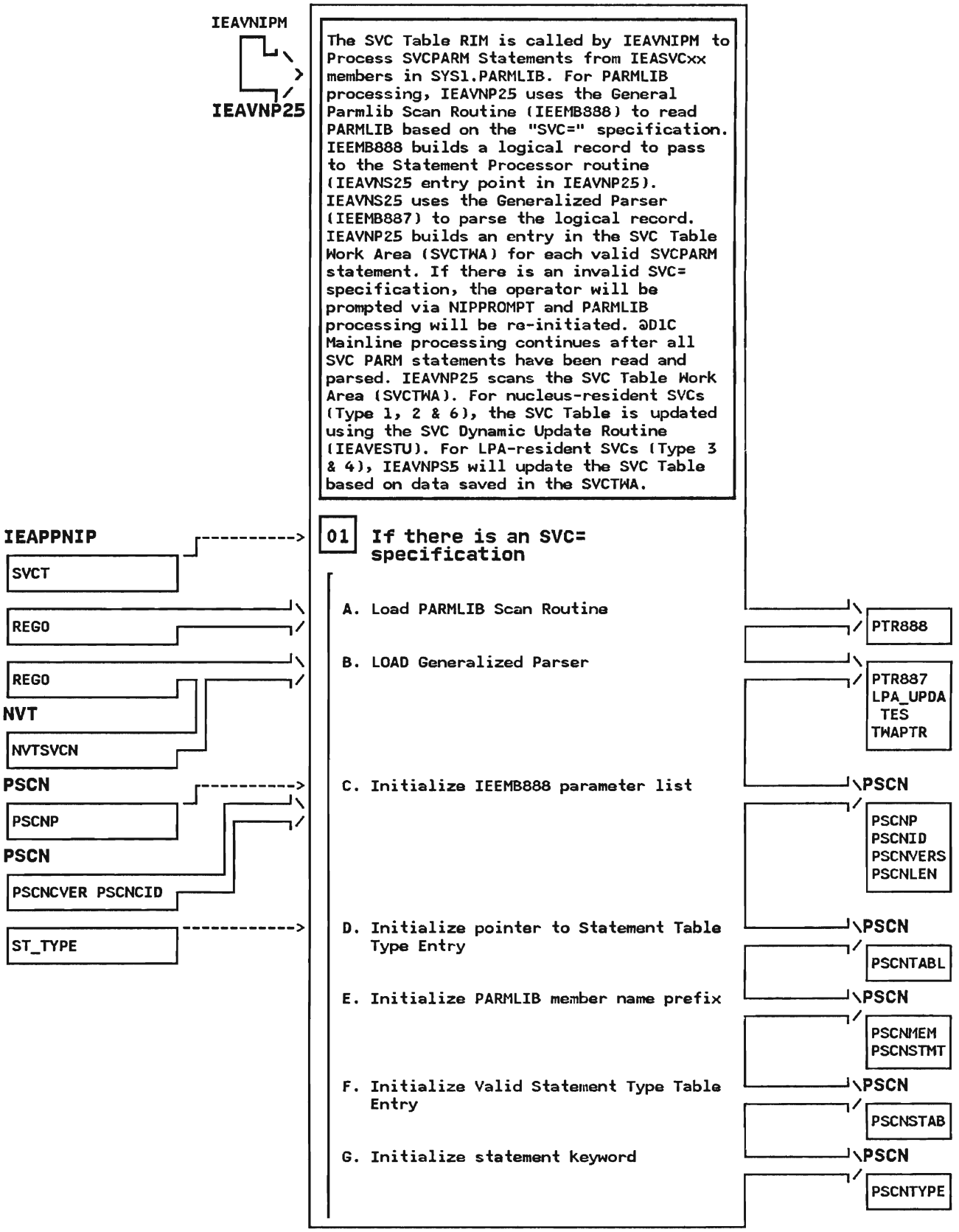


Diagram 31. SVC PARMLIB Processing (IEAVNP25) Part 11 of 23

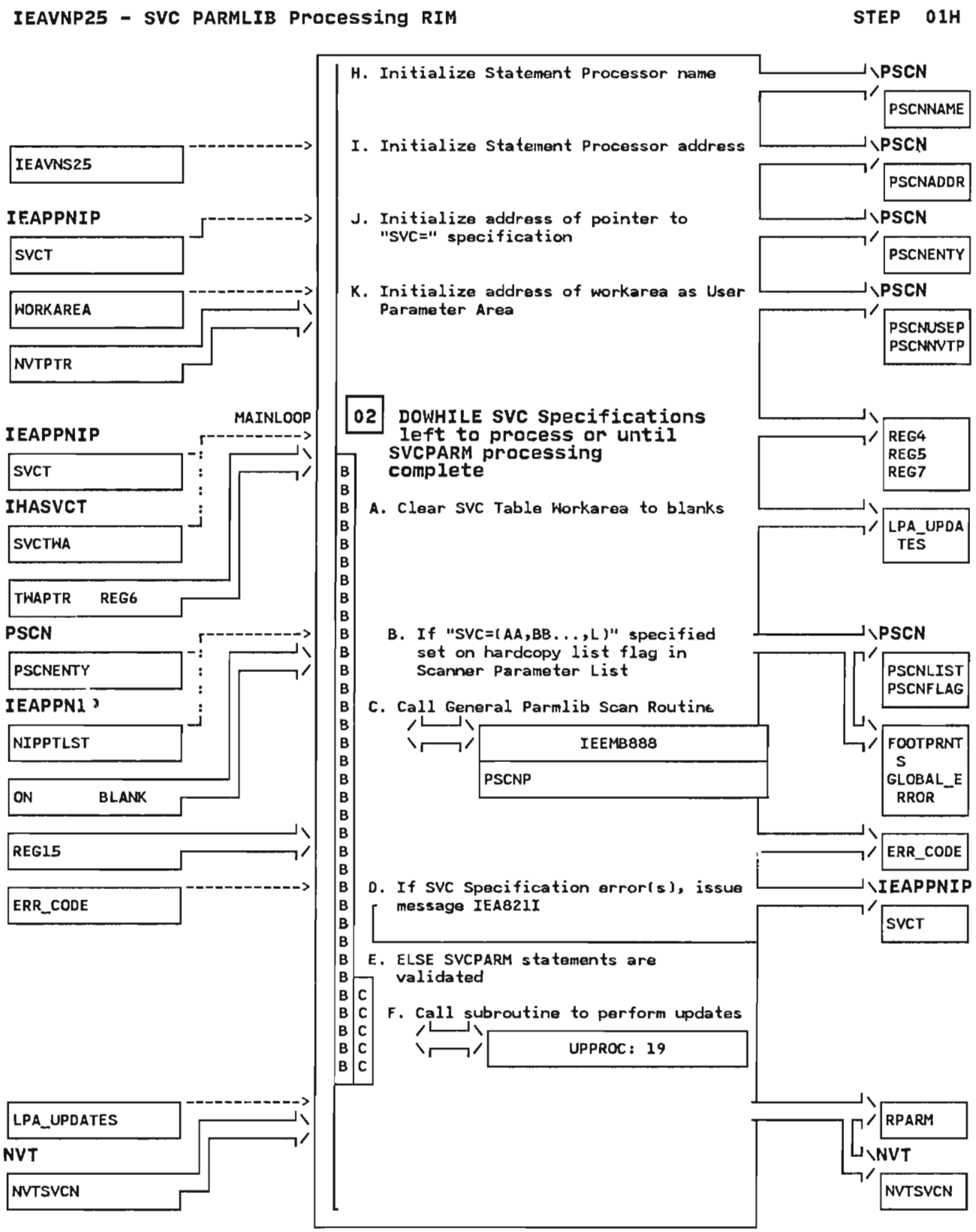


Diagram 31. SVC PARMLIB Processing (IEAVNP25) Part 12 of 23

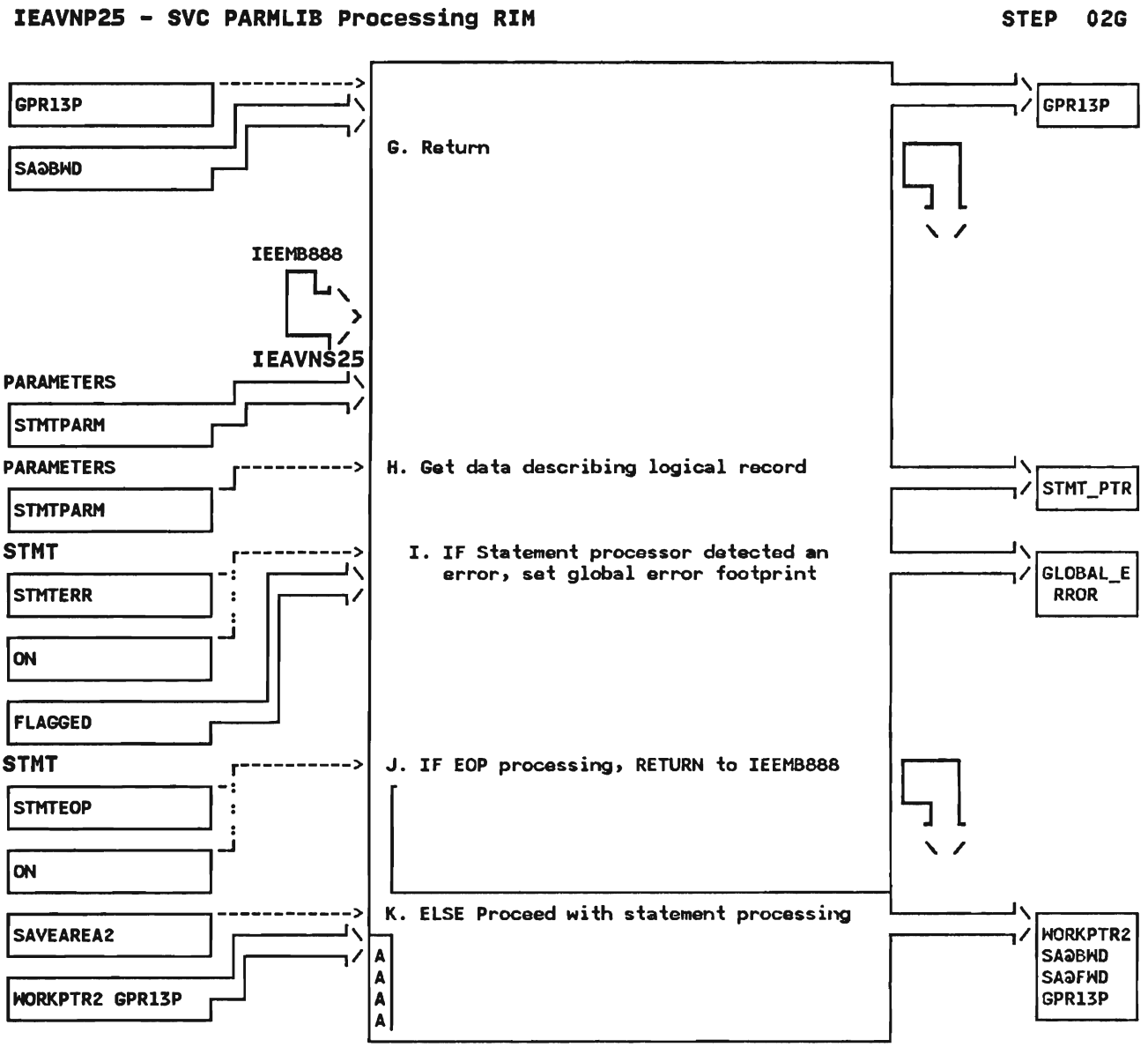


Diagram 31. SVC PARMLIB Processing (IEAVNP25) Part 13 of 23

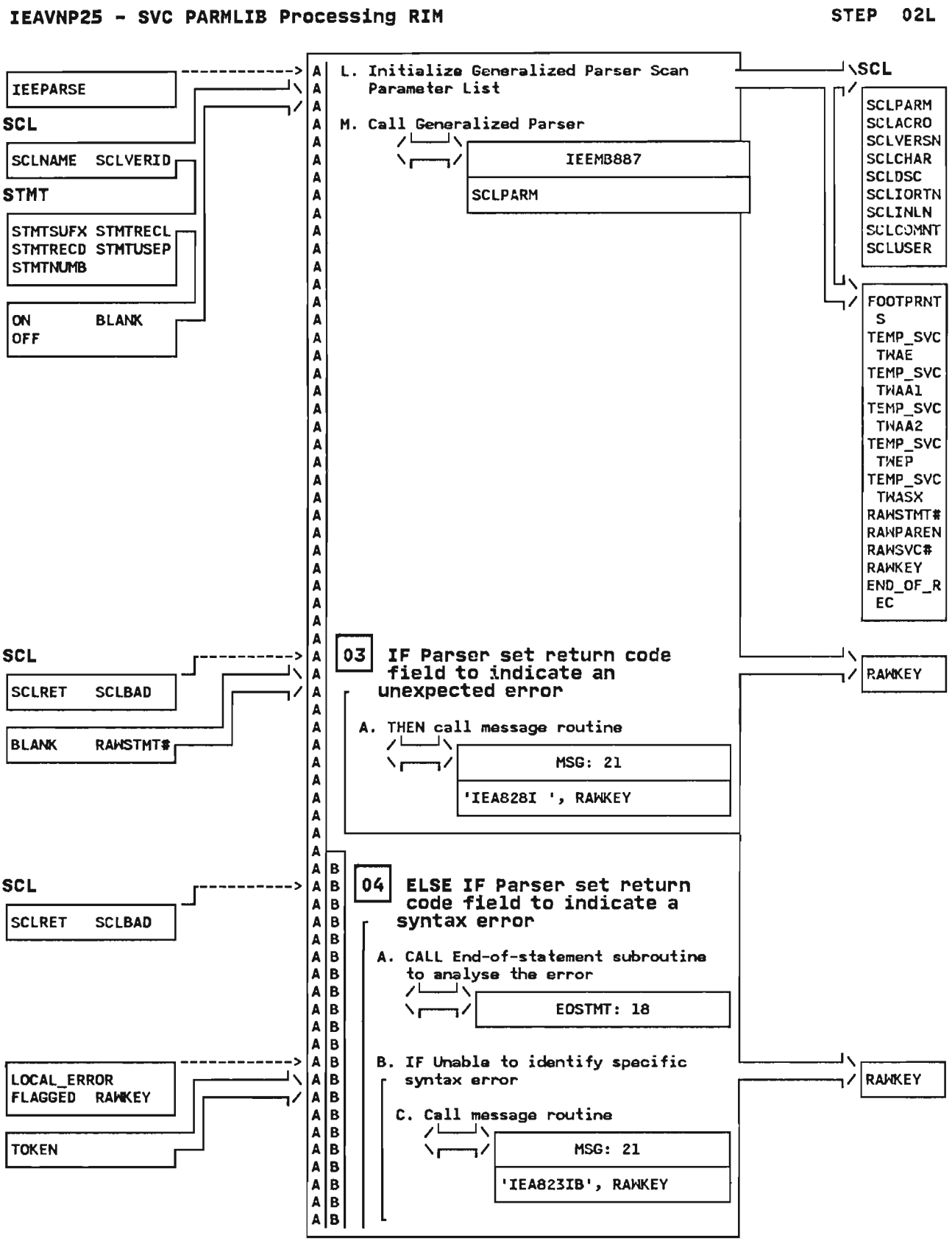


Diagram 31. SVC PARMLIB Processing (IEAVNP25) Part 14 of 23

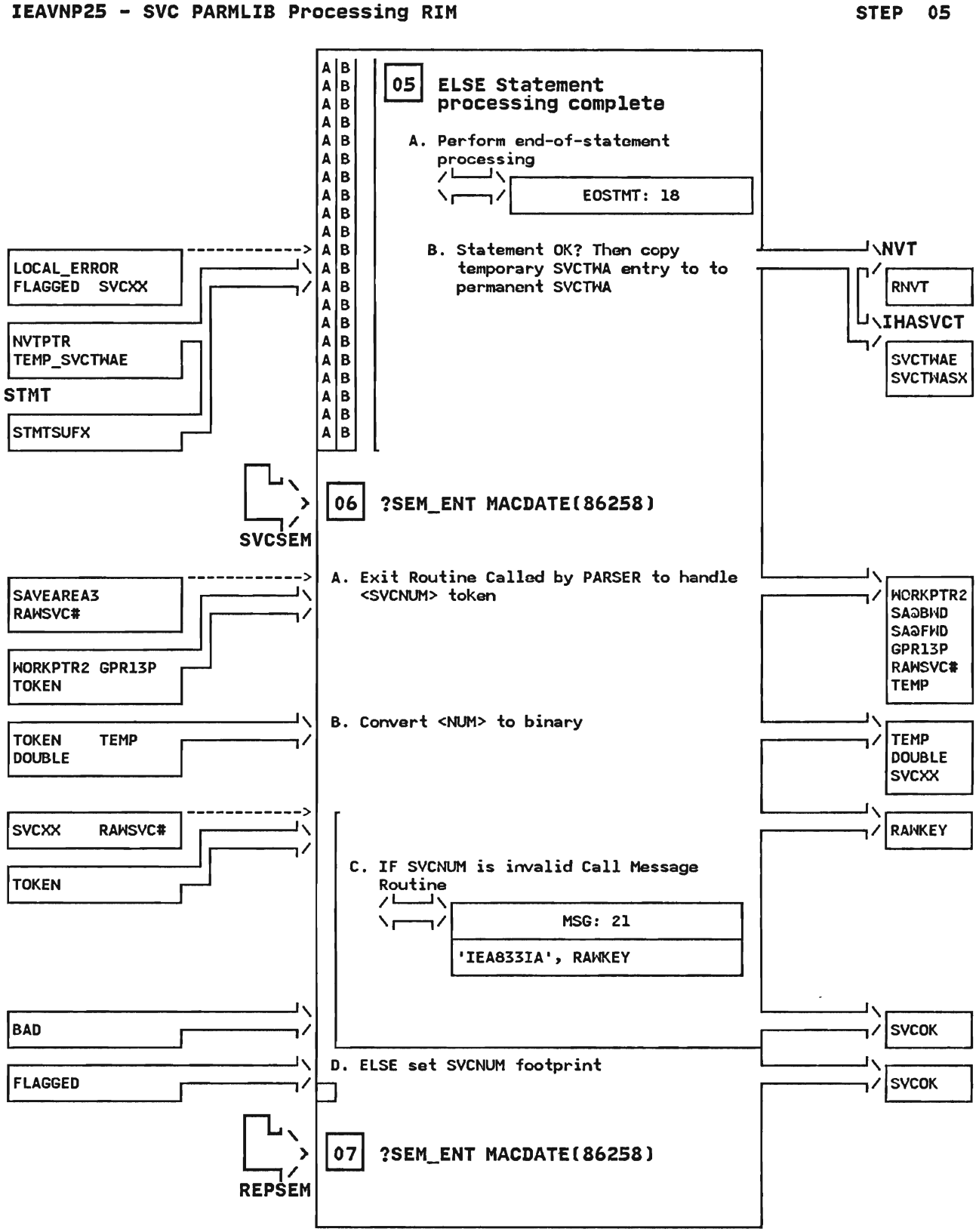


Diagram 31. SVC PARMLIB Processing (IEAVNP25) Part 15 of 23

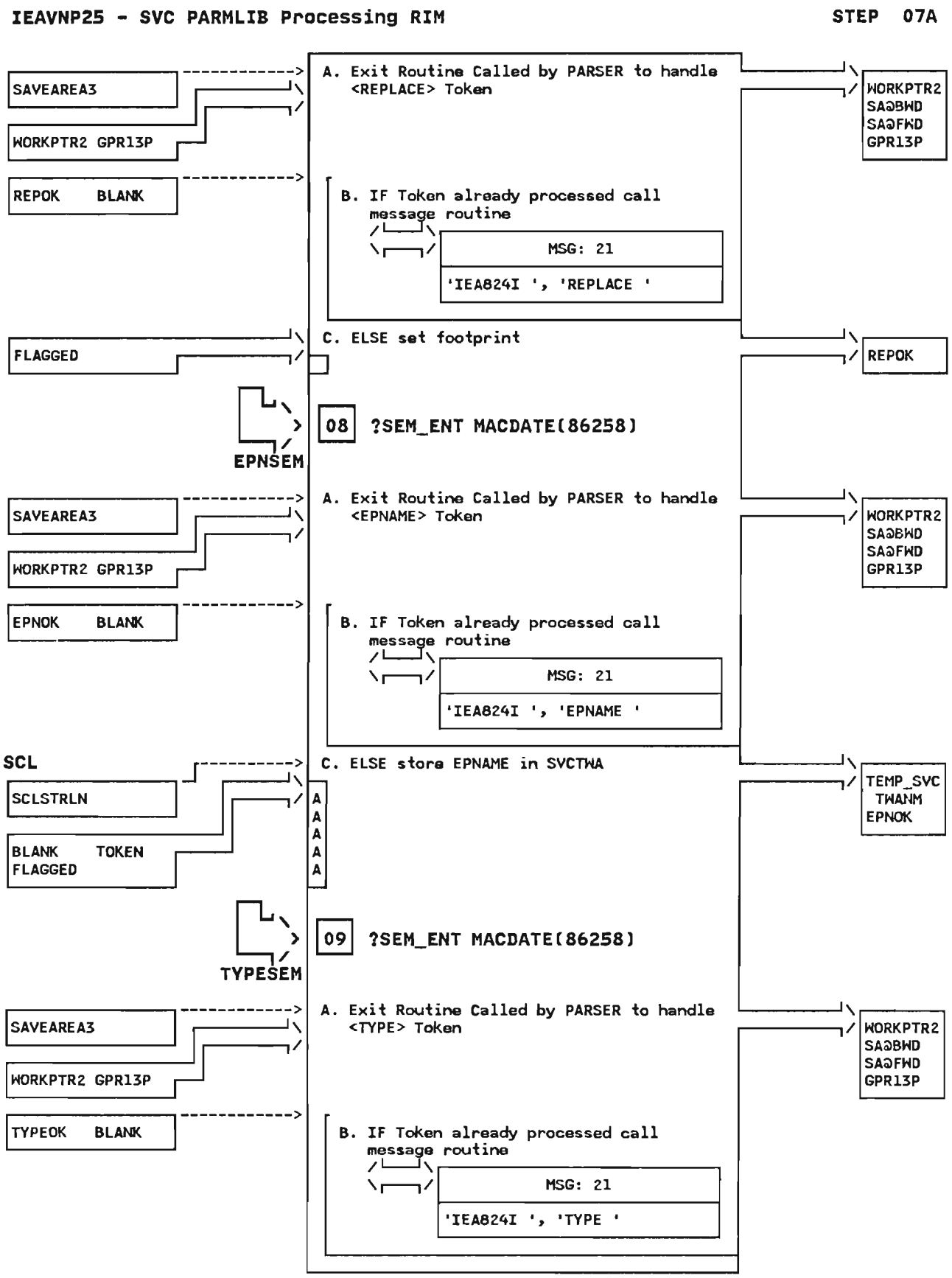


Diagram 31. SVC PARMLIB Processing (IEAVNP25) Part 16 of 23

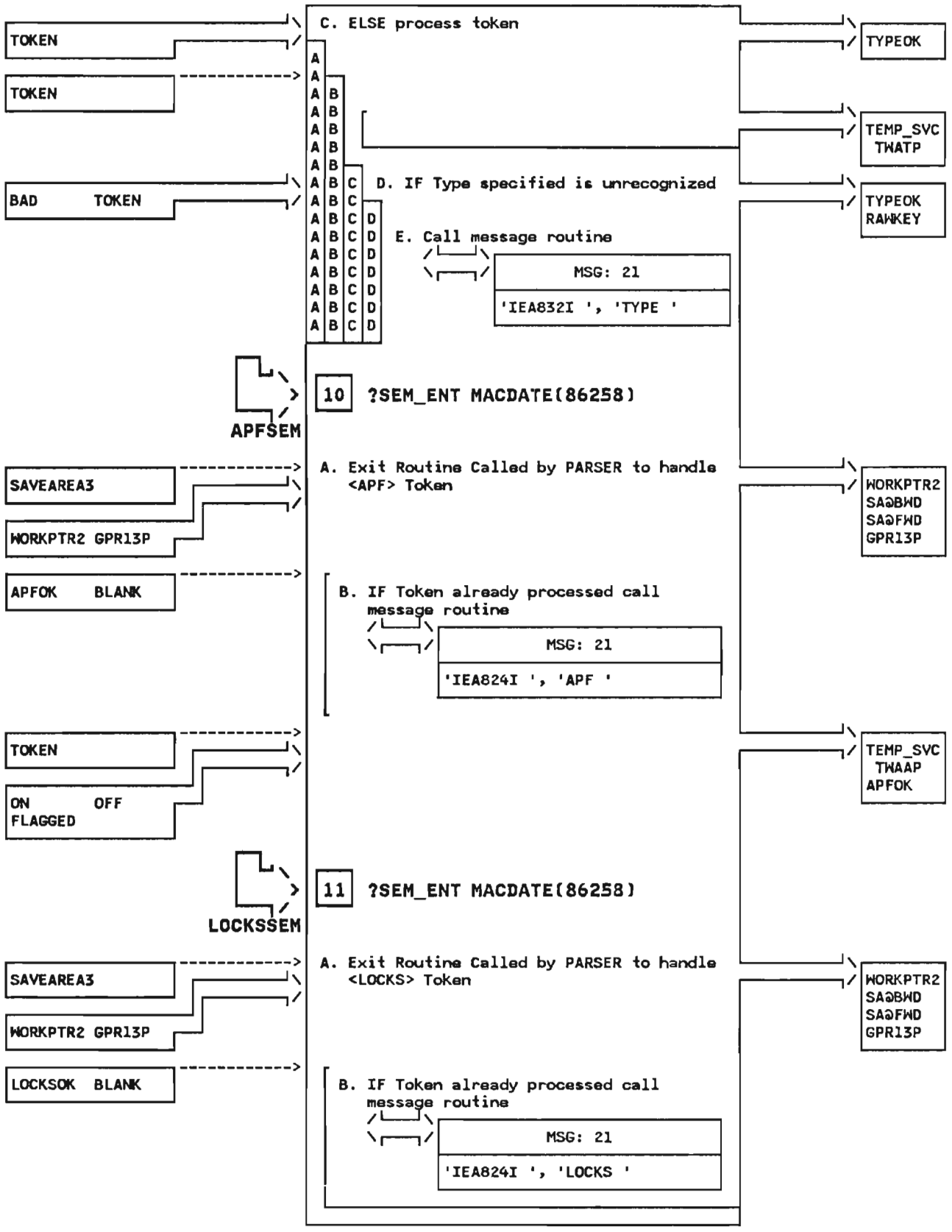


Diagram 31. SVC PARMLIB Processing (IEAVNP25) Part 17 of 23

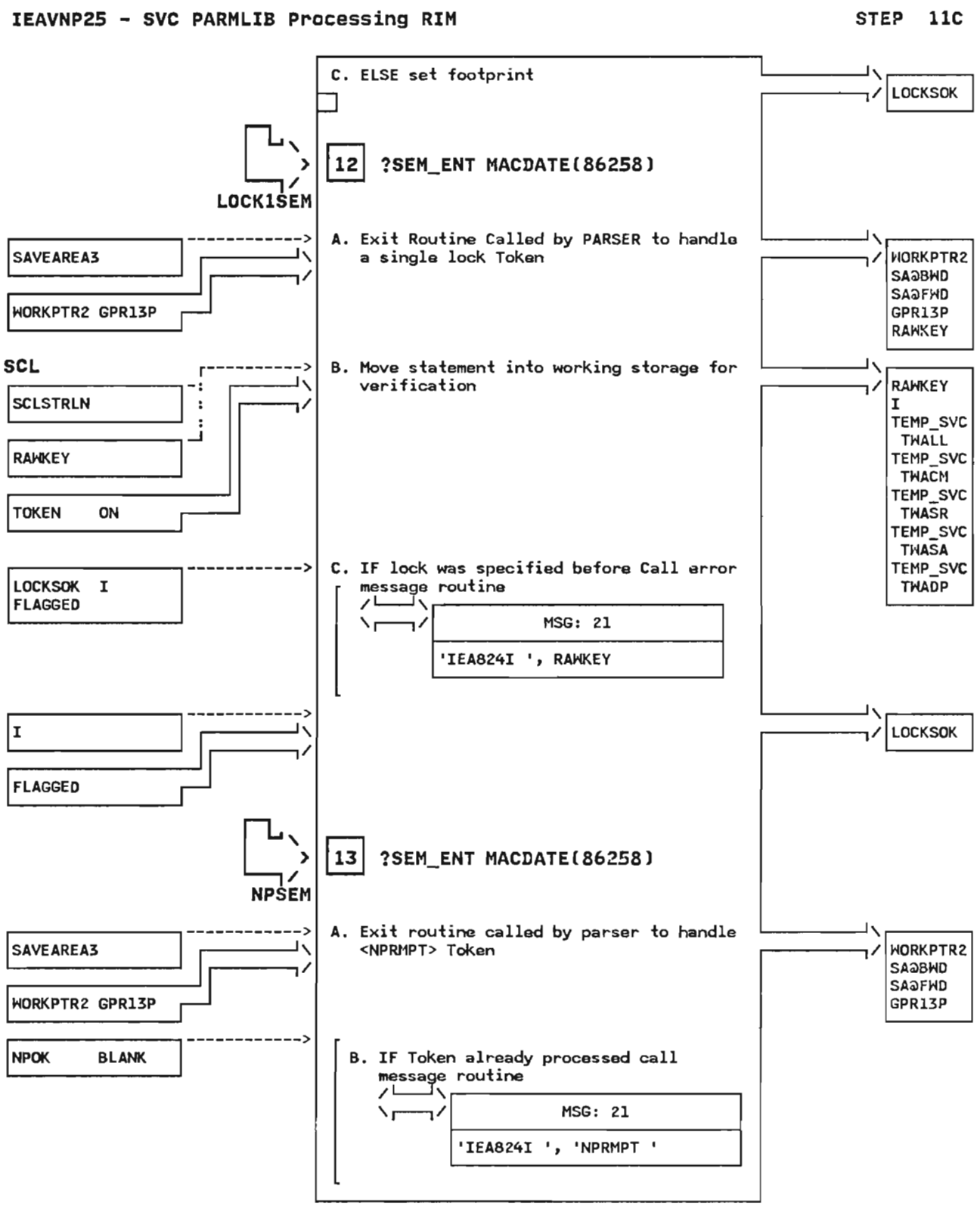


Diagram 31. SVC PARMLIB Processing (IEAVNP25) Part 18 of 23

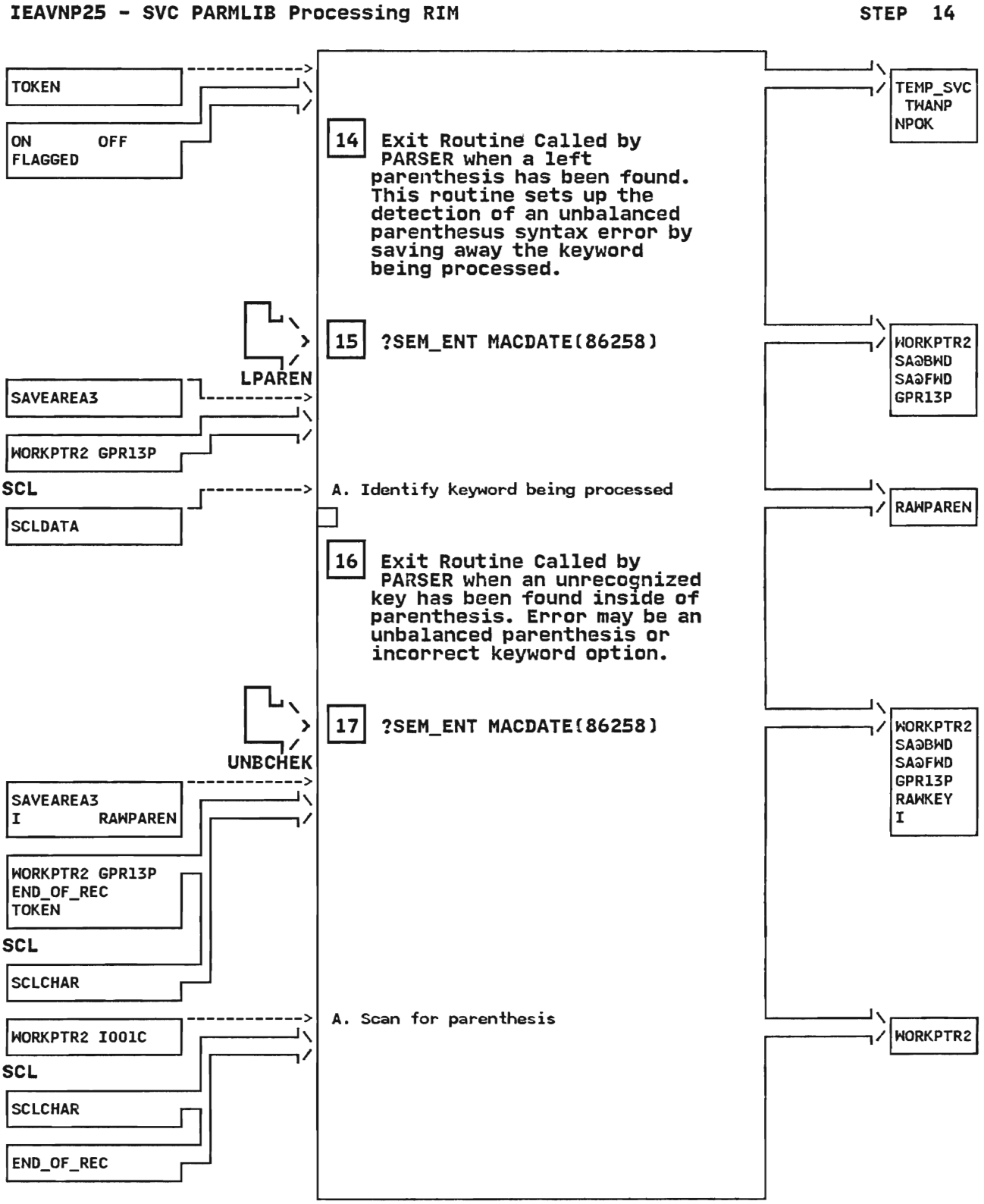


Diagram 31. SVC PARMLIB Processing (IEAVNP25) Part 19 of 23

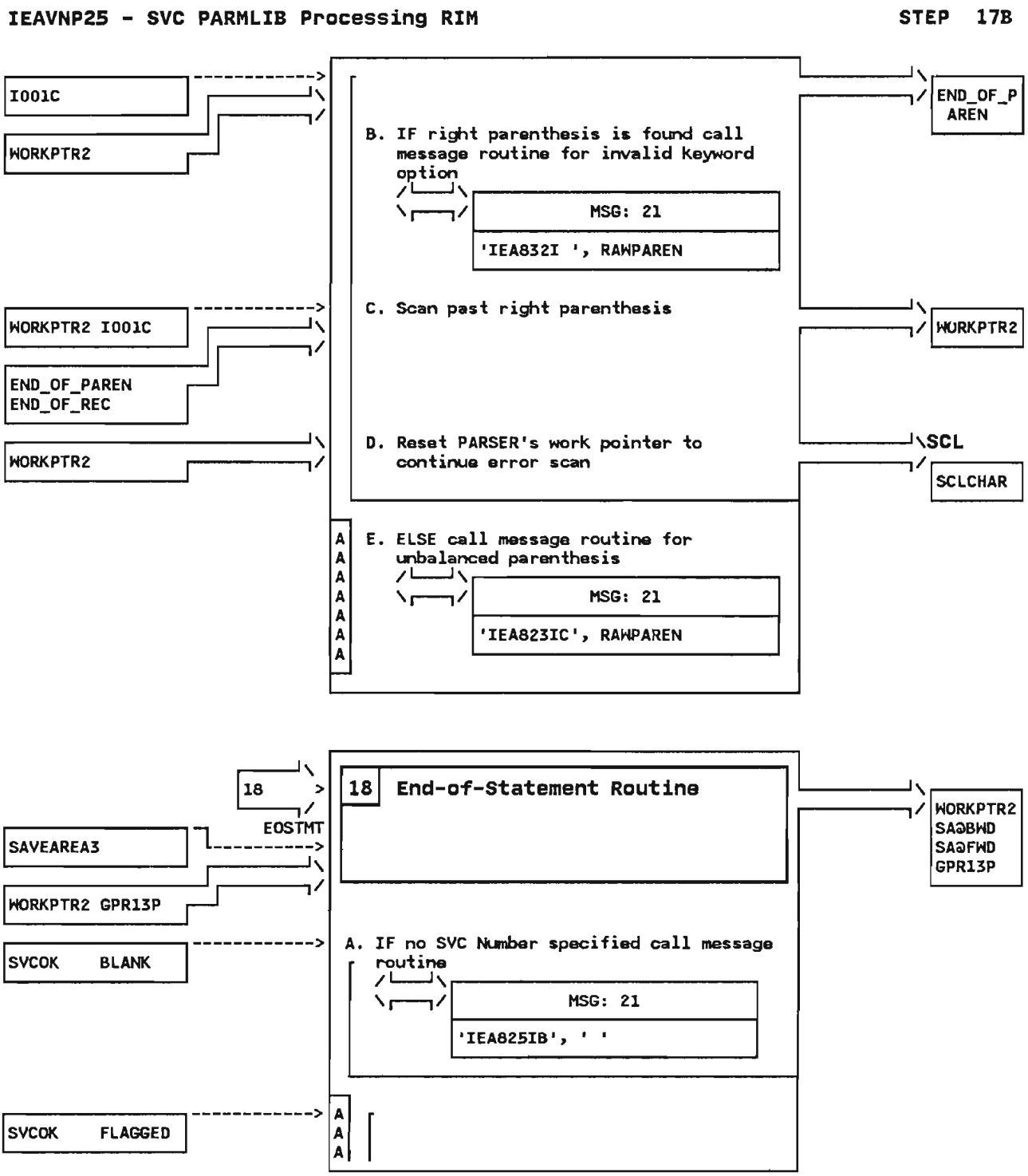


Diagram 31. SVC PARMLIB Processing (IEAVNP25) Part 20 of 23

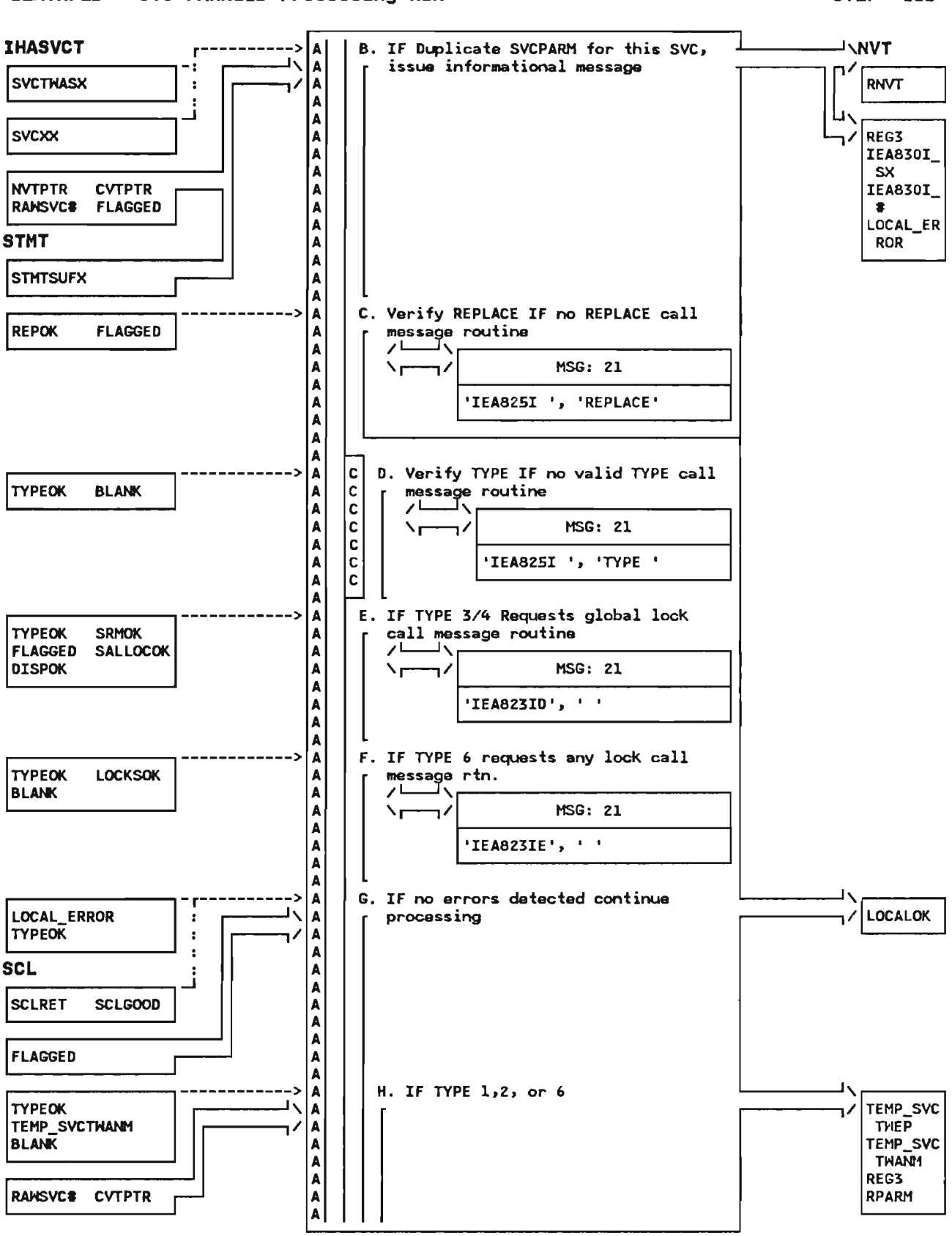


Diagram 31. SVC PARMLIB Processing (IEAVNP25) Part 21 of 23

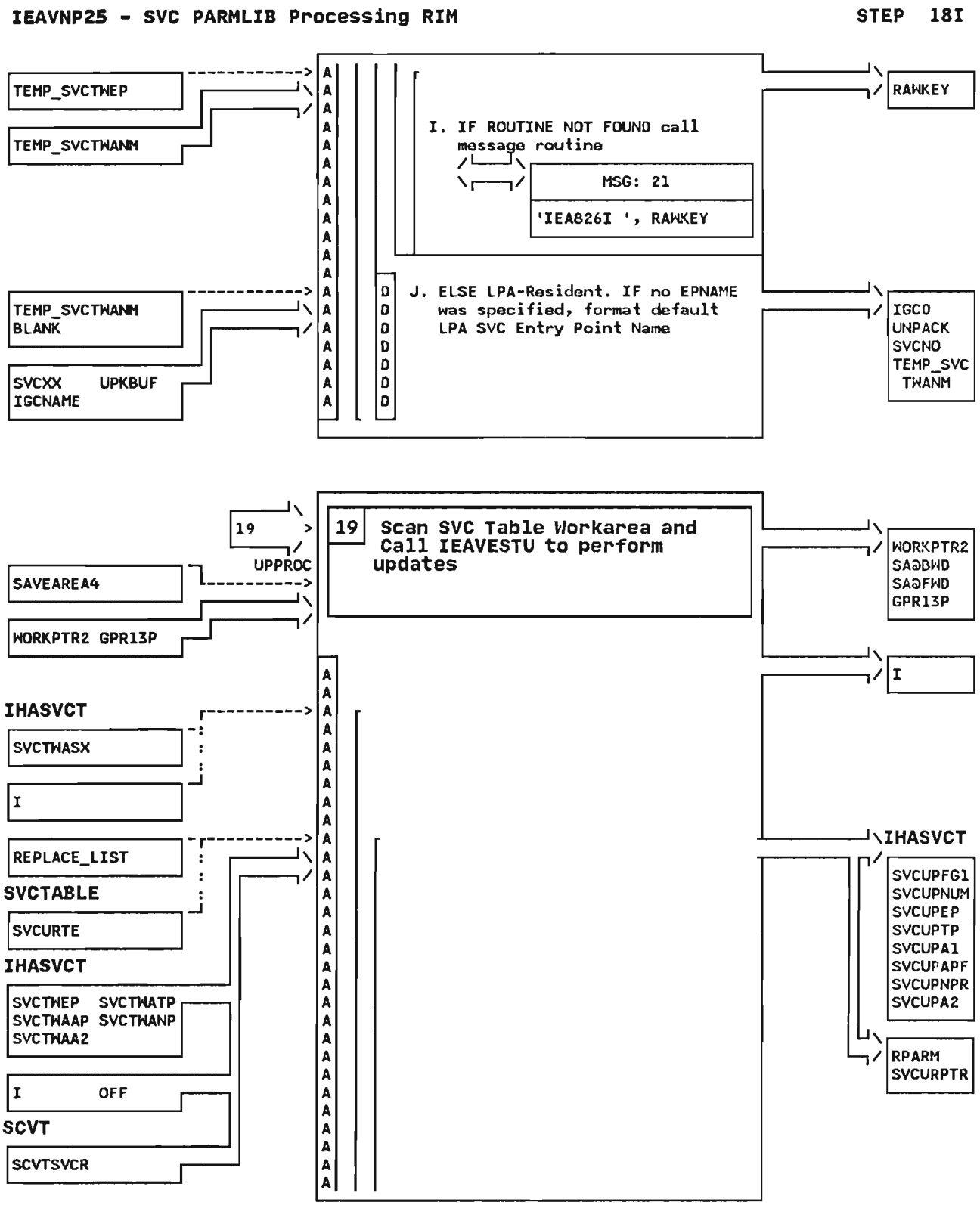


Diagram 31. SVC PARMLIB Processing (IEAVNP25) Part 22 of 23

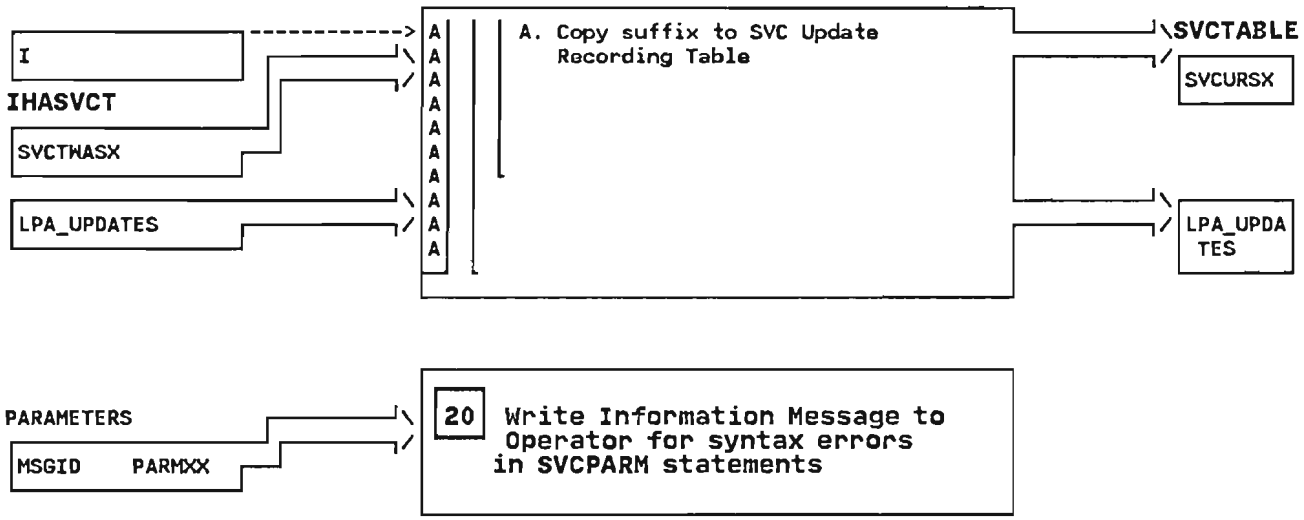


Diagram 31. SVC PARMLIB Processing (IEAVNP25) Part 23 of 23

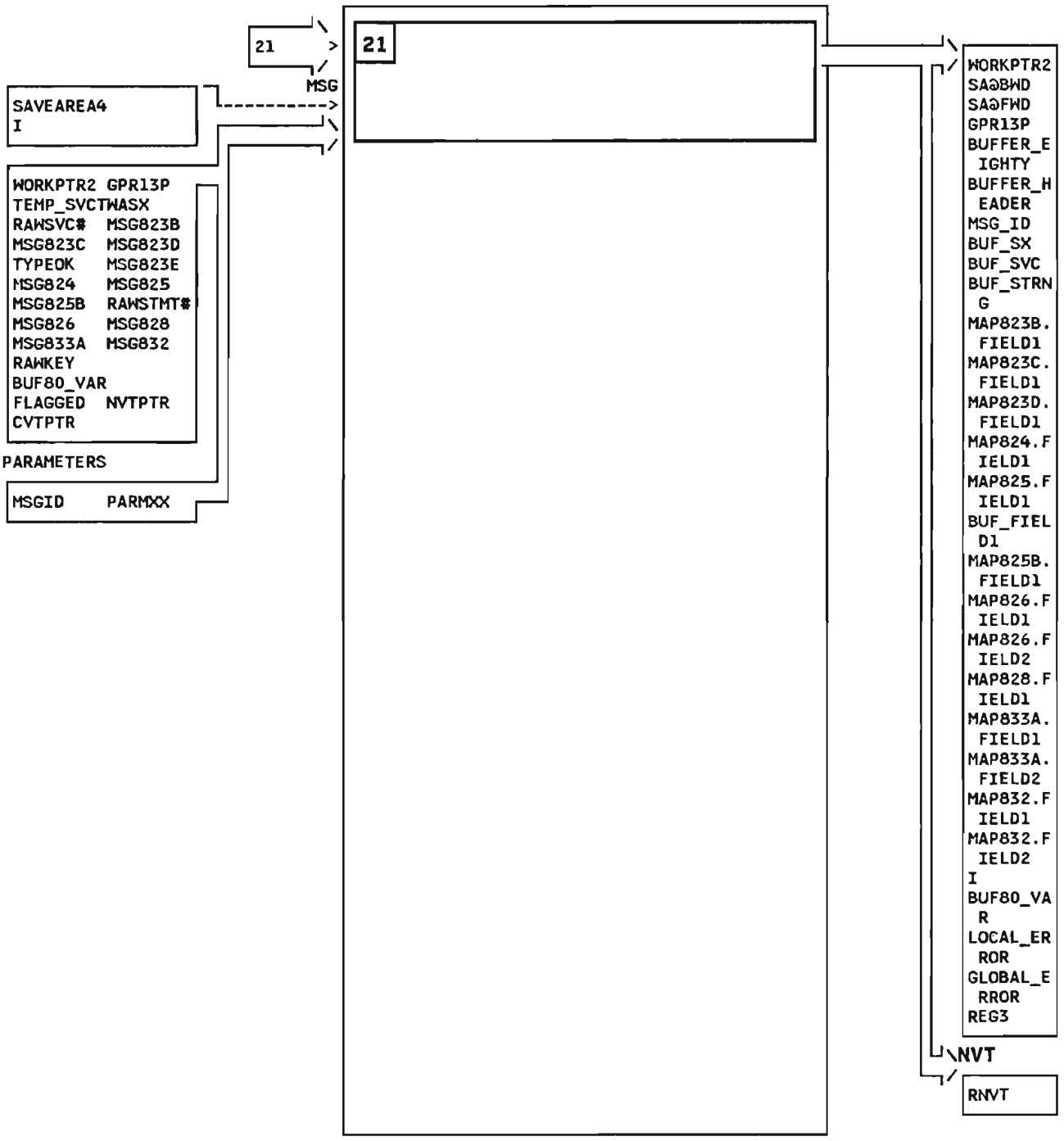


Diagram 32. Link Pack Area Initialization (IEAVNP05) Part 1 of 8

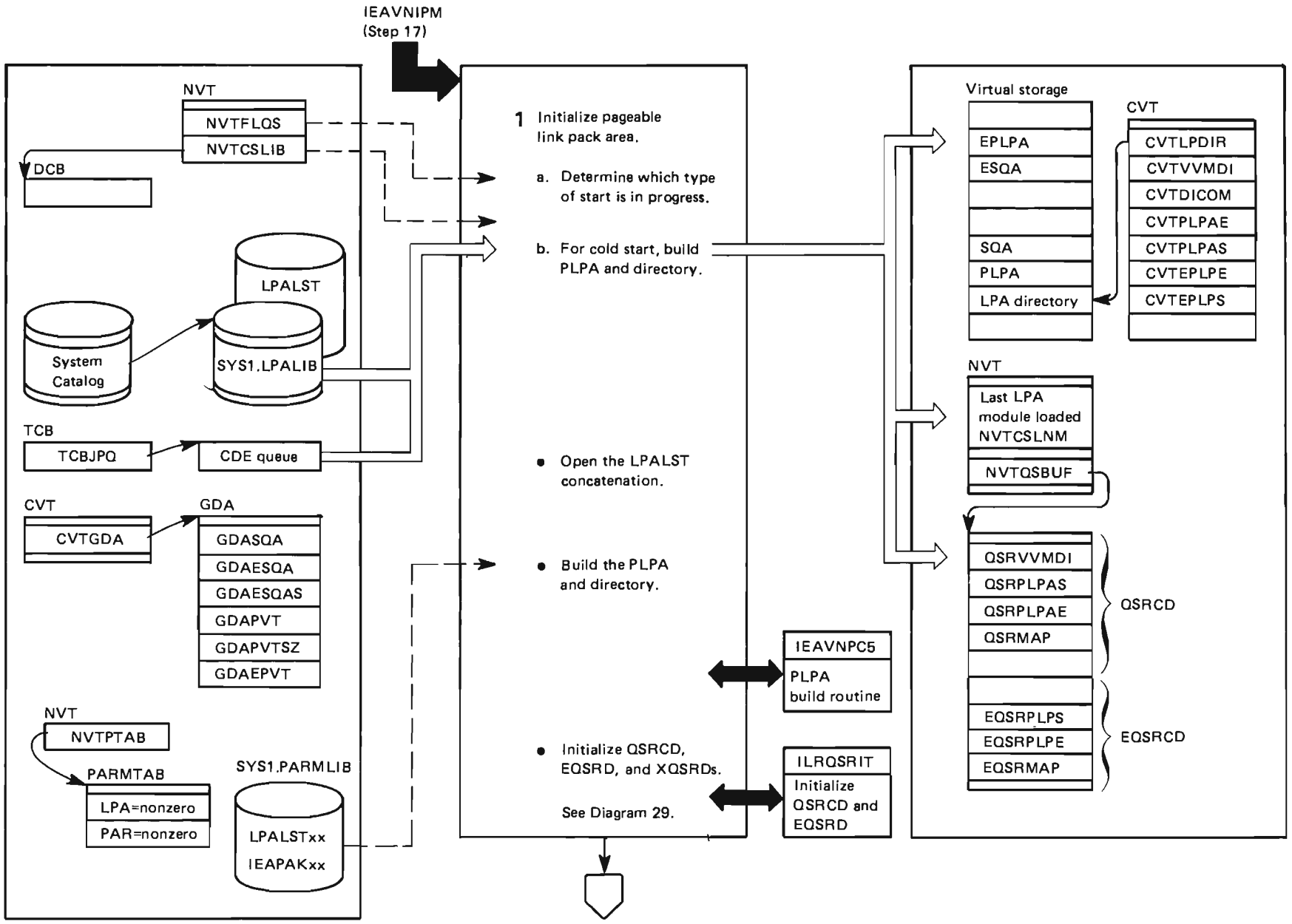


Diagram 32. Link Pack Area Initialization (IEAVNP05) Part 2 of 8

Extended Description	Module	Label
IEAVNIPM calls the contents supervisor RIM, IEAVNP05, to initialize the link pack area (LPA). See Figure 1-6 for a description of virtual storage at the conclusion of the NIP process.		
1 The link pack area (LPA) is an area of virtual storage containing reenterable routines that can be used concurrently by all tasks in the system. The contents supervisor RIM initializes the pageable LPA (PLPA) as follows:		
a. If the initialization is a quick or warm start (flag NVTFLQS is on), a PLPA and EPLPA exist from the previous system initialization. For information on building the PLPA and EPLPA, see Diagram 29, ILRQSRIT.	IEAVNP05	
b. If the initialization is a cold start (flag NVTFLQS is off) the contents supervisor module, IEAVNPC5 creates a PLPA, EPLPA and a directory in V=V storage.	IEAVNPC5	
	IEAVNPC5	NPC5HOUS
IEAVNPC5 uses fields in the global data area (GDA) to define the boundaries for the common storage area. It ensures that the LPA module library, SYS1.LPALIB is mounted and opened. If the "LPA" system parameter was specified, IEAVNPC5 validates the syntax. IEAVNPC5 then returns to IEAVNP05. IEAVNP05 opens the LPALST concatenation and calls IEAVNPC5 at entry point IEAVNPCL to build the PLPA. IEAVNPC5 obtains the starting address of each LPALST data set (the directories) from the LPALSTs DEB, then reads the LPALST directory blocks. From the directory blocks, it builds a temporary LPA directory (INFOTAB) entry for each LPALST module and constructs INFOTAB starting at the low end of the extended private area of storage. Later, IEAVNPC5 will use the INFOTAB entries to load the modules and to construct the LPA directory entries.	IEAVNPM3 IEAVNPC5	NIPMOUNT NIPOPEN NPACLIN NPA5INFO
After IEAVNPC5 builds the INFOTAB entries for LPALST, it processes the "PAK=" parameter. IEAVNPC5 calls IEAVNPM4 to read data from the specified IEAPAKxx members of SYS1.PARMLIB. The IEAPAKxx members contain groups of modules in the LPALST that are executed together or in sequence. IEAVNPC5 uses IEAPAK00 to determine the order in which listed modules are to be loaded from the LPALST into the pageable LPA.	IEAVNPM4 IEAVNPC5	NPA5BLMQ
IEAVNPC5 computes the load addresses for the modules, beginning with any pack groups specified in the IEAPAKxx members. The load addresses of modules in the PLPA are calculated, beginning with the upper boundary of the PLPA assigning downwards until all addresses are calculated. Similarly, the load addresses		NPA5GLOD
for modules in the EPLPA are calculated from the low boundary of the EPLPA, assigning upwards. The RMODE (residence mode) attribute in the PDS entry for a module determines whether the module is loaded into PLPA or EPLPA. IEAVNPC5 loads modules contained in the same group into single or contiguous pages. If any member of IEAPAKxx does not exist, IEAVNPC5 issues a message to the operator and continues processing.		NPA5ILOD
When it has calculated all the load addresses, IEAVNPC5 calls IEAVNPC8 to initialize the page tables for the PLPA and the EPLPA. It then loads the modules from the LPALST into V=V storage.	IEAVNPC8	NPA5LOAD
As each module is loaded, IEAVNPC5 stores the module name in field NVTCSLNM; in case of an error, IEAVNPC5 uses this field to identify the last attempted LPA load. Also, as each module is loaded, IEAVNCP5 converts the associated INFOTAB entry to a link pack directory entry (LPDE) and overlays the INFOTAB entry with LPDE.	IEAVNPC5	NPA5MLOD
		NPA5CLPD
After the last LPA module has been loaded, IEAVNPC5 calls IEAVNPC8 to allocate storage for the directory, moves the directory to the next address lower than the PLPA, and places a pointer to the directory in field CVTLPDIR. IEAVNPC5 moves the LPA hash value from the NVT to field CVTVVMDI.	IEAVNPC8 IEAVNPC5	NPA5BDIR
IEAVNPC5 updates CVT fields CVTPLPAS, CVTPLPAE, CVTEPLPS, and CVTEPLPE with the low and high end addresses of the PLPA and EPLPA and pages the PLPA and the EPLPA out.		
IEAVNPC5 calls the ASM quick start record initialization module, ILRQSRIT, to initialize the quick start record (QSRCD), the extended quick start record (EQSRD), and the quick start record extensions (XQSRDs). On subsequent warm starts, the ASM RIM, ILRASRIM reads the QSRCD, EQSRD, and XQSRDs from the PLPA data set to rebuild the PLPA and the EPLPA. ILRQSRIT moves the CVTVVMDI, CVTPLPS, CVTPLPE to the QSRCD, all the PLPA XPTLPIDs to XQSRDs, the CVTEPLPS, CVTPLPE, and CVTRWNS to the EQSRD, and all the EPLPA XPTLPIDs to the XQSRDs. ILRQSRIT then calls ILRPREAD to write them to the PLPA data set.	ILRQSRIT	
Upon return from ILRQSRIT, the contents supervisor RIM enables the system to use the LPA directory by setting the directory available flag CVTDICOM.	IEAVNP05	

Diagram 32. Link Pack Area Initialization (IEAVNP05) Part 3 of 8

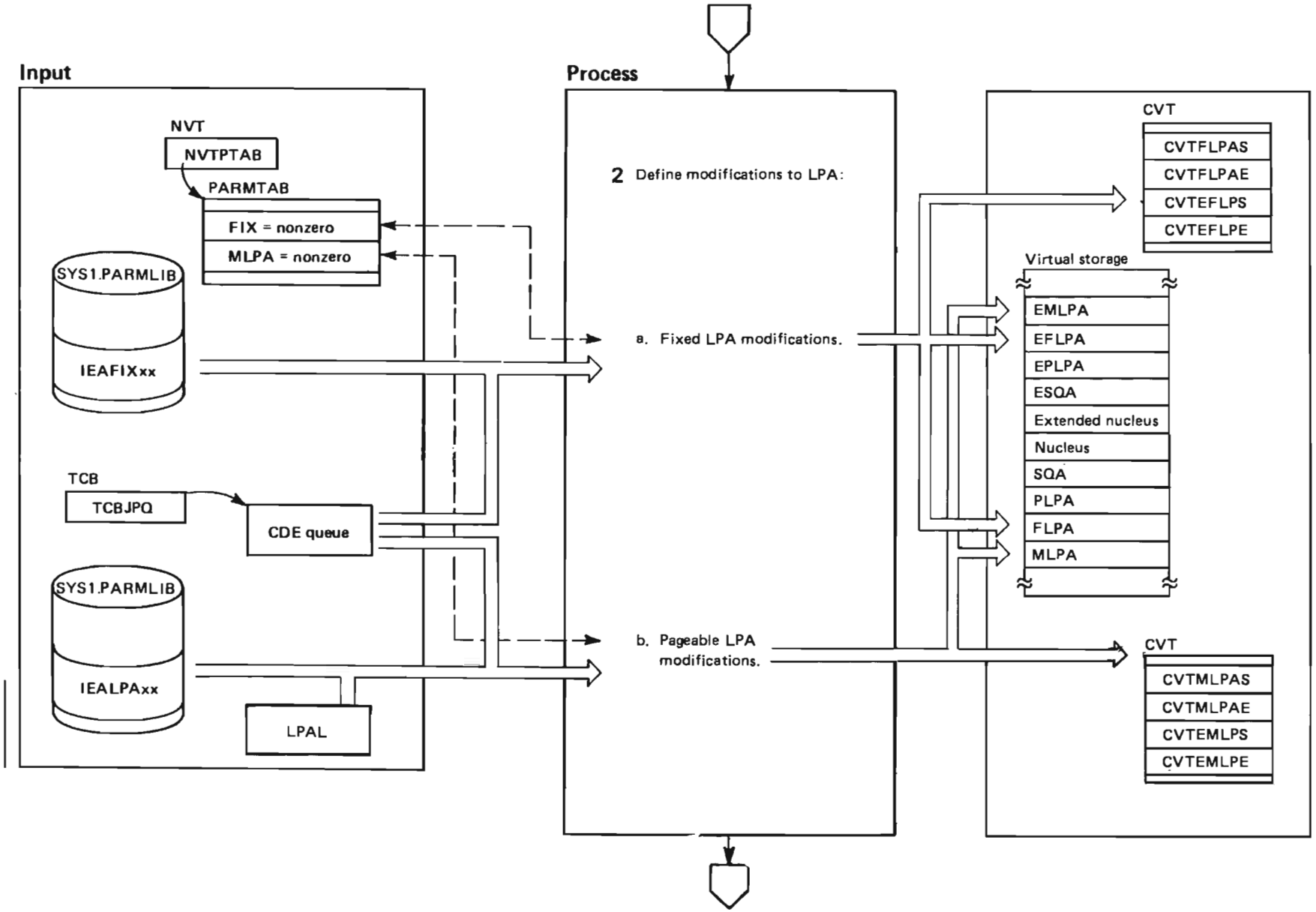


Diagram 32. Link Pack Area Initialization (IEAVNP05) Part 4 of 8

Extended Description	Module	Label
2 By using the FIX and MLPA system parameters, the installation can modify the LPA as follows:		
a. The contents supervisor RIM checks the FIX entry in NIP's PARMTAB to determine whether the installation has specified modules to be added to the fixed LPA (FLPA). If the FIX entry is not zero, it contains the suffix list for an IEAFIXxx member(s) of SYS1.PARMLIB, which IEAVNP05 reads by calling IEAVNPM4. The IEAFIXxx member contains the names of modules that are to be loaded into the FLPA.		NP5FIX
	IEAVNPM4 IEAVNP05	
The EFLPA is located in virtual storage directly above the EPLPA, and the FLPA is situated directly below PLPA. The RIM loads the modules from IEAFIXxx into the FLPA and EFLPA, using the RMODE attribute in the PDS entry for each module to determine whether the module is loaded into FLPA or EFLPA. IEAVNP05 calls IEAVNPC8 to initialize the page tables for the LPA storage. The contents supervisor RIM uses the CDE queue pointed to by NIP's TCB to define any additions to the fixed LPA.		NP5MLPRM NP5FIX
	IEAVNPC8 IEAVNP05	
As the RIM loads the modules into the LPA, it queues the load requests from the NIP TCB. After all modules have been loaded, the RIM places the address of this queue in field CVTQLPAQ. It also updates the CVT fields CVTFLPAS, CVTFLPAE, CVTEFLPS, and CVTEFLPE with the low and high end addresses of the FLPA and EFLPA. The RIM now invokes paging services to cause the FLPA and EFLPA storage to be page fixed.		NP5QLPAQ
b. The contents supervisor RIM checks the MLPA entry in NIP's PARMTAB to determine whether the installation has specified modules to be added to the PLPA. If the MLPA entry is not zero, it contains the suffix list for an IEALPAxx member(s) of SYS1.PARMLIB which IEAVNP05 reads by calling IEAVNPM4. The IEALPAxx member contains the names of the modules that are to be loaded into the PLPA.		NP5MLPA
The RIM loads the PLPA and EPLPA modifications into virtual storage, using the RMODE attribute in the PDS entry for each module to determine whether the module is loaded into MLPA or EMLPA. The MLPA is located directly below the FLPA (if FLPA is present), and the EMLPA is directly above the EFLPA (if the EFLPA is present). Once the user-specified modules have been loaded, IEAVNP05 builds separate areas of the MLPA and EMLPA containing the device support modules that are named in the LPA device support module list (LPAL) and that reside in SYS1.LINKLIB. IEAVNP05 calls IEAVNPC8 to initialize the page tables for the LPA storage. The RIM uses the CDE queue pointed to by NIP's TCB to define any additions to the PLPA.		NP5MLPRM
	IEAVNPC8 IEAVNP05	NP5QLPAQ
The RIM updates the CVT fields CVTMLPAS, CVTMLPAE, CVTEMLPS, and CVTEMLPE with the low and high end addresses of the MLPA and EMLPA.		NP5MLPA

Diagram 32. Link Pack Area Initialization (IEAVNP05) Part 5 of 8

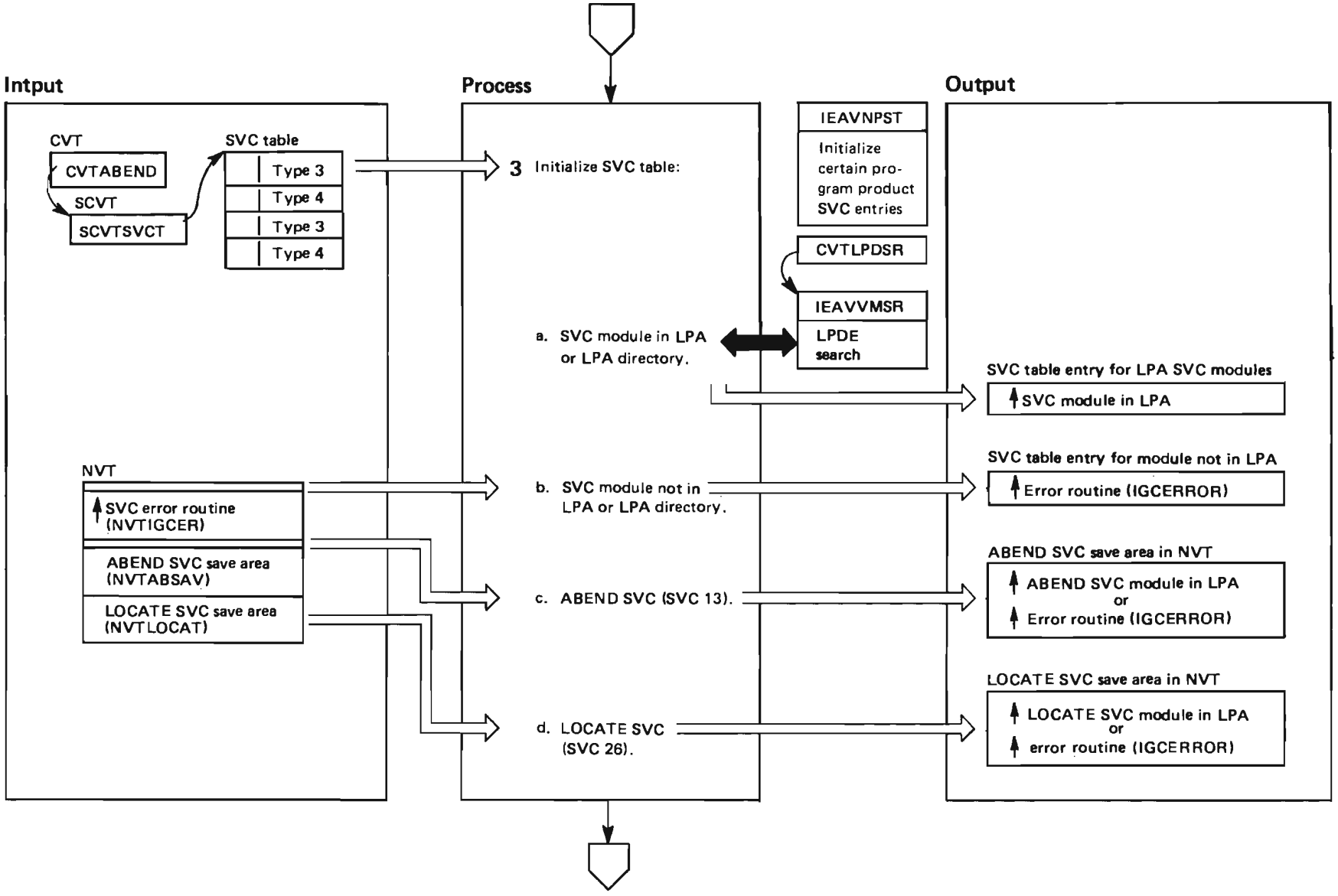


Diagram 32. Link Pack Area Initialization (IEAVNP05) Part 6 of 8

Extended Description	Module	Label
3 The SVC table contains one 8-byte entry for each system SVC routine. The contents supervisor RIM initializes entries for type 3 and type 4 SVC routines. For each entry that represents a type 3 or type 4 SVC routine, the RIM constructs the associated SVC name and determines whether the SVC routine is in the active link pack area. That is, the CDE for the SVC routine is on the CDE queue pointed to by IEAQLPA. (This pointer resides in the nucleus and is pointed to by CVTQLPAQ.)	IEAVNPS5	NPS5SVCT
a. If so, the RIM places the address of the SVC routine into the SVC table entry. If the RIM cannot find the SVC routine in the active LPA, it passes control to the supervisor LPDE search routine (IEAVVMSR), which resides in the nucleus; the entry point address of IEAVVMSR is in field CVTLPDSR. If IEAVVMSR finds the SVC routine in the LPA directory, the RIM places the address of the SVC routine into its SVC table entry. In either case, if the entry point address is resolved, the RIM sets an indicator (SVCAMODE) in the SVC entry to indicate in what addressing mode the SVC is to receive control.		NPFNDSVC
b. If the RIM cannot find a non-ESR SVC routine in either the active LPA or the LPA directory, the RIM moves the address of the non-ESR SVC error routine (IGCERROR) from field NVTIGCER to the SVC table entry. IGCERROR will cause an ABEND for a task that issues a request for an unavailable SVC routine. Finally, the RIM writes a message to the operator indicating the module that was not found. By initializing the type 3 and type 4 SVC entries the program management RIM terminates the trapping of type 3 and type 4 SVCs that was set by IEAVNIPM.		NPFNDSVC
c. Because NIP traps requests for the ABEND SVC (SVC 13), the SVC table entry for the ABEND SVC routine does not reflect the SYSGEN value: earlier in NIP's processing, NIP saved the SYSGEN value for the ABEND SVC entry in a save area in the NVT (NVTABSAV). (For more information about NIP's ABEND trap routine, refer to the "Diagnostic Aids" section of this publication.) The RIM treates the NVTABSAV as if it were the ABEND SVC routine in either the LPA or the LPA directory, the RIM places the address of the ABEND SVC routine in the save area; otherwise, the RIM stores the address of IGCERROR in the save area. As part of NIP's exit processing, NIP moves the saved ABEND entry from the NVT into the SVC table.		NPS5SVCT
d. Because NIP traps requests for the LOCATE SVC (SVC 26), the SVC table entry for the LOCATE SVC routine does not reflect the SYSGEN value: earlier in NIP's processing, NIP saved SYSGEN value for the LOCATE SVC entry in a save area in the NVT (NVTLOCAT). (For more information about NIP's LOCATE trap routine, refer to the "Diagnostic Aids" section of this publication.) The RIM treates the LOCATE save area (NVTLOCAT) as if it were the LOCATE SVC routine in either the LPA or the LPA directory, the RIM places the address of the LOCATE SVC routine in the save area; otherwise, the RIM stores the address of IGCERROR in the save area. As part of NIP's exit processing, NIP moves the saved LOCATE entry from the NVT into the SVC table.		

Diagram 32. Link Pack Area Initialization (IEAVNP05) Part 7 of 8

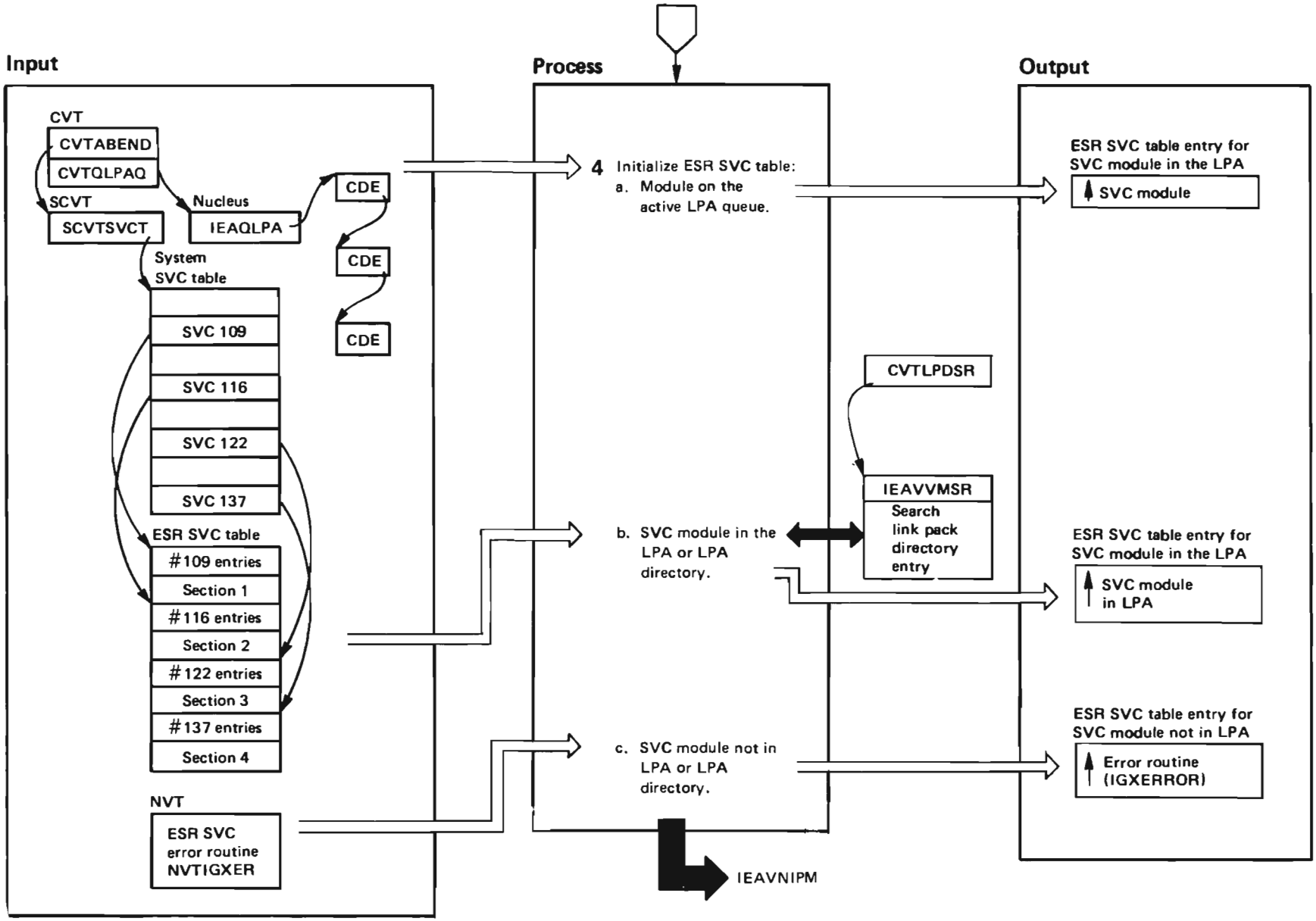


Diagram 32. Link Pack Area Initialization (IEAVNP05) Part 8 of 8

Extended Description	Module	Label
4 The extended SVC router (ESR) SVC table contains one 8-byte entry for each ESR SVC routine. It differs from the system SVC table in that it consists of 4 variable-length sections. An 8-byte prefix to each section of the ESR table contains the highest ESR SVC number for that section.		NPS5ESRT
a. The RIM initializes entries in each section for type 3 and type 4 SVCs. For each type 3 or 4 entry, the RIM constructs the name and determines whether the module is on the active LPA queue, that is, whether the CDE for the SVC routine is on the CDE queue pointed to by IEAQLPA (this pointer resides in the nucleus and is pointed to by CVTQLPAQ).		NPFNDSVC
b. If the module is on the active LPA queue, the RIM places the address of the SVC routine into the SVC table entry. If the RIM cannot find the SVC routine in the active LPA queue, it passes control to the supervisor link pack directory entry (LPDE) search routine (IEAVVMSR). If IEAVVMSR finds the SVC routine in the LPA directory, the RIM places the SVC routine's address into its SVC table entry.		NPFNDSVC
c. If the RIM cannot find the SVC routine in either the active LPA or the LPA directory, the RIM moves the address of the ESR SVC error routine (IGXERROR) from field NVTIGXER to the SVC table entry. IGXERROR causes an ABEND for a task that issues a request for an unavailable ESR SVC routine. Because there may be a considerable number of unresolved type 3 and type 4 ESR SVC entries, no message is written to the operator (as is done by the system SVC table routine), indicating the not-found condition.		

Diagram 33. Auxiliary Storage Management Quick Start Record Initialization (ILRQSRIT) Part 1 of 2

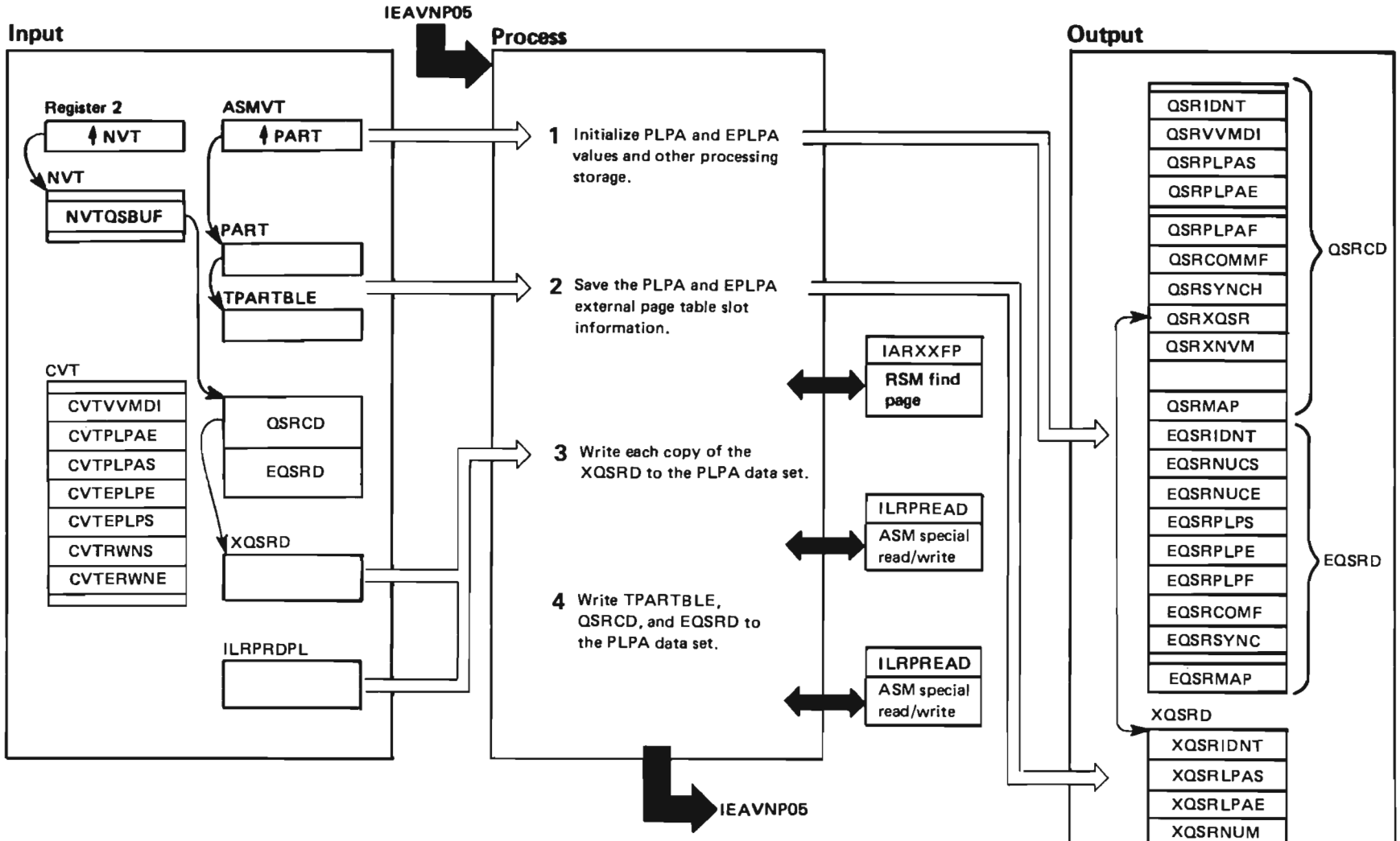


Diagram 33. Auxiliary Storage Management Quick Start Record Initialization (ILRQSRIT) Part 2 of 2

Extended Description	Module	Label
During a cold start initialization, the contents supervisor RIM, IEAVNP05, invokes ILRQSRIT to save logical slot IDs (LSIDs) in the quick start record extensions (XQSRDs). The XQSRDs are written to the PLPA data set and the logical slot ID of each XQSRD is saved in the quick start record (QSRCD) for PLPA pages and in the extended quick start record (EQSRD) for the EPLPA pages. ILRQSRIT then calls ILRPREAD to write the QSRCD, EQSRD and TPARTBLE to the PLPA data set. During subsequent quick and warm starts, the ASM RIM, ILRASRIM, will read these control blocks from the PLPA data set to rebuild the PLPA and EPLPA.		
1 Using the NVT pointer to the QSRCD (EQSRD follows the QSRCD in contiguous extended SQA), store the PLPA and EPLPA values from the CVT (CVTVVMDI, CVTPLPAE, CVTPLPAS, CVTEPLPE, CVTEPLPS, CVTRWNS, CVTERWNE) into the QSRCD and EQSRD. (IEAIPL05 has written PLPA and EPLPA to auxiliary storage and saved the PLPA and EPLPA hash values before invoking ILRQSRIT.)	ILRQSRIT	
Check the ASMVT to determine if the PLPA and common data sets are full. If they are, indicate this condition in the QSRCD (QSRCOMMF='1'). In any case, zero the QSRMAP and EQSRMAP sections of the QSRCD and EQSRD. Initialize the XQSRD.		
2 The number of PLPA or EPLPA pages in the lowest segment is the number of pages specified in QSRPLPAS (or EQSRPLPS) subtracted from the number of pages in a segment (256). Compute the segment boundary address that contains the first page. Starting with the segment boundary address for the first page, invoke IARXXFP for the first page in the segment until the segment address is equal to QSRPLPAE (or EQSRPLPE) (IARXXFP returns an external page table address.) Copy the logical slot id	ILRQSRIT IARXXFP ILRQSRIT	

Extended Description	Module	Label
2 *(continued)* (LSID) portions of each page table (PLPA slot and duplex slot information) to the LSID portion of the XQSRD. As each XQSRD is completed, write it to the PLPA data set or, if no space is available on the PLPA data set to the common data set. Save the LSID of the XQSRD in the QSRCD or EQSRD. Process all pages of PLPA and EPLPA. Continue with step 3. If IARXXFP fails to return an external page table, issue message IEA943W to indicate that an error occurred while processing external page tables and then put the system into a X'060' wait state.		
3 After all PLPA and EPLPA page information has been copied to the QSRCD, EQSRD, and XQSRDs mark the PLPA data set as read-only and the common data set as active.		
4 Write the TPARTBLE, the QSRCD, and the EQSRD to PLPA. If ILRPREAD fails, issue message IEA936D to notify the operator that future quick or warm starts for this IPL will fail. If ILRPREAD is successful, free the QSRCD and EQSRD storage and delete ILRPREAD. The TPARTBLE is not freed; it is still needed and will be freed by ILRTMI00 during master scheduler initialization.	ILRPREAD	
Recovery Processing: None.		

Diagram 34. CSA Parameter Initialization (IEAVNPB8) Part 1 of 8

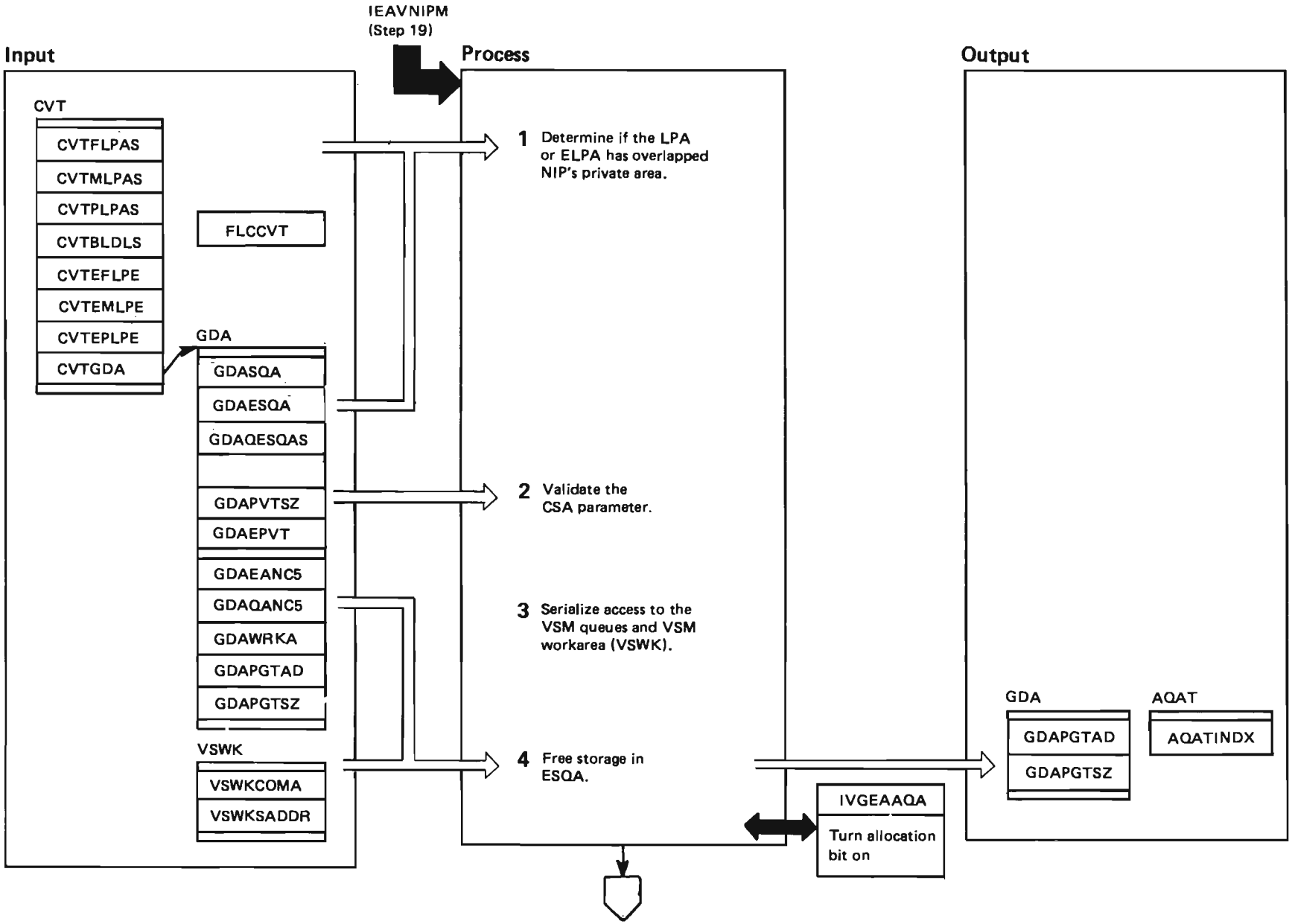


Diagram 34. CSA Parameter Initialization (IEAVNPB8) Part 2 of 8

Extended Description	Module	Label
IEAVNIPM calls the virtual storage management (VSM) RIM, IEAVNPB8, to process the CSA system parameter and build the VSM control blocks and page tables necessary to describe the common storage area (CSA) and the extended CSA (ECSA). The syntax of the parameter is CSA=(a, b), where 'a' and 'b' are values that are multipled by 1K to determine the total amount of storage that VSM is to allocate for the CSA and the ECSA. If the CSA parameter is not specified, the RIM uses the default CSA=(100, 100).		
1 The VSM RIM searches the CVT for the start address of the LPA and the ending address of the ELPA. (These addresses were established by the contents supervisor RIM, IEAVNP05.) The VSM RIM determines whether the LPA and the ELPA overlap into NIP's private areas by comparing these two addresses with the end address of the private area (GDAPVT+GDAPVTSZ) and the start address of the extended private (GDAEPVT). If there is an overlap, the RIM issues message IEA140W to inform the operator that there is no space available for CSA or ECSA and places the system in a wait state with a wait state code of X'0E3'.	IEAVNPB8	IEAVNPB8
2 The VSM RIM validates the CSA parameter and ensures that the CSA and ECSA will not overlap the NIP private area. If invalid vlaues for 'a' or 'b' were specified, the RIM issues message IEA320I to inform the operator of the error. If a value was specified that would cause an overlap, the RIM issues message IEA909I to inform the operator that the value is too large. For both these error conditions, the RIM prompts the operator for a new value. If the value the operator specified for 'a' causes the start address of the CSA to be below 8 megabytes, the RIM issues message IEA913I to inform the operator that the private area is possibly constrained.		VALIDATE
3 The VSM RIM issues a SETLOCK macro instruction to obtain the LOCAL and VSMFIX locks. There locks serialize access to the VSM queues and VSM workarea (VSWK).		
4 The VSM RIM frees some storage in ESQA that IEAIPL04 allocated for common page tables. The RIM frees all this sotrage except the area that contains the page tables that describe the CSA and the ECSA. Before the RIM issues the FREEMAIN macro instruction for these areas, it calls IGVEAAQA to set on the address queue anchor table (AQAT) allocation bits to temporarily assign these areas to subpool 245. The RIM then issues the FREEMAIN macro instruction and ajusts GDAPGTAD and GDAPGTSZ to reflect the new address and size of the common area page tables.	IGVEAAQA IEAVNPB8	 IEAVNPB8

Diagram 34. CSA Parameter Initialization (IEAVNPB8) Part 3 of 8

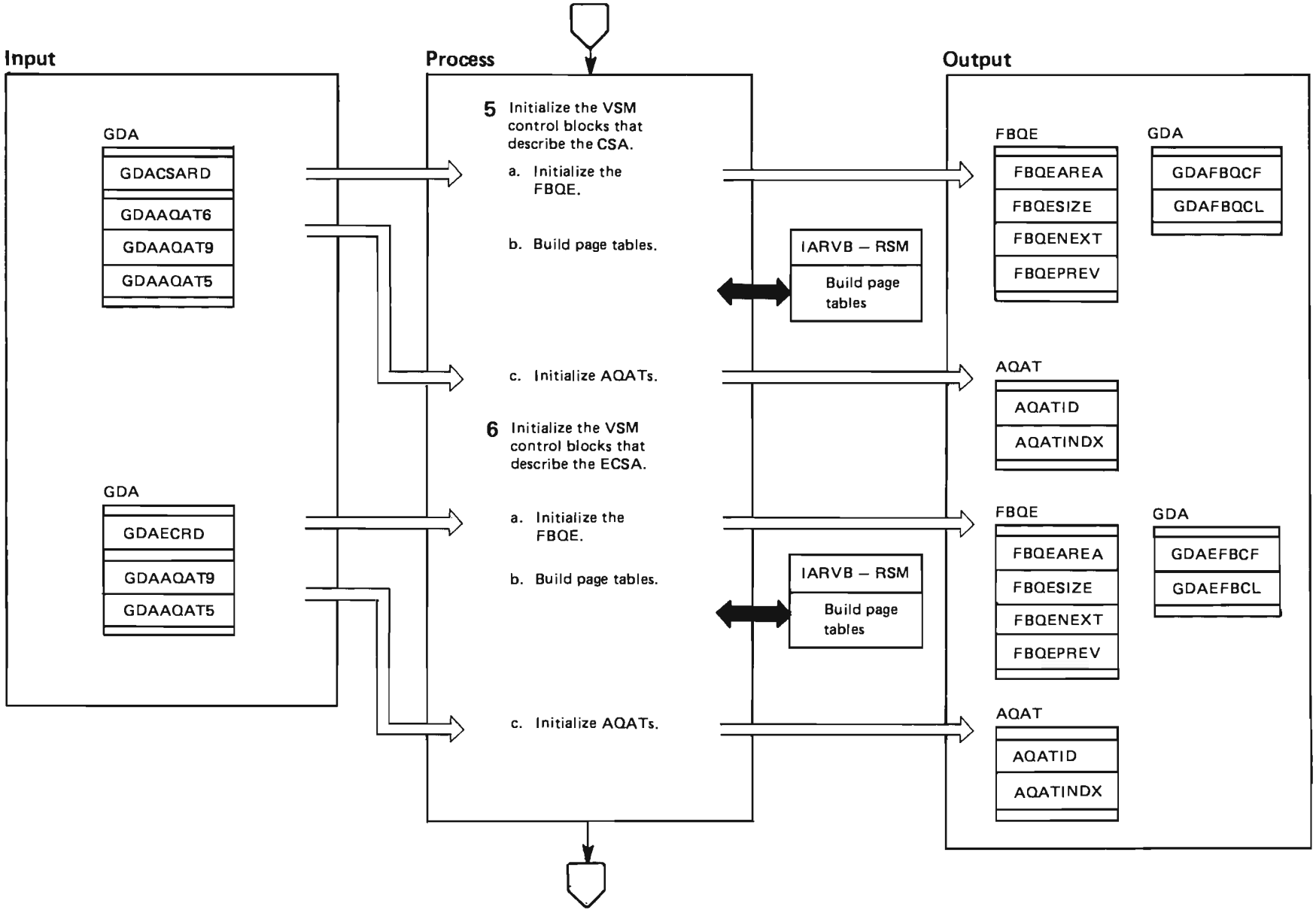


Diagram 34. CSA Parameter Initialization (IEAVNPB8) Part 4 of 8

Extended Description	Module	Label
5 The VSM RIM allocates and initializes the VSM control blocks and page tables to describe the additional CSA.		
a. Now that the bounds of the non-extended private area are established, the RIM builds the free block queue element (FBQE) to describe the CSA and enqueues it on the CSA FBQE queue anchored in the GDA.		
b. The RIM calls the RSM module IARVB to build page tables for the CSA. If there is not enough real storage available for the page tables, the VSM RIM issues message IEA907W to inform the operator that it is unable to initialize the page table for CSA and places the system in a wait state with a wait state code of X'03D'.	IARVB IEAVNPB8	IARVBCSG IEAVNPB8
c. The VSM RIM issues a GETMAIN macro instruction to obtain ESQA or SQA storage to build the subpool 226 AQATs to describe the CSA. (This step and step 6c are necessary because, when there is not enough storage available in the SQA or the ESQA, VSM converts storage from the CSA or the ECSA to the SQA or ESQA. These AQATs exist for this situation.) The RIM zeroes the AQATs, sets the AQAT index table entries (AQATINDXs) to point to the appropriate AQATs, and initializes the control block ID (AQATID). The VSM RIM initializes the subpool 239 and subpool 245 AQATs for the CSA in the same way.		BUILDAQT
6 The VSM RIM allocates and initializes the VSM control blocks and page tables to describe the additional ECSA.		
a. Now that the bounds of the extended private area are established, the RIM builds the free block queue element (FBQE) to describe the ECSA and enqueues it on the ECSA FBQE queue anchored in the GDA.		
b. The RIM calls the RSM module IARVB to build page tables for the ECSA. If there is not enough real available storage for the page tables, the VSM RIM issues message IEA907W to inform the operator that it cannot initialize the page table for ECSA and places the system in a wait state with a wait state code of X'03D'.	IARVB IEAVNPB8	IARVBCSG IEAVNPB8
c. The VSM RIM issues a GETMAIN macro instruction to obtain ESQA or SQA storage to build the subpool 239 AQATs to describe the ECSA. The RIM zeros the AQATs, sets the AQAT index table entries (AQATINDXs) to point to the appropriate AQATs, and initializes the control block ID (AQATID). The VSM RIM initializes the subpool 245 AQATs for the ECSA in the same way. (Note that the VSM RIM does not initialize subpool 226 AQATs because subpool 226 is only in non-extended virtual storage.)		BUILDAQT

Diagram 34. CSA Parameter Initialization (IEAVNPB8) Part 5 of 8

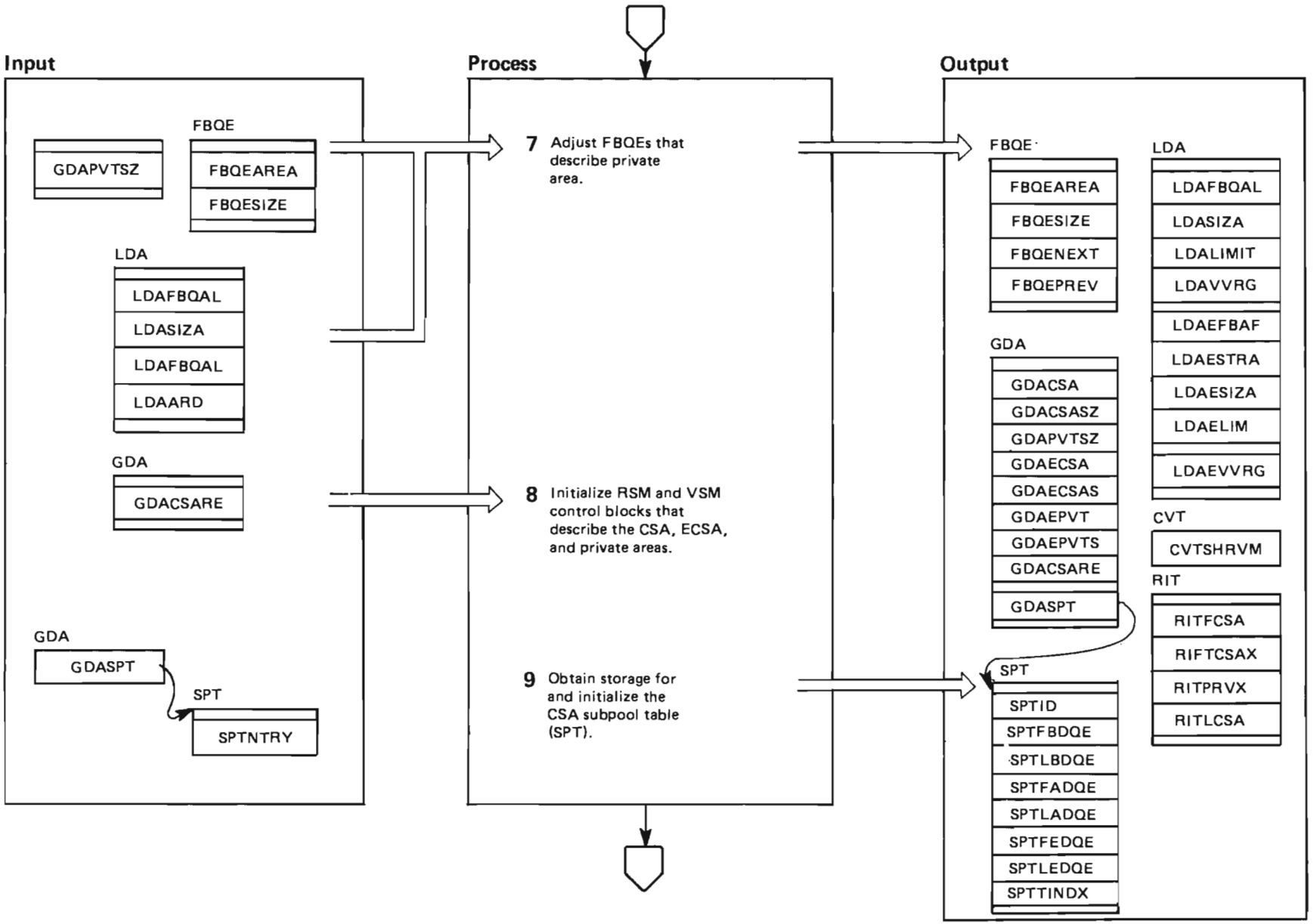


Diagram 34. CSA Parameter Initialization (IEAVNPB8) Part 6 of 8

Extended Description	Module	Label
7 If, after the RIM allocates the CSA and ECSA, there is any space left between the CSA and the non-extended private area or between the ECSA and the extended private area, the VSM RIM adjusts the FBQE or builds a new FBQE to include the unused areas within the private areas. It also initializes fields in the LDA that contain information about the FBQE queues.		IEAVNPB8
8 The RIM initializes RSM and VSM control blocks that contain information about the bounds of the CSA, the ECSA, and the private areas.		
9 The VSM RIM issues a GETMAIN macro instruction to obtain storage in subpool 245 for the CSA subpool table (SPT). This table contains DQE queue anchors for each CSA subpool/key combination for each of keys 0 through 15. It initializes the control block ID (SPTID) and the DQE anchors in the SPT.		

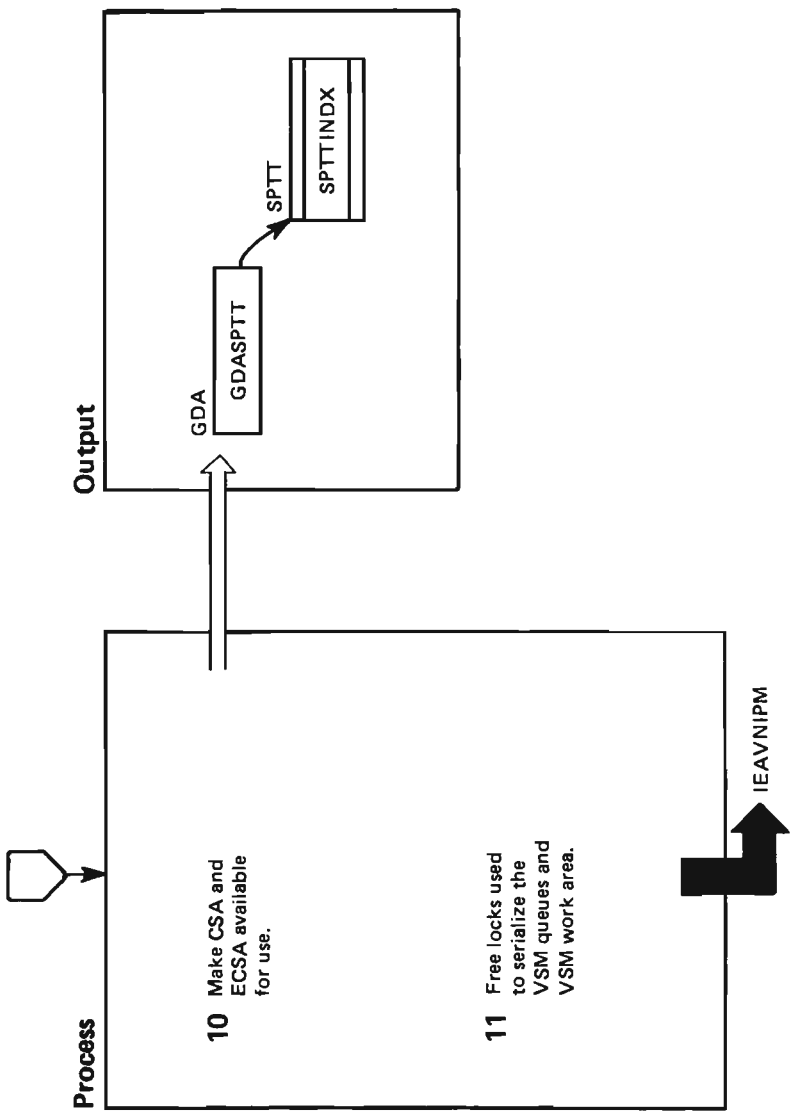


Diagram 34. CSA Parameter Initialization (IEAVNPB8) Part 7 of 8

Diagram 34. CSA Parameter Initialization (IEAVNPB8) Part 8 of 8

Extended Description	Module	Label
10 Until this point in the initialization process, NIP and the RIMs are not able to issue GETMAIN and FREEMAIN macro instructions for storage in the CSA and ECSA because the SPTTINDX entry for these subpools points to a subpool translation table (SPTT) entry that is marked invalid. This will cause any GETMAIN macro instruction that requests storage in a CSA or ECSA subpool to fail. The VSM RIM resets the SPTTINDX entries to point to the correct SPTT entries.		
11 The VSM RIM issues a SETLOCK macro instruction to free the LOCAL and VSMFIX locks and returns control to IEAVNIPM.		

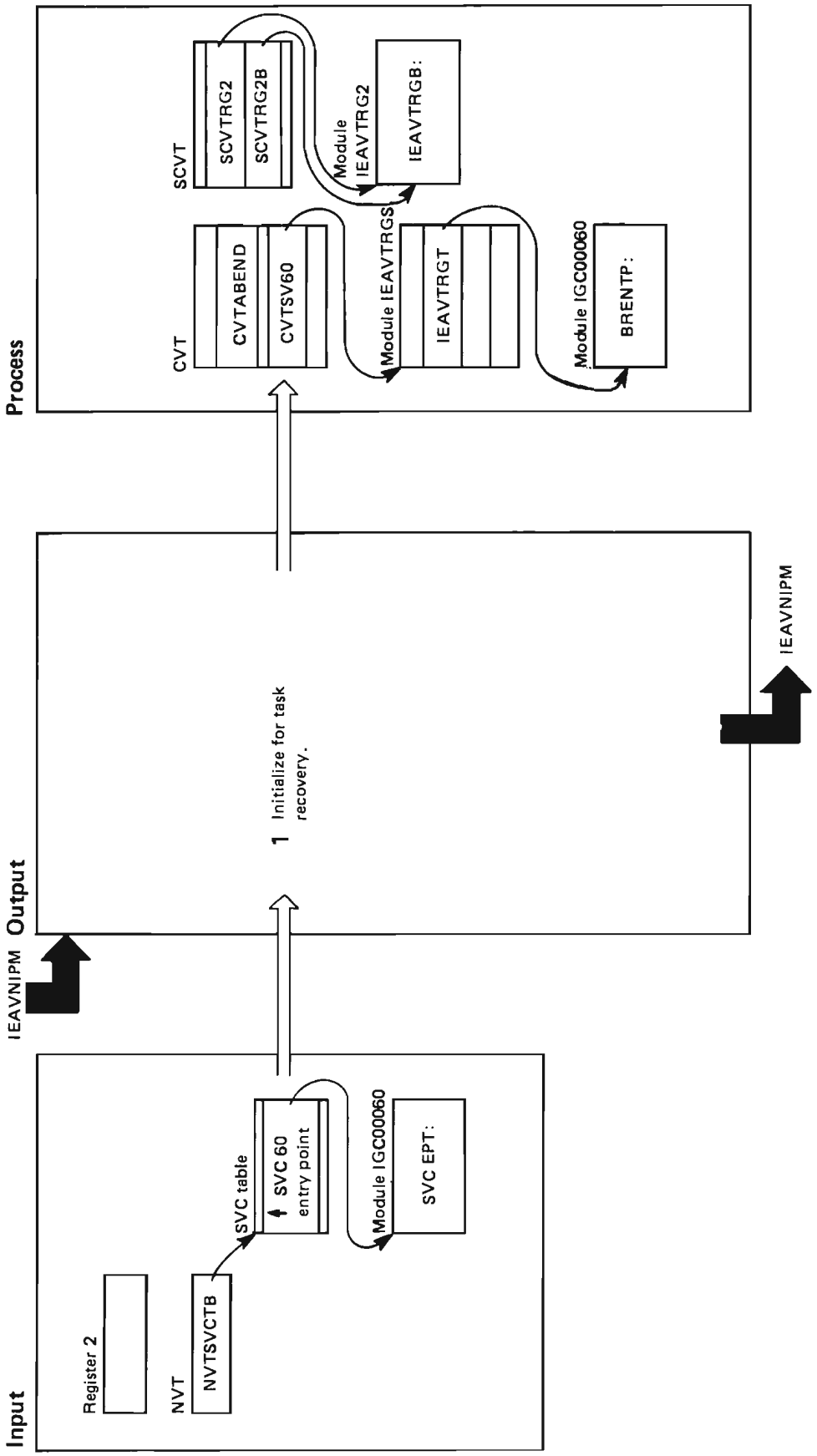


Diagram 35. Task Recovery Initialization (IEAVNPD6) Part 1 of 2

Diagram 35. Task Recovery Initialization (IEAVNPD6) Part 2 of 2

Extended Description	Module	Label
The recovery termination manager (RTM) RIM, IEAVNPD6 initializes the recovery termination functions that support task recovery (ESTAE) and calls the SVC dump and ABDUMP RIMs to initialize the SVC dump and ABDUMP dumping services.		
1 For task recovery, the RTM RIM initializes CVTSV60 with the branch entry point address of the nucleus CSECT IEAVTRGS. This CSECT resides below the 16-megabyte line and contains the entry point for all BRANCH=YES requests for the ESTAE/ESTAI service routine. The RTM RIM then initializes the field IEAVTRGT (an address constant in IEAVTRGS) with the address of the branch-entered ESTAE/ESTAI service routine, which resides above the 16-megabyte line. (IGC00060 is also named IEAVSTA0) The IEAVTRGT address constant enables IEAVTRGS to pass control to IEAVSTA0 in 31-bit addressing mode while preserving the addressing mode of the branch caller of ESTAE/ESTAI.	IEAVNPD6	
The RTM RIM also initializes SCVTRG2 with the address of of the entry point to the interface routine (IEAVTRG2) that RTM2 (task recovery) uses to call the resource managers that must get control in 24-bit addressing mode. The RIM also initializes SCVTRG2B with the address of location IEAVTRGB in routine IEAVTRG2. IEAVTRGB is the return point for calls to the resource managers; it ensures that control returns to RTM2 in 31-bit addressing mode.		

Diagram 36. Supervisor Control RIM (IEAVNP09) Part 1 of 2

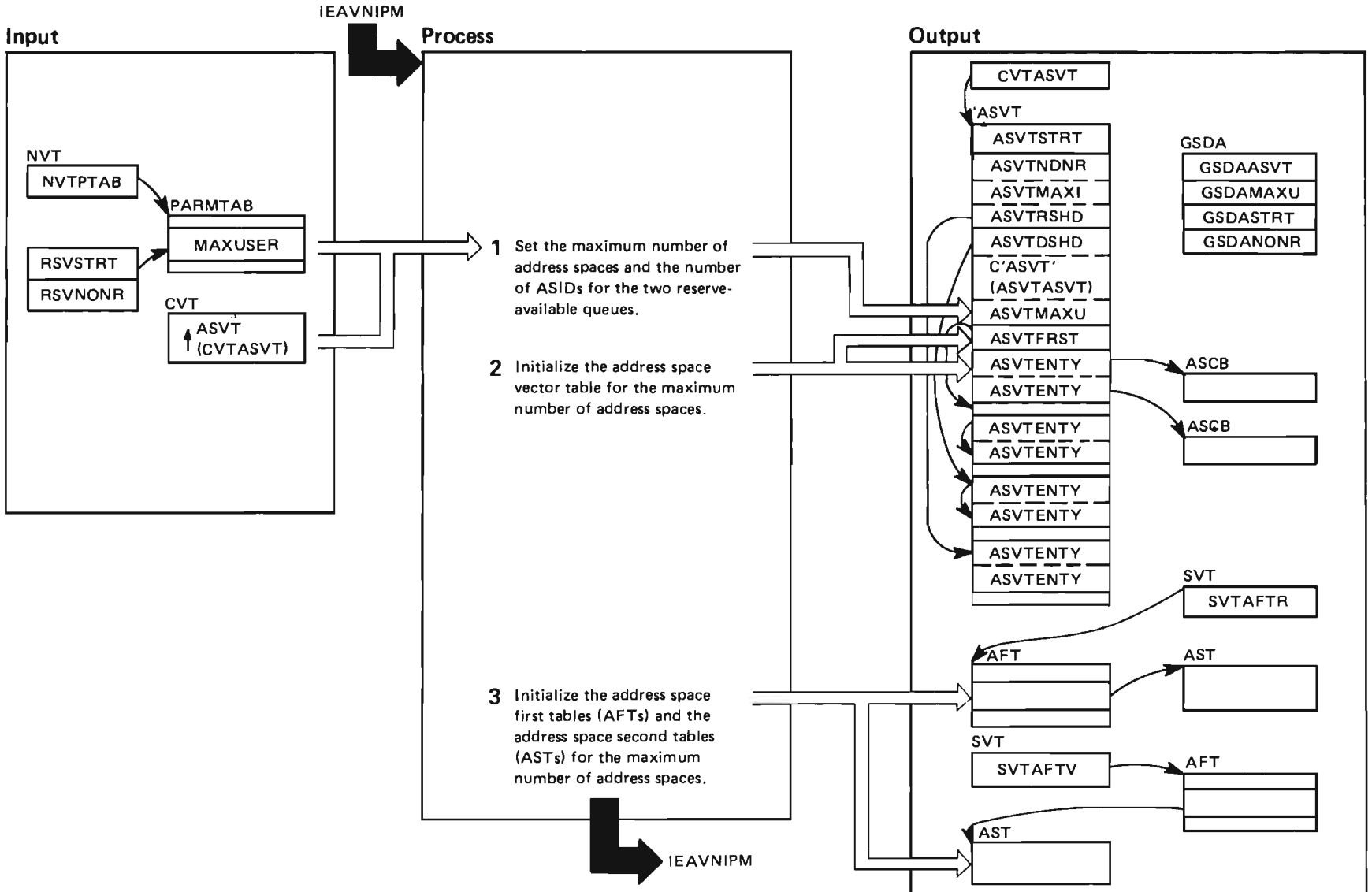


Diagram 36. Supervisor Control RIM (IEAVNP09) Part 1 of 2

Extended Description	Module	Label
Supervisor control controls the execution of tasks in an address space. The supervisor control RIM, IEAVNP09, initializes task management by initializing the address space vector table (ASVT), the address space first tables (AFTs). and the address space second tables (ASTs), as follows:		
1 Supervisor control initializes the ASVT with the maximum number of address spaces that may be created. Using the MAXUSER system parameter, the installation can specify the maximum number of address spaces. Using the RSVNONR system parameter, the installation can specify the number of ASIDs reserved as replacements for non-reusable address sapces. Using the RSVSTRT system parameter, the installation can specify the number of ASIDs reserved for address spaces created in response to the START command. The RIM checks fields MAXUSER, RSVNONR, and RSVSTRT in PARMTAB to determine whether the installation specified a value. If the installation specified one or more invalid values, the RIM uses the NIPPRMPT service routine to prompt the operator for new values. If the installation did not specify MAXUSER, the RIM uses a default of 256 and issues message IEA891I. If the installation did not specify RSVNONR, the RIM uses a default of 5 and issues message IEA736I. If the installation did not specify RSVSTRT, the RIM uses a default of 5 and issues message IEA737I.	IEAVNP09	
The total number of ASVT entries is the sum of the three system parameters. If this sum is greater than X'7FFF', the RIM issues message IEA738I and prompts the operator to respecify all three system parameters. If the sum is within this limit, the RIM obtains storage for the ASVT and ASTs. If the installation specified a value too large for the available SQA space, the RIM issues message IEA892I and prompts the operator to respecify all three system parameters.		

Extended Description	Module	Label
2 The RIM creates an entry in the ASVT for each possible address space. For an address space already created, the RIM sets to zero the high-order bit in its associated ASVT entry and places the address of the address space's ASCB into the entry. For those entries for which address spaces have not been created, the RIM sets the high order bit to 1 and places the address of the next available entry into the current entry. The RIM initializes field ASVTFRST to point to the first available ASVT entry. The RIM initializes field ASVTRSHD to point to the first available entry for those ASIDs reserved for started/SASI address spaces. The RIM initializes field ASVTDSHD to point to the first available entry for those ASIDs reserved as replacements for ASIDs which become non-reusable. The RIM initializes the CVT field, CVTASVT to point to the newly created permanent ASVT.		
If there is a valid pointer in the CVT to the global system data area, the RIM stores copies of the system parameters MAXUSER, RSVNONR, and RSVSTRT into the fields GSDAMAXU, GSDANONR, and GSDASTRT. The RIM saves the address of the permanent ASVT in GSDAASVT.		
3 The RIM creates and initializes AFTs and ASTs. There are two AFTs created: the hardware AFT, which is pointed to by SVTAFTR, and the backup AFT used, if required, to recreate the hardware AFT. SVTAFTV points to the backup AFT. The hardware AFT contains real addresses; the backup AFT contains virtual addresses. Both AFTs are located in the private area of the master scheduler's address space.		
The ASTs are located in SQA. Each AST contains entries for 64 ASIDs. Each AFT entry is to contain the address of an AST. If the high order bit of an AFT entry is 0, there is a valid AST address in that entry; if the high order bit is 1, there is no AST address in that entry.		
After the AFTs and ASTs have been initialized, the RIM loads control register 14 on the IPL processor with the real AFT address. The RIM returns control to IEAVNIPM.		

Diagram 37. REAL, VVREGN, and RSU Parameter Initialization (IEAVNPD8) Part 1 of 8

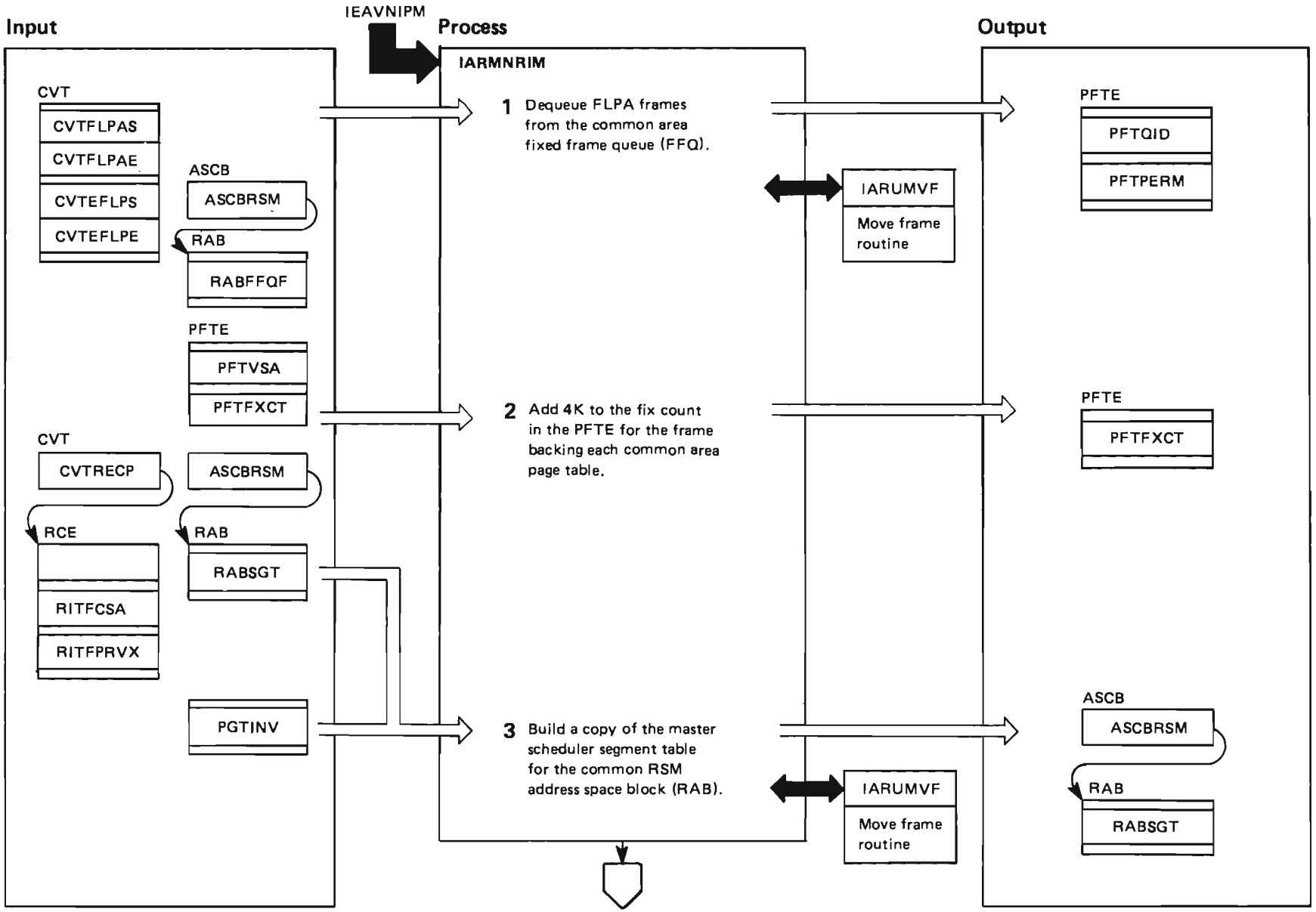


Diagram 37. REAL, VVREGN, and RSU Parameter Initialization (IEAVNPD8) Part 2 of 8

Extended Description	Module	Label
IEAVNIPM calls the real storage management (RSM) RIM, IEAVNPD8, to initialize RSM control blocks and process the REAL, VRREGN, and RSU system parameters. (The first module within IEAVNPD8 is IARMN).	IEAVNPD8	
1 The RSM RIM searches the common area fixed frame queue for frames in the fixed link pack area (FLPA), or extended FLPA regions. It calls the RSM service routine, IARUMVF, to dequeue these frames. The RSM RIM marks the frames permanently resident (PFTPERM=1) and places in PFTQID the ID that identifies the frames in FLPA.	IARMN IARUM	IARMNRIM IARUMVF
2 The RSM RIM locates the PFTE for the frame backing the page table for each common segment. It adds 4K to the fix count for the page table so that common segments will always appear to have valid pages.		
3 The RSM RIM issues a GETMAIN macro instruction for two contiguous pages of extended system queue area (ESQA) for the common segment table. If the two pages are valid in storage (PGTINV=0), the RSM RIM will not use these frames. It calls IARUMVF to move the frames backing the pages to the available frame queue (AFQ). The RSM RIM places the real addresses of the frames backing the master scheduler segment table in the page table entries for the pages in (ESQA). The RSM RIM puts the virtual address of the common segment table in ESQA into the RSM address space block (RAB) for the common area.	IARUM	IARUMVF

Diagram 37. REAL, VVREGN, and RSU Parameter Initialization (IEAVNPD8) Part 3 of 8

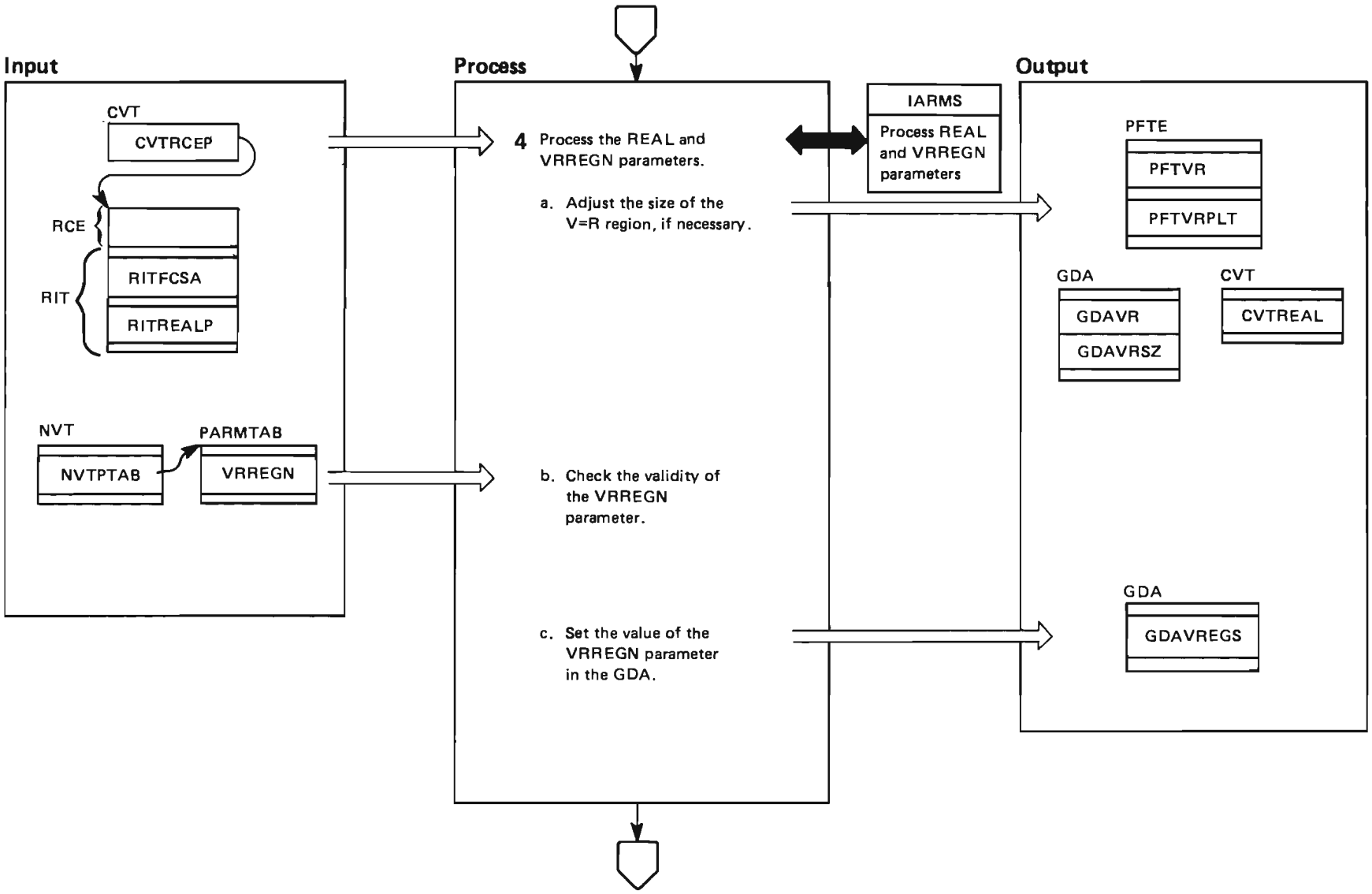


Diagram 37. REAL, VVREGN, and RSU Parameter Initialization (IEAVNPD8) Part 4 of 8

Extended Description	Module	Label
4 The RSM RIM calls IARMSRIM to process the REAL and VRREGN system parameters.	IARMS	IARMSRIM

a. IARMSRIM checks the size of the V=R region set up by IEAVNPE8. If the V=R regions includes real addresses that the VSM RIM, IEAVNPB8, allocated to the common area (non-extended CSA) in virtual storage, the operator must reduce the size of the V=R region. IARMSRIM issues message number IAR002A to inform the operator that the size of V=R is too large and prompts the operator to respecify a new value for the REAL parameter. IARMSRIM checks the operator's response for syntax errors and, if the response is not valid, issues message number IAR006A to inform the operator that the parameter is invalid and again prompts the operator to respecify a new value.

   IARMSRIM marks each frame that is removed from the V=R region as not V=R candidates and not V=R polluted and puts the beginning real address and the size of the V=R region in the global data area (GDA) for use by VSM. IARMSRIM puts the address of the first frame above the V=R area in the CVT.

b. IARMSRIM checks the value specified by the operator for the VRREGN parameter. The VRREGN parameter is used as the default V=R region size for any job with the ADDRSPC=REAL parameter that does not have a REGION parameter. If the value is not valid, IARMSRIM issues message number IAR006A to request that the operator respecify a new value.

c. IARMSRIM places the default V=R region size in the GDA for use by VSM.

Diagram 37. REAL, VVREGN, and RSU Parameter Initialization (IEAVNPD8) Part 5 of 8

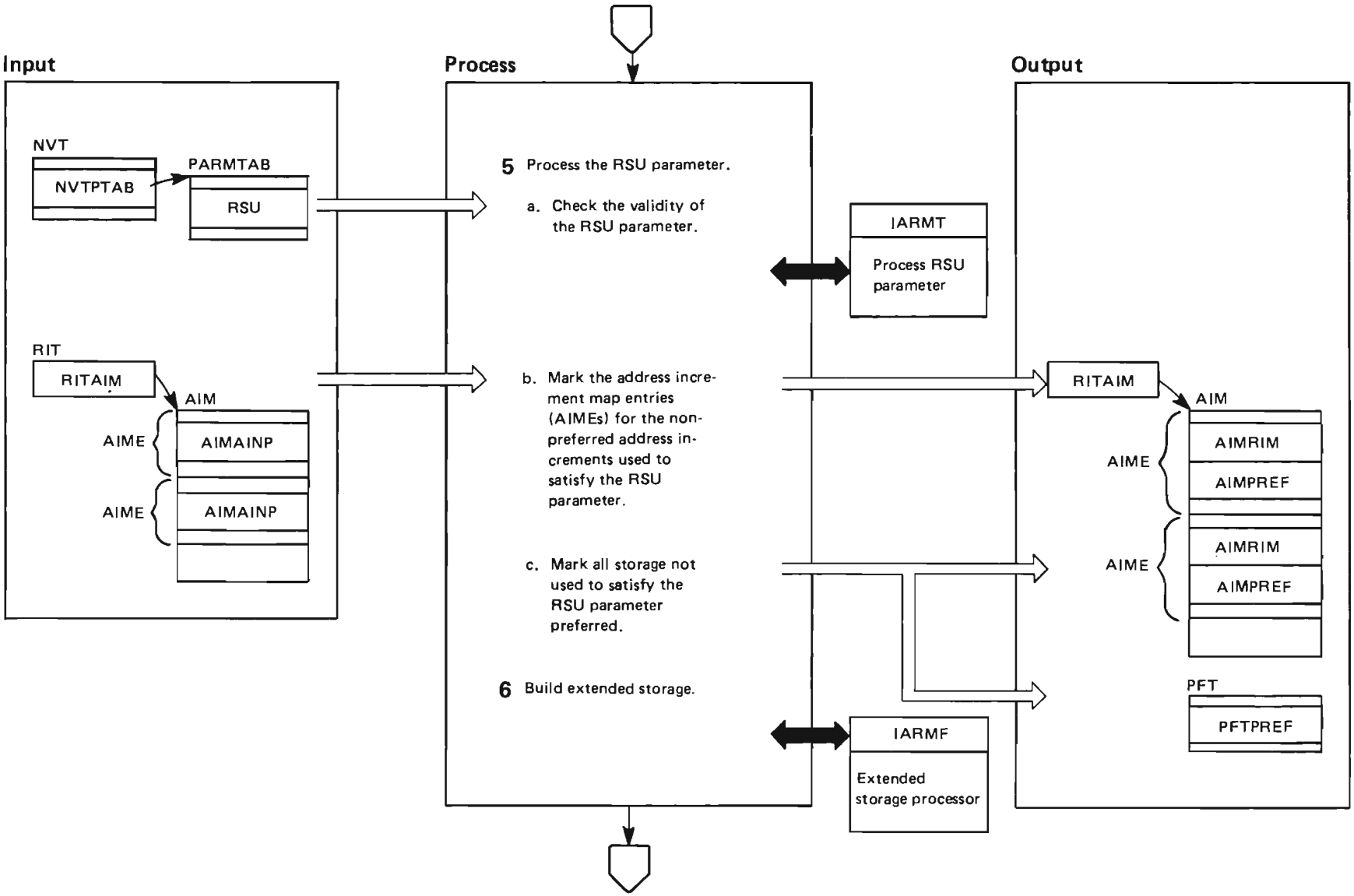


Diagram 37. REAL, VVREGN, and RSU Parameter Initialization (IEAVNPD8) Part 6 of 8

Extended Description	Module	Label
5 The RSM RIM calls IARMTRIM to process the RSU parameter. The RSU parameter specifies the amount of storage that RSM is to keep available for reconfiguration. It allots this area in increments based on the storage unit size.	IARMT	IARMTRIM
a. IARMTRIM checks the RSU parameter. If the value is not valid, IARMTRIM issues message number IAR006A to request that the operator respecify a new value for the RSU parameter.		
b. IARMTRIM selects address increments that contain all offline, non-preferred storage to satisfy the RSU parameter, beginning with the lowest address increment. If the RSU parameter has not been fully satisfied when the highest offline address increment has been selected, IARMTRIM selects online address increments beginning with the highest online address. If there are not enough address increments, IARMTRIM issues the message number IAR004I to inform the operator that the RSU parameter was not processed as specified.		
c. IARMTRIM marks the bands of address increments not used to satisfy the RSU parameter as preferred (AIMPREF=1) and marks each PFTE for the frames in the bands as preferred (PFTPREF=1).		
6 The RSM RIM calls IARMFRIM to build and initialize the extended storage control blocks and queues that RSM needs to control any extended storage that might be configured in the system. If extended storage can be configured in the system, IARMFRIM performs the following:	IARMF	IARMFRIM
a. Calculates the number of extended storage E-frames that can be configured in the system.		
b. Builds the extended storage table (EST) and the extended storage increment map (EIM).		
c. Marks the status of each extended storage frame in its corresponding extended storage table entry (ESTE).		
d. Builds the available ESTE queue and initializes the applicable fields in the RSM internal table (RIT) and the RSM control and enumeration area (RCE).		

Diagram 37. REAL, VVREGN, and RSU Parameter Initialization (IEAVNPD8) Part 7 of 8

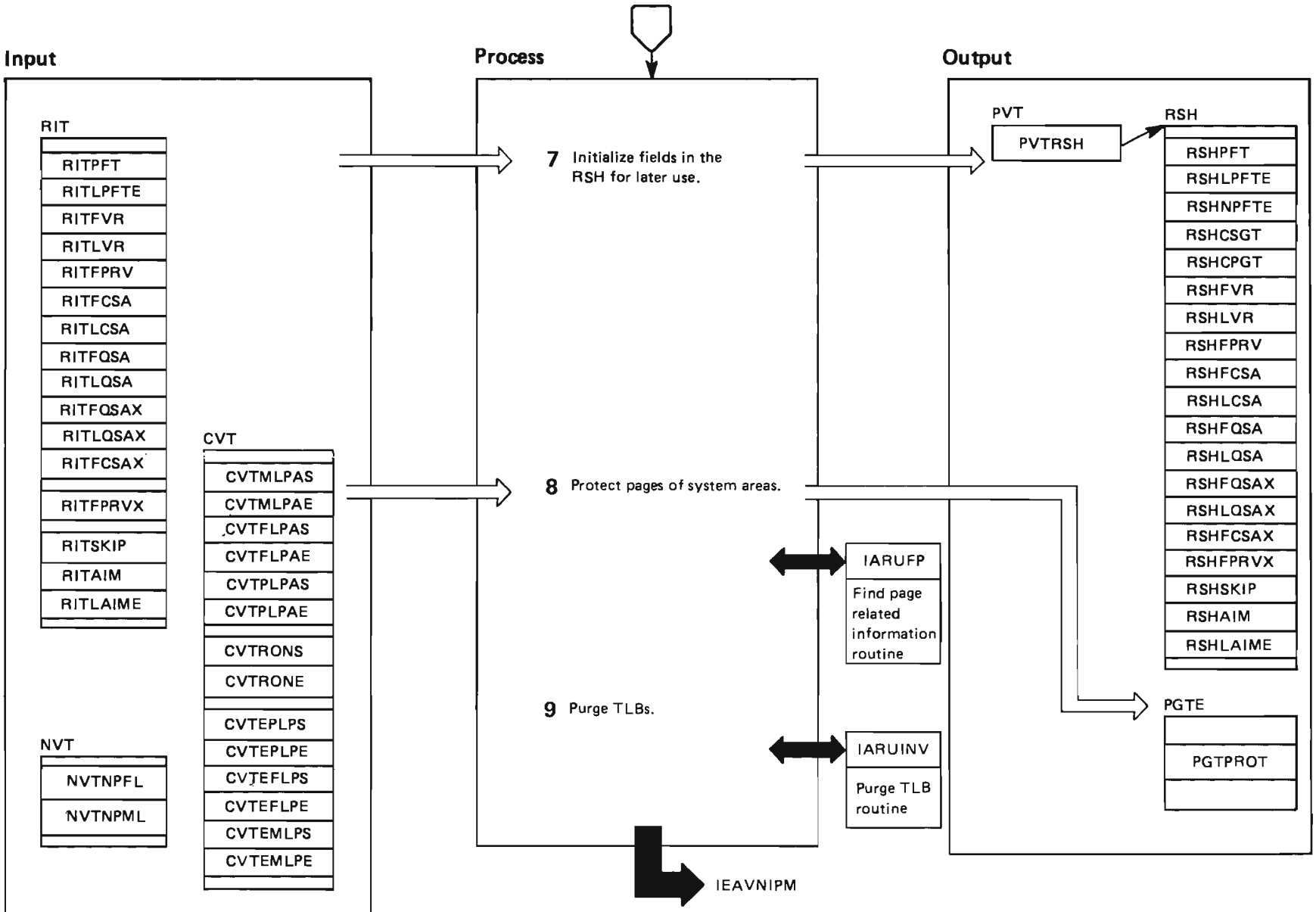


Diagram 37. REAL, VVREGN, and RSU Parameter Initialization (IEAVNPD8) Part 8 of 8

Extended Description	Module	Label
7 The RSM RIM copies values from the RSM internal table (RIT) into the RSM refresh table (RSH) that is in the read-only nucleus. RSM uses these values to refresh the RIT.	IARMN	IARMNRIM
8 The RSM RIM calls the RSM service routine, IARUFP, to locate the PGTE for each page in the system area that is to be protected. It sets the protect bit in each PFTE. The following areas are protected: • the read-only nucleus • the nucleus map (NUCMAP) • the pageable link pack area (PLPA) • the extended PLPA • the fixed link pack area (FLPA) and extended FLPA, unless requested otherwise on the FIX system parameter • the modified link pack area (MLPA) and extended MLPA, unless requested otherwise on the MLPA system parameter	IARUF	IARUFP
9 The RSM RIM then calls the RSM service routine, IARUINV, to purge the translation lookaside buffers (TLBs). The RIM returns control to IEAVNIPM.	IARUI	IARUINV

Diagram 38. System Resources Manager Initialization (IEAVNP10) Part 1 of 12

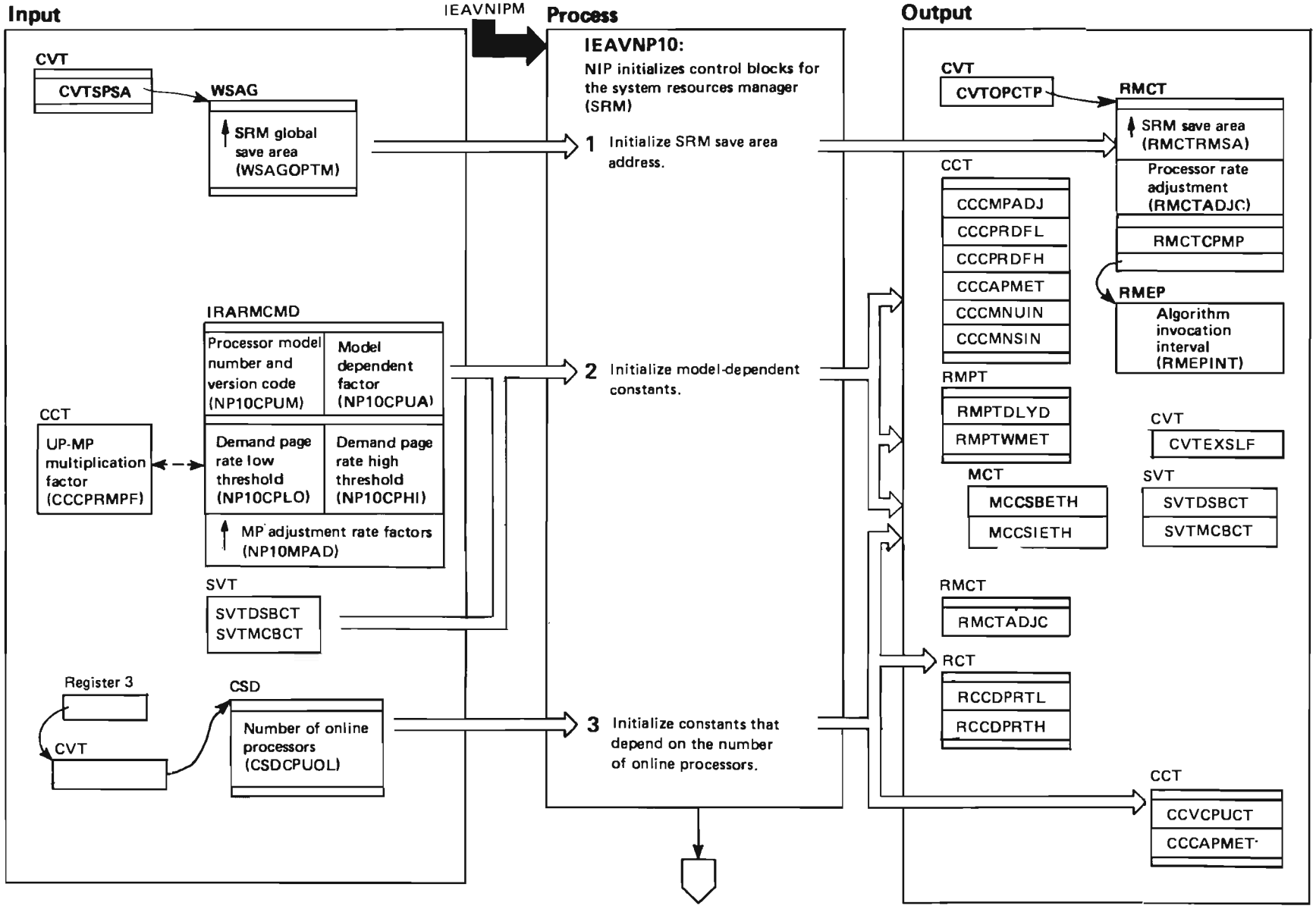


Diagram 38. System Resources Manager Initialization (IEAVNP10) Part 2 of 12

Extended Description	Module	Label
The system resources manager (SRM) supervises system resource usage by keeping those address spaces in real storage that best use system resources and meet the installation's performance specifications (IPS), at any time and under any workload. Other system components communicate with the SRM by means of the SYSEVENT macro instruction. The issuing of a SYSEVENT prior to SRM RIM processing results in a direct return to the issuer without any SRM processing. The SRM RIM initializes the system resource manager as follows:		
1 The SRM RIM obtains the address of the SRM global save area from field WSAGOPTM in the global work/save area vector table (WSAG). The SRM RIM initializes field RMCTRMSA of the SRM control table (RMCT) with the address of the SRM global save area. For a description of the RMCT, refer to *Data Areas*.	IEAVNP10	
2 Several SRM constants related to processor execution time depend on the processor model number and version code. The SRM RIM obtains the processor model number and version code of the IPL processor by issuing the store CPU ID instruction and uses them to find the model-dependent factor in the processor model table (IRARMCMD). The RIM saves the model-dependent factor in the field RMCTCPMP.	IEAVNP10	
The processor model table entry points to an array of multiprocessing adjustment rate factors. The SRM RIM copies the factors into the CCT (CCCMPADJ) and copies the demand paging rate default thresholds for MPL and adjustment from the model table entry to the CCT (CCCPRDFL and CCCPRDFH).	IRARMCPU	IRARMCMD
The SRM RIM uses the model dependent factor saved in RMCTCPMP to compute the following: • The number of instructions executed before a CPU-spin loop time out condition should be detected. The RIM saves this number in CVTEXSLF. • The number of address spaces the dispatcher will be able to process in one second, which is when a timeout condition should occur. • The number of address spaces that memory switch and chap will be able to process in one second, which is when a timeout should occur. • The processor rate adjustment value (field RMCTADJC) in the RMCT.		

Extended Description	Module	Label
• The processor control time interval (fields CCCAPMET, CCCMNUIN, and CCCMNSIN in the processor control table (CCT). • The execution time thresholds (fields MCCSBETH and MCCSIETH) in the MCT. • The algorithm invocation interval (RMEPINT) in the SRM algorithm entry point blocks (RMEP) for the following routines: IRARMCL1, IRARMEQ1, IRARMCAP, IRARMIL1, IRARMAP1, IRARMASM, IRARMMS6, IRARMWM2, IRARMRM2. • The delay interval threshold (RMPTDLYD) in the RMPT. The workload manager evaluation interval threshold (field RMPTWMET) in the RMPT. For a description of the MCT, RMEP, RMPT, and CCT, refer to *OS/VS2 Data Areas*.		
3 Several SRM constants depend on the number of online processors. The SRM RIM obtains the number of online processors from field CSDCPUOL and places the value in field CCVCPUCT of the CCT. The SRM RIM uses the number of online processors and the multiprocessing adjustment rate factors to adjust the following: • The processor rate adjustment factor (RMCTADJC) • The processor control timer interval (CCCAPMET) in the CCT • The execution time thresholds (MCCSBETH and MCCSIETH) in the MCT • The algorithm invocation interval (RMEPINT) in the RMEP for routine IRARMRM2. The RIM also initializes the following fields in the SRM resource control table (RCT): the demand page rate low threshold (RCCDPRTL) and the demand page rate high threshold (RCCDPRTH). For a description of the RCT, see *Data Areas*.		

Diagram 38. System Resources Manager Initialization (IEAVNP10) Part 3 of 12

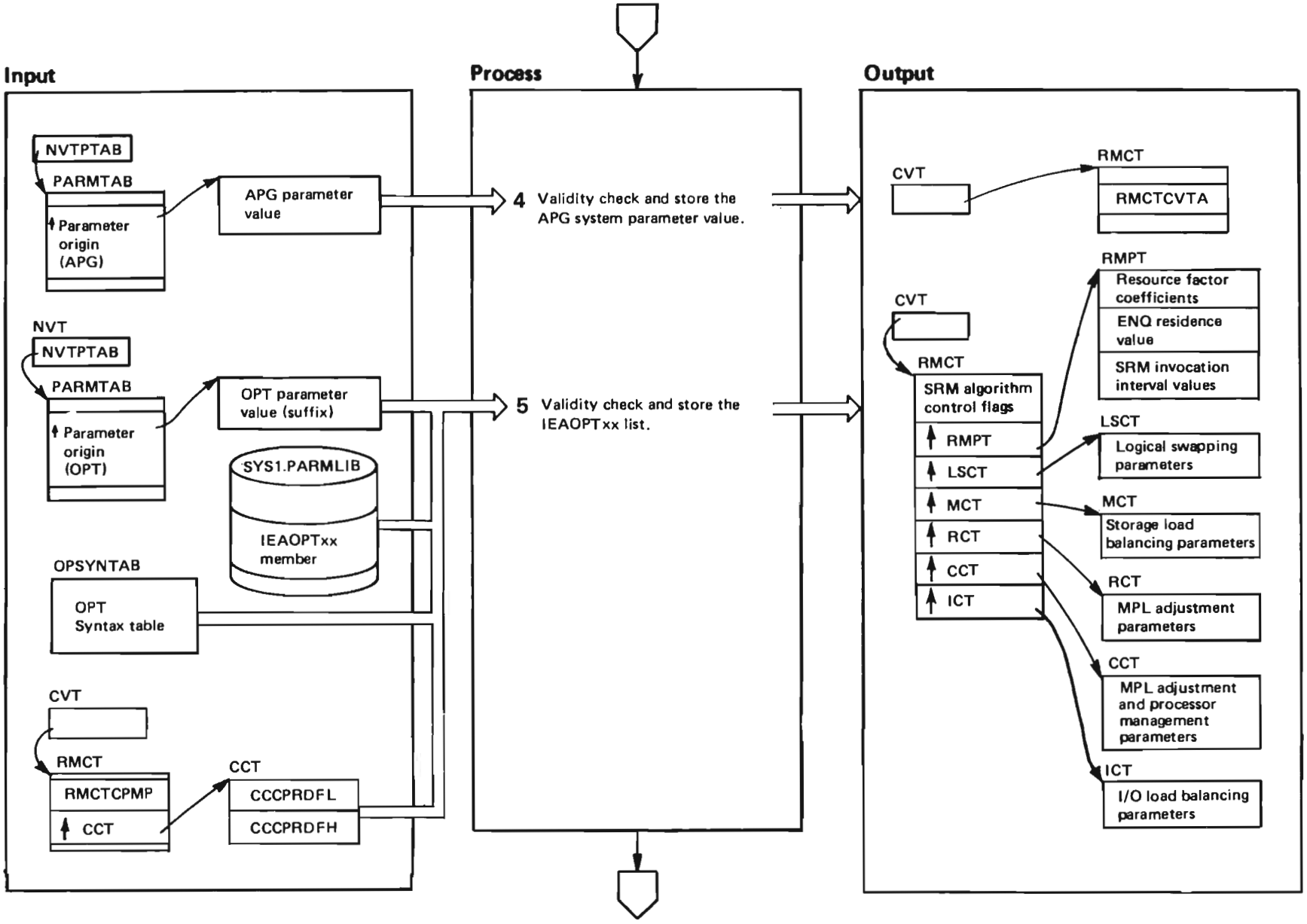


Diagram 38. System Resources Manager Initialization (IEAVNP10) Part 4 of 12

Extended Description	Module	Label
4 The APG system parameter specifies an automatic priority group for use by the SRM. The SRM RIM finds the APG value using PARMTAB. If no value was specified (that is, the address of the APG value is zero), the SRM RIM uses a value of six. This APG value is then multiplied by 16 and saved in RMCTCVTA.	IEAVNP10	NP10APGP
5 The installation can supply tuning parameters to the SRM. These parameters are supplied in a member of SYS1.PARMLIB named IEAOPTxx. The installation uses the OPT system parameter to indicate the suffix for IEAOPTxx. The SRM RIM obtains the OPT-supplied suffix using PARMTAB. If no suffix was specified, the SRM RIM uses default values for the OPT parameters. If a suffix was specified, the RIM checks it for validity.	IEAVNP10	NP10OPTP
Next the RIM checks the data in IEAOPTxx for validity using the OPT syntax table (OPSYNTAB) and stores the data temporarily in the OPT parameter list (OLST).	IRARMOPT IRARMANL IRARMOPT	IRARMOPT RMOPINIT
The SRM RIM must adjust several of these parameters before placing them in the OLST: • The resource factor coefficients are scaled to match the SRM's recommendations. • The enqueue residence value and the CPU significant-user threshold depend on the processor model and the number of processors online. Therefore, the SRM RIM must adjust them based on the model-dependent factor saved in the field RMCTCPMP, the number of processors (CCVCPUCT), and the multiprocessing adjustment factor (CCCMPADJ). • The SRM invocation interval depends on the processor number; therefore, it is adjusted based on RMCTCPMP. • The UP or MP paging rate thresholds must be specified in the OPT parmlib member or the SRM RIM takes the default values from CCCPRDFL and CCCPRDFH.	IRARMOPT	
These values are then copied from the OPT parameter list into SRM control blocks RMCT, RMPT, LSCT, MCT, RCT, CCT, and ICT. RMPTTOL and MCCSBSCF are based on the new values of RMPTTOM and MCCSBSTL respectively. For a more detailed description of IRARMOPT see *System Logic Library.*	IEAVNP10	NP10OPTP

Diagram 38. System Resources Manager Initialization (IEAVNP10) Part 5 of 12

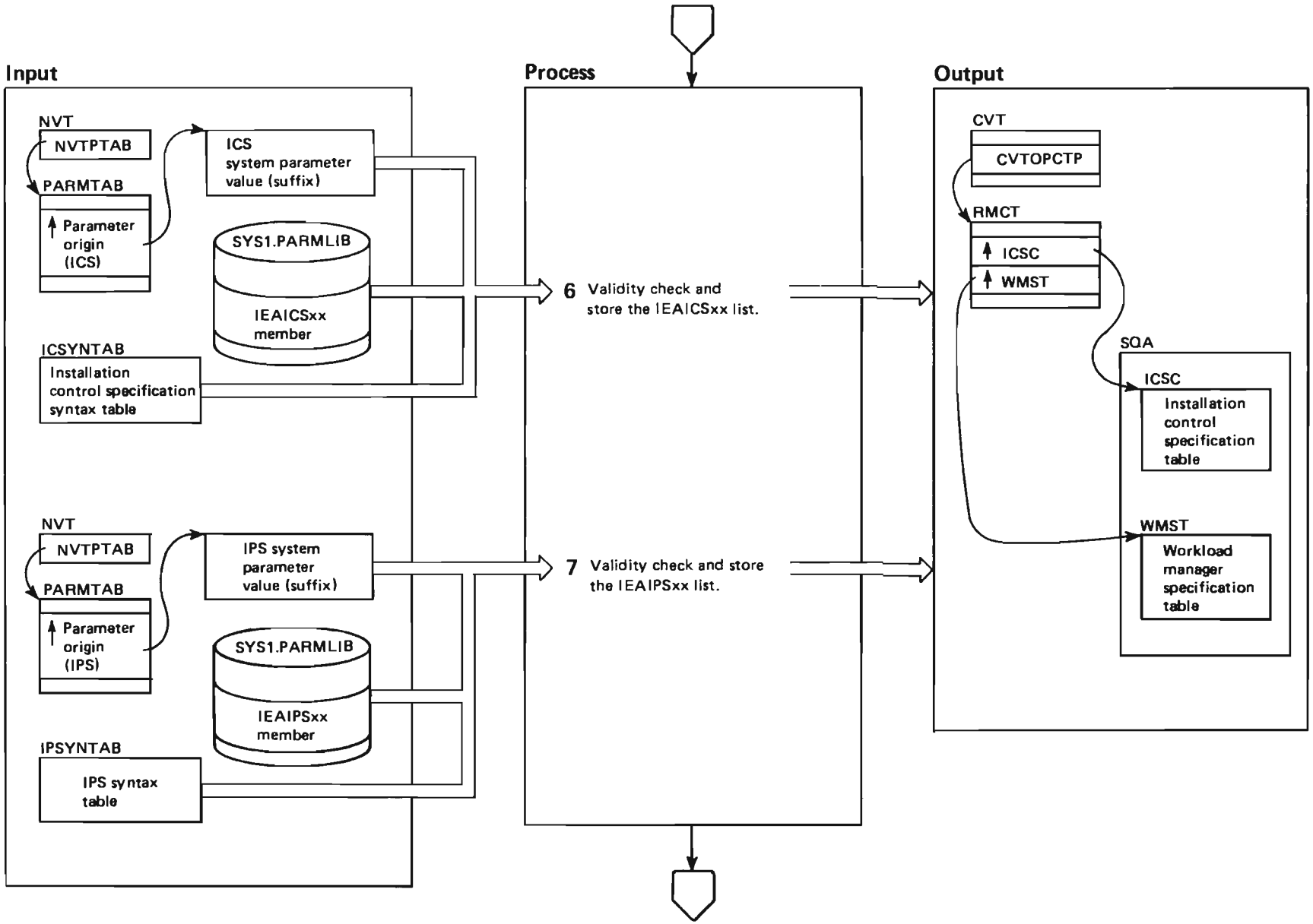


Diagram 38. System Resources Manager Initialization (IEAVNP10) Part 6 of 12

Extended Description	Module	Label
6 The installation can provide SRM with the information needed to assign a performance group number (PGN) to a user based on the user ID, transaction name, or transaction class. Also, a separate report performance group number (RPGN) can be assigned so that user statistics can be collected independently at the same time as workload manager control. The information provided by the installation is called the installation control specification and is placed in a member of SYS1.PARMLIB named IEAICSxx. The ICS system parameter is used to indicate the suffix for IEAICSxx.		
The SRM RIM obtains the ICS system parameter value (the suffix) using PARMTAB. If no value has been specified, then SRM assumes that there is no installation control specification in effect. If a value was specified, the RIM checks it for validity, then stores the IEAICSxx member name in field NIPPANAM of PARMAREA. The RIM then checks the IEAICSxx list for validity using the installation control specification syntax table (ICSYNTAB), and builds the installation control specification table (ICSC) in SQA. The address of the ICSC is stored in RMCTICST.	IEAVNP10	IEAVICSP
	IRARMICS	IRARMICS
	IRARMANL	
	IRARMICS	RMICINIT
For a more detailed description of IRARMICS, see the system resource manager section of *System Logic Library*.		

Extended Description	Module	Label
7 The installation specifies to the SRM's workload manager an installation performance specification (IPS) consisting of: 1) domain service rates, and 2) priorities to be associated with users during thair processing. The installation places its IPS in a member of SYS1.PARMLIB named IEAIPSxx and uses the IPS system parameter to indicate the suffix for IEAIPSxx.		
The SRM RIM obtains the IPS parameter value (the suffix) using PARMTAB. If no value was specified, the RIM assumes that the default IPS in member IEAIPS00 is to be used. If a value was specified, the RIM checks it for validity, then stores the IEAIPSxx member name in field NIPPANAM of PARMAREA. The RIM checks the IEAIPSxx list for validity using the IPS syntax table (IPSYNTAB) and stores the IPS data in the workload manager specification table (WMST), which the RIM constructs in the SQA.	IEAVNP10	NP10IPSP
	IRARMIPS	IRARMIPS
	IRARMANL	
	IRARMIPS	RMIPINIT
For a description of the workload manager specification table, refer to *Data Areas*.		
For a more detailed diagram of IRARMIPS, see *System Logic Library*.		

Diagram 38. System Resources Manager Initialization (IEAVNP10) Part 7 of 12

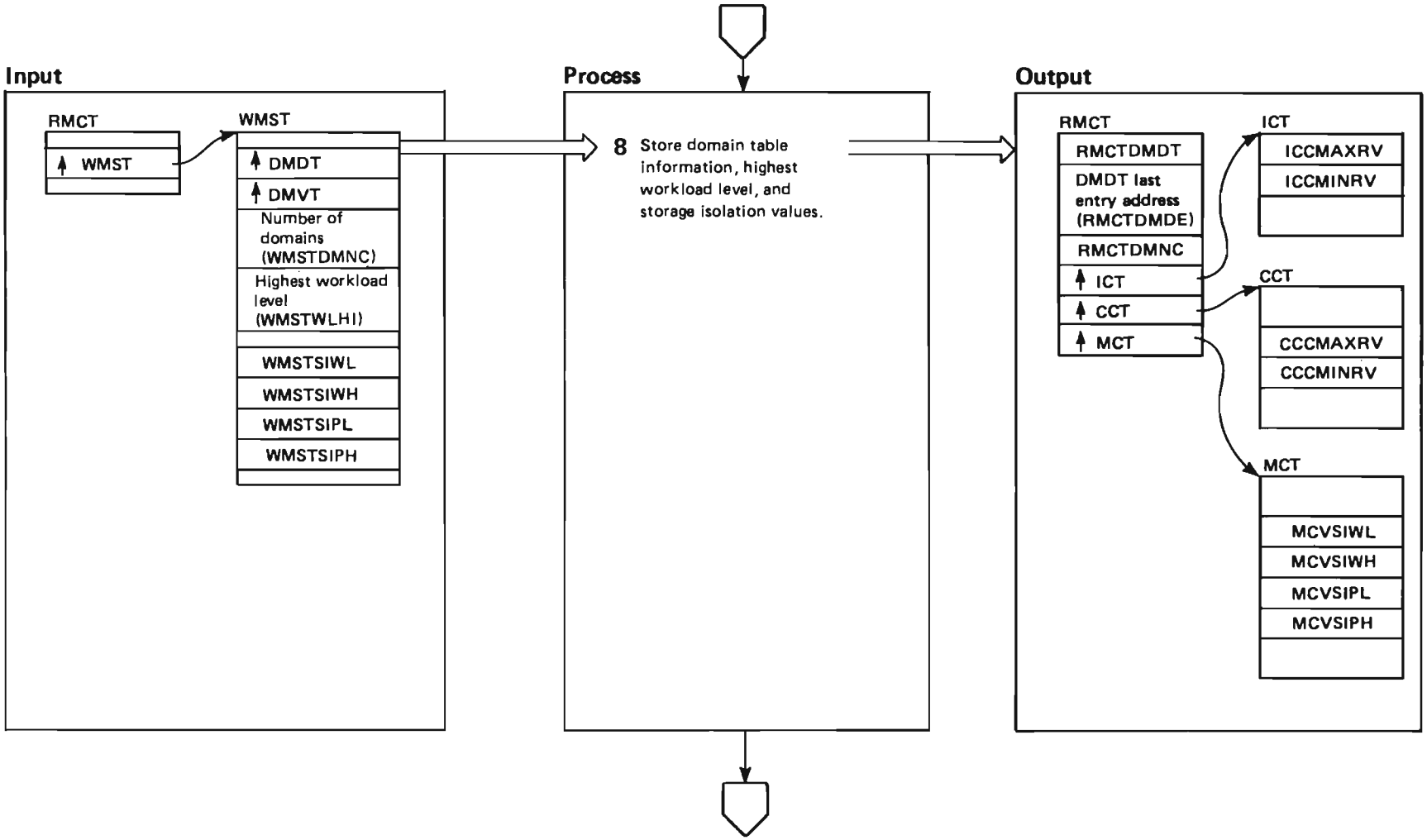


Diagram 38. System Resources Manager Initialization (IEAVNP10) Part 8 of 12

Extended Description	Module	Label
8 The domain specification in the IPS enables the installation to discriminate between aggregates of users competing for resources. The SRM RIM extracts domain specification information from the WMST and stores it in the RMCT. Also, the positive and negative highest workload level in the WMST is stored in the ICT and CCT. Finally, the common storage isolation values are copied from the WMST to the MCT.	IEAVNP10	NP10IPSP

Diagram 38. System Resources Manager Initialization (IEAVNP10) Part 9 of 12

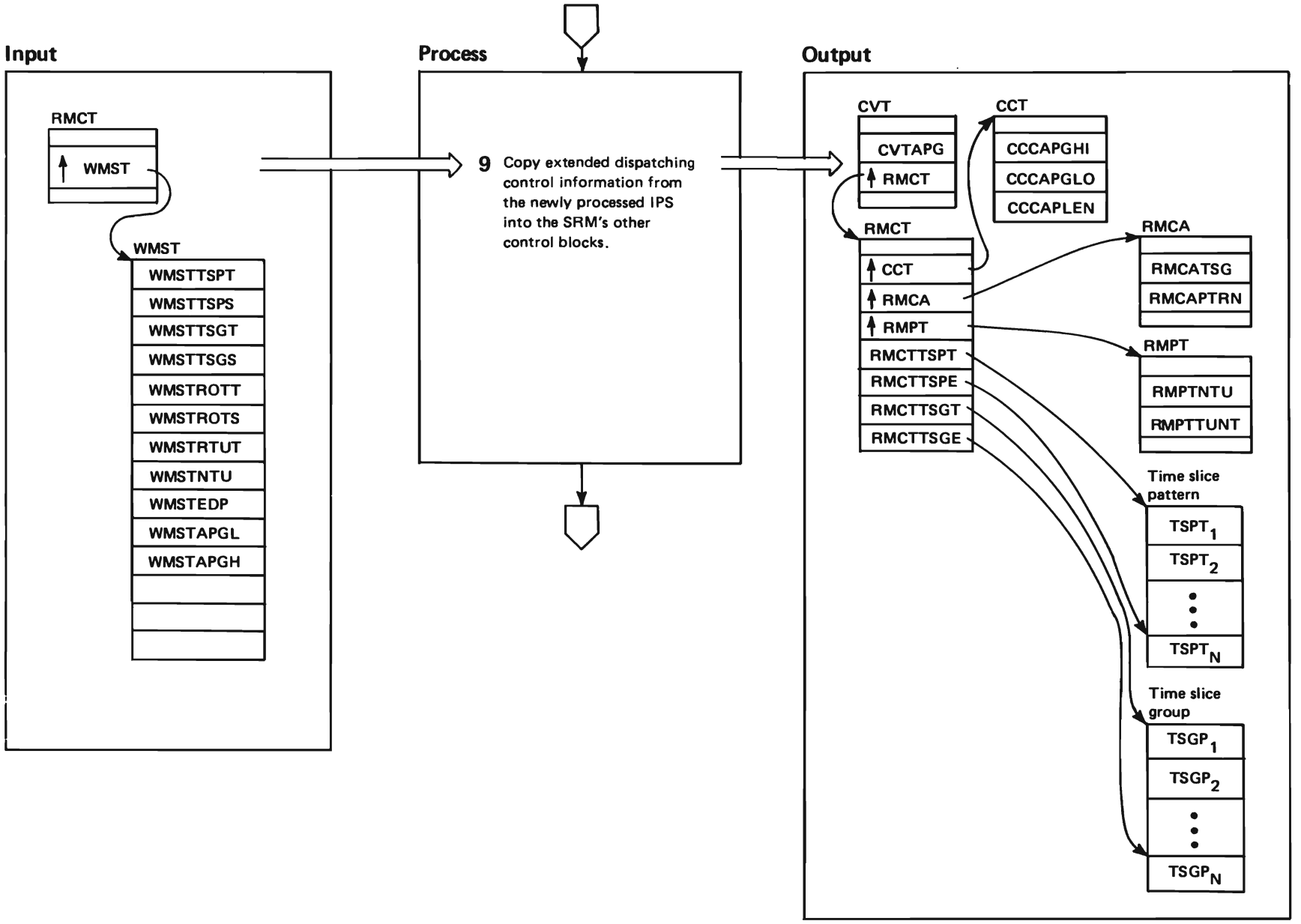


Diagram 38. System Resources Manager Initialization (IEAVNP10) Part 10 of 12

Extended Description	Module	Label
9 The SRM RIM extracts extended dispatching control information from the WMST and stores it in the RMCT, CCT, RMPT, CVT, and RMCA.	IEAVNP10	NP10IPSP

Diagram 38. System Resources Manager Initialization (IEAVNP10) Part 11 of 12

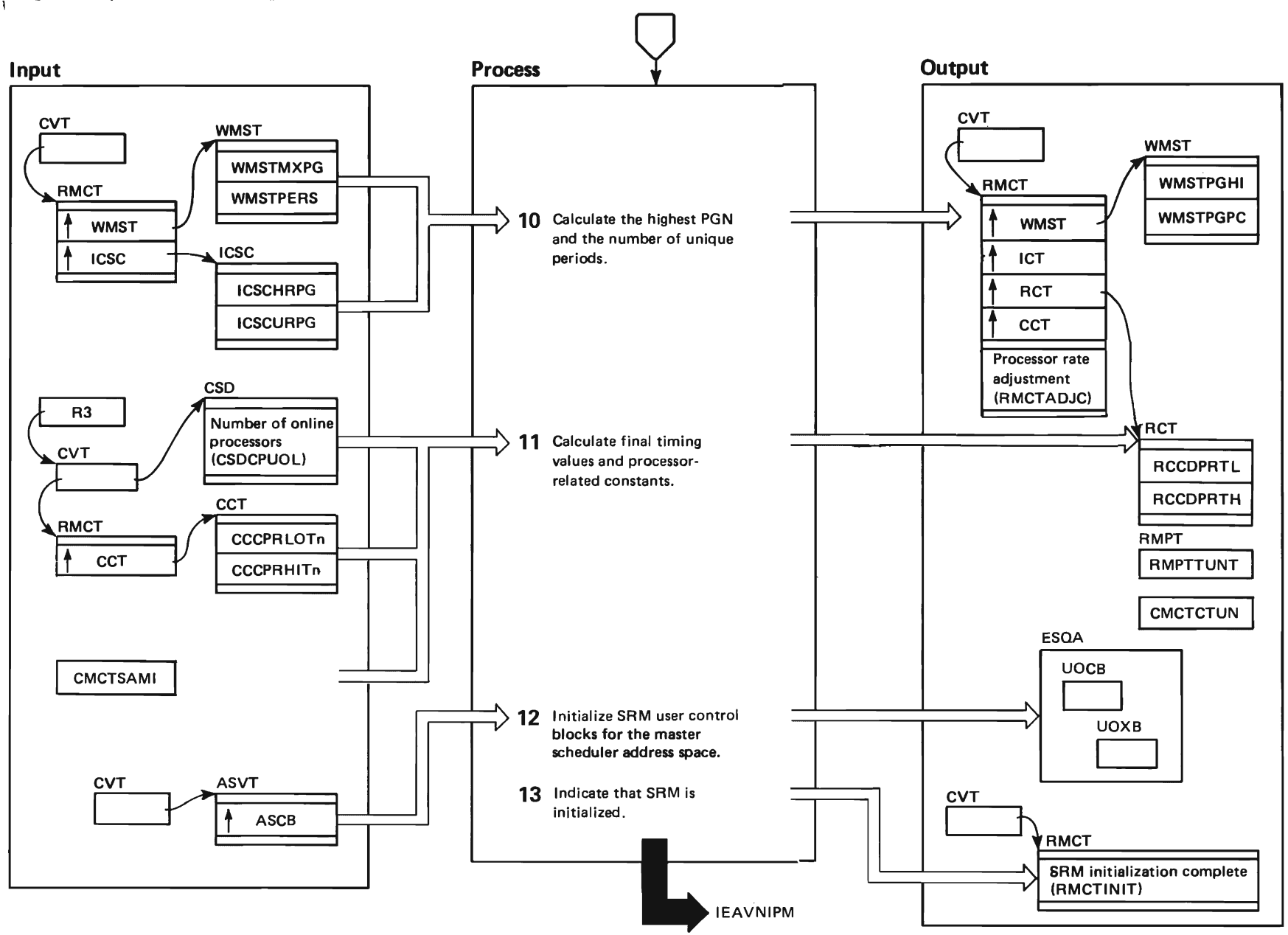


Diagram 38. System Resources Manager Initialization (IEAVNP10) Part 12 of 12

Extended Description	Module	Label
10 The limits of the IPS table are calculated. These will be used later by the measurement facility to determine the size of the table needed for workload data collection. The highest performance group number specified in either the IPS or the installation control specification is the larger of WMSTMXPG or ICSCHRPG. This value is placed in WMSTPGHI. The number of unique periods in the combined IPS and installation control specification is the number of periods in the IPS (WMSTPERS) plus the number of report performance groups in the installation control specification (ICSCURPG) minus the number of report performance groups in the installation control specification that are also defined as performance groups in the IPS. The result of this calculation is stored in WMSTPGPC.	IEAVNP10	
11 After the SRM RIM has processed the parameters in the IPS and the OPT that control the SRM timer entry frequency, the RIM completes the calculation and setting of the processor model dependent values. The RIM calculates the amount of time in a timer unit and saves this value in RMPTTUNT. It determines the frequency of sampling for channel measurements based on the target time between samples (CMCTSAMI) and the time in a timer unit. It saves this value in CMCTCTUN. Because the interval between channel measurement samples can be less than the length of a timer unit, the RIM saves the frequency as the number of timer units times a scaling factor of 16. (For example, a value of 8 in CMCTCTUN means that there is 8/16ths or one half of a timer unit is to elapse between channel samples.) The RIM saves the demand paging rate thresholds for MPL adjustment in RCT fields RCCDPRTL and RCCDPRTH. The RIM finds these values in CCCPRLOTn and CCCPRHITn, where n=1 or 2 depending on the number of processors online.	IEAVNP10	
12 SRM user control blocks are created for each address space that is created. Because NIP creates the address space for the master scheduler, the SRM RIM must initialize the user control block (UOCB) and a user extension block (UOXB) for the master scheduler address space. For a description of the UOCB and UOXB, refer to *Data Areas*.	IEAVNP10	
13 The SRM RIM indicates that it has completed SRM initialization by setting flag RMCTINIT in the RMCT.	IEAVNP10	

Diagram 39. System Resources Manager Channel Measurement Initialization (IEAVNP1F) Part 1 of 6

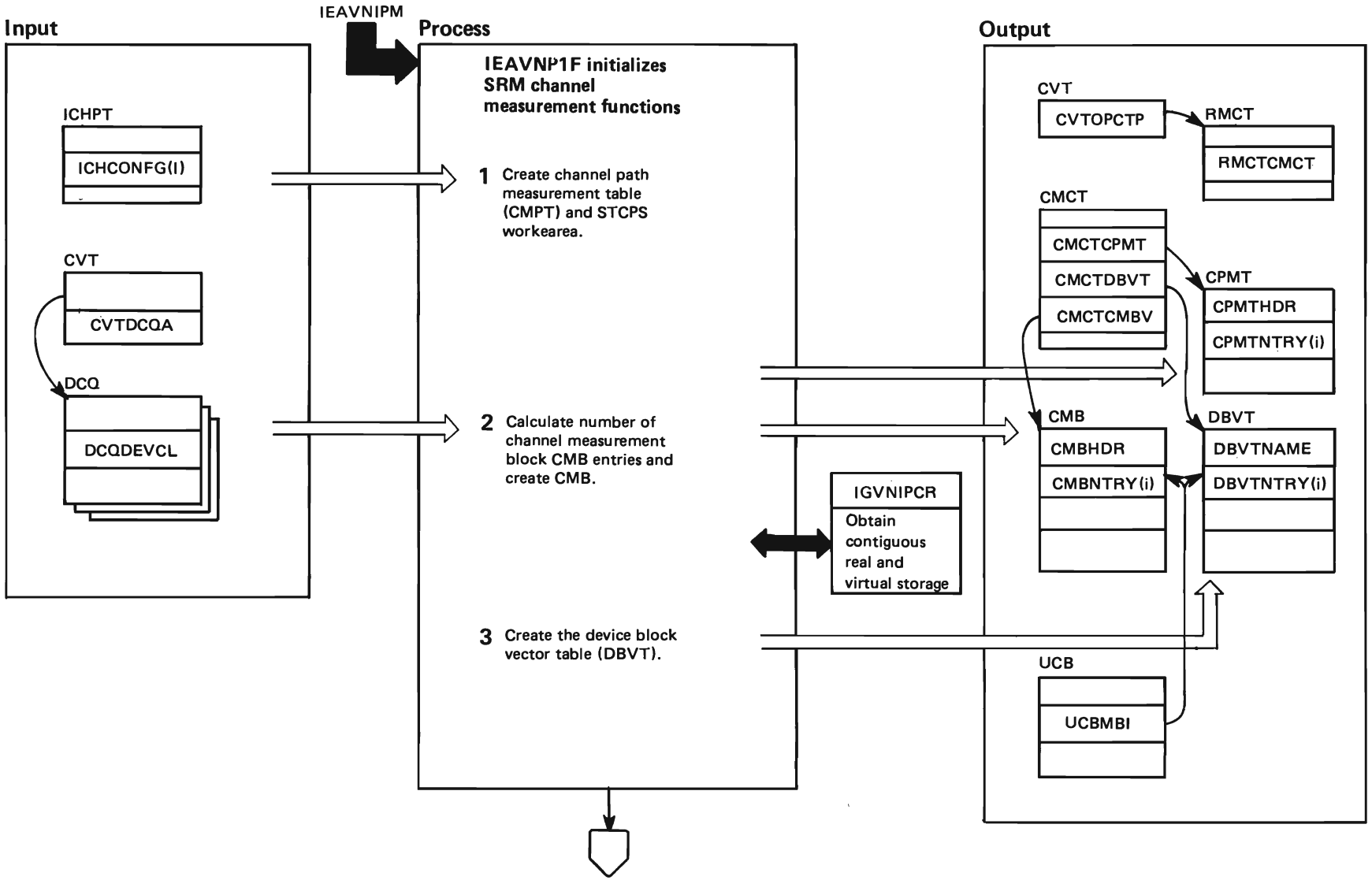


Diagram 39. System Resources Manager Channel Measurement Initialization (IEAVNP1F) Part 2 of 6

Extended Description	Module	Label
The system resource manager (SRM) RIM, IEAVNP1F, builds and maintains five types of tables associated with channel measurements. The tables are: • The channel path measurement table (CPMT), which describes the activity on the channel paths. It has one entry for each channel path installed, regardless of whether the channel path is active at this installation. • The channel measurement block (CMB), which contains subchannel counts and timing data on a device basis. • The logical path control block (LPB), which contains data about the logical paths. A logical path represents a set of physical paths to a device or a group of devices. • The device measurement block (DMB), which contains the device statistics that SRM maintains for device allocation and I/O load balancing. For device allocation, there is one entry for each online device for which there has been a previous request. For I/O load balancing, there is one entry for each online tape device and DASD. • The device block vector table (DBVT), which contains pointers to the LPB table (LPBT) entries and the DMBs. These SRM channel measurement functions are activated in module IRARMCHM as a result of a request by periodic entry point scheduling. The SRM RIM also initializes the CMB system parameter. The CMB (channel measurement block) system parameter allows an installation to designate the types of I/O devices (in addition to tape devices and DASDs) for which measurement data might be collected. The SRM uses this information to calculate the required size for the CMB.		
1 The SRM RIM indicates the channel measurement control block structure is valid by setting bits (CMCTMFOK, CMCTDCOK, CMCTPAOK, and CMCTCMOK) on. The RIM obtains storage for the CPMT, with an entry for each channel path to the highest installed channel path ID, and the STCPS workarea. If it cannot obtain storage, the RIM turns the CMCTPAOK and CMCTMFOK bits off and continues processing with step 6.	IEAVNP1F	
2 The SRM RIM calculates the number of CMB entries to include one for each tape device and DASD that was initialized or defined during the system generation process. It analyzes the installation-defined CMB keyword parameters to determine if the installation has requested additional entries for the channel-to-channel (CTC) adapters, communication devices, graphics devices, and/or character readers or requested a specific number of entries. The SRM RIM then obtains storage for the required CMBs. Because the channel stores into the CMB without dynamic address translation, the virtual pages must reside in contiguous real storage. The SRM RIM calls module IGVNIPCR to obtain contiguous virtual storage, and IGVNIPCR calls IARXTIGU to obtain the contiguous real storage for the CMB. If IGVNIPCR is unable to obtain storage for the tape device and DASD CMB slots, the RIM turns CMCTMFOK off and continues processing with step 6.	ANLCMB IGVNIPCR IARXT	 IARXTIGU
3 The SRM RIM obtains storage for the DBVT. There is one DBVT entry for every CMB entry. If it cannot obtain storage, the SRM RIM turns CMCTMFOK off and continues processing with step 6. The UCB field UCBMBI provides a common index for the corresponding entries in these two control blocks.	IEAVNP1F	

Diagram 39. System Resources Manager Channel Measurement Initialization (IEAVNP1F) Part 3 of 6

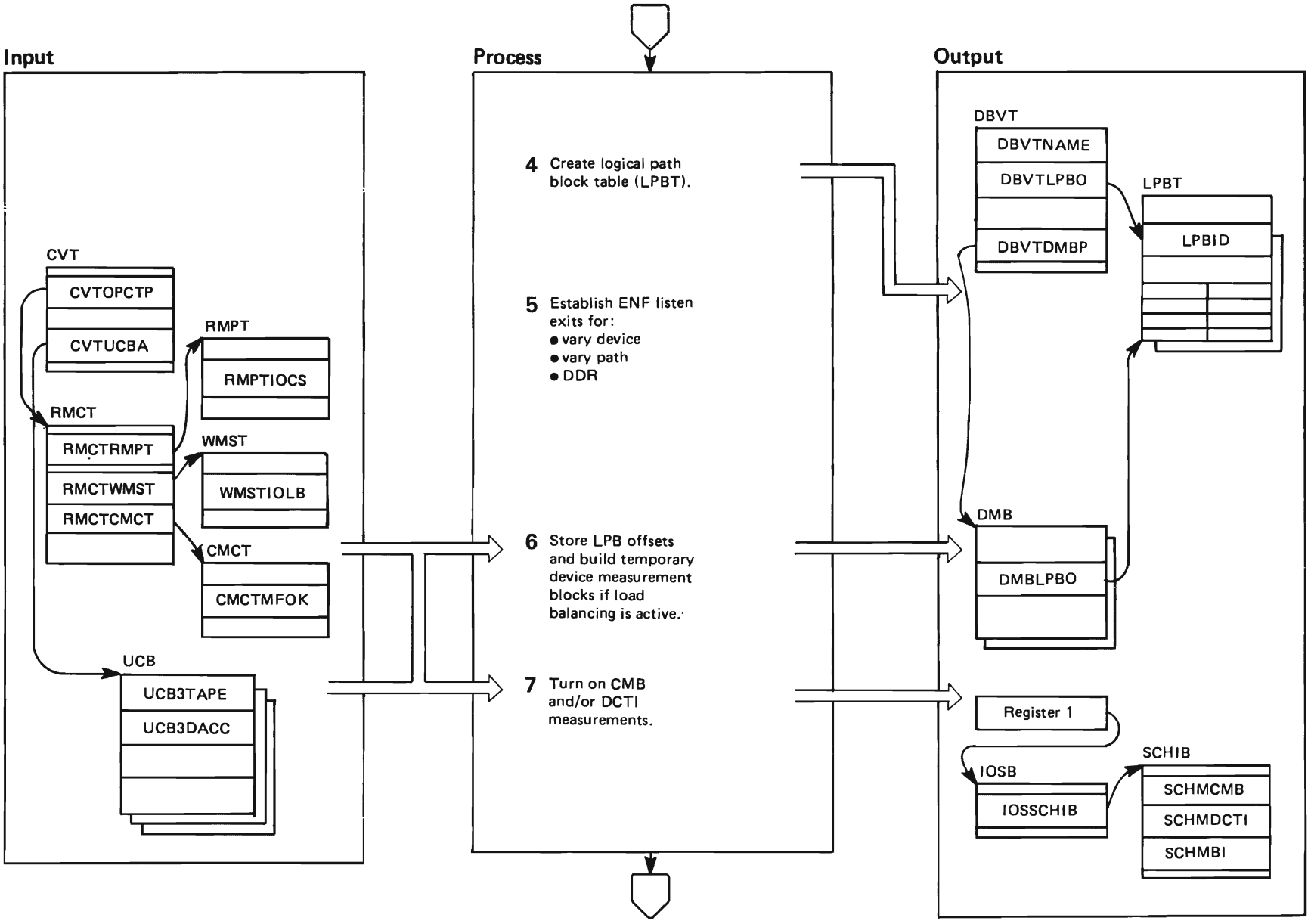


Diagram 39. System Resources Manager Channel Measurement Initialization (IEAVNP1F) Part 4 of 6

Extended Description	Module	Label
4 The SRM RIM obtains one page of storage for the LPBT. If it cannot obtain storage, the RIM turns CMCTMFOK off and continues processing with step 6.		
5 The SRM RIM establishes ENF exits to listen for three events: vary device online or offline, vary path, and DDR.		
6 The SRM RIM scans the UCB chain. If the UCB is for a tape device or DASD and CMCTMFOK is on, the RIM associates the device with an existing LPB table entry or obtains storage for another entry in the LPBT, builds the entry, and assigns the device class index. If the LPBT is too small to accommodate the new entry, the RIM gets a larger LPBT, copies the old LPBT to the new one, and frees the old one. If the RIM cannot obtain storage for expansion of the LBPT, it turns CMCTMFOK off. If load balancing is active, (that is, WMSTIOLB=1 and RMPTIOCS¬=0), the RIM builds a temporary DMB for each tape device and DASD that is online. If the RIM cannot get storage for the DMBs, it turns CMCTMFOK off.		LPBINDEX DMBBUILD
7 The RIM obtains storage for an SRB, IOSB, and subchannel information block (SCHIB). (The SRB is scheduled whenever a system routine requests asynchronous recording.) If CMCTMFOK is on and the device is a tape device (UCB3TAPE=1) or a DASD (UCB3DACC=1), the RIM turns both CMB and device connect time interval (DCTI) measurements on and stores the MBI index in the UCB. Otherwise, it turns on only the DCTI measurements. The RIM issues a synchronous modify subchannel request for the device. If the device is not available (that is, there is a non-zero return code from the modify subchannel request, MSCH), the RIM queues an asynchronous request and obtains storage for the next modify subchannel request. If the request for storage is unsuccessful, the RIM turns CMCTMFOK and CMCTDCOK off. If the device is boxed, in error, or not connected, IOS will store the information in the UCB for later placement in the subchannel.		IOCMSCQE IEAVNP1F

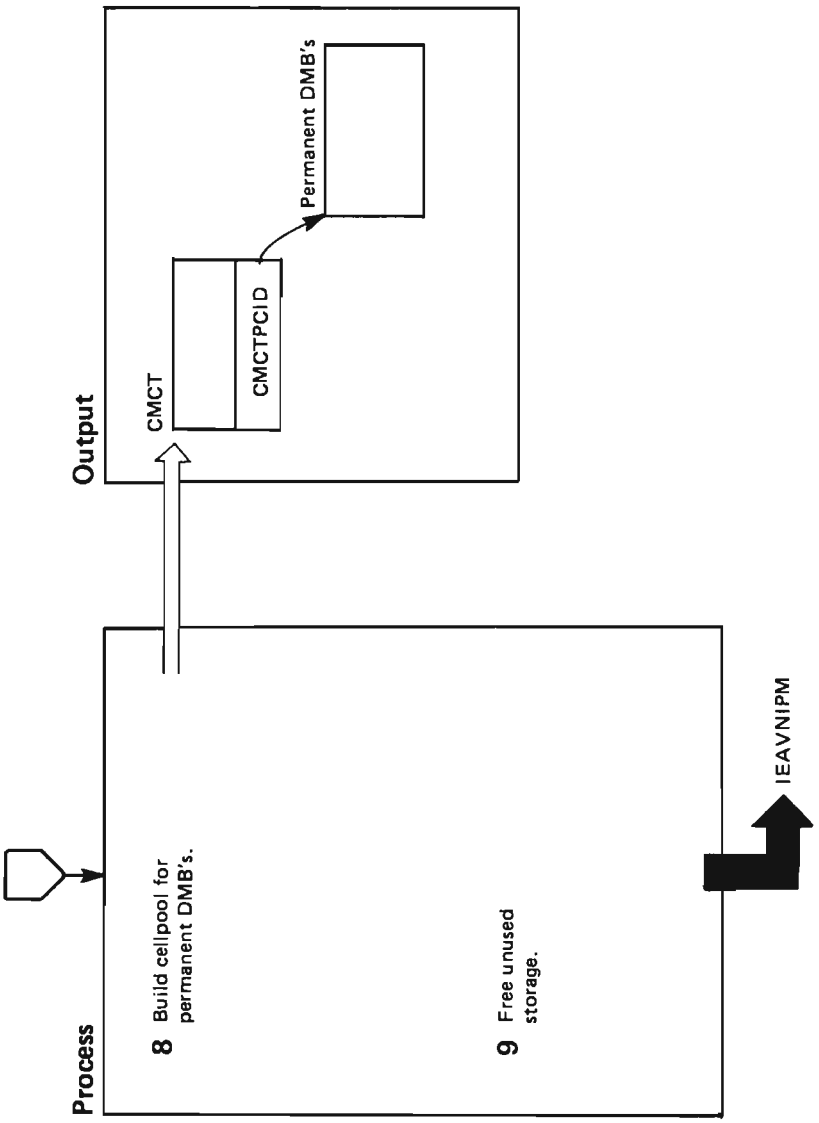


Diagram 39. System Resources Manager Channel Measurement Initialization (IEAVNP1F) |

Diagram 39. System Resources Manager Channel Measurement Initialization (IEAVNP1F) Part 6 of 6

Extended Description	Module	Label
8 The RIM frees storage associated with the modify sub-channel request and the unused portion of the LPBT. If CMCTMFOK is on, the RIM obtains storage for permanent DMBs.		
9 If CMCTMFOK is off, the RIM frees the storage for the DBVT, LPBT, and DMB control blocks and issues message IEA966I. If CMCTDCOK is off, the RIM issues message IEA991I.		
IEAVNP1F returns control to IEAVNIPM.		

Diagram 40. ABDUMP Initialization (IEAVNPD1) Part 1 of 2

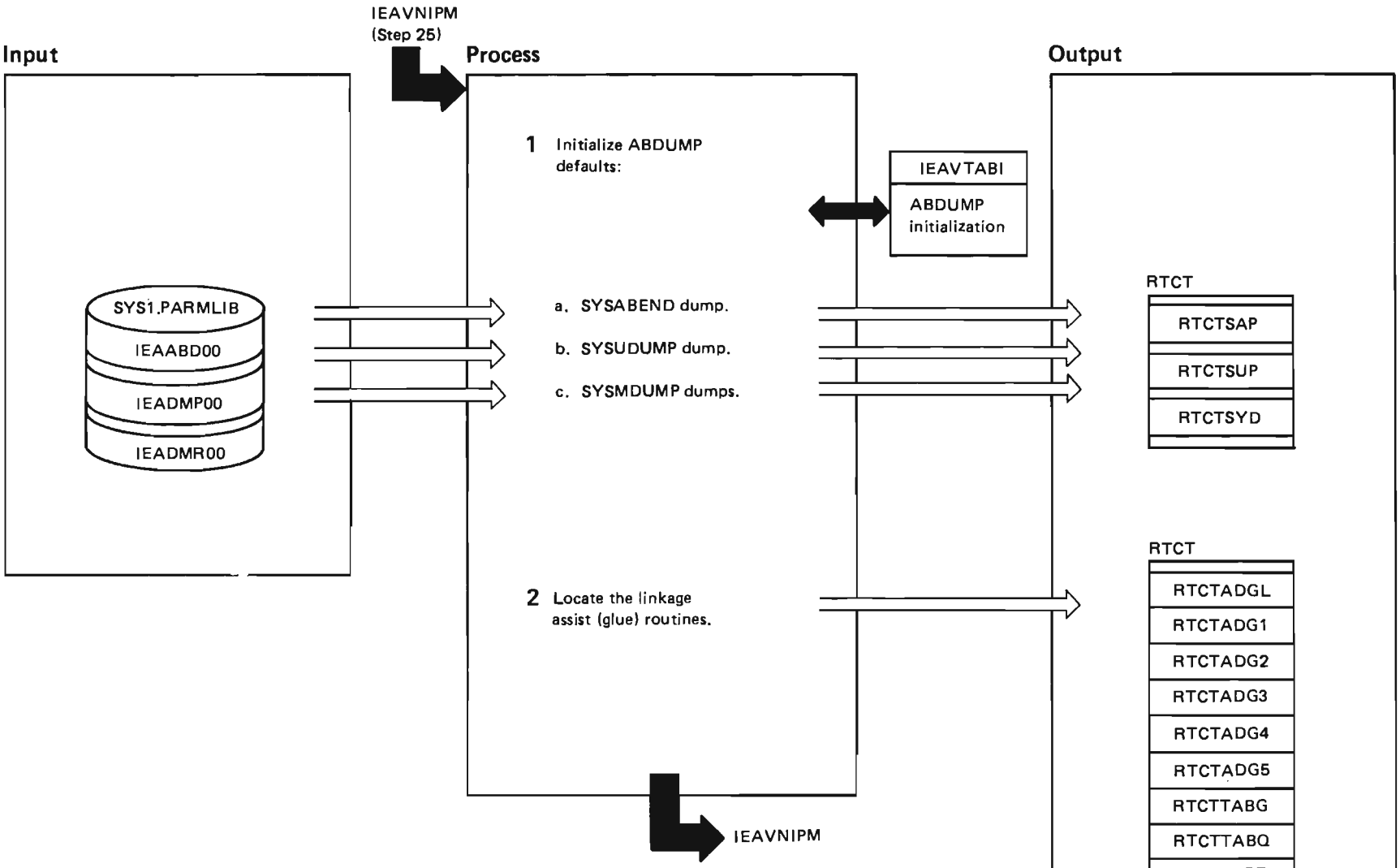


Diagram 40. ABDUMP Initialization (IEAVNPD1) Part 2 of 2

Extended Description	Module	Label
1 The installation can replace or modify IBM-supplied defaults for ABDUMP dumps by placing the default values in members of SYS1.PARMLIB. The ABDUMP RIM, IEAVTABI initializes ABDUMP control blocks and work areas.		
a. The installation specifies defaults for SYSABEND dumps by placing default values in IEAABD00. The ABDUMP RIM searches SYS1.PARMLIB for IEAABD00, scans the default values, then places the values in field RTCTSAP of the RTCT. If the operator does not use the CHNGDUMP operator command to specify other options (via the add or override option), ABDUMP will use these default options and merge them with other dump options specified (via the CALLRTM, SETRP, or ABEND macro instructions) before the dump occurs.	IEAVTABI	
b. The installation specifies defaults for SYSUDUMP dumps by placing default values in IEADMP00. The ABDUMP RIM searches SYS1.PARMLIB for IEADMP00, scans the default values, then places the values in field RTCTSUP of the RTCT. If the operator does not use the CHNGDUMP operator command to specify overrides or adds, ABDUMP will use these default options and merge them with other dump options specified (via the CALLRTM, SETRP, or ABEND macro instructions) before the dump occurs.		
c. The installation specifies defaults for SYSMDUMP dumps by placing default values in IEADMR00. The ABDUMP RIM searches SYS1.PARMLIB for IEADMR00, scans the default values, then places the values in field RTCTSYD of the RTCT. If the operator does not use the CHNGDUMP operator command to specify overrides or adds, ABDUMP will use these default options and merge them with other dump options specified (via the CALLRTM, SETRP, or ABEND macro instructions) before the dump occurs.		
2 The ABDUMP RIM issues a LOAD macro instruction to locate the linkage-assist (glue) routines that are needed for SNAP/ABDUMP processing. These routines are: IEAVADGL, IEAVADG1, IEAVADG2, IEAVADG3, IEAVADG4 and IEAVAD12 (which are aliases of IGC0805A) and IEAVTABG, IEAVTABQ and IEAVTABR (which are aliases of IEAVTRG2).		
The RTM RIM returns control to IEAVNIPM.		

Diagram 41. SVC Dump Initialization (IEAVNPDI) Part 1 of 12

IEAVTSDI - MODULE DESCRIPTION

DESCRIPTIVE NAME: SDUMP INITIALIZATION RIM

FUNCTION:
THE SVC DUMP NIP RIM INITIALIZES THE SVC DUMP DATA AREAS.

ENTRY POINT: IEAVTSDI

PURPOSE: INITIALIZE SVC DUMP DATA AREAS.

LINKAGE: PLS CALL STATEMENT

CALLERS: None

INPUT: None

OUTPUT:
CONTROL BLOCKS ARE GETMAINED AND INITIALIZED, AND SVC DUMP MODULES ARE LOADED.

EXIT NORMAL: RETURNS TO CALLER

EXTERNAL REFERENCES:

ROUTINES: NIPWTO - WRITE MESSAGES DURING NIP

CONTROL BLOCKS:

COMMON NAME	LIBRARY NAME	ACCESS
ASCB	IHAASCB	READ
CVT	CVT	READ
DFL	ADYDFLM	CREATE,READ,WRITE
DSCA	ADYDSCA	CREATE,READ,WRITE
DSPD	ADYDSPD	CREATE
DSTAT	ADYDSTAT	CREATE
DSVCB	IHADSVCB	CREATE,READ,WRITE
ERRWORK	IHASDERR	CREATE,READ
GDA	IHAGDA	READ
LDA	IHALDA	READ
NNTOHDR	IEAPMNIP	READ
NIPPAHDR	IEAPPNIP	READ
NVT	IHANVT	READ
PSA	IHAPSA	READ
RTCT	IHARTCT	READ,WRITE
RTSD	IHARTSD	CREATE,READ,WRITE
SDCON	IHASDCON	CREATE,READ,WRITE
SDDAT	IHASDDAT	CREATE,READ,WRITE
SDDIE	IHASDDIE	CREATE,WRITE
SDEXPARM	IHASDEXP	CREATE,WRITE
SDMOD	IHASDMOD	CREATE,READ,WRITE
SDTAB	IHASDTAB	CREATE,READ,WRITE
SDUMP	IHASDUMP	READ
SDWA	IHASDWA	READ
SDWORK	IHASDWRK	CREATE,READ,WRITE
SMDLR	IHASMDLR	REFERENCED BY SMWK ONLY
SMWK	IHASMWK	CREATE,READ,WRITE
SMWKRSCB	IHASDRSB	CREATE
SRB	IHASRB	READ
TQE	IHATQE	READ
TRCT	IHATRCT	READ
TRSN	IHATRSN	READ
VRA	IHAVRA	READ
VSMDSP	IGVVSMD	REFERENCED BY SDTAB ONLY
UCB	IEFUCBOB	REFERENCE CONSTANTS ONLY

Diagram 41. SVC Dump Initialization (IEAVNPDI) Part 2 of 12

IEAVTSDI - MODULE DESCRIPTION (Continued)

SERIALIZATION:
RTCTSDPL AND CVTSDBF(HIGH-ORDER BIT) ARE SET
TO SERIALIZE ALL OF THE SVC DUMP PROCESS.

Diagram 41. SVC Dump Initialization (IEAVNPDI) Part 3 of 12

IEAVTSDI - MODULE OPERATION

1. GETMAINS THE FOLLOWING STORAGE AREAS AND SAVES THE ADDRESSES IN THE INDICATED CONTROL BLOCKS:

CONTROL BLOCK	ADDRESS STORED	SUBPOOL	SIZE	DESCRIPTION
RANGE TABLE	SDDRNGTB	239-SQA	8K	RANGE TABLE START
	SDDRNGTE			RANGE TABLE END
SDDAT	RTCTSDAT	239-SQA	SDDATLEN	SDUMP DATA AREA
SDDAT	RTCTDATC	239-SQA	LENGTH(SDDAT)	COPY OF SDDAT
ERRWORK	SDDERWRK	239-SQA	LENGTH(ERRWORK)	ERROR WORK AREA
TRSN	SDDTRSN	239-SQA	LENGTH(TRSN)	SYSTEM TRACE SNAPTRC PARAMETER LIST
TRCT	SDDTRCT	239-SQA	LENGTH(TRCT)	SYSTEM TRACE COPYTRC PARAMETER LIST
SDMOD	RTCTSMOD	239-SQA	LENGTH(SDMOD)	SDUMP MODULE PTRS
	SDDSDMOD			PTR IN SDDAT
SDCON	RTCTSCON	239-SQA	LENGTH(SDCON)	SDUMP CONSTANTS
	SDDSDCON			PTR IN SDDAT
SDDIE	RTCTDIND	239-SQA	SDDIELEN	SDUMP DIE CONTROL BLOCK FOR SYSTEM NON-DISPATCHABILITY
SDDIE	RTCTDIRS	239-SQA	SDDIELEN	SDUMP DIE CONTROL BLOCK FOR THE REAL STORAGE BUFFER
SMWK	RTCTSDSW	239-SQA	LENGTH(SMWK)	SUMDUMP WORK AREA
	SDDSMWK			FOR SUMDUMP FORMATTER
SDWORK	RTCTSDWK	227-CSA	LENGTH(SDWORK)	SDUMP WORK AREA
	SDDSDWRK			
OUTPUT & 4K BUFFER	SDDSRBUF	231-CSA	8224	SINGLE RECORD OUTPUT BUFFERS
	CVTSDBF			SDUMP 4K BUFFER
RSB	SMWKRSM	239-SQA	LENGTH(SMWKRSCB)	SUMDUMP REAL STORAGE BUFFER CONTROL BLOCK
RTSD	RTCTRTSD	239-SQA	LENGTH(RTSD)	RECOVERY TERMINATION SDUMP EXTENSION
SDWA	SDDSDWA	239-SQA	SDWAMLEN	COPY OF SDWA RECORDABLE AREAS
SDTAB	SDDSDTAB	231-CSA	LENGTH(SDTAB)	SDUMP VSMLIST TABLE
SDEXPARM	SDDSDEXP	231-CSA	SDEXLEN	SDUMP EXIT PARAMETER LIST
DSVCB	RTCTDSV	231-CSA	DSVLEN	DUMPSRV ADDRESS SPACE CONTROL BLOCK
DSCA	RTCTDSCA	239-SQA	DSCALEN	DAE COMMUNICATION AREA
DSPD	DSCDSPDP	239-SQA	LENGTH(DSPD)	DAE PRE AND POST DUMP
	SDDDSPD			PARAMETER LIST
DFL	DSCDFLT	239-SQA	LENGTH(DFL)	DAE DEFAULT OPTIONS
HEADER	DSCHDRP	239-SQA	HDRRECLN	COPY OF DUMP HEADER RECORD
	SDDCPYHD			COPY HEADER PTR

2. DETERMINES THE ADDRESSES OF SVC DUMP LPA AND NUCLEUS MODULES AND SAVES THE ADDRESSES IN THE SVC DUMP CONTROL BLOCK SDMOD. THE SDMOD INITIALIZATION LIST (SDMILIST) CONTAINS A DESCRIPTION OF EACH MODULE PROCESSED.

3. LOADS IEAVTSXT WHICH CONTAINS A TABLE OF SVC DUMP EXIT MODULES. IEAVTSDI COUNTS THE NUMBER OF EXITS, GETS STORAGE FOR THE SVC DUMP EXIT TABLE (RTSDEXIT) FROM FIXED SQA, SUBPOOL 239, AND SETS RTSDEXTB TO POINT TO IT. SVC DUMP USES THIS EXIT TABLE TO INVOKE THE DUMP EXIT MODULES.

4. LOADS EACH EXIT MODULE LISTED IN IEAVTSXT. IF THE LOAD WAS SUCCESSFUL, STORES THE ADDRESS IN THE SVC DUMP EXIT TABLE

Diagram 41. SVC Dump Initialization (IEAVNPDI) Part 4 of 12

IEAVTSDI - MODULE OPERATION (Continued)

(RTSDEXIT) AND DELETES THE MODULE WHICH WILL FREE THE CDE ENTRY ASSIGNED TO IT. IF THE LOAD FAILED, ISSUES MESSAGE IEA883I. IF NO VALID EXITS ARE FOUND, FREES THE SVC DUMP EXIT TABLE AND SET POINTER (RTSDEXTB) TO ZERO.

5. CALLS MODULE IEAVTSDD TO PARSE THE SYSTEM PARAMETERS DUMP OPTION, AND TO INITIALIZE THE SYS1.DUMP DATA SETS.

6. GETMAINS THE FOLLOWING STORAGE AREAS AND SAVES THE ADDRESSES IN THE INDICATED CONTROL BLOCKS:

CONTROL BLOCK	ADDRESS STORED	SUBPOOL	SIZE	DESCRIPTION
CCWIDAW	SDDFTCCW	227-CSA	(24*(MAX RECS PER TRACK))+32	CCWS & IDAWS FOR FULL TRACK WRITE
OUTPUT	SDDFTBUF	231-CSA	8224*(MAX RECS PER TRACK)	FULL TRACK WRITE OUTPUT BUFFERS

@D1A

Diagram 41. SVC Dump Initialization (IEAVNPDI) Part 5 of 12

IEAVTSDI - DIAGNOSTIC AIDS

ENTRY POINT NAME: IEAVTSDI

MESSAGES:

```
       MESSAGE ID:    IEA883I
       MODULE XXXXXXXX NOT FOUND
THIS MESSAGE IS ISSUED FOR ANY SVC DUMP MODULE
WHICH CANNOT BE LOADED AS WELL AS
ANY OTHER DUMP EXIT MODULES SPECIFIED IN IEAVTSXT
```

ABEND CODES: None

WAIT STATE CODES: None

RETURN CODES:

EXIT NORMAL:

0 - PROCESSING COMPLETE

REGISTER CONTENTS ON ENTRY:

```
REG 2  - ADDRESS OF NVT
REG 3  - ADDRESS OF CVT
REG 13 - ADDRESS OF STANDARD SAVE AREA
REG 14 - RETURN ADDRESS
REG 15 - ENTRY POINT ADDRESS
```

REGISTER CONTENTS ON EXIT:

EXIT NORMAL:

SAME AS ENTRY REGISTERS

Diagram 41. SVC Dump Initialization (IEAVNPDI) Part 6 of 12

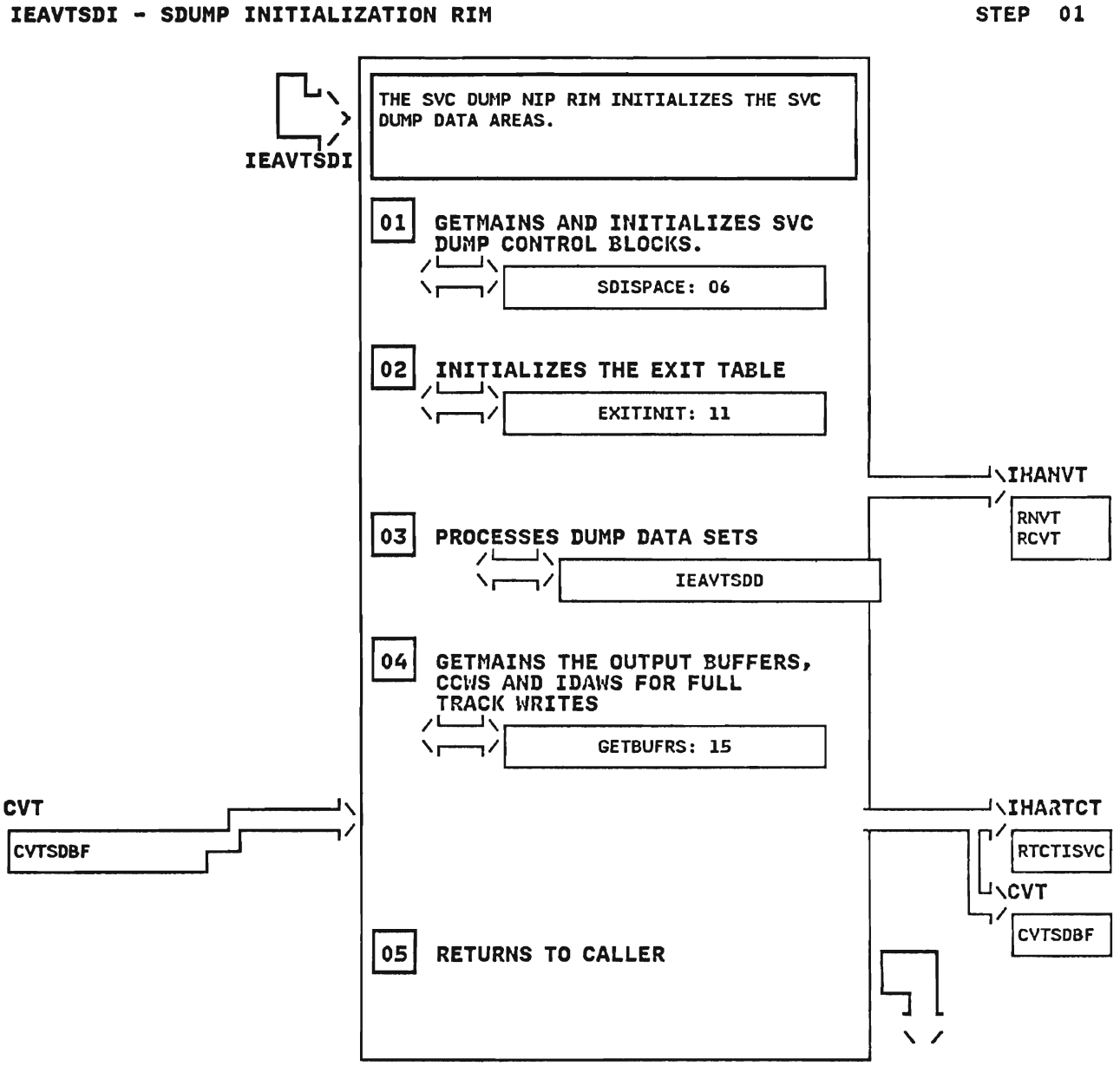


Diagram 41. SVC Dump Initialization (IEAVNPDI) Part 7 of 12

IEAVTSDI - SDUMP INITIALIZATION RIM STEP 06

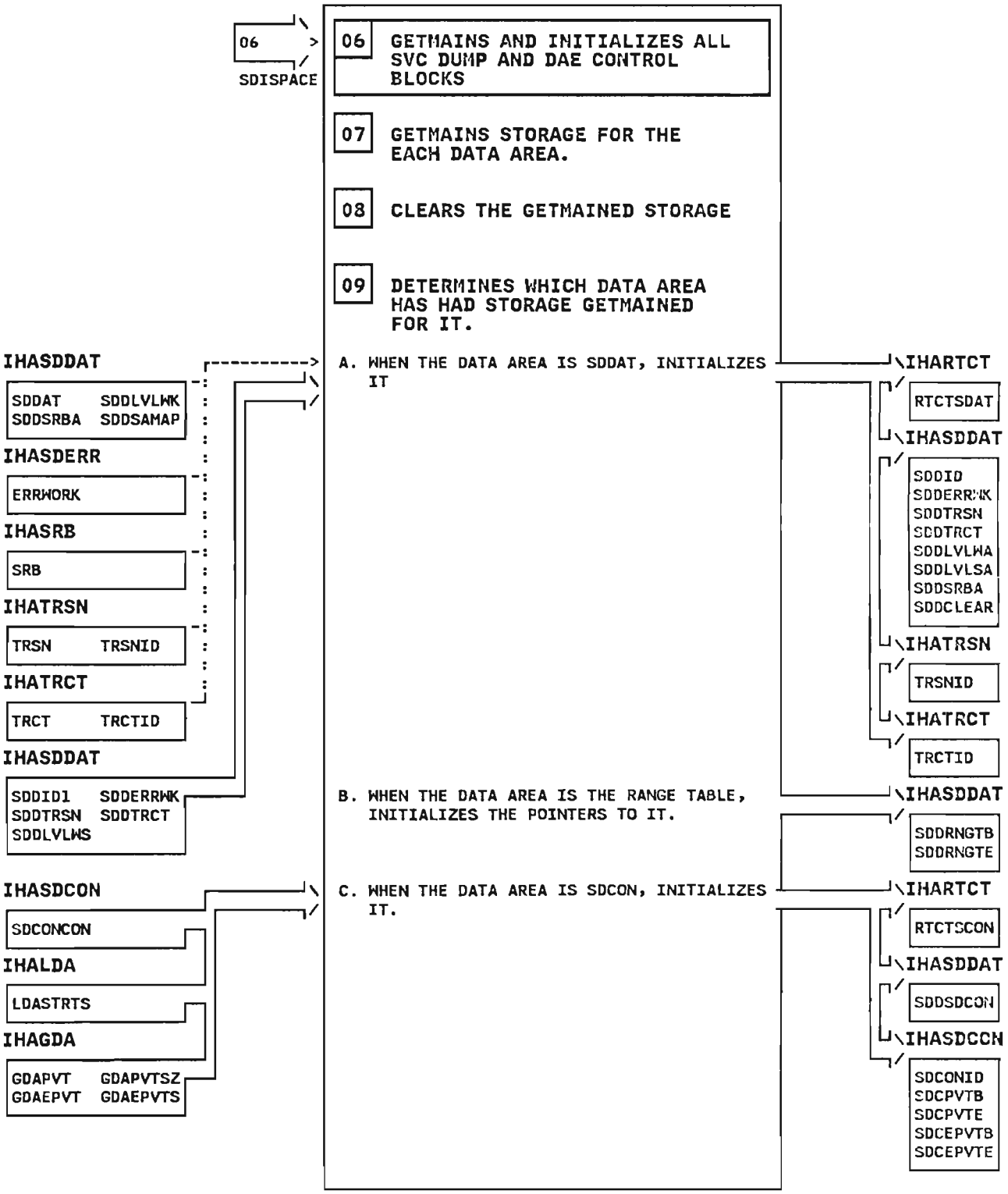


Diagram 41. SVC Dump Initialization (IEAVNPDI) Part 8 of 12

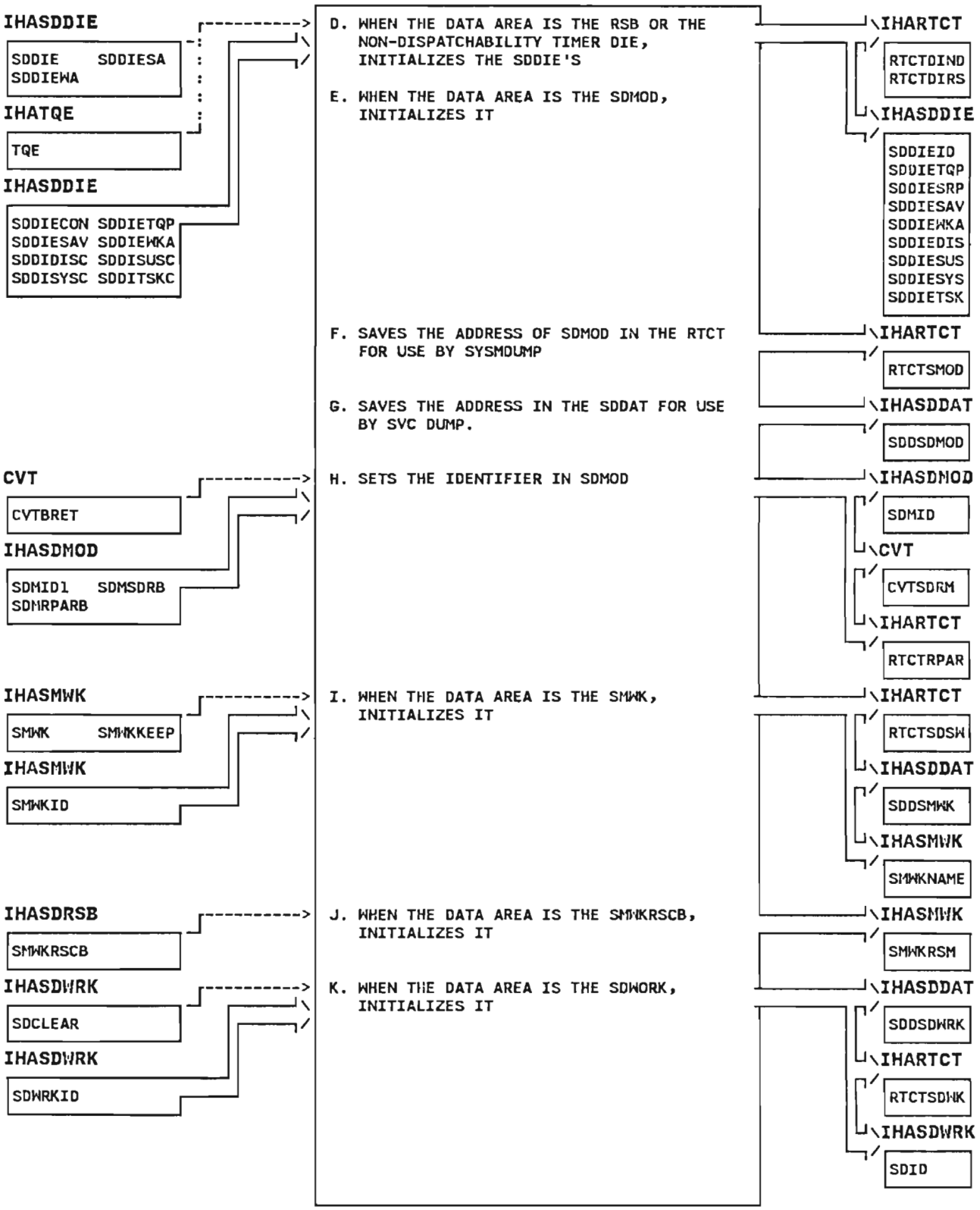


Diagram 41. SVC Dump Initialization (IEAVNPDI) Part 9 of 12

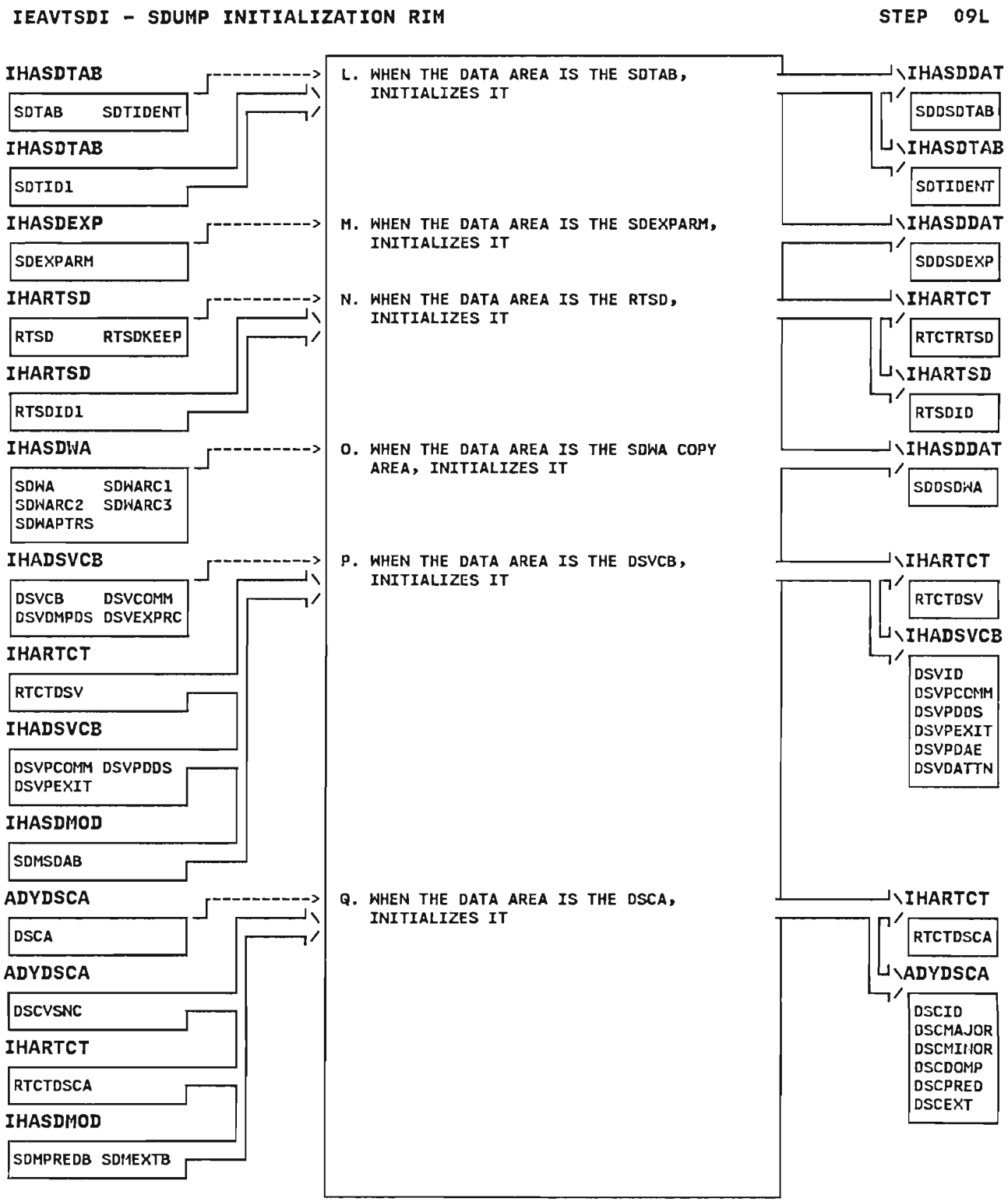


Diagram 41. SVC Dump Initialization (IEAVNPDI) Part 10 of 12

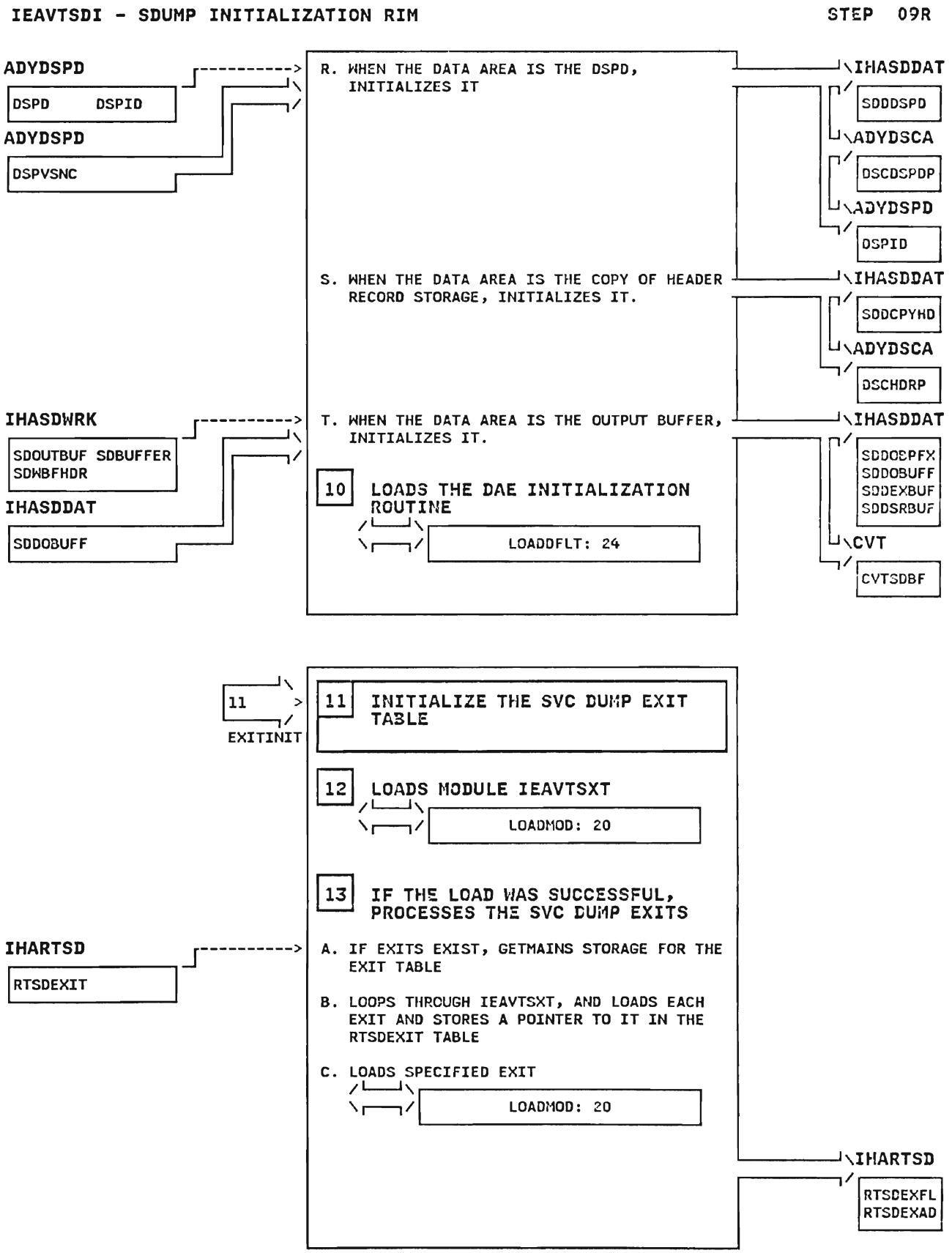


Diagram 41. SVC Dump Initialization (IEAVNPDI) Part 11 of 12

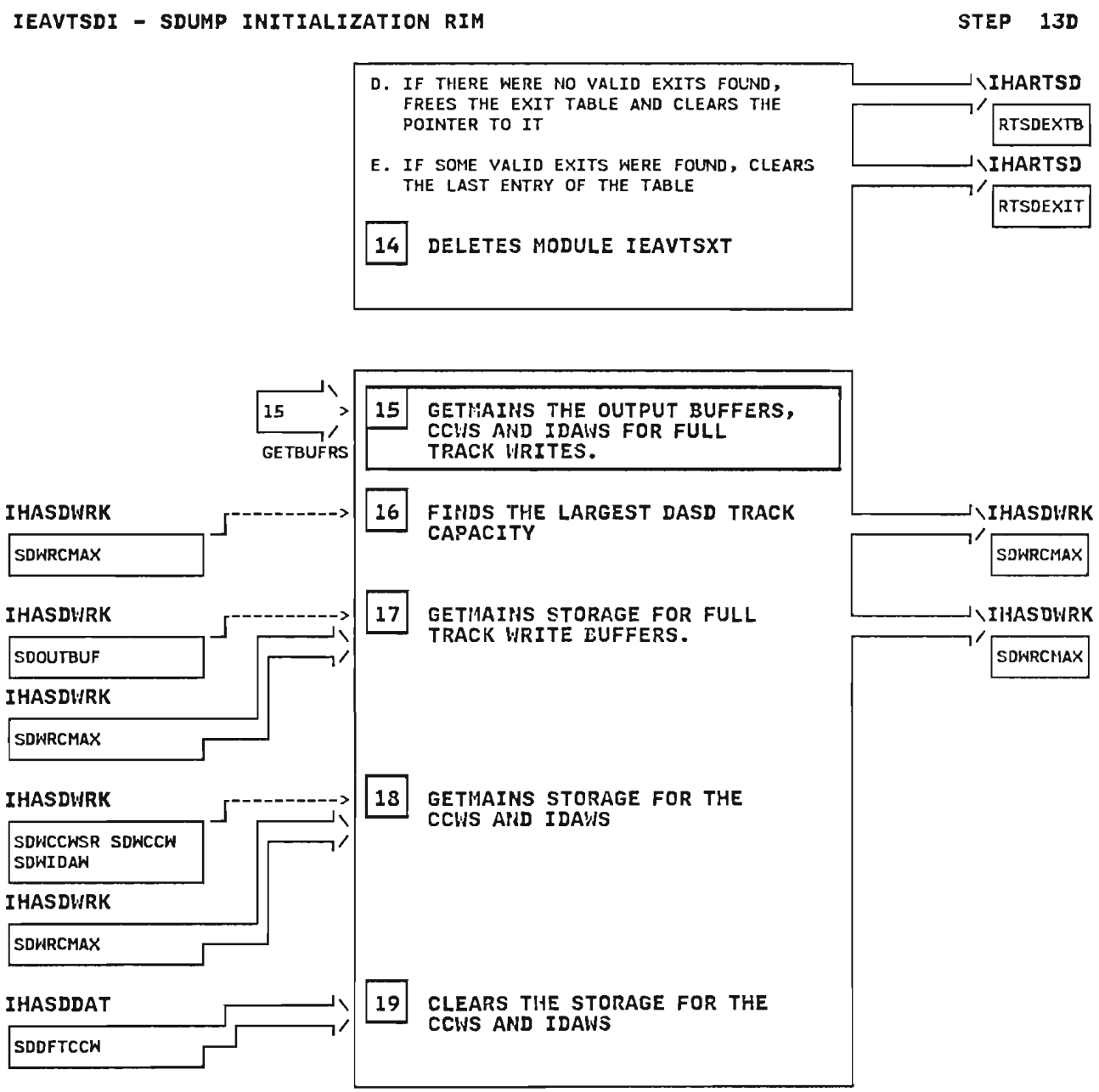


Diagram 41. SVC Dump Initialization (IEAVNPDI) Part 12 of 12

IEAVTSDI - SDUMP INITIALIZATION RIM STEP 20

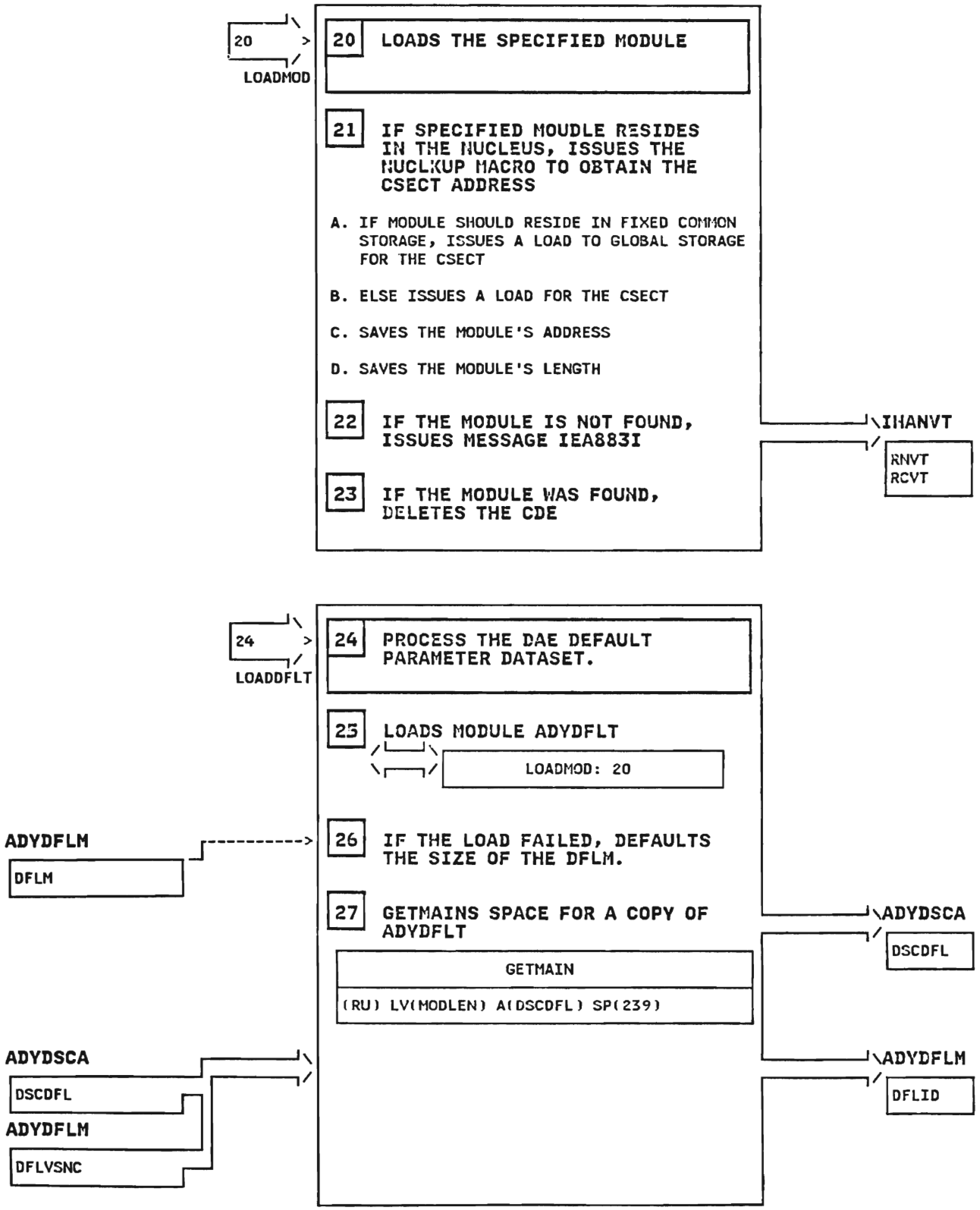


Diagram 42. Program Call/Authorization Initialization (IEAVNPF5) Part 1 of 2

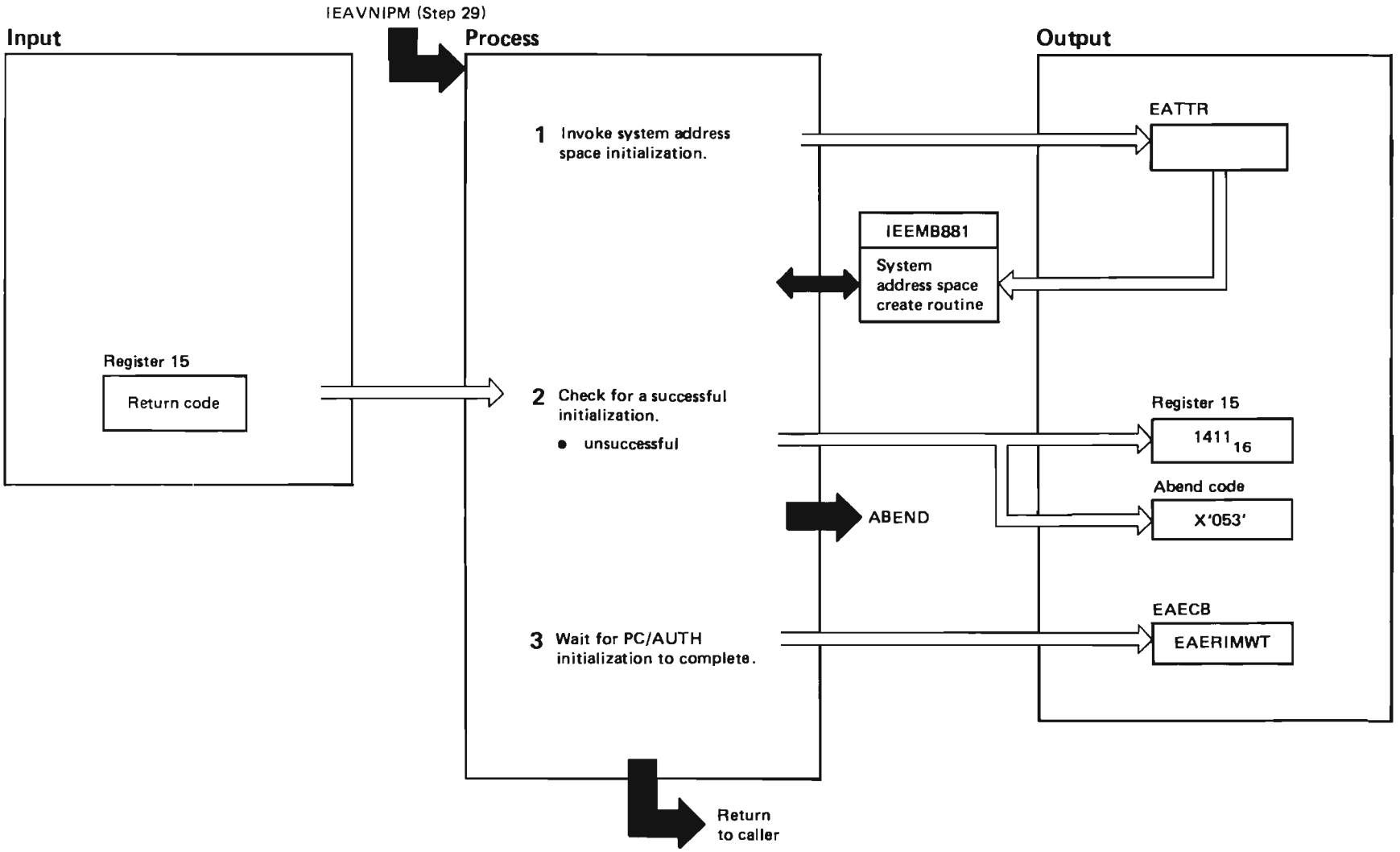


Diagram 42. Program Call/Authorization Initialization (IEAVNPF5) Part 2 of 2

Extended Description	Module	Label
The program call/authorization RIM creates the PC/AUTH address space and initializes it, including the control blocks and tables necessary to allow cross memory communication among address spaces to take place.		
1 Initialize the attribute list (EATTR) that describes the attributes of the PC/AUTH address space: it is to be exempt from system non-dispatchability (that is, it is to be in limited function start mode), it cannot be terminated even for DAT errors, and it is to be used for data only. Invoke the system address space create routine (IEEMB881), passing the attribute list as input. Indicate on the call to IEEMB881 that the PC/AUTH address space initialization routine, IEAVXMAS, is to get control to initialize the PC/AUTH address space after it is created. (IEAVXMAS is an example of system address space initialization, depicted in Diagram 39.)	IEAVNPF5	
2 Examine the return code from IEEMB881 to determine if the address space creation was successful. If so, continue with step 3. If the return code indicates that the creation was not successful, then place the return code from register 15 in register 2, place the reason code from register 0 in register 3, and issue an ABEND with a code of X'053'.	IEAVNPF5	
3 With address space initialization underway, issue the WAIT macro instruction to wait until IEAVXMAS finishes initializing the PC/AUTH address space. IEAVXMAS will post the ECB IEAVNPF5 is waiting on (EAERIMWT) to indicate that the PC/AUTH address space is initialized. IEAVNPF5 then returns to its caller.	IEAVNPF5	

Recovery Processing

None.

Diagram 43. Program Call/Authorization Address Space Initialization (IEAVXMAS) Part 1 of 8

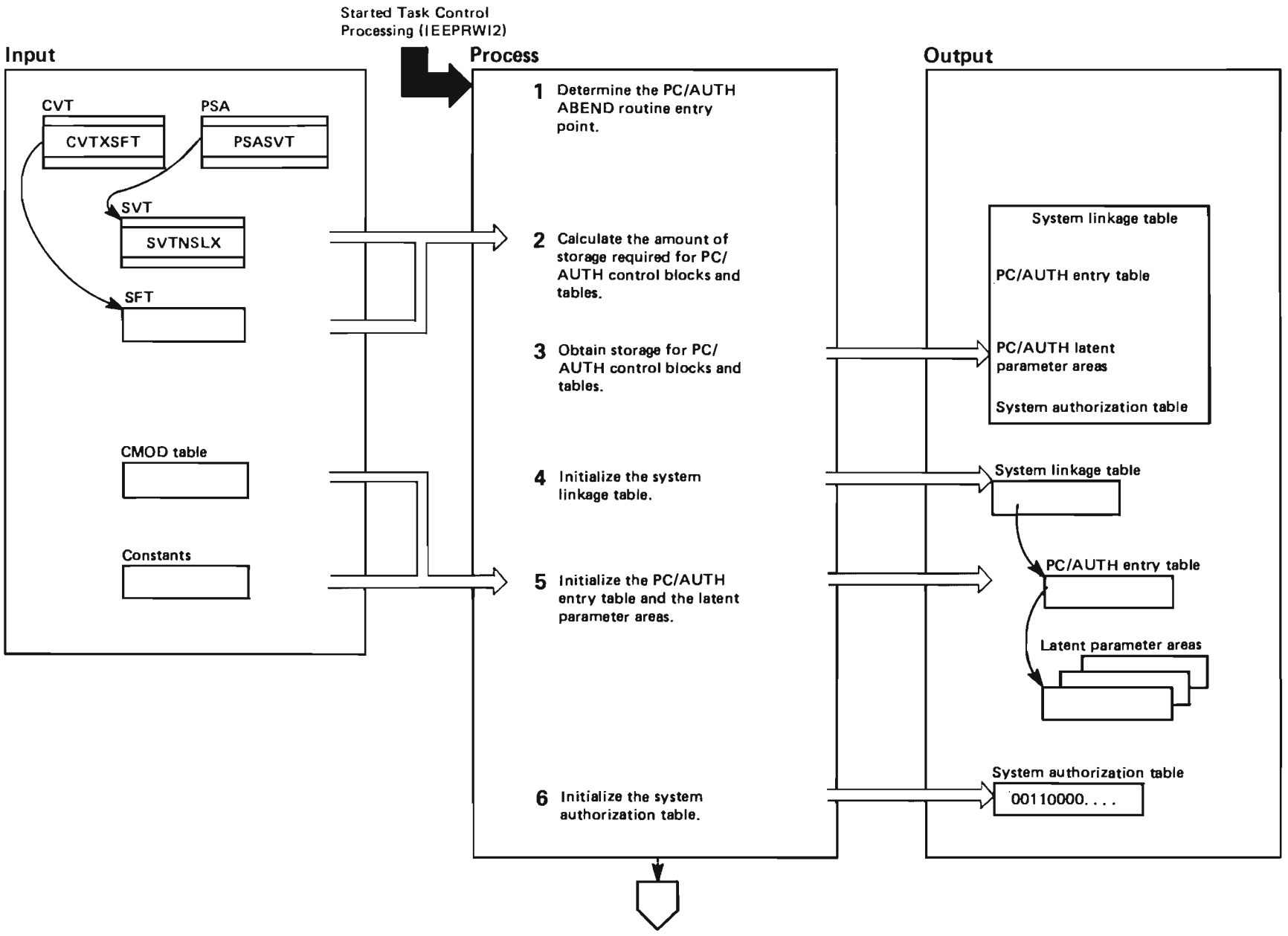


Diagram 43. Program Call/Authorization Address Space Initialization (IEAVXMAS) Part 2 of 8

Extended Description	Module	Label
PC/AUTH address space initialization is an example of model system address space initialization routine (depicted in Diagram 61). IEAVXMAS initializes the PC/AUTH address space in response to a request from the PC/AUTH RIM, IEAVNPF5 during NIP. This initialization includes obtaining and initializing the PC/AUTH control blocks and tables required for the PC/AUTH address space. IEAVXMAS also initializes other control information for other existing address spaces so that PC/AUTH services can be used during the remainder of NIP processing. Input to IEAVXMAS consists of: • The ECBs (IEZEAECB) used to communicate between IEAVNPF5 and IEAVXMAS during PC/AUTH address space initialization. • The address of the system address space initialization WAIT/POST routine, IEEMB883, which IEAVXMAS uses to complete the PC/AUTH address space initialization and wait for master scheduler initialization to complete. • The PC/AUTH ASCB. Other control blocks and tables accessed by IEAVXMAS include: • A system function table (SFT) identifying the PC/AUTH entry tables. • A CMOD table identifying the PC/AUTH services corresponding to the PC/AUTH entry table entries. • An address space second table entry (ASTE) pointed to by the PC/AUTH ASCB. • A supervisor vector table (SVT) for indicating to the other system components that PC/AUTH services are available and active.		
1 Invoke the PC/AUTH nucleus entry point search routine, IEAVXEPM, to determine the PC/AUTH ABEND routine entry point address. If IEAVXEMP returns a non-zero return code, issue an ABEND with code X'053' and an appropriate reason code. If IEAVXEPM returns a zero return code, then the PC/AUTH ABEND routine's entry point address is in register 2 and is saved for later use.	IEAVXEPM	
2 Determine the number of PC/AUTH entry indexes, the highest linkage index in the SFT, and the highest possible system linkage index. If the highest linkage index in the SFT is too high, issue an ABEND with code X'053', and an appropriate reason code; otherwise calculate the amount of storage needed for the system linkage table, PC/AUTH entry table and latent parameter areas, and the system authorization table.	IEAVXMAS	
3 Issue a GETMAIN macro instruction to obtain storage from subpool 255 for the system linkage table, PC/AUTH entry table and latent parameter areas, and the system authorization table.		
4 Initialize the system linkage table by setting all the linkage table entries invalid. Then initialize the appropriate linkage table entry for the PC/AUTH entry table. To initialize the entry, mark it as valid and set it to point to the PC/AUTH entry table.		
5 Use the CMOD table and constants within this module to initialize the PC/AUTH entry table and latent parameter areas. Issue the LOAD macro instruction to load the PC/AUTH load module, IEAVXPCA. IEAVXPCA permanently resides in the PC/AUTH address space and contains the PC/AUTH service routines. The CMOD table contains the entry point names of the various PC/AUTH services. Issue the LOAD macro instruction for each entry point name. For each address returned from the LOAD macro instruction, initialize a PC/AUTH entry table entry and its corresponding latent parameter area.		
6 Set the system authorization table to its base state, identifying authorized and unauthorized authorization indexes.		

Diagram 43. Program Call/Authorization Address Space Initialization (IEAVXMAS) Part 3 of 8

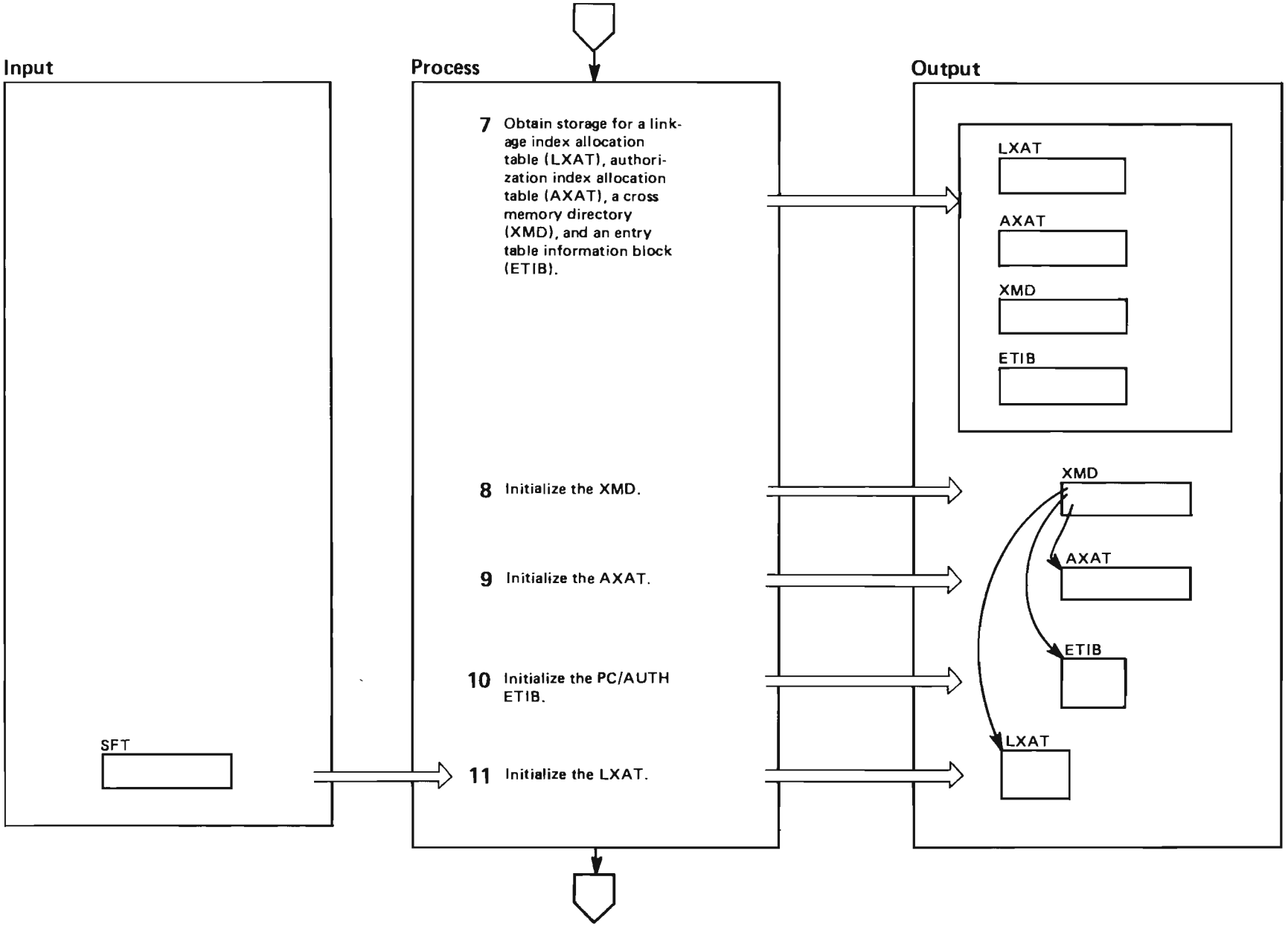


Diagram 43. Program Call/Authorization Address Space Initialization (IEAVXMAS) Part 4 of 8

Extended Description	Module	Label
7 Determine the amount of storage required for a linkage index allocation table (LXAT), an authorization index allocation table (AXAT), e cross memory directory (XMD), and an entry table information block (ETIB). Acquire the local lock and issue a GETMAIN macro instruction to obtain this storage from subpool 229; then release the local lock. Performing a GETMAIN in this way ensures that the storage will be associated with the RCT so that the AXAT and LXAT can be freed by PC/AUTH service routines.		
8 Initialize the XMD. The XMD contains data for all PC/AUTH services.		
9 Indicate in the AXAT that authorization indexes of 0 and 1 are owned by the PC/AUTH address space.		
10 The ETIB contains pointers to the entry table it describes, the latent parameter areas for that entry table, and other information such as the owning ASCB and entry table attributes.		
11 Indicate in the LXAT that all linkage indexes reserved in the SFT are system linkage indexes and are owned. Initially, all linkage indexes contained in the SFT are marked as being owned by the PC/AUTH address space. When another address space performs the entry table connect for a system linkage index, that address space becomes the owner of the linkage index.		

Diagram 43. Program Call/Authorization Address Space Initialization (IEAVXMAS) Part 5 of 8

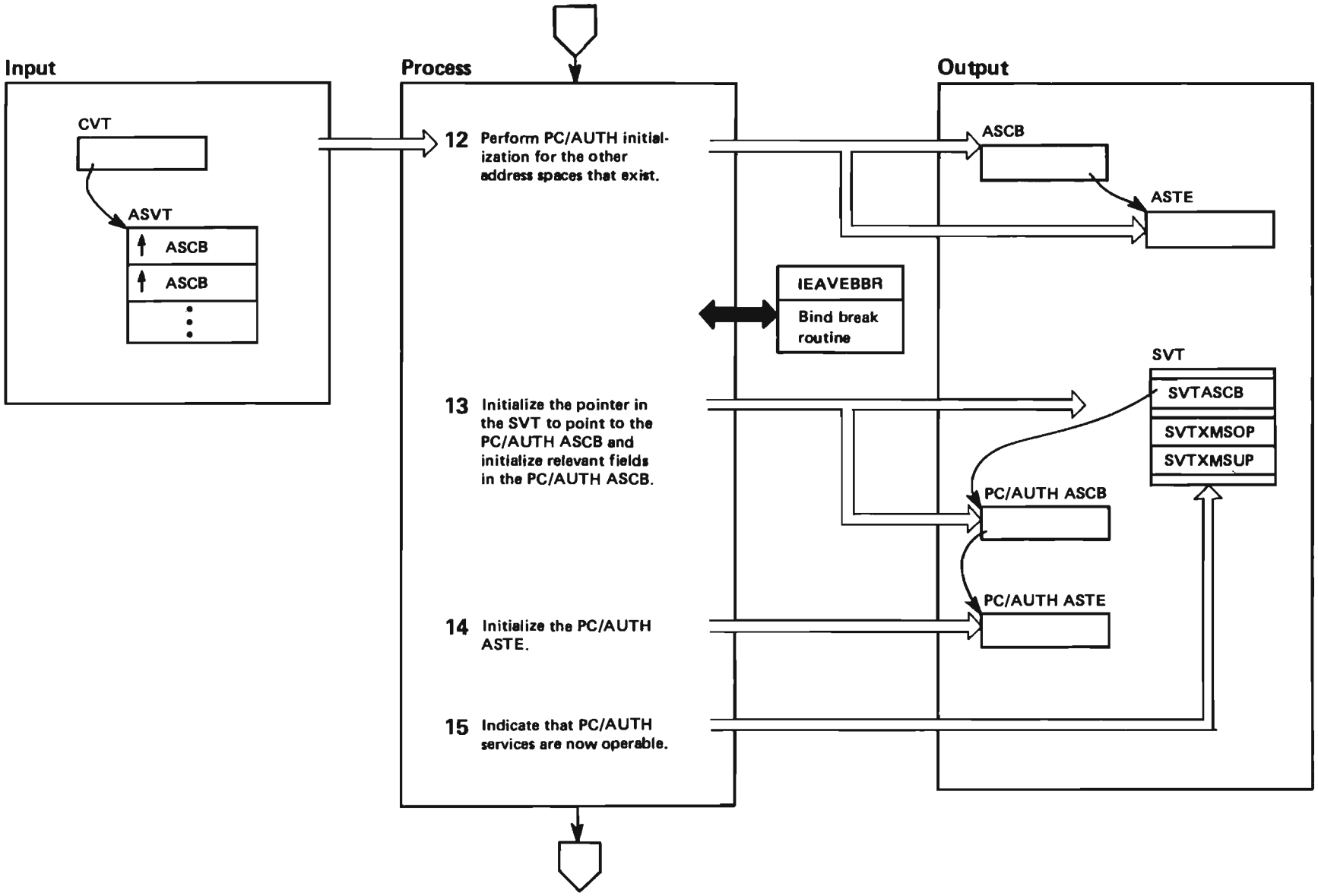


Diagram 43. Program Call/Authorization Address Space Initialization (IEAVXMAS) Part 6 of 8

Extended Description	Module	Label
12 Perform the following for each address in the ASVT except the current one. • Set the AX to 0 or 1 if the ASID=1. • Set up to use the system linkage table. • Set up to use the system authorization table. • Reset the ASTEICMA field. (When any ASTE is created, ASTEICMA is set on to prevent cross memory accesses before the other fields of the ASTE are initialized.)		
Invoke the bind break routine (IEAVEBBR) to perform re-translation by initializing the control registers of any active address spaces so that they can use PC/AUTH services. If the return code from IEAVEBBR is not zero, ABEND with a code of X'053'.	IEAVEBBR	
13 Indicate in the PC/AUTH ASCB the number of authorization indexes, linkage indexes, and entry tables owned and the virtual addresses of the system authorization table and system linkage table. Chain the PC/AUTH ASCB to the SVT.	IEAVXMAS	
14 Perform the following for the PC/AUTH ASTE: • Set the authorization index to 1. • Set up to use the system linkage table. • Set up to use the system authorization table. • Reset the ASTEICMA field after the other fields have been updated.		
15 Turn bits SVTXMSOP and SVTXMSUP on to indicate that PC/AUTH services are available.		

Diagram 43. Program Call/Authorization Address Space Initialization (IEAVXMAS) Part 7 of 8

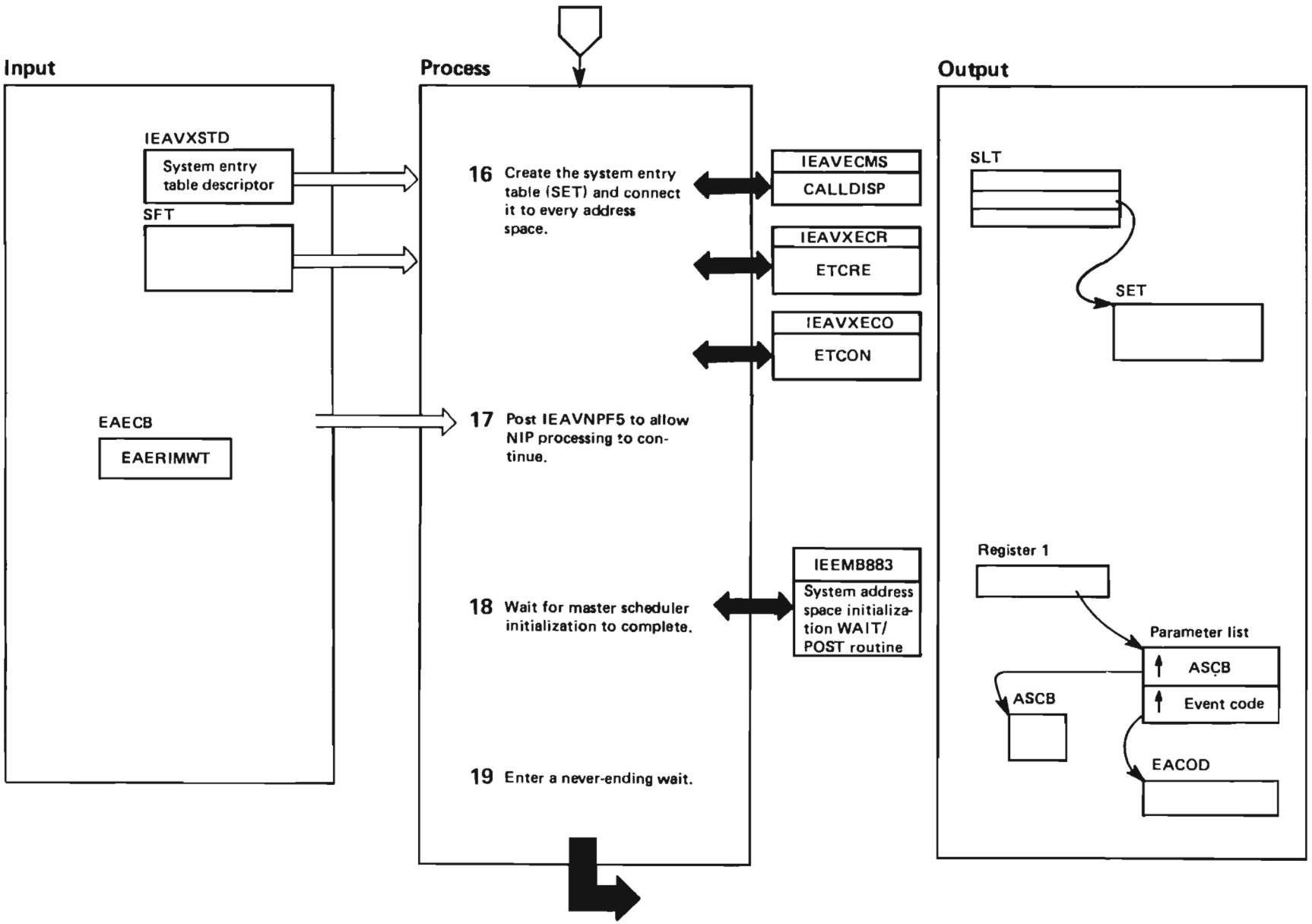


Diagram 43. Program Call/Authorization Address Space Initialization (IEAVXMAS) Part 8 of 8

Extended Description	Module	Label
16 Issue the CALLDISP macro instruction to retranslate the control registers for the processor on which IEAVXMAS is active. Using the system entry table descriptor (IEAVXSTD) in module IEAVXSEM as input, issue the ETCRE macro instruction to create the system entry table (SET). To connect the system entry table to all address spaces, issue the ETCON instruction macro.	IEAVECMS IEAVXECR IEAVXECO	
17 Post the PC/AUTH RIM, IEAVNPF5, using the ECB named EAERIMWT, tp permit NIP processing to continue.		
18 Call the system address space intitialization WAIT/POST routine, IEEMB883, to indicate that control should be returned when master scheduler initialization has completed. Invokes IEEMB883 in order to allow the master scheduler to fully initialize PC/AUTH's control blocks, such as the pointer in the TCB to the TIOT, which will be the master scheduler's TIOT. Pass to IEEMB883 a parameter list containing the PC/AUTH ASCB address and the event code address.	IEEMB883	
19 When IEEMB883 returns, enter a never-ending wait. (The PC/AUTH address space is thus in limited function start mode.)	IEAVXMAS	

Recovery Processing

None.

Diagram 44. System Trace Initialization (IEAVNP51) Part 1 of 2

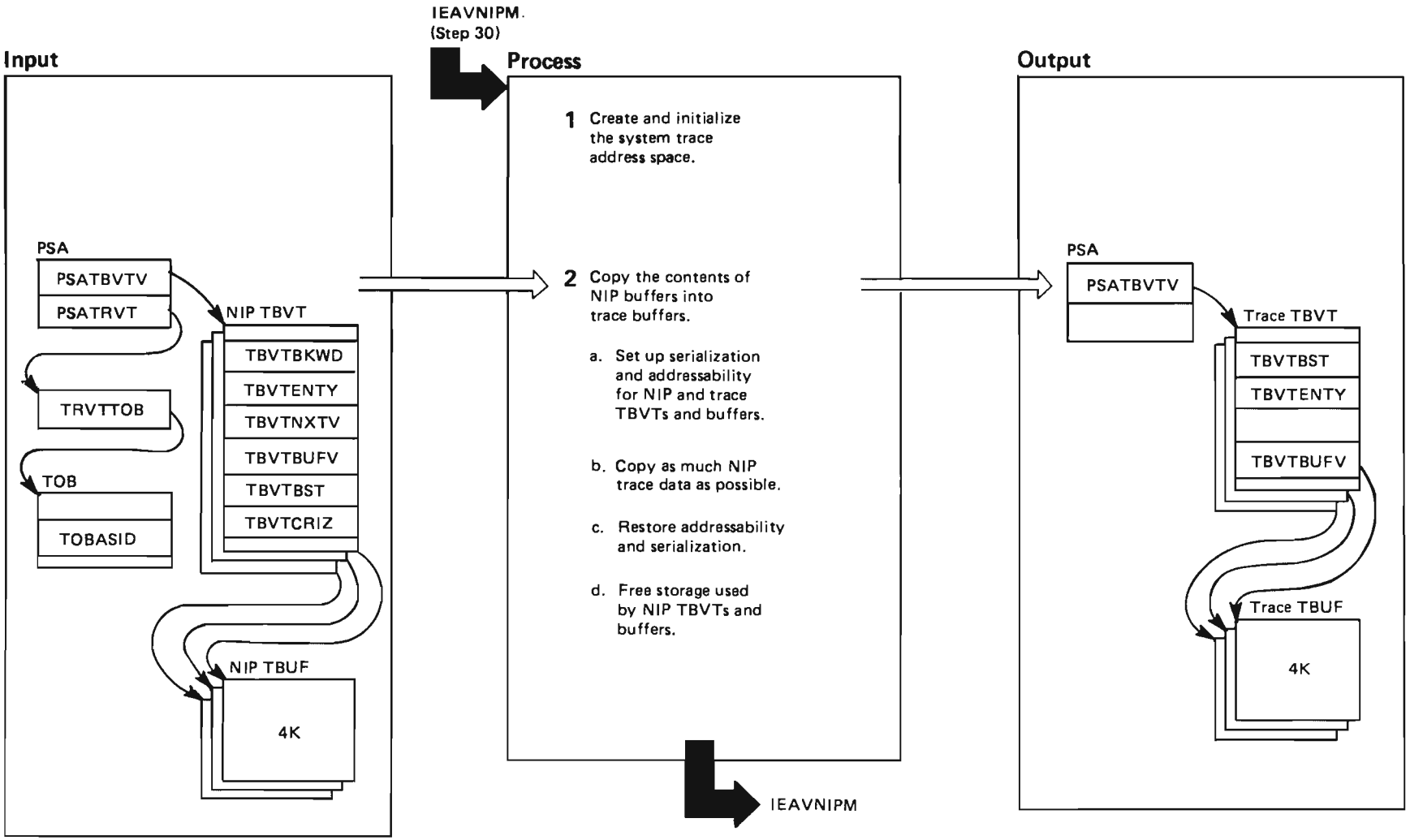


Diagram 44. System Trace Initialization (IEAVNP51) Part 2 of 2

Extended Description	Module	Label
IEAVNIPM calls the system trace RIM, IEAVNP51, to create the system trace address space and initialize the trace function.		
1 The trace RIM calls the alter trace environment service (ALTRTRC) to initialize the trace address space and initialize the trace function. It checks the return code from ALTRTRC. If the return code is X'0', processing continues. Otherwise, ALTRTRC did not initialize the trace function, and the system enters a wait state with a code X'151'. For more information on the ALTRTRC service, see the system trace section in *System Logic Library*.	IEAVNP51	
2 The trace RIM preserves the most recent data from NIP buffer vector tables (TBVTs) and buffers (TBUFs) by copying the most recent NIP trace data to the system trace TBVTs and buffers. Note that the NIP TBVTs and buffers are in the system queue area (SQA) and that the trace TBVTs and buffers are in the private area of the trace address space.		
If NIP has not created a trace structure for NIP tracing (that is, PSATBVTV = 0), the trace RIM returns to IEAVNIPM.		
If NIP has created a trace structure for NIP tracing (that is, PSATBVTV was non-zero to entry), then the trace RIM proceeds as follows:		
a. To prevent the loss of any trace entries, the trace RIM calls the PGSER service to fix the pages this module occupies and obtains exclusive use of the trace spin lock for disablement and serialization on the TBVT queue. To obtain access to the trace data, it issues the ESAR instruction to set up secondary addressability to the trace address space. (TOBASID gives the ASID of the trace address space.) The trace RIM invokes ALTRTRC to suspend recording of system events before the trace RIM copies the trace data. If ALTRTRC was unable to suspend recording system events, the trace RIM returns control to IEAVNIPM.		
b. The trace RIM loops through the NIP TBVT queue. Starting with the most recent NIP TBVT, it copies trace data information from each NIP data buffer and NIP TBVTs into the new trace buffers (addressed by PSATBVTV) and trace TBVTs. If there is more NIP data than the system TBVTs and buffers can hold, then the earliest NIP trace data is lost. TBVTBUFV and TBVTENTY identify the start and end address of the area the trace RIM copies. TBVTPREV and TBVTNEXT are the forward and backward pointers for the TBVT queue.		
c. The trace RIM calls the ALTRTRC service to resume recording system events and resets secondary addressability to that entry. It issues a SETLOCK macro instruction to release the trace spin lock to enable for interrupts and release the serialization.		
d. The trace RIM issues a FREEMAIN macro instruction to release the SQA areas that NIP used for its TBVTs and buffers and returns to IEAVNIPM with a return code of zero.		

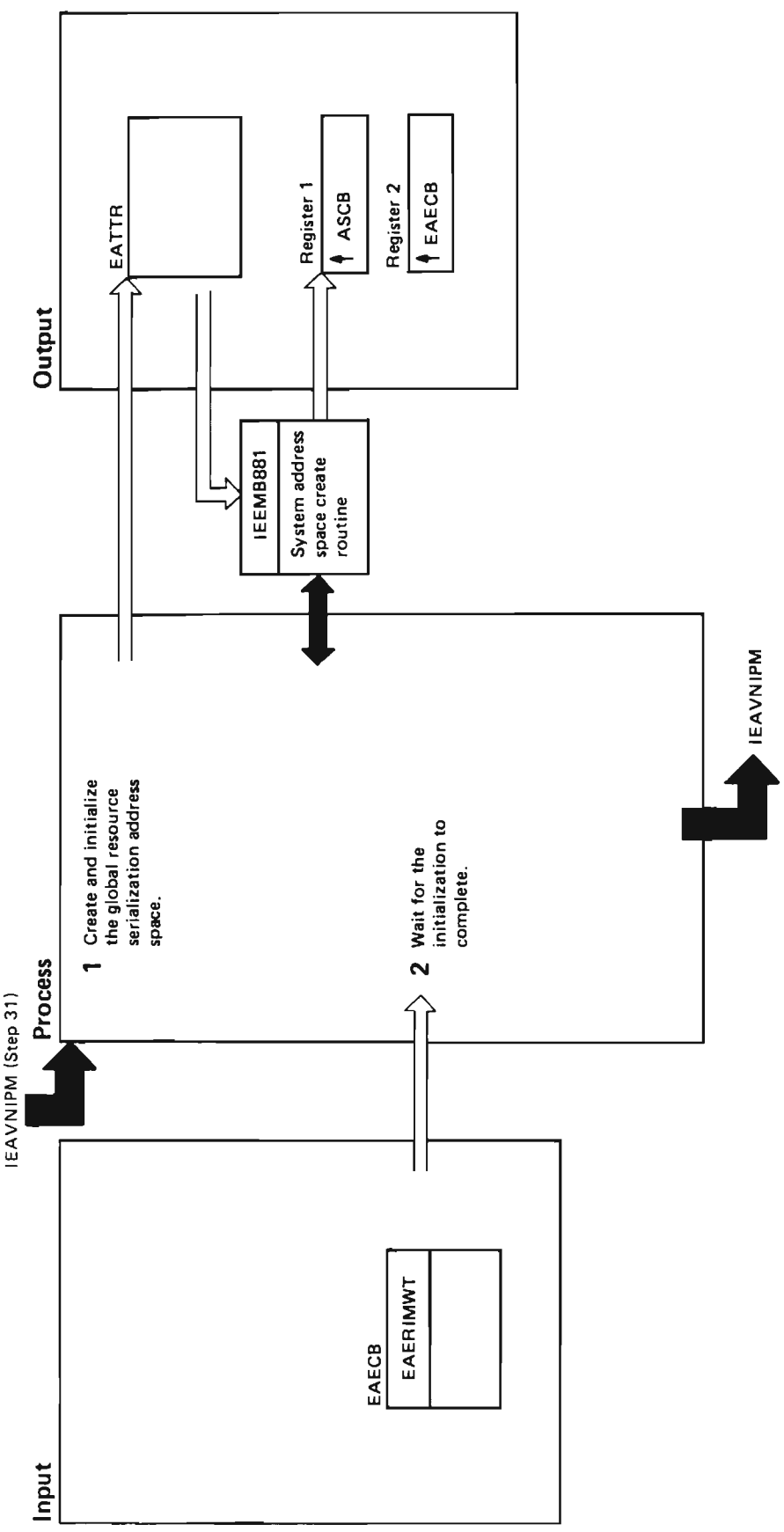


Diagram 45. Global Resource Serialization Trigger Address Space Creation (IEAVNP33/ISGNTASC) Part 1 of 2

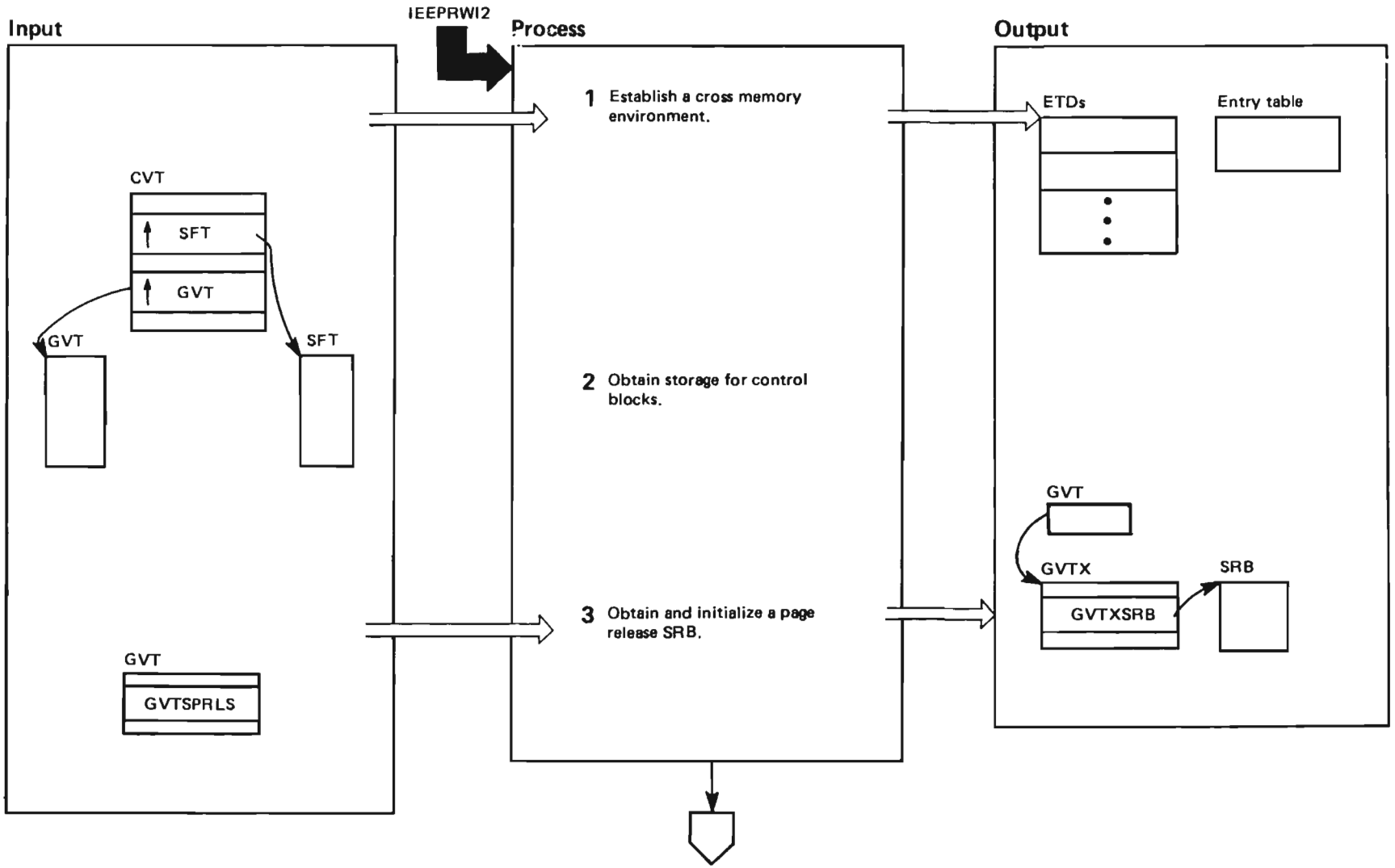
Diagram 45. Global Resource Serialization Trigger Address Space Creation (IEAVNP33/ISGNTASC) Part 2 of 2

Extended Description	Module	Label
ISGNTASC is the second global resource serialization RIM to get control and starts the creation of the global resource serialization address space. ISGNTASC receives control from IEAVNIPM in supervisor state and in key zero. IEAVNIPM uses the standard interface to RIMs. • Register 2 contains the address of the NIP vector table (NVT). • Register 3 contains the address of the communications vector table (CVT). Note that IEAVNIPM knows ISGNTASC as IEAVNP33. ISGNTASC is an entry point in load module IEAVNP33.		
1 Build the three-word parameter list to pass to the system address space create routine (IEEMB881). The first word of the parameter list points to the system address space initialization attributes (The address space is exempt from system non-dispatchability and termination. It has the highest possible dispatching priority, it is non-swappable, non-cancellable, and non-forceable; and its page fixes are in preferred storage.) The second word points to the program name (ISGNASIM) to receive control from IEEPRWI2 in order to initialize the address space with global resource serialization information. ISGNASIM is an example of a model address space initialization routine depicted in diagram 56. The third word points to the length and character string containing the START parameters. Call IEEMB881 to create the address space. If system address space creation is successful, IEEMB881 returns the ASCB address in register 1 and the address of the ECB (EAERIMWT) in register 2.	IEEMB881	
2 Issue a WAIT on the ECB provided by IEEMB881 (EAERIMWT). ISGNASIM will post the ECB when the global resource serialization address space has been completely initialized. ISGNTASC then returns to IEAVNIPM.		

Recovery Processing

None.

Diagram 46. Global Resource Serialization Trigger Address Space Initialization (ISGNASIM) Part 1 of 12



Extended Description	Module	Label
ISGNASIM initializes the global resource serialization address space so that the global resource serialization services of the system will be operational. ISGNASIM is an example of model system address space initialization (depicted in Diagram 61. Input to ISGNASIM is a parameter list containing: • The address of the ASCB for the global resource serialization address space. • The address of an ECB to use to inform global resource serialization trigger address space creation module, ISGNTASC, that the global resource serialization address space has been initialized. • The address of the system address space initialization WAIT/POST routine, IEEMB883, used to wait until master scheduler initialization is complete.		
1 Establish a cross memory environment by using the ETCRE, AXSET, and ETCON macro instructions. First, build an entry table description (ETD) for each service: • ENQ/DEQ/RESERVE. • ENQ/DEQ/RESERVE (SQA QWB overflow). • Global resource serialization termination resource manager. • Fast path DEQ. • Fast path ENQ. • Global resource serialization SVC dump services. • Queue scanning services (for SCOPE=STEP, SYSTEM, and SYSTEMS). • Queue scanning services (for SCOPE=LOCAL or GLOBAL). • Mainline ESTAE. • ENQ/DEQ/RESERVE mainline FRR. • Global resource serialization storage manager. From the system function table (SFT), save the program call number of the following services in the global resource serialization vector table (GVT). • Global resource serialization SVC dump services. • ENQ/DEQ/RESERVE. • ENQ/DEQ/RESERVE (SQA QWB overflow). • Mainline ESTAE. • ENQ/DEQ/RESERVE mainline FRR. • Fast path ENQ. • Fast path DEQ. • Global resource serialization storage manager. • Global resource serialization termination resource manager. • ENQ/DEQ resource manager.	ISGNASIM	

Extended Description	Module	Label
After creating all ETDs, issue an ETCRE macro instruction to create the entry table, issue the AXSET macro instruction to set the authorization index for the global resource serialization address space, and issue the ETCON macro instruction to connect the global resource serialization entry table to the proper list of linkage index values. Other address spaces can now access the global resource serialization services defined in the entry table.		
2 Calculate the amount of storage needed for the resource queue area (RQA), the extended resource queue area (ERQA), a bit map and the ERQA describing the RQA, and the following control blocks: • Global resource serialization vector table extension (GVTX). • Local queue hash table (LQHT). • Global queue hash table (GQHT). • Local resource pool table (LRPT). • Global resource pool table (GRPT). • System/ASID hash table (SAHT). Issue the GETMAIN macro instruction with the LOC=(BELOW, ANY) operand to obtain this storage from subpool 229. Clear the storage containing the control blocks and the bit maps and issue the PGFIX macro instruction to fix all the pages describing this storage. Issue the GETMAIN macro instruction to obtain storage for the RQA from subpool 229. Issue the GETMAIN macro instruction with the LOC=(ANY, ANY) operand to obtain storage for the ERQA from subpool 127. Finally, issue the GETMAIN macro instruction for SQA storage (subpool 245) to obtain permanent storage for the queue work block (QWB). This storage is not cleared. The permanent QWB will describe local resource requests within the global resource serialization address space.	ISGNASIM	
3 Obtain SQA storage (subpool 245) for an SRB. The global resource serialization storage manager schedules this SRB to release pages fixed no longer needed in the resource queue area step 2 because the serialization required for a page release cannot be obtained in cross memory mode. GVTSPRLS contains the entry point address of the routine that will perform the page release. The SRB is initialized with: • The address in the GVTSPRLS field of the GVT. • An FRR address from GVTGFRR0 in the GVT. • The address of the global resource serialization address space ASCB and its ASID. • The address of a resource manager routine (BR 14).		

Diagram 46. Global Resource Serialization Trigger Address Space Initialization (ISGNASIM) Part 3 of 12

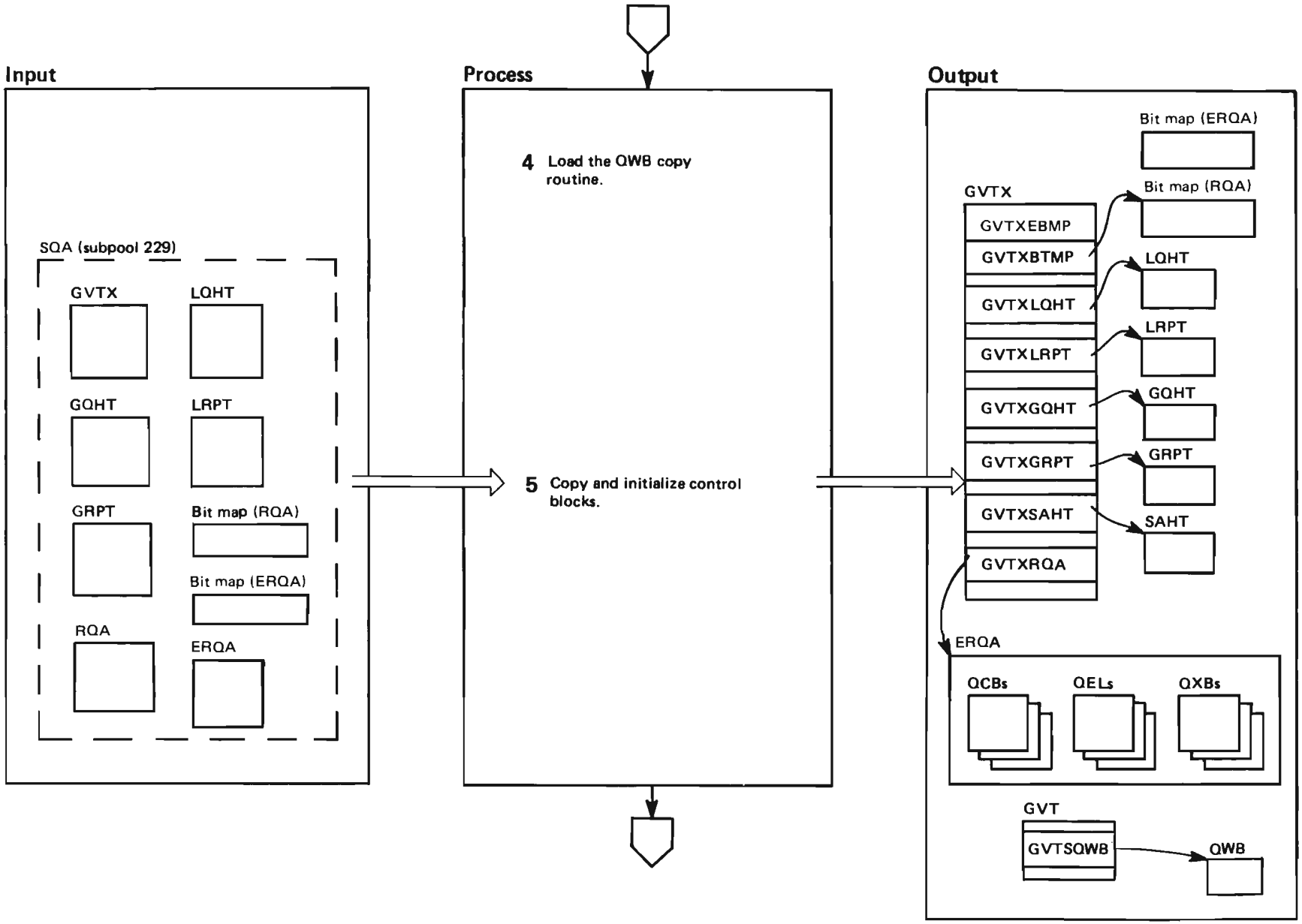


Diagram 46. Global Resource Serialization Trigger Address Space Initialization (ISGNASIM) Part 4 of 12

Extended Description	Module	Label
4 Issue the LOAD macro instruction to load ISGGQWBC, the QWB copy routine into the global resource serialization address space, and save its address. The address is stored later in the GVT. The QWB copy routine is used to extract information from the RSA-message.	ISGNASIM	
5 To prepare to copy the control blocks, obtain the local lock for the global resource serialization address space and the CMS ENQ/DEQ lock. These locks serialize the use of the GVT, GVTX, and other local control blocks in the global resource serialization address space. Store in the GVT, the address of the QWB copy routine that resides in the global resource serialization address space. Save the address of the SQA version of ISGQWBC. Use this address to release the storage later. Initialize the GVTX in the global resource serialization address space's private area. Store in the GVTX the address of the page release SRB. Put the address of the RQA and its size into the GVTX. Put the address of the ERQA and its size into GVTX. Save the RQA and ERQA bit map address in the GVTX. Build and format the LQHT and LRPT and store their addresses in the GVTX. Store the address of the new permanent QWB in the GVT. (Save the address of the old permanent QWB so its storage can be released later.) Copy all queue control blocks (QCBs), queue elements (QELs), and queue extension blocks (QXBs) from the SQA RQA to the ERQA in the global resource serialization address space's private area. Use the storage manager routine, ISGSALC, to obtain QCB, QEL, and QXB cells as needed during the copy process. Use the global/local hash routine (ISGGHASH) to obtain indexes into the LQHT for the new QCBs. Format the GQHT, GRPT, and SYSID/ASID hash table (SAHT) in the private area of the global resource serialization address space and save their addresses in the GVTX.	ISGNASIM	

Diagram 46. Global Resource Serialization Trigger Address Space Initialization (ISGNASIM) Part 5 of 12

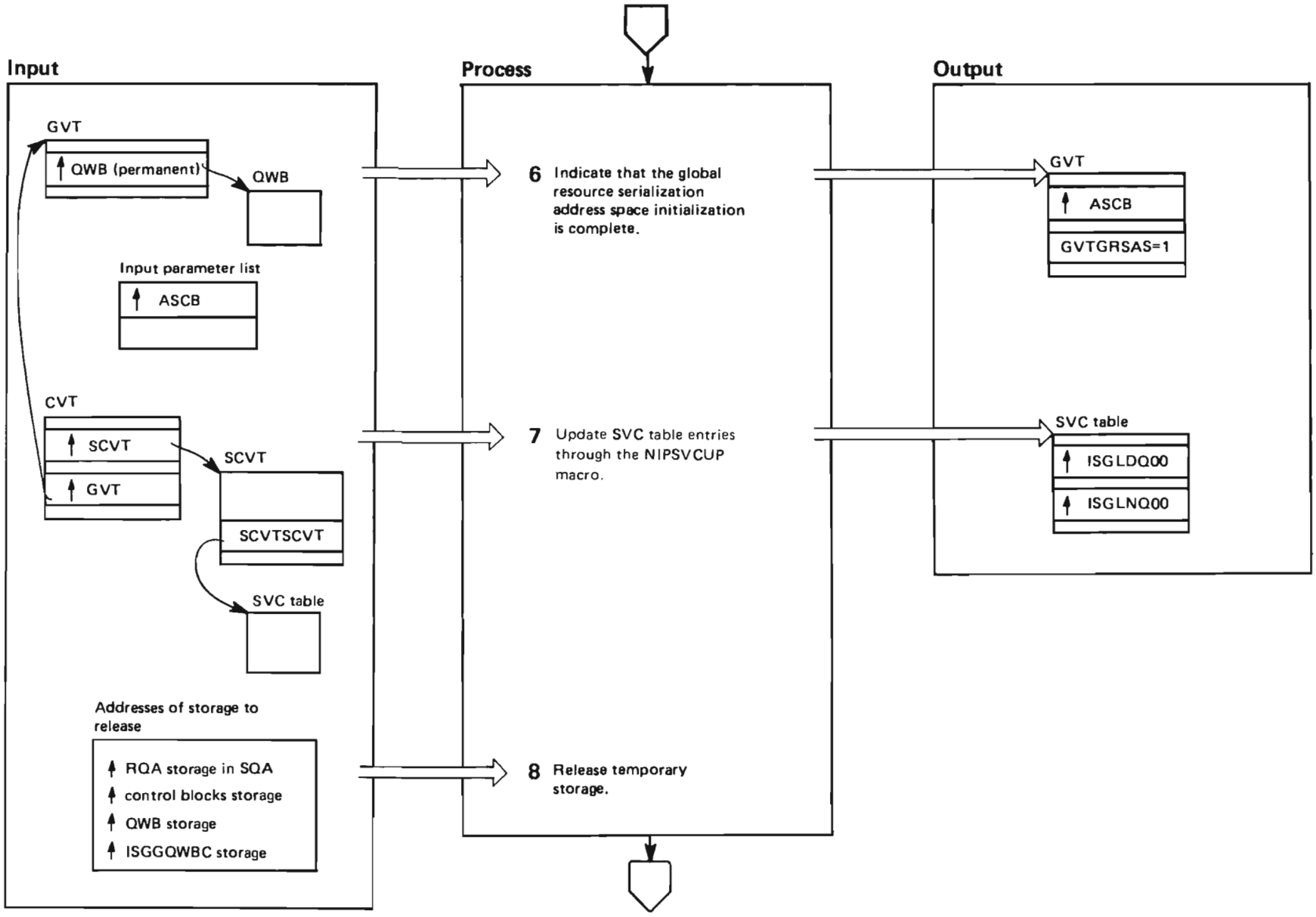


Diagram 46. Global Resource Serialization Trigger Address Space Initialization (ISGNASIM) Part 6 of 12

Extended Description	Module	Label
6 Place the ASCB address passed as input into the GVT. The ASCB is the one for the global resource serialization address space. Then indicate that the global resource serialization address space is initialized (GVTGRSAS='1').	ISGNASIM	
7 Update the SVC table entries through the NIPSVCUP macro for ENQ and DEQ so that control is now given to the fast path DEQ function (entry point ISGLDQ00) and the fast path ENQ function (entry point ISGLNQ00) with AMODE(31). These SVC table entries are initialized at this time because these fast path functions require the global resource serialization address space to be initialized. After initializing the SVC table entries, release CMS ENQ/DEQ lock and the local lock of the global resource serialization address space.	ISGNASIM	
8 Release temporary storage acquired by ISGNCBIM, which initialized global resource serialization control blocks used by local services before the creation of the global resource serialization address space: • Release the entire RQA in SQA unless one or more QWB cells are allocated, in which case, only release the RQA storage that doesn't include the QWB pool of storage. • Release global resource serialization control blocks in SQA. • Release the old permament QWB storage in SQA. • Release the SQA version of ISGGQWBC. All this storage is released by issuing the FREEMAIN macro instruction for SQA (subpool 245).	ISGNASIM	

Diagram 46. Global Resource Serialization Trigger Address Space Initialization (ISGNASIM) Part 7 of 12

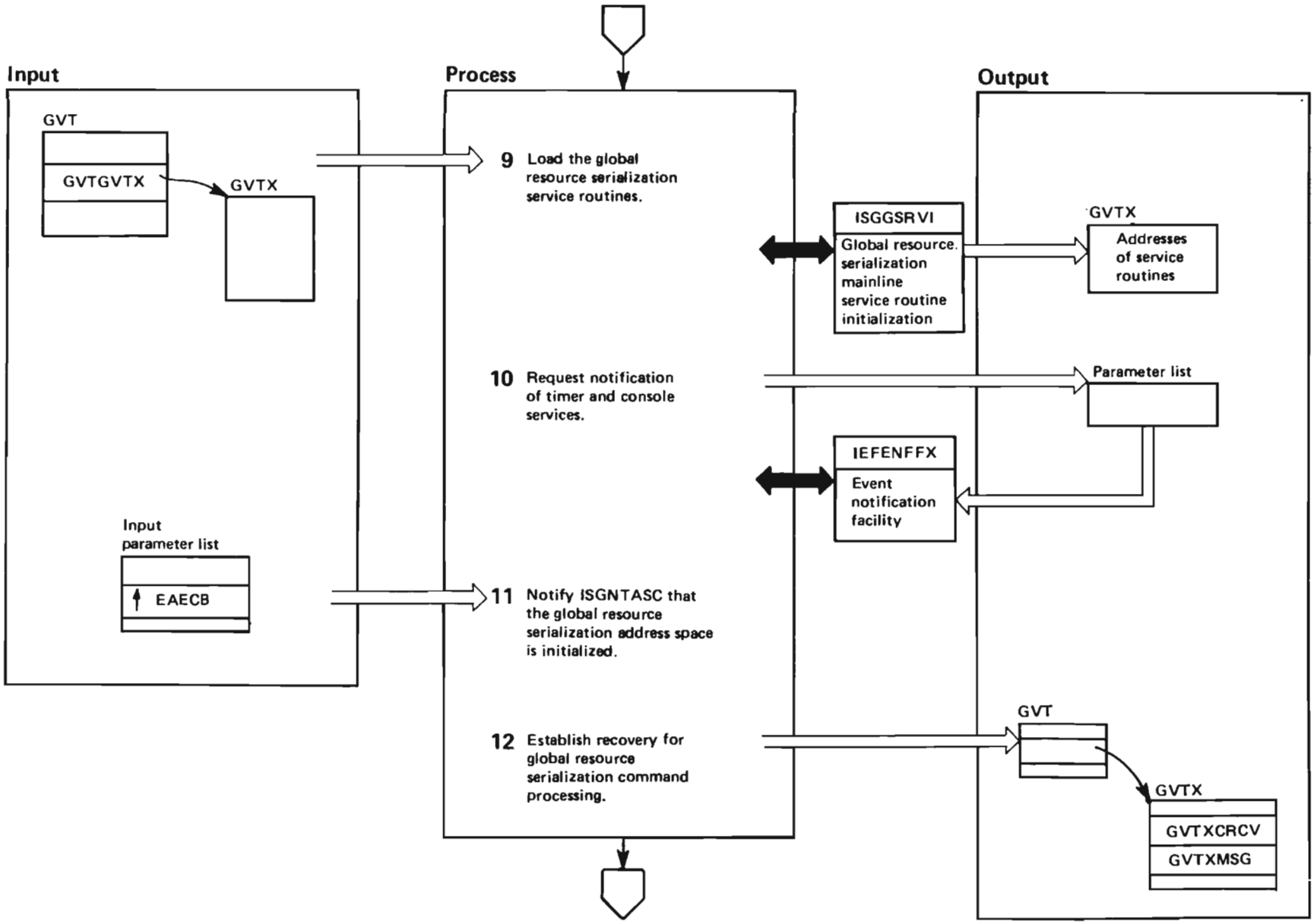
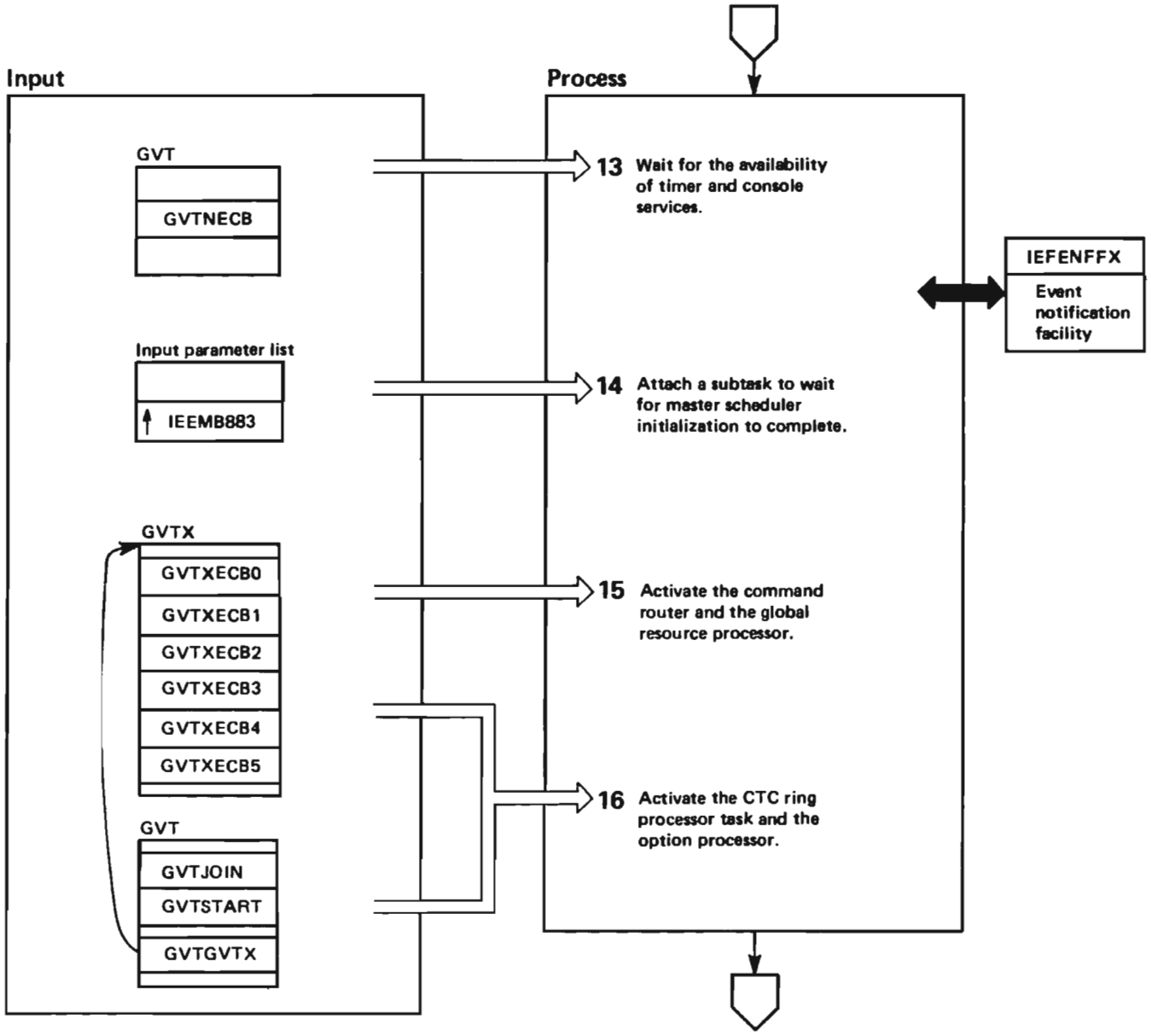


Diagram 46. Global Resource Serialization Trigger Address Space Initialization (ISGNASIM) Part 8 of 12

Extended Description	Module	Label
9 Issue the LOAD macro instruction for the global resource serialization service routines located in the ISGGSRV0 load module. Call ISGGSRVI to place the entry point addresses of the various service routines into the GVTX.	ISGNASIM ISGGSRVI	
10 Issue the LOAD macro instruction to load ISGNPGIM into CSA storage. Build a parameter list for the event notification facility (ENF) and invoke the ENF (IEFENFFX) to establish ISGNPGIM as the "listener" for the availability of timer and console services. (Timer and console services are needed in order to process global resource requests.) The event notification facility invokes ISGNPGIM when timer and console facilities become available. The parameter list indicates: • A listen request. • ISGNPGIM as the listener. • An event code indicating timer and console services availability as the event. If ISGNPGIM is not established as the "listener" (the return code from IEFENFFX is not zero) then issue an ABEND macro instruction with a system completion code of X'09A'. (A specific reason code identifies the nature of the error.); if ISGNPGIM is established at the "listener", then continue with step 11.	ISGNASIM IEFENFFX	
11 Using the ECB passed in the input parameter list, issue the POST macro instruction in order to notify global resource serialization address space creation, ISGNTASC, that the address space has been initialized. ISGNTASC can then finish its processing.	ISGNASIM	
12 Load the recovery routine required by the command processing modules and the message module. Then establish recovery for ISGNASIM. Specifically. • Load ISGCRCV and store the entry point address in the GVTX (GVTXCRCV). • Load ISGMSG00, message module, and store the entry point address in the GVTX (GVTXMSG). • Establish ISGCRCV as an ESTAE for ISGNASIM. If establishing the ESTAE was unsuccessful, issue an ABEND with a system completion code of X'09A' and a reason code identifying the specific nature of this error.		

Diagram 46. Global Resource Serialization Trigger Address Space Initialization (ISGNASIM) Part 9 of 12



Extended Description	Module	Label
13 Using the ECB in the GVT (GVTNECB), issue the WAIT macro instruction in order to wait until timer and console services are available. ISGNPGIM, established as the "listener" for the availability of timer and console services in step 10, posts this ECB when these services are available. Once ISGNPGIM posts GVTNECB, ISGNPGIM is no longer needed at "the listener" for the availability of timer and console services (they are now available). Invoke the event notification facility (IEFENFFX) to delete ISGNPGIM as a "listener" for these services. Upon return from IEFENFFX, issue the DELETE macro instruction to delete ISGNPGIM from storage.	ISGNASIM IEFENFFX	
14 Issue the ATTACH macro instruction for the global resource serialization wait for master scheduler initialization routine (ISGNWMSI), passing the address of the system address space initialization WAIT/POST routine (IEEMB883). If the attach is successful, ISGNWMSI, executing as a subtask, will invoke IEEMB883 in order to determine when master scheduler initialization completes. If the attach is unsuccessful, issue an ABEND with a system completion code of X'09A' and a reason code identifying the specific nature of the error.	ISGNASIM	
15 Issue the ATTACH macro instruction for the global resource serialization command router (ISGCMDR), passing the ECB, GVTXECB0. Issue the ATTACH macro instruction for the global resource serialization resource processor (ISGGRP00), passing the ECB, GVTXECB2. If either ATTACH is unsuccessful, issue an ABEND with a system completion code of X'09A' and a reason code identifying the specific nature of the error. If both are successful, issue a WAIT macro instruction for the ECB list containing GVTXECB1 and GVTXECB2. • GVTXECB0 indicates whether ISGCMDR finished processing normally or abnormally. • GVTXECB1 indicates whether ISGGRP00 finished its initialization successfully or unsuccessfully. • GVTXECB2 indicates whether ISGGRP00 finished its processing normally or abnormally.	ISGNASIM	
If GVTXECB1 and GVTXECB2 do not indicate that ISGGRP00 completed successfully, issue an ABEND with a system completion code of S'09A' and a reason code identifying the specific nature of the error. GVTXECB0 is checked in step 17.		
16 If the GRS system parameter was specified as JOIN (GVTJOIN = 1) or START (GVTSTART = 1), issue the ATTACH macro instruction, for the global resource serialization CTC ring processor (ISGBTC) passing the ECB, GVTXECB4. In addition, issue the ATTACH macro instruction for the global resource serialization option processor (ISGNGRSP) passing the ECB, GVTXECB5. Issue an ABEND with a system completion code of X'09A' and an appropriate reason code if either ATTACH is unsuccessful; otherwise, issue the WAIT macro instruction for the list of ECBs, GVTXECB3, GVTXECB4, and GVTXECB5. • GVTXECB3 indicates whether ISGBTC finished its initialization successfully or unsuccessfully. • GVTXECB4 indicates whether ISGBTC finished processing abnormally or normally. • GVTXECB5 indicates whether ISGNGRSP finished processing abnormally or normally.	ISGNASIM	
Issue an ABEND with a system completion code of X'09A' and an appropriate reason code if GVTXECB3 or GVTXECB4 indicate unsuccessful ISGBTC processing or if GVTXECB5 indicates unsuccessful ISGNGRSP processing.		

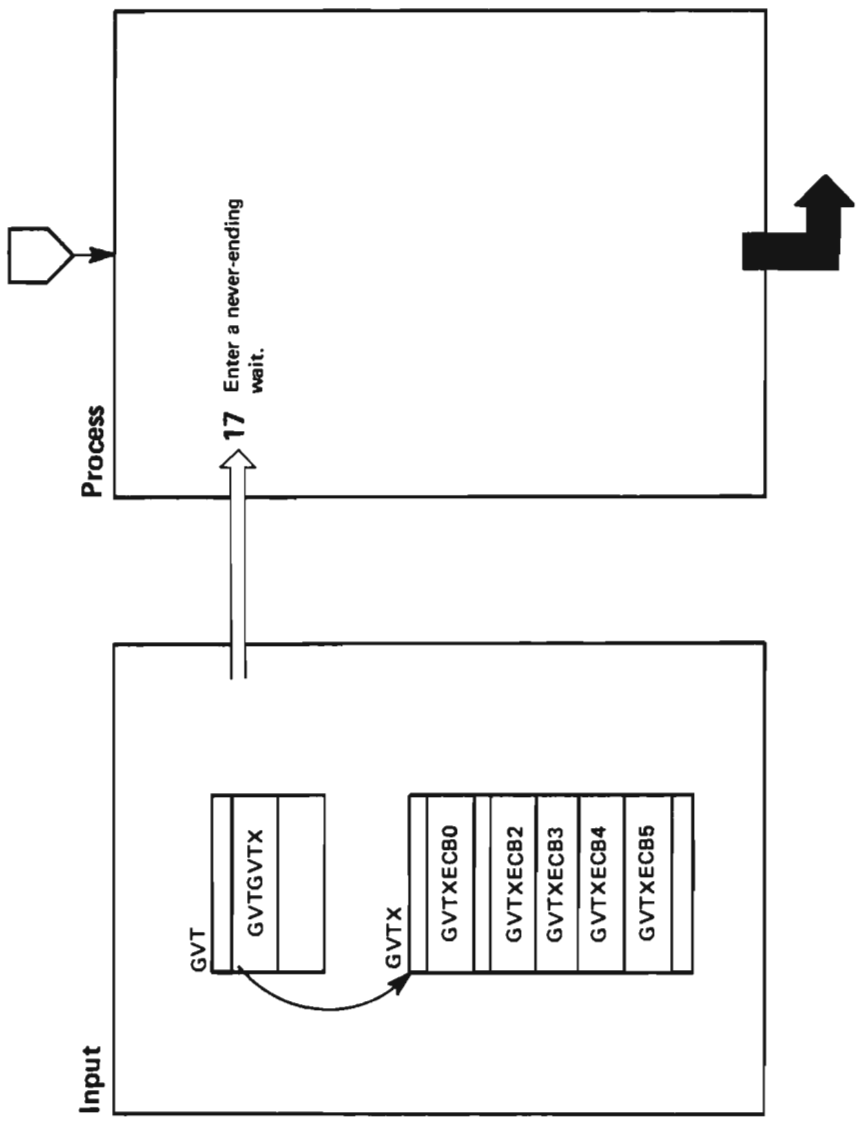


Diagram 46. Global Resource Serialization Trigger Address Space Initialization (ISGNASIN

Diagram 46. Global Resource Serialization Trigger Address Space Initialization (ISGNASIM) Part 12 of 12

Extended Description	Module	Label
17 In an infinite loop, perform the following processing:	ISGNASIM	ISGNASR2
a. Issue a WAIT macro instruction for four ECBs: GVTXECB0, GVTXECB2, GVTXECB4, and GVTXECB5. • GVTXECB0 indicates whether the processing of ISGCMDR ended normally or abnormally. • GVTXECB2 indicates whether the processing of ISGGRP00 ended normally or abnormally. • GVTXECB4 indicates whether the processing of ISGBTC ended normally or abnormally. • GVTXECB5 indicates whether the processing of ISGNGRSP ended normally or abnormally.		
b. When any of the four ECBs are posted, determine which ECB was posted. If GVTXECB0 was posted, check to see if ISGCMDR completed successfully. If not, issue a DETACH macro instruction to detach ISGCMDR, clear GVTXECB0, and issue the ATTACH macro instruction to reattach ISGCMDR as a subtask. If this reattach is unsuccessful, process the command request queue, terminating each requester, and finally issue an ABEND with system completion code of X'09A' and a reason code identifying the specific nature of the error. If the reattach is successful, check the processing status of the global resource processor, ISGGRP00. Check the ECB, GVTXECB2, to see if ISGGRP00 processed successfully. If ISGGRP00 processed successfully, issue the DETACH macro instruction to detach ISGGRP00 and release GQWA and GGSA storage. If ISGGRP00 was unsuccessful, issue an ABEND with system completion code of X'09A' and a reason code identifying the specific nature of the error. Check the processing status of the CTC ring processor task, ISGBTC, using the ECB, GVTXECB4. If ISGBTC was unsuccessful, issue an ABEND with a system completion code of X'09A' and a reason code identifying the specific nature of the error; otherwise, continue checking the processing status. Check the ECB, GVTXECB5, to determine the processing status of the global resource serialization option processor, ISGNGRSP. Issue an ABEND with system completion code of X'09A' and a reason code identifying the specific nature of the error if ISGNGRSP was unsuccessful; otherwise, continue the infinite loop by proceeding back to the beginning of this step to issue the WAIT macro instruction for the four ECB: GVTXECB0, GVTXECB3, GVTXECB4, and GVTXECB5. (Clear these ECBs before issuing the WAIT.)		

Recovery Processing

A recovery section of ISGNASIM, starting at entry point ISGNASRV provides the primary recovery. The ESTAE, ISGCRCV (established in step 12), initially intercepts the error and passes control to ISGNASRV. ISGNASRV determines where the error occurred and whether a retry is possible. If a retry is possible, ISGNASRV attempts the retry. Input to ISGNASRV is:

- The address of the GVT.
- The address of the GVTX.

Output from ISGNASRV is:

- Message ISG010E, if the GRS system parameter is JOIN or START (ISGMSG00 is used to print the message and the message has not already been issued.)
- A reinitialized ECB used to check processing status at the point where retry is to be attempted.
- An indication whether retry is to be attempted or not.

Then, ISGNASRV returns to its caller.

Notes:

- Each ABEND issue by ISGNASIM with the system completion code of X'09A' has a unique reason code associated with it. The reason code indicates the specific error that occurred in ISGNASIM.
- The never-ending wait keeps the global resource serialization address space from termination; it is in limited function start mode.

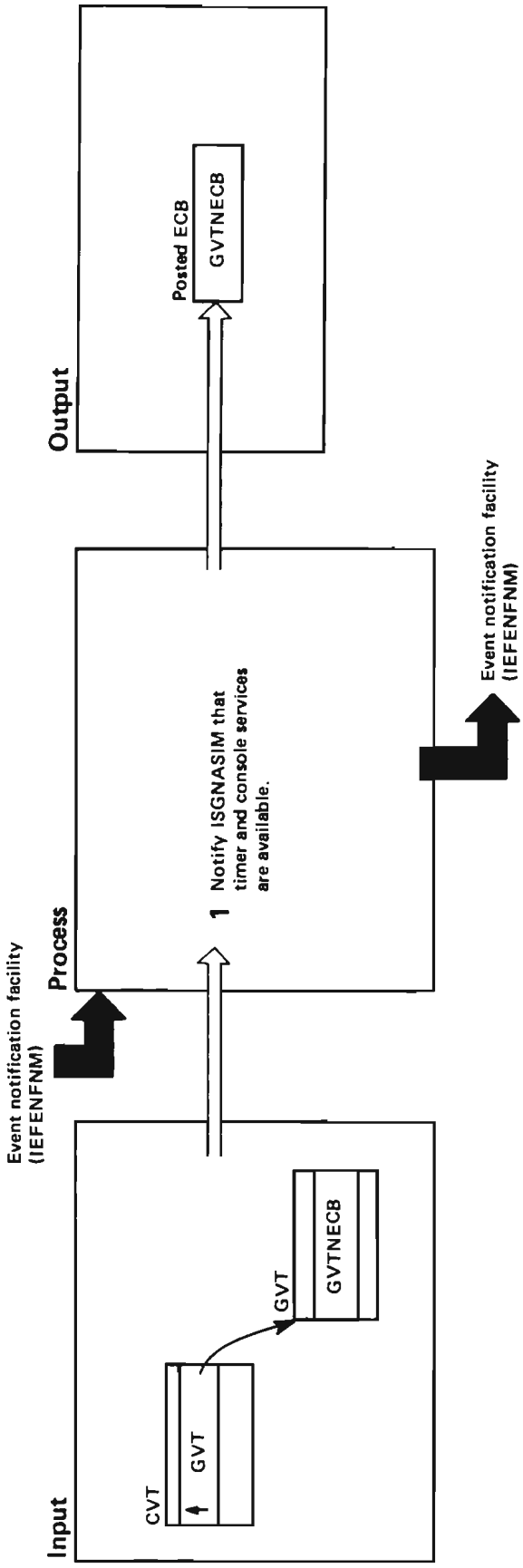


Diagram 47. Post Global Resource Serialization Initialization (ISGNPGIM Part 1 of 2

Diagram 47. Post Global Resource Serialization Initialization (ISGNPGIM Part 2 of 2

Extended Description	Module	Label
ISGNPGIM uses a cross memory post to notify ISGNASIM that timer and console services are available. The event notification facility invokes ISGNPGIM when the timer and console services are available. ISGNPGIM executes in key zero and in supervisor state.		
1 Issue the POST macro instruction passing the address of ISGNPRET the routine to be given control when an error condition results during POST processing. The ISGNPGIM post notifies the global resource serialization address space initialization module (ISGNASIM) that timer and console services are available.	ISGNPGIM	
Recovery Processing If the POST issued in step 1 fails, ISGNPRET issues a CALLRTM macro instruction to abend with a completion code of X'09A' the task representing ISGNASIM. ISGNASIM's recovery routine will subsequently get control to put the global resource serialization address space into a permanent wait state. (ISGNPRET is an internal entry point in ISGNPGIM, specified on the ERRET keyword of the POST macro instruction.)		

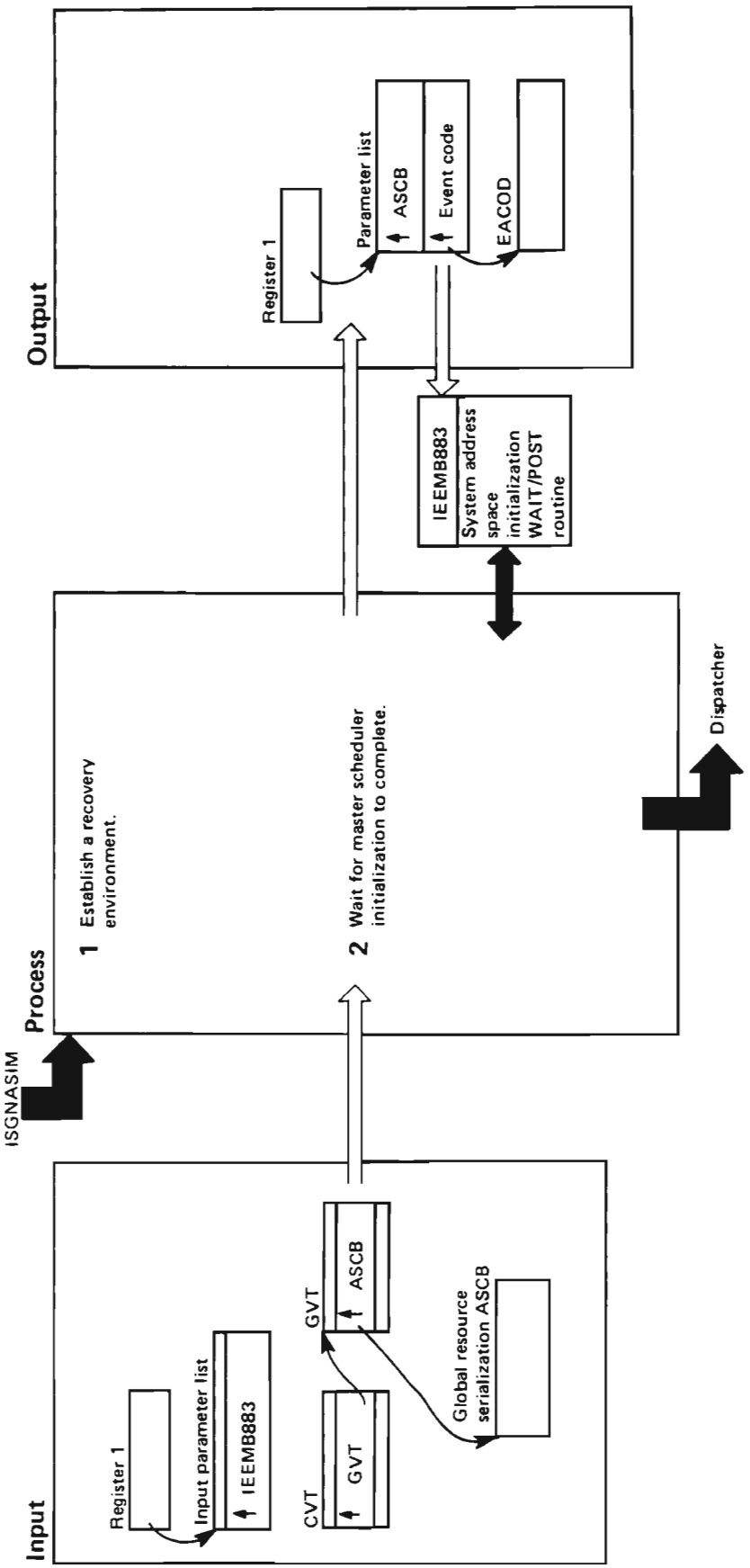


Diagram 48. Global Resource Serialization Wait for Master Scheduler Initialization (ISGNWMSI) Part 1 of 2

Diagram 48. Global Resource Serialization Wait for Master Scheduler Initialization (ISGNWMSI) Part 2 of 2

Extended Description	Module	Label
ISGNWMSI waits for master scheduler initialization to complete. Global resource serialization address space initialization (ISGNASIM) attaches ISGNWMSI to run in key 0 and in supervisor state, passing as input the address of the system address space initialization WAIT/POST routine (IEEMB883), the address of the global resource serialization ASCB, and address of the ECB IEEMB883 uses to post completion of master scheduler initialization.		
1 Establish an ESTAE, ISGCRCV, so that a dump can be taken if IEEMB883 fails in its processing.	ISGNWMSI	
2 Create the input parameter list for IEEMB883. The input parameter list contains: • The address of the global resource serialization ASCB. • The address of an event code; the event code indicates that ISGNWMSI is waiting for master scheduler initialization to complete.	ISGNWMSI	
Invoke IEEMB883, passing the input parameter list. When IEEMB883 returns, examine the return code.	IEEMB883	
• If the return code is not zero, issue an ABEND with system completion code of X'09A' and a reason code indicating the IEEMB883 was unsuccessful. • If the return code is zero, indicating that IEEMB883 was successful, cancel the ESTAE recovery environment and return to the dispatcher as a completed task.	ISGNWMSI	

Recovery Processing

The ESTAE, ISGCRCV takes a dump when an error occurs while waiting for master scheduler initialization to complete.

Diagram 49. Global Resource Serialization Option Processor (ISGNGRSP) Part 1 of 10

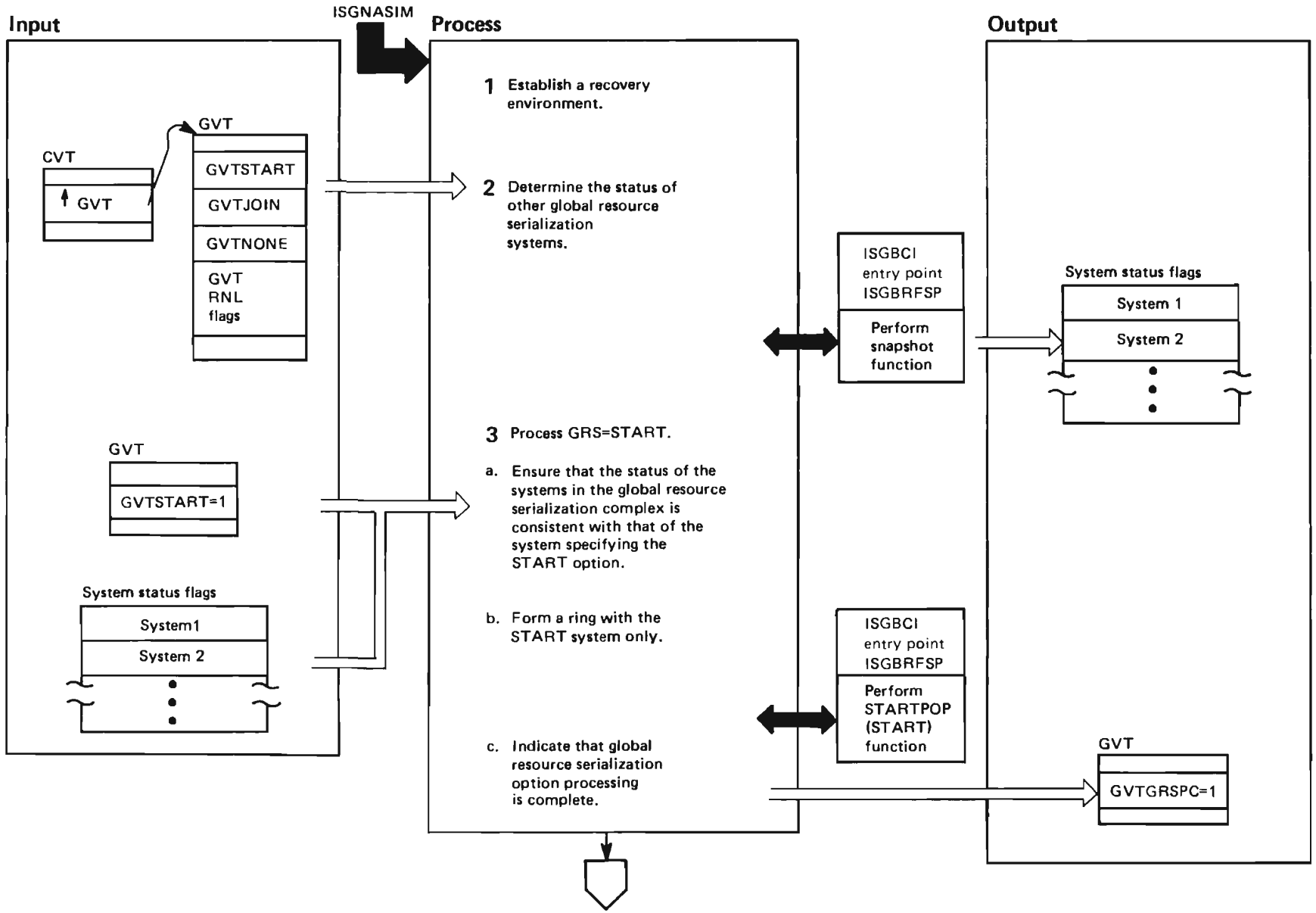


Diagram 49. Global Resource Serialization Option Processor (ISGNGRSP) Part 2 of 10

Extended Description	Module	Label
ISGNGRSP processes the system parameter GRS=START or GRS=JOIN as part of the overall process of global resource serialization address space initialization. ISGNASIM attaches ISGNGRSP. • If GRS=START, then before this system can process global requests for resources, ISGNGRSP verifies that no other rings exist and that no error flags have been set. • If GRS=JOIN, ISGNGRSP invokes global resource serialization queue merge (ISGCQMRG) to get a copy of existing global resources that are held before permitting global requests from this system to be processed. If errors occur during ISGNGRSP processing or if the GRS system parameter is inconsistent or invalid, ISGNGRSP prompts the operator for a new GRS option. • If the operator responds with GRS=NONE, ISGNGRSP invokes the STARTPOP (NONE) function (step 7). • If the operator responds with GRS=START or JOIN, ISGNGRSP again attempts option processing (step 2). After ISGNGRSP finishes its processing, control returns to the dispatcher to end the task, causing an ECB to be posted. ISGNASIM examines the ECB in order to determine whether or not to continue initializing the global resource serialization address space.		
1 Identify module, ISGCRCV, as the ESTAE routine.	ISGNGRSP	
2 Steps 2-6 are performed until processing of GRS=START or GRS=JOIN is successful or until the operator switches the option to GRS=NONE (step 7). If an error occurs in steps 2-6 because of any calls to external routines, control passes to step 5 to issue an error message. Invoke ISGBCI, which invokes entry point ISGBRFSN for the snapshot function to determine the status of other global resource serialization systems. ISGBCI returns either information about other global resource serialization systems or an indication that there are none.	ISGBCI	ISGBRFSN (SNAPSHOT)
If other systems exist, use the information about them in subsequent steps to determine whether or not the specified option (START or JOIN) for the GRS system parameter can be processed. Use this information also to determine the names of any other active global resource serialization systems. If ISGBCI was unsuccessful, issue an ABEND with a completion code of X'09A' and a reason code identifying the snapshot failure. Also, issue message ISG007I to the operator indicating that the prompt option is to be used to restart the system or to reply "NONE". If ISGBCI was successful, analyze the status of the systems being started and of the systems that already exist.		
3 Check the GVT to see if the system parameter is GRS=START. If so, verify that start processing for the system still needs to be done. If it does, then continue with step 3a. If the system is started, then issue a message to the operator indicating that the system is started, cancel the ESTAE, and return to the caller. If GRS=JOIN was specified, proceed to step 4.	ISGNGRSP	
a. If any other global resource serialization system exists or any error flags were set during the snapshot function of ISGBCI, then GRS=START is inconsistent with this environment. Set up message ISG005I with the appropriate informational message option related to this condition and go to step 5.		
b. If GRS=START is consistent with the environment, invoke the ring processing/command interface to start ring processing activities (STARTPOP, START) so that global resource requests can be processed.	ISGBCI	ISGBCFSP (STARTPOP)
c. Indicate in the global resource serialization vector table (GVT) that global resource serialization option processing is complete (GVTGRSPC='1'). Prepare informational message ISG004I and proceed to step 5.		

Diagram 49. Global Resource Serialization Option Processor (ISGNGRSP) Part 3 of 10

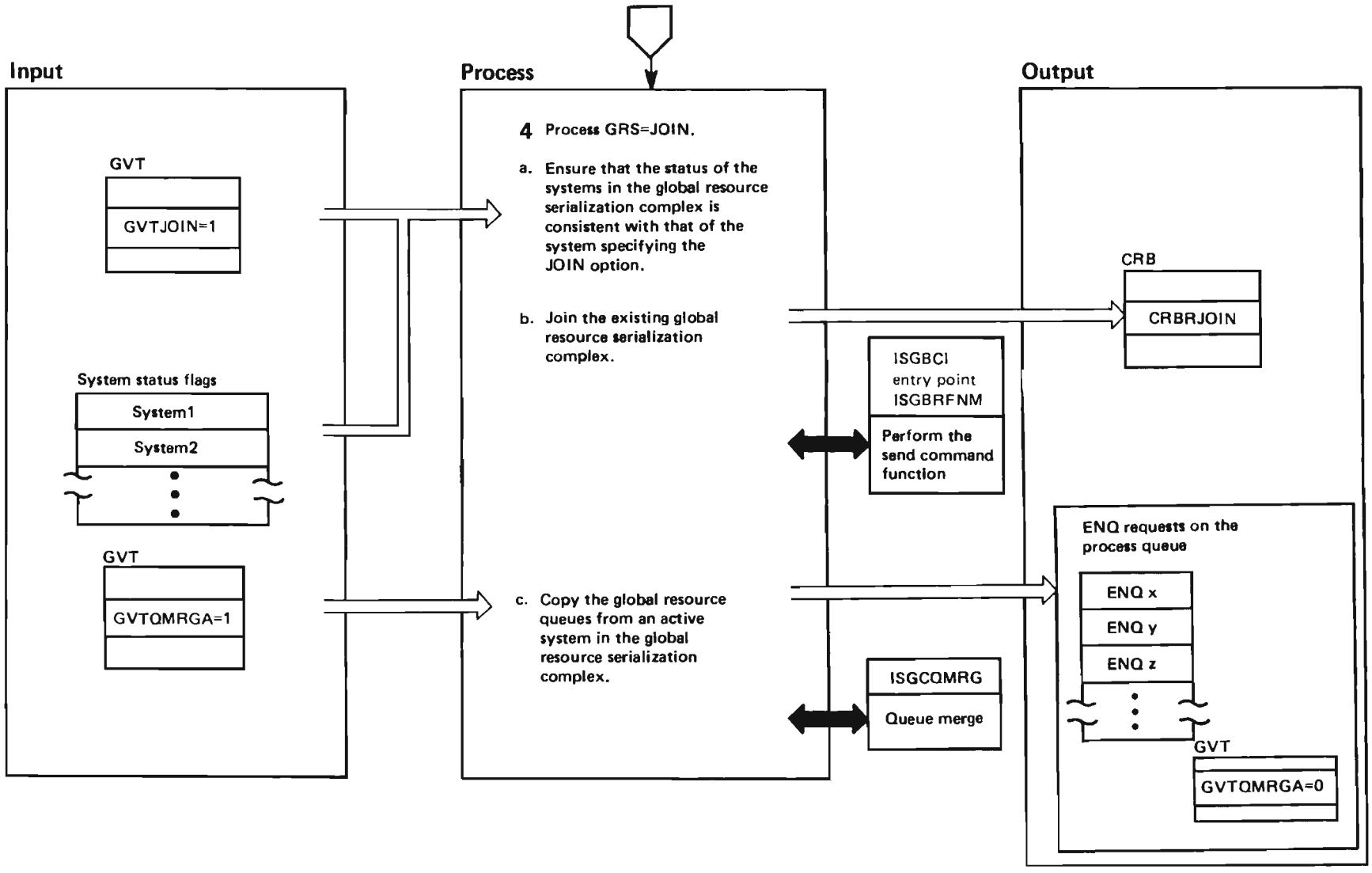


Diagram 49. Global Resource Serialization Option Processor (ISGNGRSP) Part 4 of 10

Extended Description	Module	Label
4 Verify that join processing for this system still needs to be done. If join processing needs to be done, continue with step 5a. If the system is already joined, then broadcast to all systems in the ring that a new system has joined, cancel the ESTAE, and return to the caller.	ISGNGRSP	
a. If no active global resource serialization system exists that this system can join or if any error flags were set during the snapshot function of ISGBCI, GRS=JOIN is inconsistent with this environment. Set up message ISG006I with the appropriate informational message option that is specifically related to this condition, then go to Step 5; otherwise continue with step 4b.		
b. Invoke ISGBCI to send the CRB (command request block) to an active global resource serialization system identified in the snapshot list. The CRB describes a command that asks another system to bring this system into the complex. Flags in the CRB indicate that the system is joining the global resource serialization complex. The system executing the command and bringing the joining system into the complex tells all other active global resource serialization systems about the joining system and sends a copy of the global resource queues to the joining system. If ISGBCI cannot send the command, issue an ABEND with completion code X'09A' and a reason code identifying the specific nature of the error.		
c. Link to ISGCQMRG to copy the global resources held by other systems in the complex. Use the copy to generate ENQ requests for those resources.		ISGCQMRG

Diagram 49. Global Resource Serialization Option Processor (ISGNGRSP) Part 5 of 10

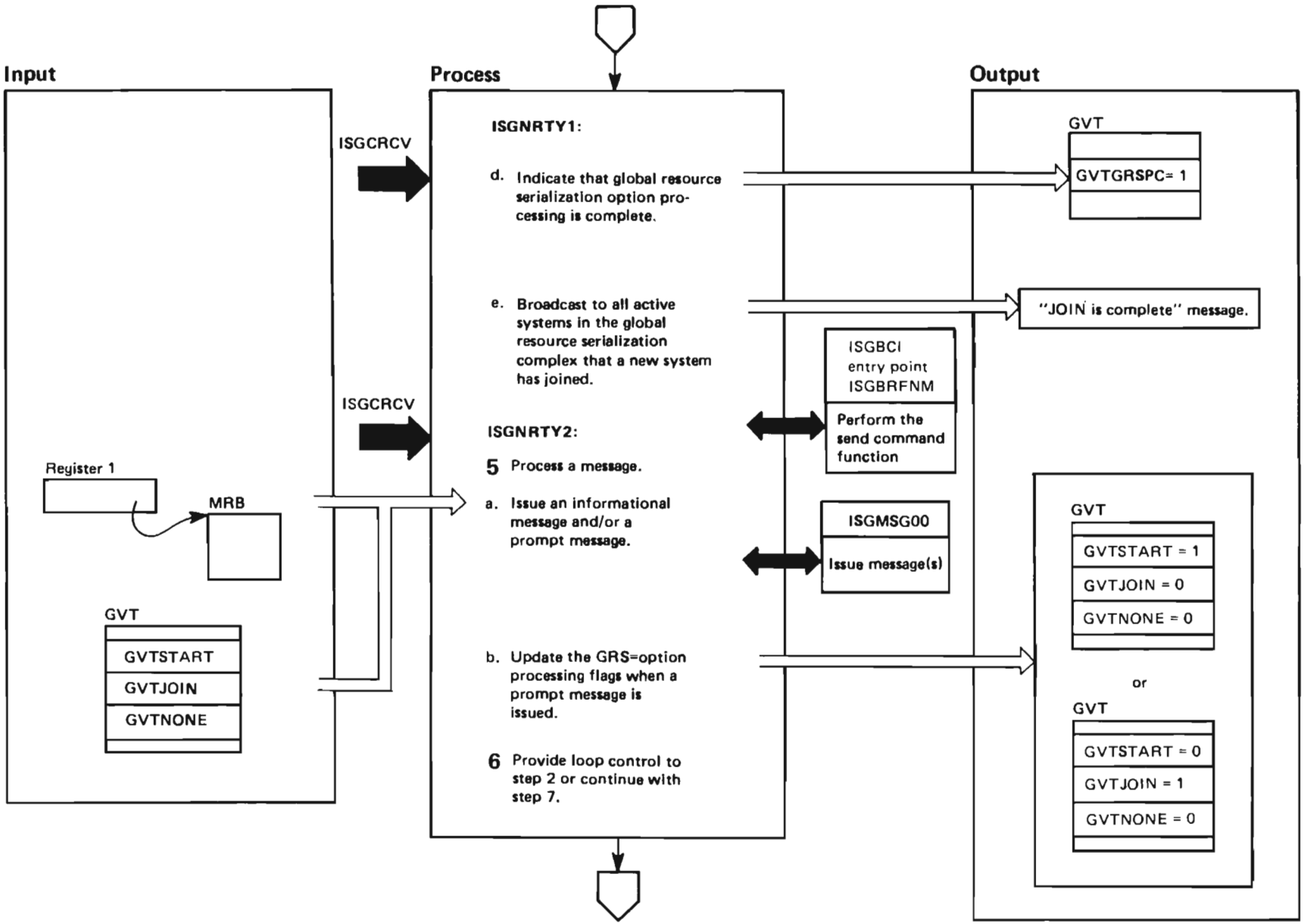


Diagram 49. Global Resource Serialization Option Processor (ISGNGRSP) Part 6 of 10

Extended Description	Module	Label
4 (continued)		
ISGNRTY1 is a retry point that receives control via ISGCRV through RTM. If an error occurs during the queue merge process (ISGCQMRG), and the queue update has completed, ISGNGRSP retries at this point.	ISGNGRSP	ISGNRTY1
d. Indicate in the GVT that global resource serialization option processing is complete (GVTGRSPC=1) for the joining system.	ISGBCI	ISGBCFNM (SENDCMD)
e. Issue a console message to tell all active global resource serialization systems that a new system has joined the global resource serialization complex. Issue an ABEND with completion code X'09A' if broadcasting the message is unsuccessful. The reason code specifically identifies the error.		
5 ISGNRTY2 is a retry entry point that receives control via ISGCRCV through RTM. If an error occurred during global resource serialization option processing, ISGNGRSP will retry at this point to issue a message about the error that occurred. ISGNRTY2 also is the place in ISGNGRSP whera all messages that were set up in earlier steps are issued.		
a. To issue an informational and/or prompt message to the operator, set up the MRB (message request block) describing the message(s). Invoke the global resource serialization message module (ISGMSG00) to issue the message.	ISGNGRSP ISGMSG00	ISGNRTY2
b. If a prompt message asked the operator to respecify the GRS=option, set the correct option processing flag (GVTSTART=1, GVTJOIN=1, or GVTNONE=1) in the GVT, based on the response to the prompt.	ISGNGRSP	
6 If GRS option processing is complete (GRS=START or GRS=JOIN) was specified and the subsequent processing had no errors), continue with step 8. If GRS=NONE is specified, continue with step 7. Otherwise, return to step 2 to continue option processing.		

Diagram 49. Global Resource Serialization Option Processor (ISGNGRSP) Part 7 of 10

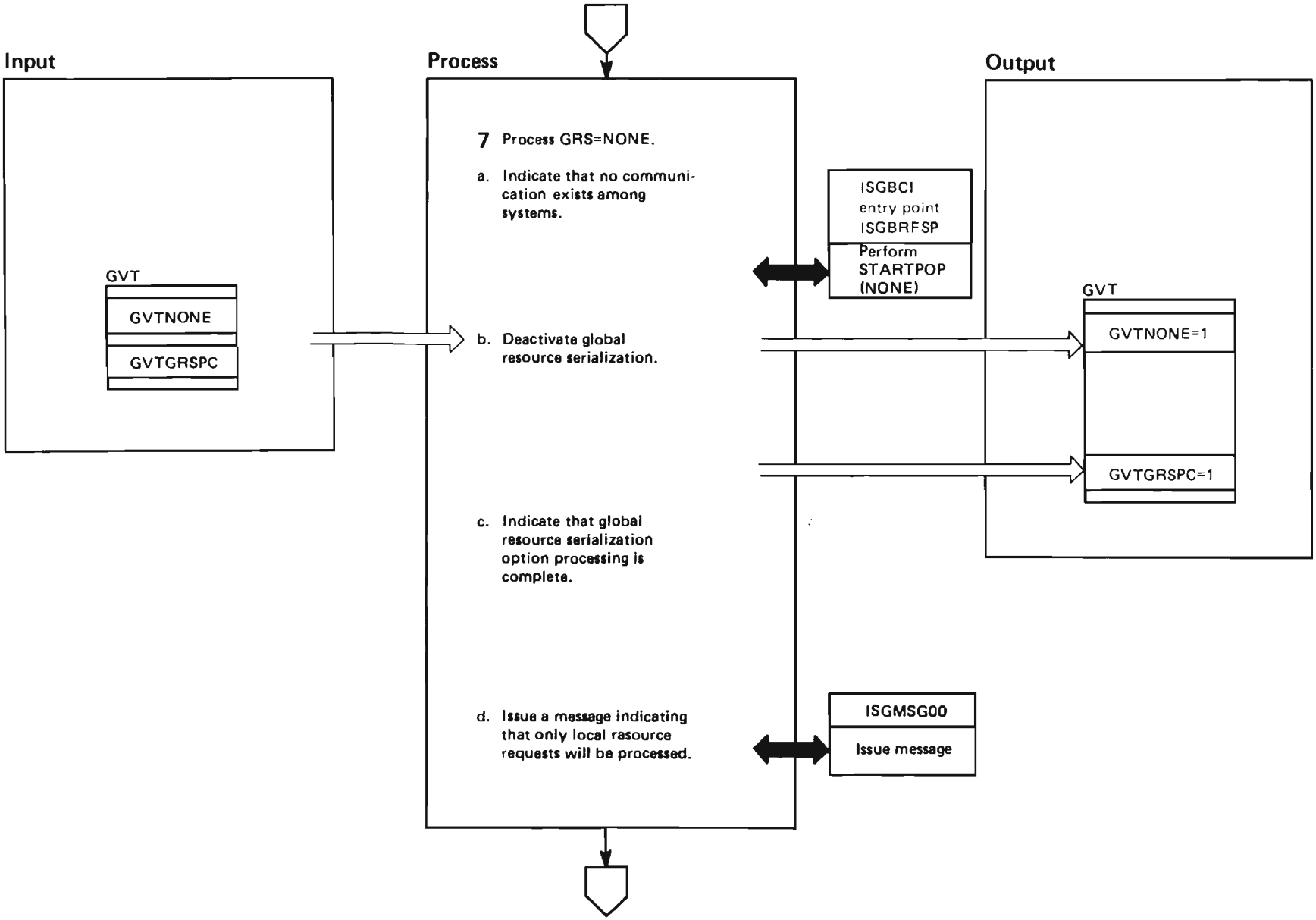


Diagram 49. Global Resource Serialization Option Processor (ISGNGRSP) Part 8 of 10

Extended Description	Module	Label
7 a. Invoke the STARTPOP (NONE) function of ISGBCI to tell the ring processing/command interface to stop ring processing activities. Communication with other global resource serialization systems is then not possible.	ISGBCI	STARTPOP
b. Deactivate global resource serialization by performing the following process steps:	ISGNGRSP	GETLL2
• Obtain the global resource serialization local lock using the SETLOCK macro instruction.		
• Set the GVTNONE flag on in the GVT.		
• POST the RB (GVTGRPRB) for ISGGRP00 using IEA0PT01 post routine. ISGGRP00 checks the GVTNONE flag. If it is set, all global requests on the process queue are treated as local requests. ISGGRP00 then exits.	IEA0PT01	
• Release the global resource serialization local lock using the SETLOCK macro instruction.	ISGNGRSP	FREELL2
c. Indicate in the GVT that global resource serialization option processing is complete (GVTGRSPC=1).		
d. Using ISGMSG00, inform the operator (with message ISG010E) that no global resource requests can be processed because the global resource serialization option has been switched to NONE.	ISGMSG00	

Diagram 49. Global Resource Serialization Option Processor (ISGNGRSP) Part 9 of 10

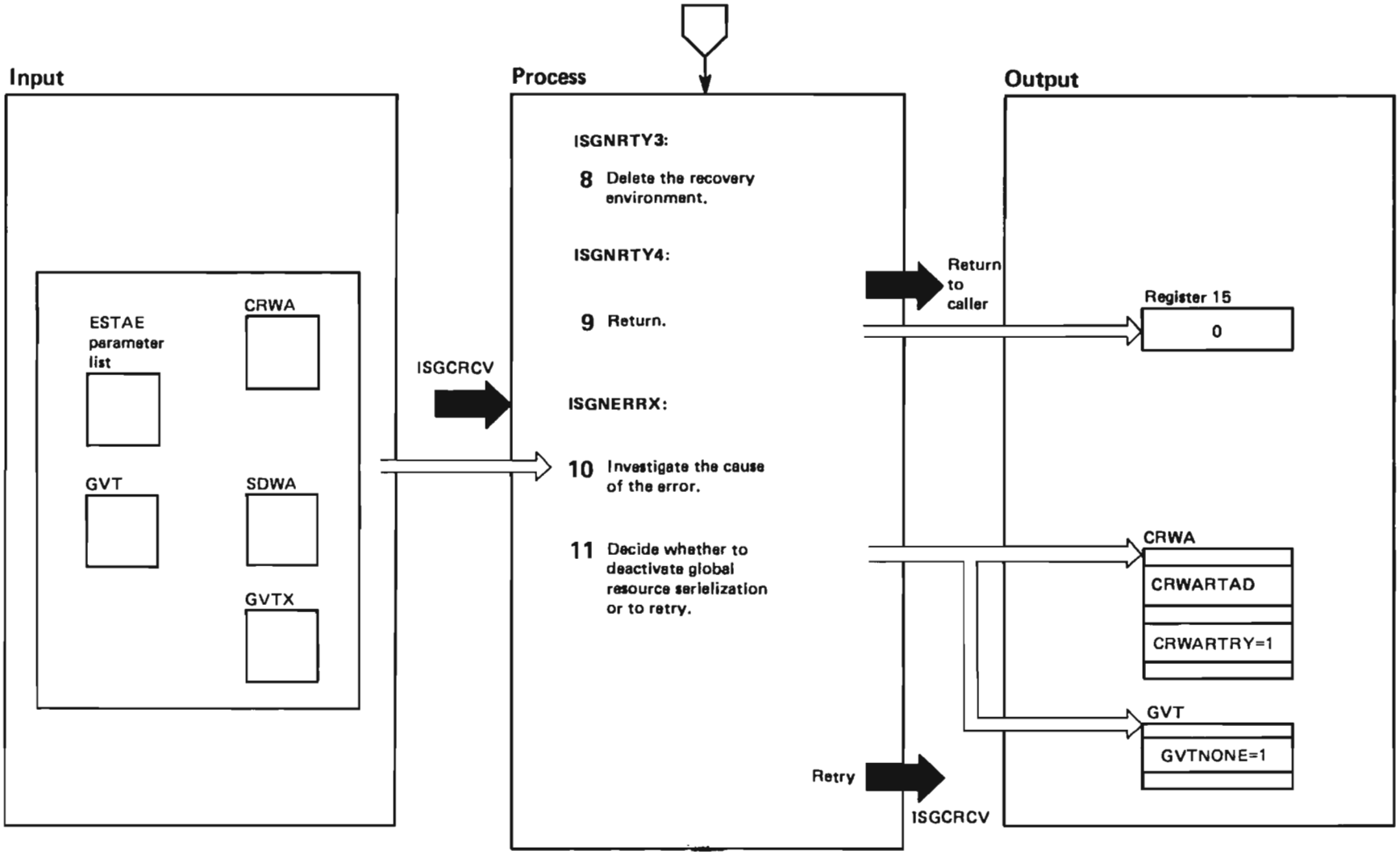


Diagram 49. Global Resource Serialization Option Processor (ISGNGRSP) Part 10 of 10

Extended Description	Module	Label
8 ISGNRTY3 is a retry entry point that receives control from ISGCRCV through RTM when an error has occurred after global resource serialization processing has completed. Cancel the ESTAE recovery environment for ISGNGRSP. Continue with step 9.	ISGNGRSP	ISGNRTY3
9 ISGNRTY4 is a retry entry point that receives control from ISGCRCV through RTM when an error occurred in trying to cancel the ESTAE for ISGNGRSP. Return to ISGNASIM with a return coda of zero.	ISGNGRSP	ISGNRTY4
Recovery Processing ISGNERRX is the special error exit for ISGNGRSP. It receives control from ISGCRCV through RTM. ISGNERRX investigates the cause of the error and decides what action to take. Input to ISGNERRX consists of the general purpose registers whose contents are as follows: Registers 0 and 1 – undefined. Register 2 – address of ESTAE parameter list. Register 3 – address of the command recovery work area (CRWA). Register 4 – base register for ISGCRCV. Register 5 – address of the SDWA or zero if it is invalid. Register 6 – address of the GVT or zero if it is invalid. Register 7 – address of the GVTX or zero if it is invalid. Register 8-13 – undefined. Register 14 – return address. Register 15 – address of ISGNERRX.		
10 Using the processing information checkpointed throughout ISGNGRSP processing in the command recovery work area (CRWA), determine the processing that was taking place at the time of the error.	ISGNGRSP	ISGNERRX

Extended Description	Module	Label
11 Decide if retry back into ISGNGRSP is feasible or if global resource serialization should be deactivated. If global resource serialization should ba deactivated, indicate no retry and perform the following steps:		
• Obtain the global resource serialization local lock using the SETLOCK macro instruction.	ISGNGRSP	GETLK1
• Set the GVTNONE flag on in the GVT.		
• Call IEA0PT01 to post the RB (GVTGRPRB) for ISGGRP00. ISGGRP00 checks the GVTNONE flag. If it is set on, all global resources on the process queue are treated as local requests. ISGGRP00 then exits.	IEA0PT01	
• Release the global resource serialization local lock using the SETLOCK macro instruction.	ISGNGRSP	FREELK1
For retry, set the correct retry entry point according to the information in the CRWA ans set up an information that is needed at the retry point.	ISGNGRSP	
Depending on the nature of the error causing ISGNERRX to get control, four retry points are possible from ISGNERRX into ISGNGRSP: • ISGNRTY1 – step 4d. • ISGNRTY2 – step 5. • ISGNRTY3 – step 8. • ISGNRTY4 – step 9.		

Diagram 50. Global Resource Serialization Queue Merge (ISGCQMRG) Part 1 of 6

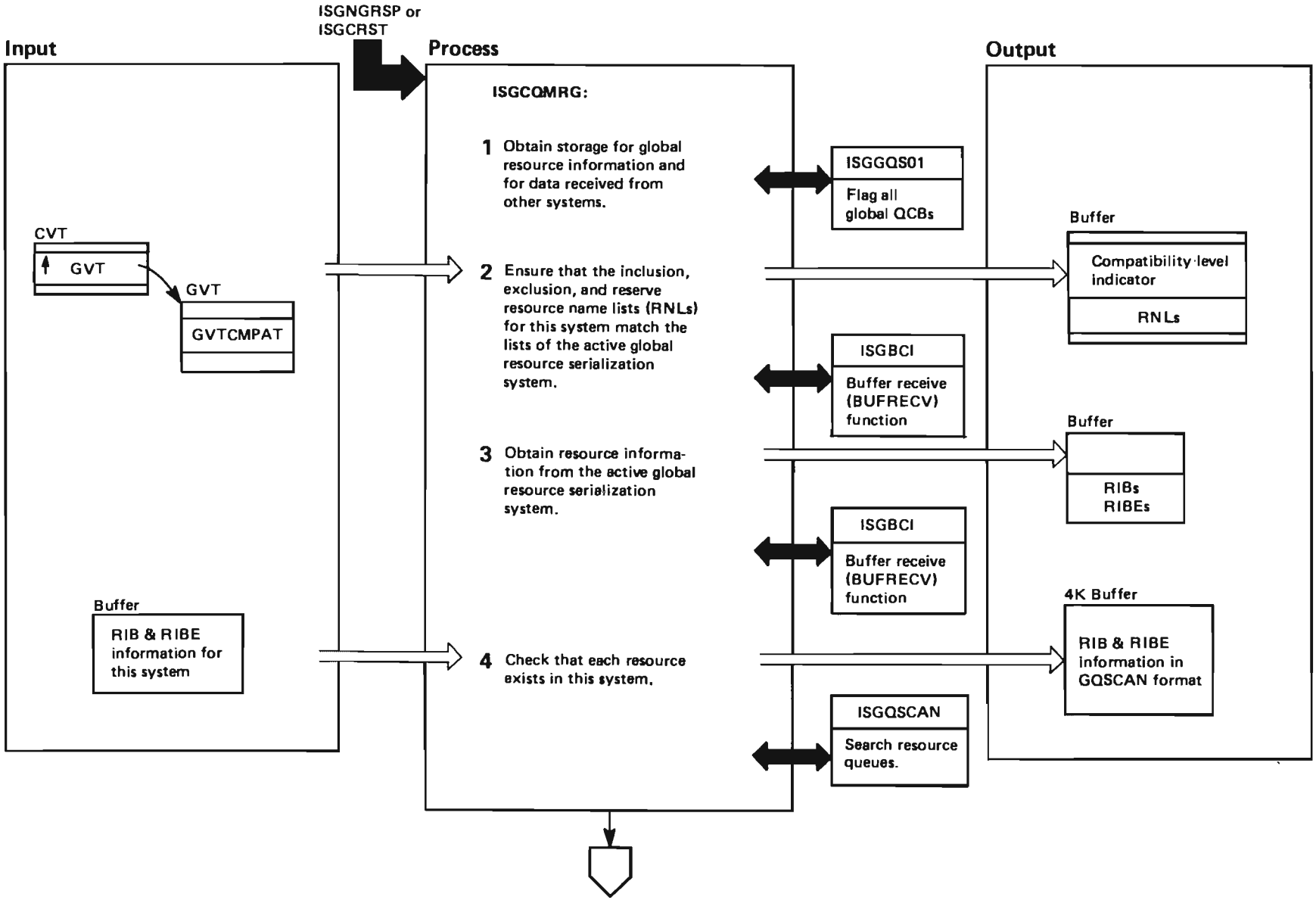


Diagram 50. Global Resource Serialization Queue Merge (ISGCQMRG) Part 2 of 6

Extended Description	Module	Label
ISGNGRSP (or ISGCRST) calls queue merge (ISGCQMRG) to verify that the inclusion, exclusion, and reserve lists in this global resource serialization system match those inclusion, exclusion, and reserve lists of the global resource serialization complex. ISGCQMRG also generates the ENQ or DEQ requests necessary so that this system's global resource queues match those of the complex. ISGCQMRG loads module ISGGQSRV to use the various global resource serialization service routines provided by ISGGQSRV. ISGCQMRG uses the various global resource serialization service routines of ISGGQSRV.		
1 After performing some initialization for subsequent processing and recovery, issue the GETMAIN macro instruction to obtain storage (64K bytes) from subpool 229. 60K of this storage is used as a buffer to hold the data sent from an active global resource serialization system. The remaining 4K is used as a work area when scanning the resource queues of this system. Invoke ISGGQS01, which sets a flag in all QCBs. The flag indicates that the QCB has not yet been processed by ISGCQMRG.	ISGCQMRG	
2 Invoke the buffer receive function of ISGBCI. The first buffer sent by the global resource serialization system contains the global resource serialization compatibility level indicator followed by the inclusion, exclusion, and conversion resource name lists (RNLs). To preserve data integrity, these lists must match the ones specified for this system. If the compatibility level indicator does not match that of the complex or if the lists do not match, issue an ABEND with completion code X'09A', and a reason code appropriate for the error condition. If the compatibility levels are the same and the lists match, continue with Step 3.	ISGBCI	BUFRECV
3 Invoke the buffer receive function of ISGBCI again to cause the active global resource serialization system to send the global resource queues to this system. ISGBCI does not return control until the data has been copied into the buffer area provided. Ensure that the resource information blocks (RIBs) and resource information block extensions (RIBEs) are constructed properly. If they are not, issue an ABEND with completion code of X'09A' and an appropriate reason code; otherwise, continue with Step 4.	ISGBCI	BUFRECV
4 Copy the qname and rname from the resource information block (RIB) for each resource and pass these names to ISGQSCAN using the GQSCAN macro instruction to obtain any information this system has describing requesters of the resource. (ISGQSCAN is the module that obtains this information.) If ISGQSCAN returns a return code indicating that no data was found, issue an X'09A' ABEND with an appropriate reason code.	ISGQSCAN	QSCAN

Diagram 50. Global Resource Serialization Queue Merge (ISGCQMRG) Part 3 of 6

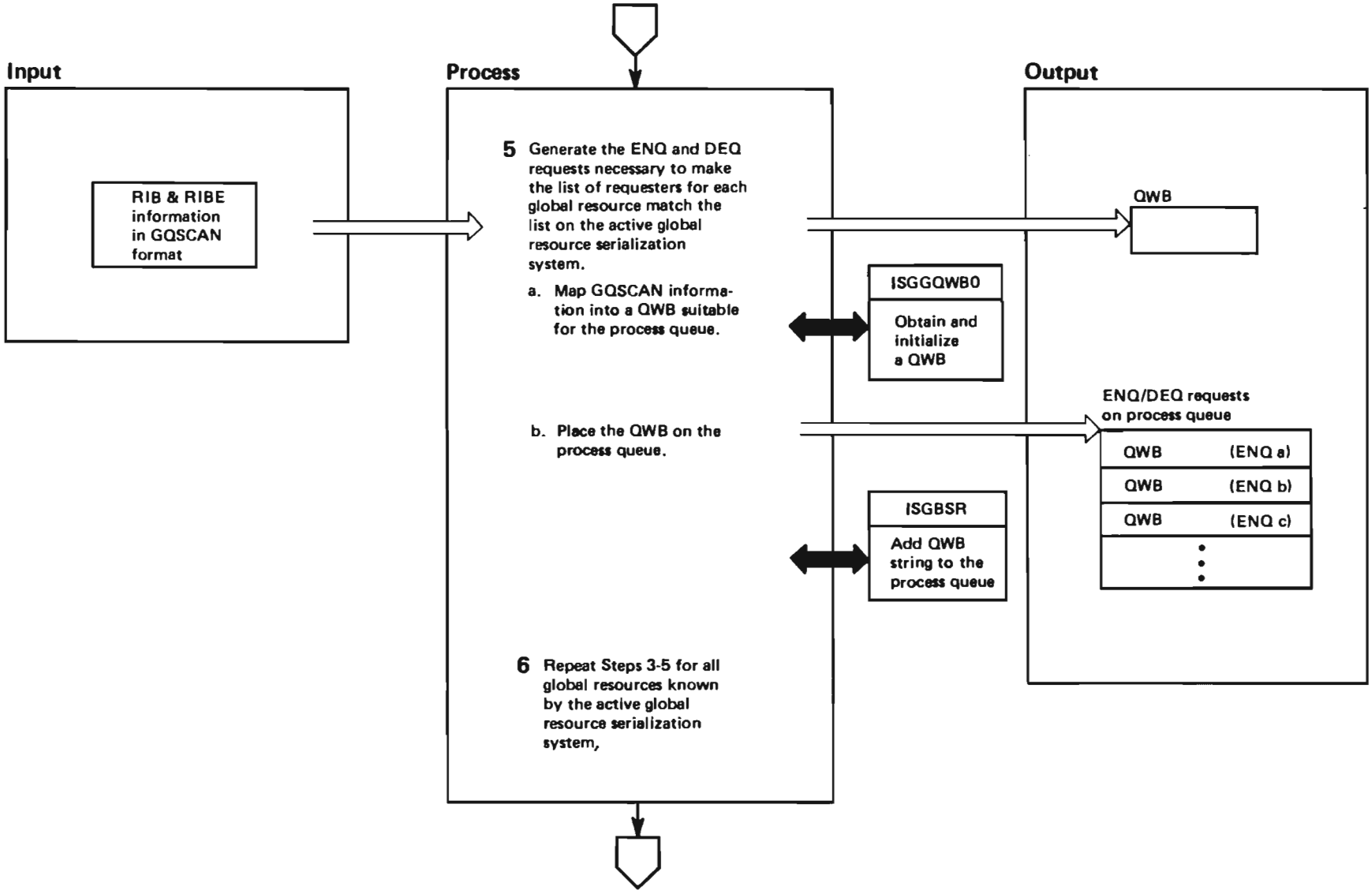


Diagram 50. Global Resource Serialization Queue Merge (ISGCQMRG) Part 4 of 6

Extended Description	Module	Label
5 Generate and queue the ENQ/DEQ requests necessary to make the system's list of requesters for global resources match the list on the active global resource serialization system.		
a. Invoke ISGGQWB0 to map the information in the GQSCAN buffers into a QWB that is suitable for the process queue.	ISGGQWB0	ISGGQWB2
b. Place this QWB on the process queue by invoking ISGBSR.	ISGBSR	ISGBBE
6 Repeat Steps 3-5 until the active global resource serialization system has sent all the information about each global resource on that system.	ISGCQMRG	

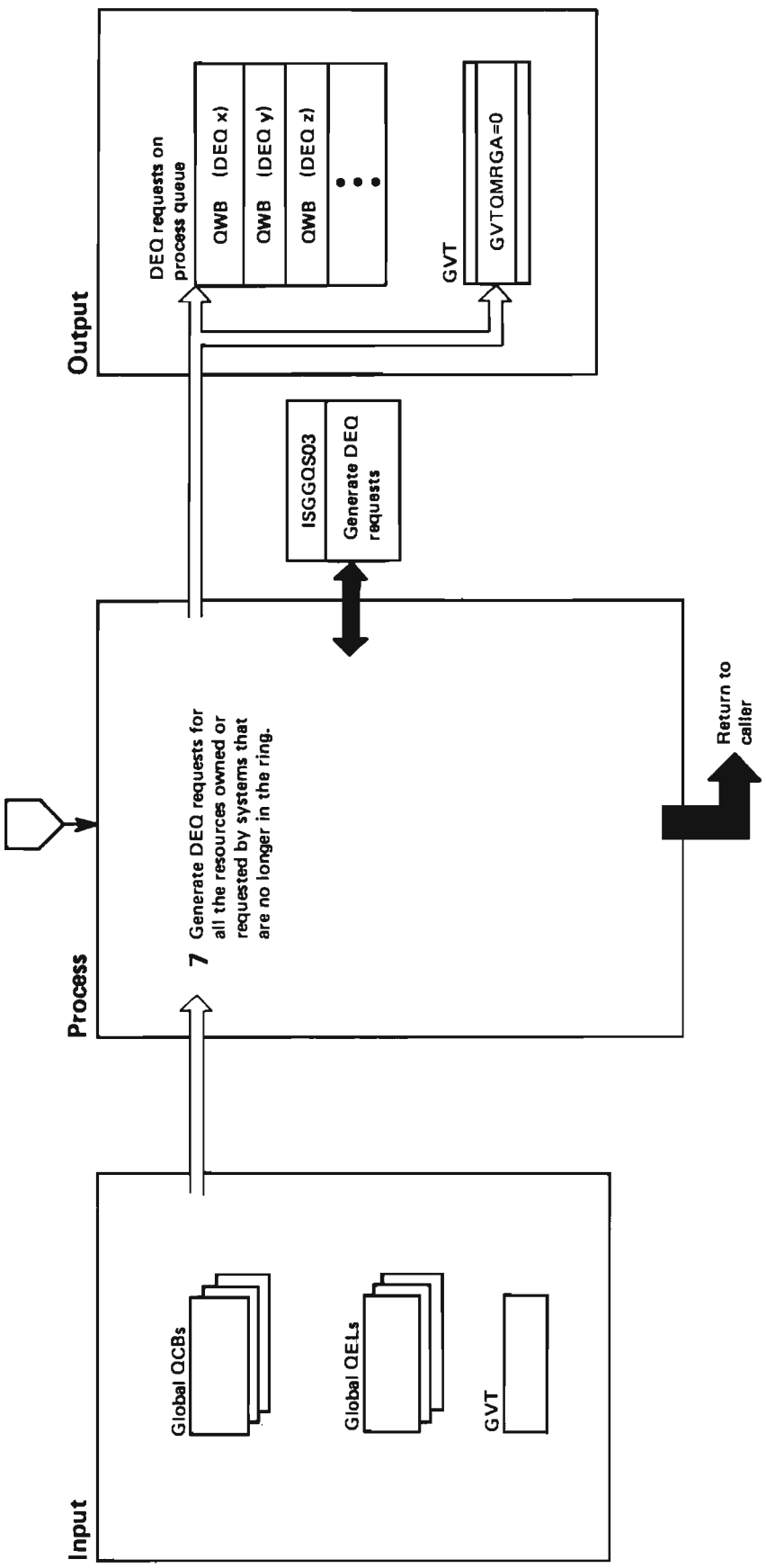


Diagram 50. Global Resource Serialization Queue Merge (ISGCQMRG) Part 5 of 6

Diagram 50. Global Resource Serialization Queue Merge (ISGCQMRG) Part 6 of 6

Extended Description	Module	Label
7 . If this system had been quiesced and is now being restarted into the complex, this system can show some resources owned by other systems that have actually released those resources. These resources were not in the list that was sent by the active global resource serialization system and must be removed from this system's global resource queues.		
Invoke ISGGQS03 to scan the QCB/QEL chains and generate DEQ requests for all requesters of resources not known to the other systems in the complex. The resource requests of this system now match those of the active global resource serialization system. Free the storage used for GQSCAN information and return to the caller with an indication in the GVT that the merge was successful (GVTQMRGA=0). If ISGGQS03 is unsuccessful, issue an ABEND with a completion code of X'09A' and a reason code identifying the specific nature of the error.	ISGGQS03 ISGCQMRG	

Recovery Processing:

When an error occurs while ISGCQMRG is executing, RTM calls ISGCRCV. ISGCRCV passes control to a special error exit routine in ISGCQMRG to perform the following:

- Release any storage obtained for QWBs
- Delete module ISGGQSRV
- Specify storage to be released by ISGCRCV

ISGCQMRG returns control to ISGCRCV to process the following recovery options:

- Retry if allowed
- Take a dump using default options
- Release dynamic area and buffer area obtained for GOSCAN and BUFRECV
- If retry is not allowed, ISGCRCV returns control to RTM to continue with termination

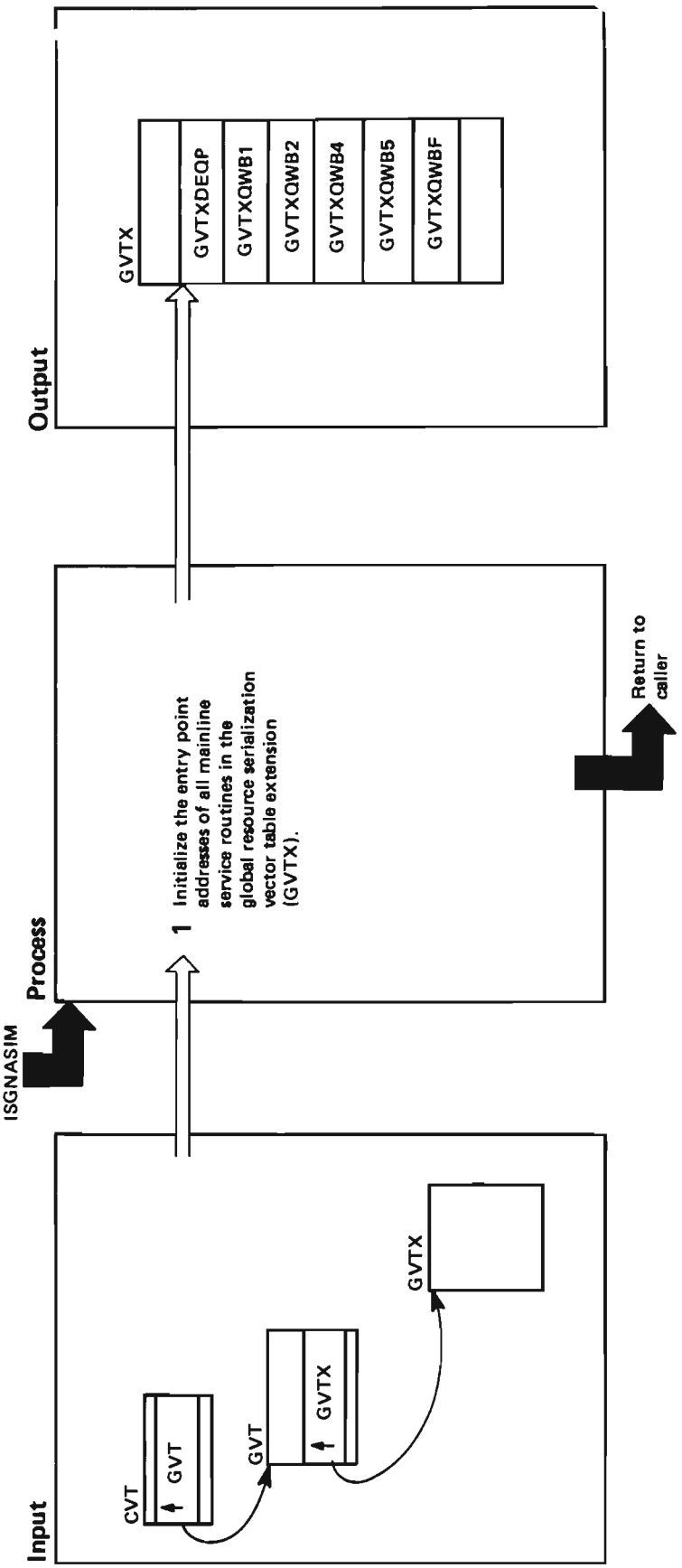


Diagram 51. Global Resource Serialization Mainline Service Routine Initialization (ISGGSRVI) Part 1 of 2

Diagram 51. Global Resource Serialization Mainline Service Routine Initialization (ISGGSRVI) Part 2 of 2

Extended Description	Module	Label
ISGGSRVI receives control from ISGNASIM in order to initialize all mainline service routine entry points.		
1 Set the following entry point addresses in the GVTX: • GVTXDEQP – entry point address of ISGGDEQP, the routine that purges local resources. • GVTXQWB1 – entry point address of ISGGQWB1, the routine that builds a queue work block (QWB) from a ring system authority (RSA) message. • GVTXQWB2 – entry point address of ISGGQWB2, the routine that builds a QWB from a resource information block (RIB) or a resource information block extent (RIBE). • GVTXQWB4 – entry point address of ISGGQWB4, the routine that builds a DEQ QWB from a queue element (QEL). • GVTXQWB5 – entry point address of ISGGQWB5, the routine that builds a SYSID, ASID, or TCB DEQ QWB. • GVTXQWBF – entry point address of ISGGQWBF, the routine that frees a QWB.	ISGGSRVI	
Note: ISGGQWB1, ISGGQWB2, ISGGQWB4, ISGGQWB5 and ISGGQWBF are all entry points of ISGGQWB0.		
Recovery Processing None.		

Diagram 52. Dumping Services Address Space RIM (IEAVNP57) Part 1 of 2

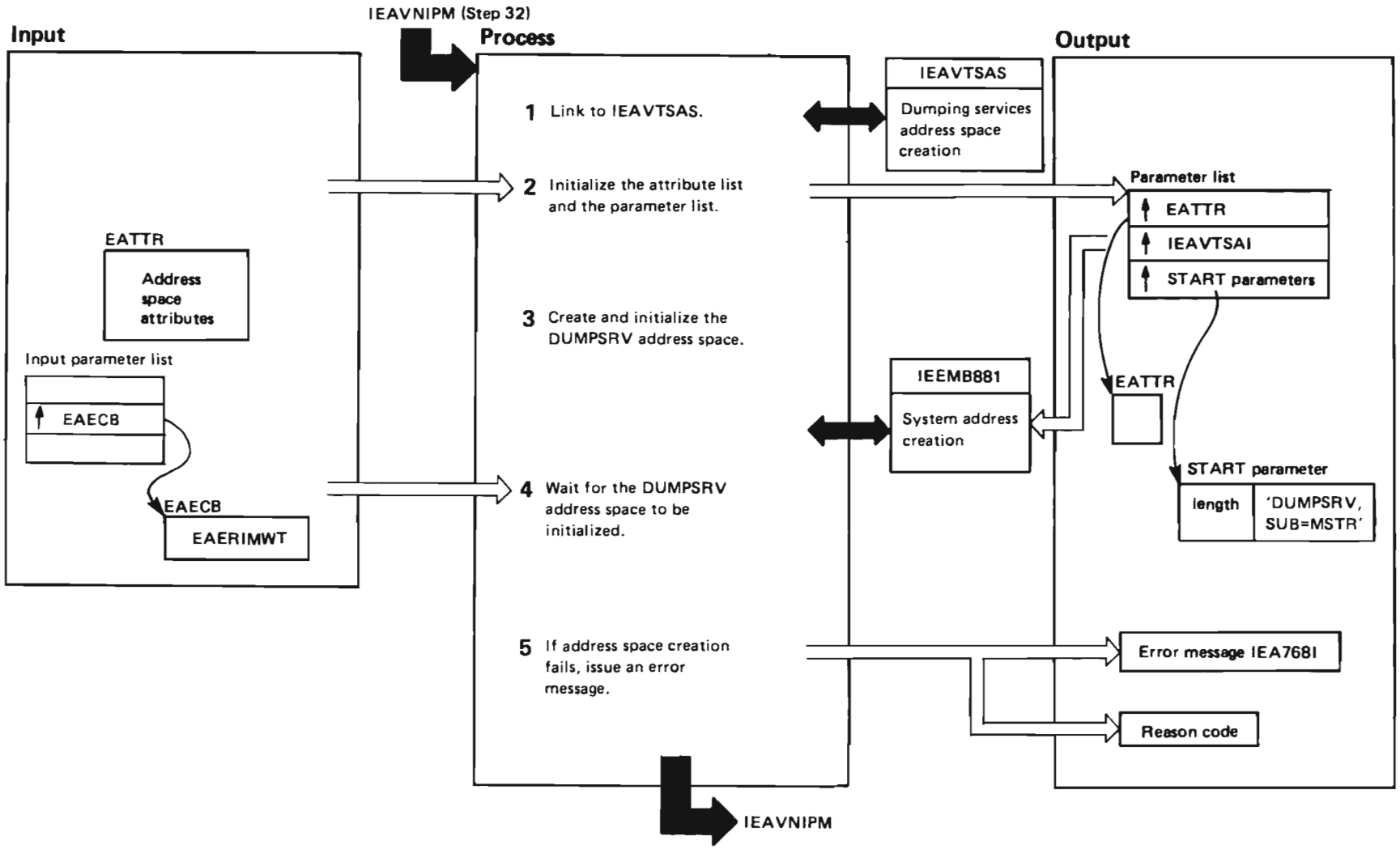


Diagram 52. Dumping Services Address Space RIM (IEAVNP57) Part 2 of 2

Extended Description	Module	Label
IEAVNIPM calls the dumping services RIM, IEAVNP57. IEAVNP57 calls IEAVTSAS to create and initialize the dumping services address space (DUMPSRV) by using the system address space create routine (IEEMB881). For an overview of dumping services address space initialization, see Figure 4-5.		
1 IEAVNP57 issues the LINK macro instruction to give IEAVTSAS control.	IEAVNP57	
2 IEAVTSAS initializes the DUMPSRV address space attribute list so that the address space is: • Exempt from system non-dispatchability. • Swappable. • Cancellable and forceable. • Allowed to be terminated. Set the START parameters to contain only the name of the dumping services address space (DUMPSRV) and set the address of dumping services address space initialization (IEAVTSAI) that will initialize DUMPSRV.		
3 IEAVTSAS issues the LINK macro instruction to give IEEMB881 control, passing the parameter list that contains the address of IEAVTSAI and the name of the DUMPSRV procedure in SYS1.PROCLIB, IEEMB881 creates the DUMPSRV address space and invokes IEAVTSAI to initialize it. IEEMB881 passes back a return code indicating whether or not the DUMPSRV address space was created.	IEEMB881	
4 If IEEMB881 is successful, wait on an ECB (EAECB) until it is posted by IEAVTSAI. After EAECB is posted, check the return code in the ECB (EAERIMWT) to see if initialization was successful. If it was, return to the caller. If initialization was not successful post the ECB (EAEASWT) to notify IEAVTSAI that it might terminate. IEAVTSAS then returns to IEAVNP57.	IEAVTSAS	
5 If IEEMB881 successfully created the DUMPSRV address space, but IEAVTSAI was unsuccessful in initializing the address space, issue error message IEA7681 with a reason code of 2. If IEEMB881 was unsuccessful in creating the DUMPSRV address space, issue the same message, IEA7681, but with a reason code of 1. (Use the IEAPMNIP service routine to issue the error message in either case.) Return to the caller whether successful or unsuccessful.	IEAVNP57	
Recovery Processing None.		

Diagram 53. Dumping Services Address Space Initialization (IEAVTSAI) Part 1 of 2

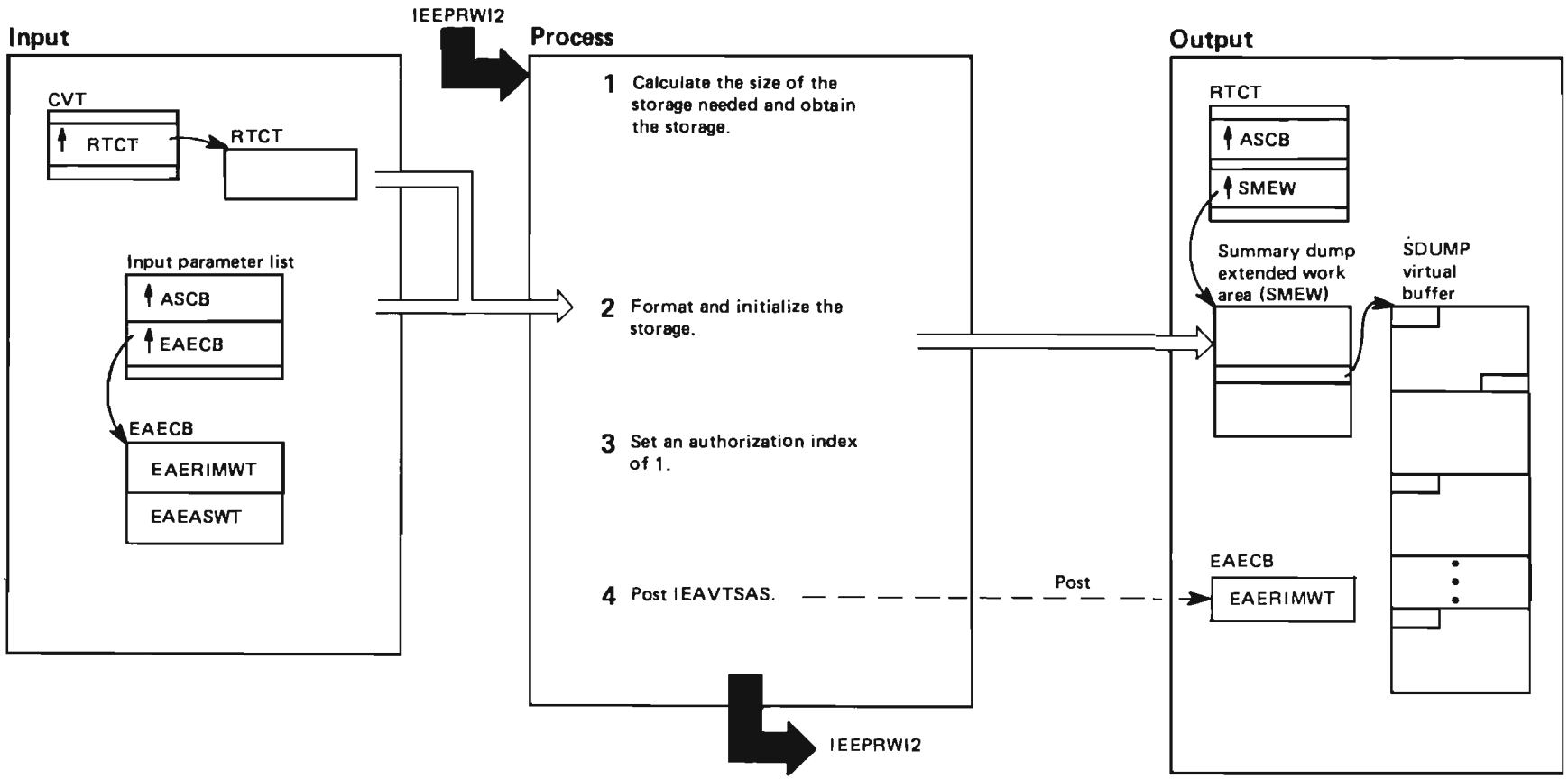


Diagram 53. Dumping Services Address Space Initialization (IEAVTSAI) Part 2 of 2

Extended Description	Module	Label
The started task control routine, IEEPRWI2, calls IEAVTSAI to initialize the dumping services address space (DUMPSRV) with a summary dump extended work area (SMEW) and a SDUMP virtual buffer. IEAVTSAI is an example of model system address space initialization		
1 Calculate the size of storage needed for a SMEW and the SDUMP virtual buffer. Issue the GETMAIN macro instruction to acquire the storage from subpool 229. If the GETMAIN is unsuccessful, post the input ECB (EAERIMWT) with a failure return code to indicate to IEAVTSAS that DUMPSRV address space initialization is unsuccessful.	IEAVTSAI	
Then wait for IEAVTSAS to POST the ECB (EAEASWT). This action ensures that the ECB (EAERIMWT) is not freed before IEAVTSAS has a chance to examine it. Then set up a return code of 4 to notify the caller, IEEPRWI2, that it should perform cleanup processing for the DUMPSRV address space and return to the caller.		
If the GETMAIN is successful, continue with step 2.		
2 Format the SDUMP virtual storage buffer and a SMEW from the acquired storage. Save the SMEW's address in the RTCT and set the address of the SDUMP virtual storage buffer in the SMEW. Make the RTCT point to the DUMPSRV address space.	IEAVTSAI	
3 Set the authorization index for the DUMPSRV address space to 1 so that SDUMP has full authorization to use cross memory instructions.	IEAVTSAI	
4 Invoke the system's cross memory post function to tell IEAVTSAS (the dumping services address space creation routine) that the DUMPSRV address space is initialized. Issue a return code of zero and return to IEEPRWI2 to continue to initialize the DUMPSRV address space.	IEAVTSAI	
For information on the DUMPSRV job steptask (IEAVTDSV), see *System Logic Library*.		
Recovery Processing		
None.		

Diagram 54. Communications Task Initialization (IEAVNPA1) Part 1 of 8

```
IEAVNPA1 - MODULE DESCRIPTION


DESCRIPTIVE NAME: Communications Task RIM.

FUNCTION:
   Verifies "CON=" system parameter and passes control to the
   front end processing of the console member of Parmlib.

ENTRY POINT: IEAVNPA1

   PURPOSE: Verifies "CON=" System Parameter

   LINKAGE: CALL statement

   CALLERS: IEAVNIPM

   INPUT:
      PARMTAB contains a pointer to the suffix value for the
      CONSOLxx member of SYS1.PARMLIB.

   OUTPUT:
      Message to prompt the operator to respecify the system
      parameter, if a new CONSOLxx member is needed.

   EXIT NORMAL: Return to the caller

EXTERNAL REFERENCES:

   ROUTINES:
      IEAVN600 - Front end processor of the CONSOLxx member of
                 SYS1.PARMLIB.

   CONTROL BLOCKS:
      Common Name           Macro Name          Usage
      -----------           ----------          -----
      CVT                   CVT                 R
      Nip Vector Table      IHANVT              R
      Nip Parm Address
          Table Entry       IEAPPNIP            R
      Nip Parm List         IEAPMNIP            C
      BASEA                 IEEBASEA            M
      SNPL                  IEEZB807            C
      UCM                   IEECUCM             M

       KEY = C - Create, D - Delete, M - Modify, R - Read

TABLES: TRTABL
```

Diagram 54. Communications Task Initialization (IEAVNPA1) Part 2 of 8

IEAVNPA1 - MODULE OPERATION

Initialize the variables and get addressability to NVT, NIPPAREA and PARMTAB.

Get storage for Scheduler NIP Parameter List.

Check that a value was specified for the "CON" parameter and that it is two characters and alphanumeric. Also, determine if the user specified the list option.

If the "CON" specification is not valid, prompt the operator, using the NIP service, to respecify the value correctly.

Call IEAVN600, the front end processor of the console member in Parmlib, passing the suffix for the Parmlib member name and a flag to indicate that the user wanted the list option. Interrogate the return code and issue a message if needed.

Store the address of NIP hardcopy message buffer (NVTMBUP) into UCMNIPTR in UCM Event Indication List.

Diagram 54. Communications Task Initialization (IEAVNPA1) Part 3 of 8

```
IEAVNPA1 - DIAGNOSTIC AIDS


ENTRY POINT NAME: IEAVNPA1

MESSAGES:

             Issued via IEAPMNIP

    IEA187I - FORMAT OF THE CON SYSTEM PARAMETER IS NOT VALID.

    IEA193I - CONSOLXX NOT USABLE. text

      text - UNABLE TO SELECT MASTER. REASON=rc

          rc=1 - No full capability console is defined
                 in the CONSOLxx PARMLIB member.
          rc=2 - No valid full capability console is defined
                 in the CONSOLxx PARMLIB member.
          rc=3 - No online full capability console is defined
                 in the CONSOLxx PARMLIB member.

      text - NO VALID CONSOLE STATEMENTS FOUND.

      text - I/O ERROR OCCURRED.

    IEA332A - SPECIFY CON

ABEND CODES: None

WAIT STATE CODES: None

RETURN CODES:

    EXIT NORMAL:

      Register 15 contains 0

REGISTER CONTENTS ON ENTRY:

    Reg  0 and 1    - Irrelevant
    RNVT (Reg 2)    - Address of NIP Vector Table (NVT)
    Reg 3 - 12      - Irrelevant
    Reg 13          - Address of Save Area
    REXIT (Reg14)   - Return Address
    RENTRY (Reg15)  - Entry Point Address

REGISTER CONTENTS ON EXIT:

    EXIT NORMAL:

      Registers 0-14 restored to contents upon entry
```

Diagram 54. Communications Task Initialization (IEAVNPA1) Part 4 of 8

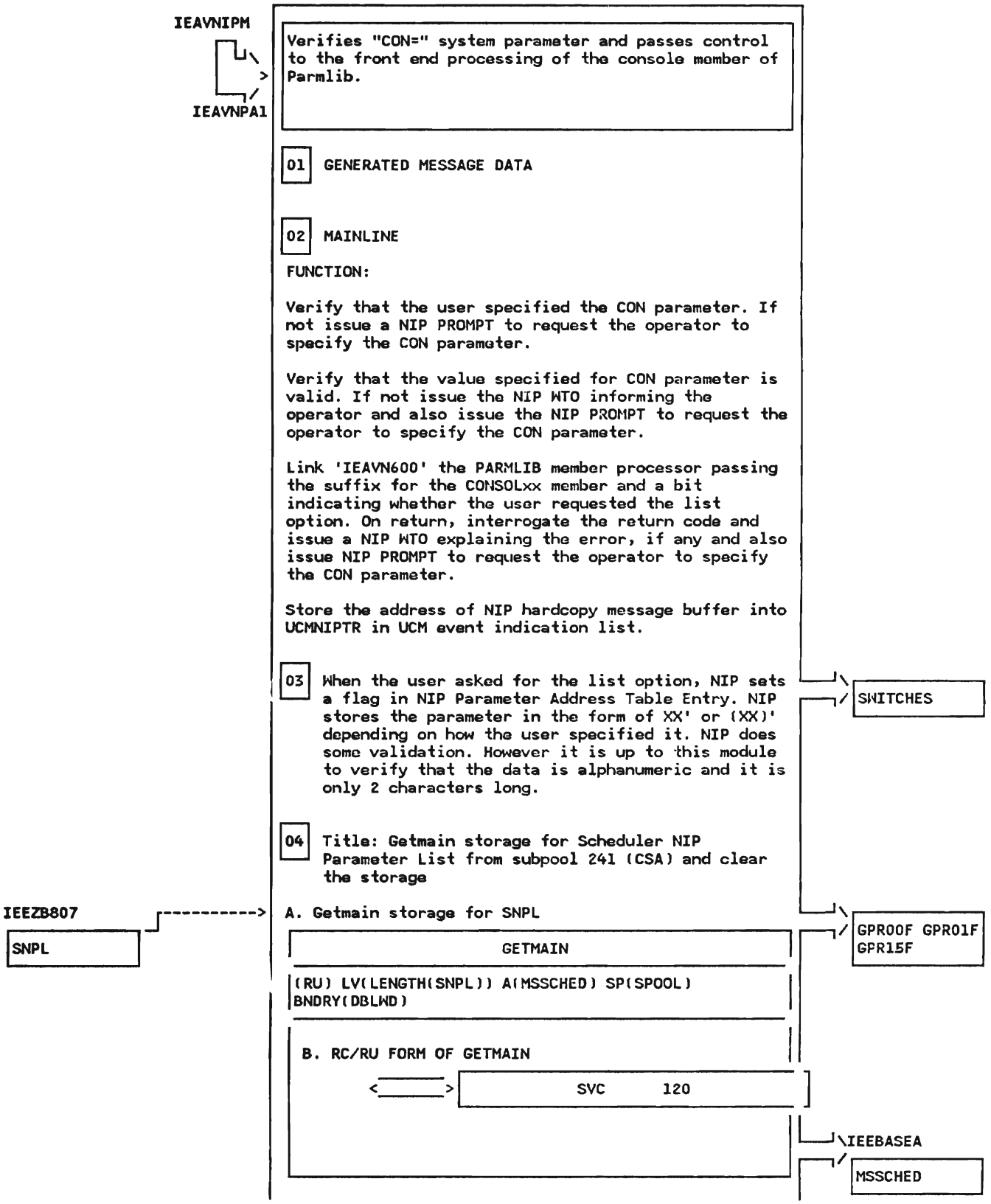


Diagram 54. Communications Task Initialization (IEAVNPA1) Part 5 of 8

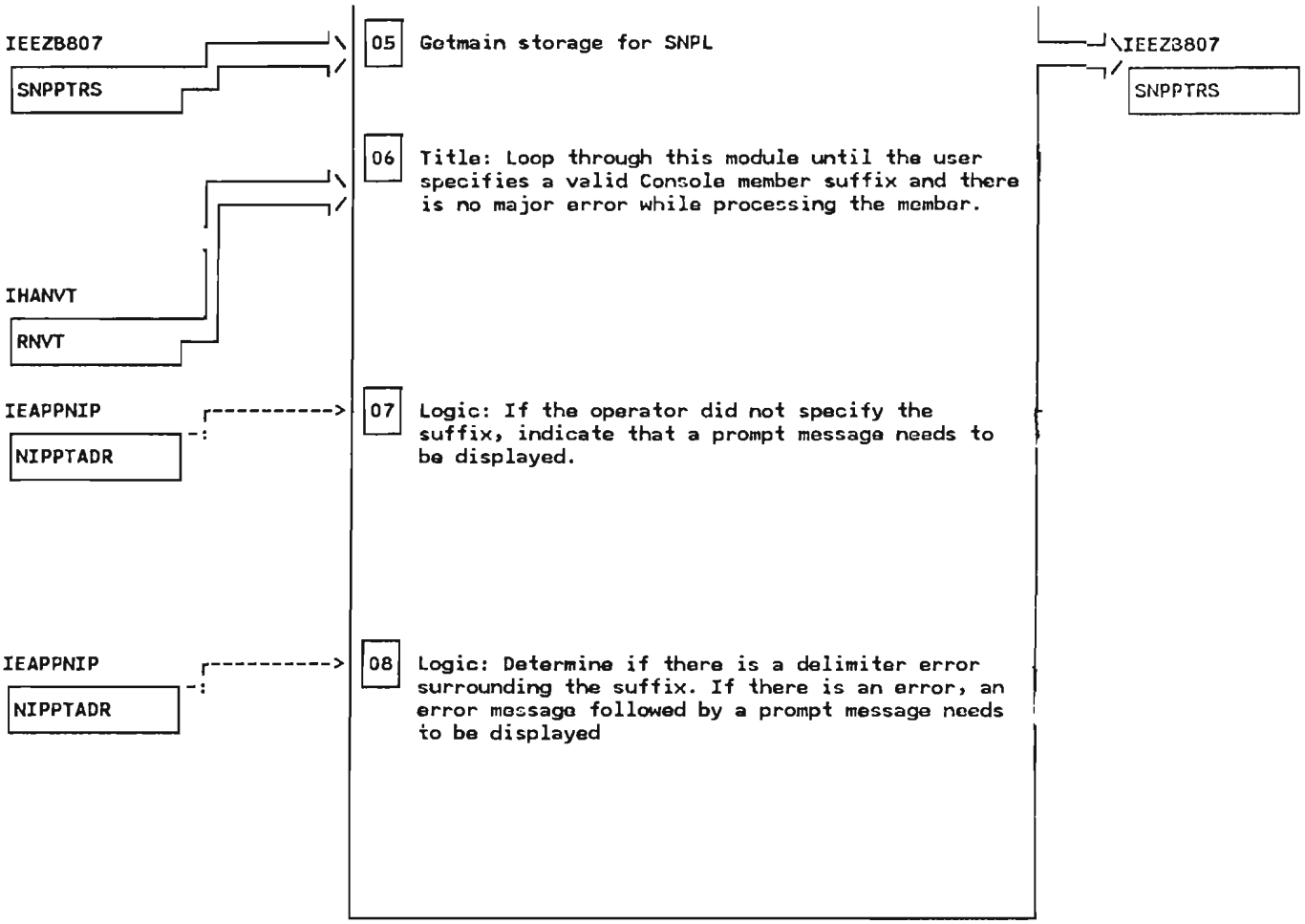


Diagram 54. Communications Task Initialization (IEAVNPA1) Part 6 of 8

IEAVNPA1 - Communications Task RIM.

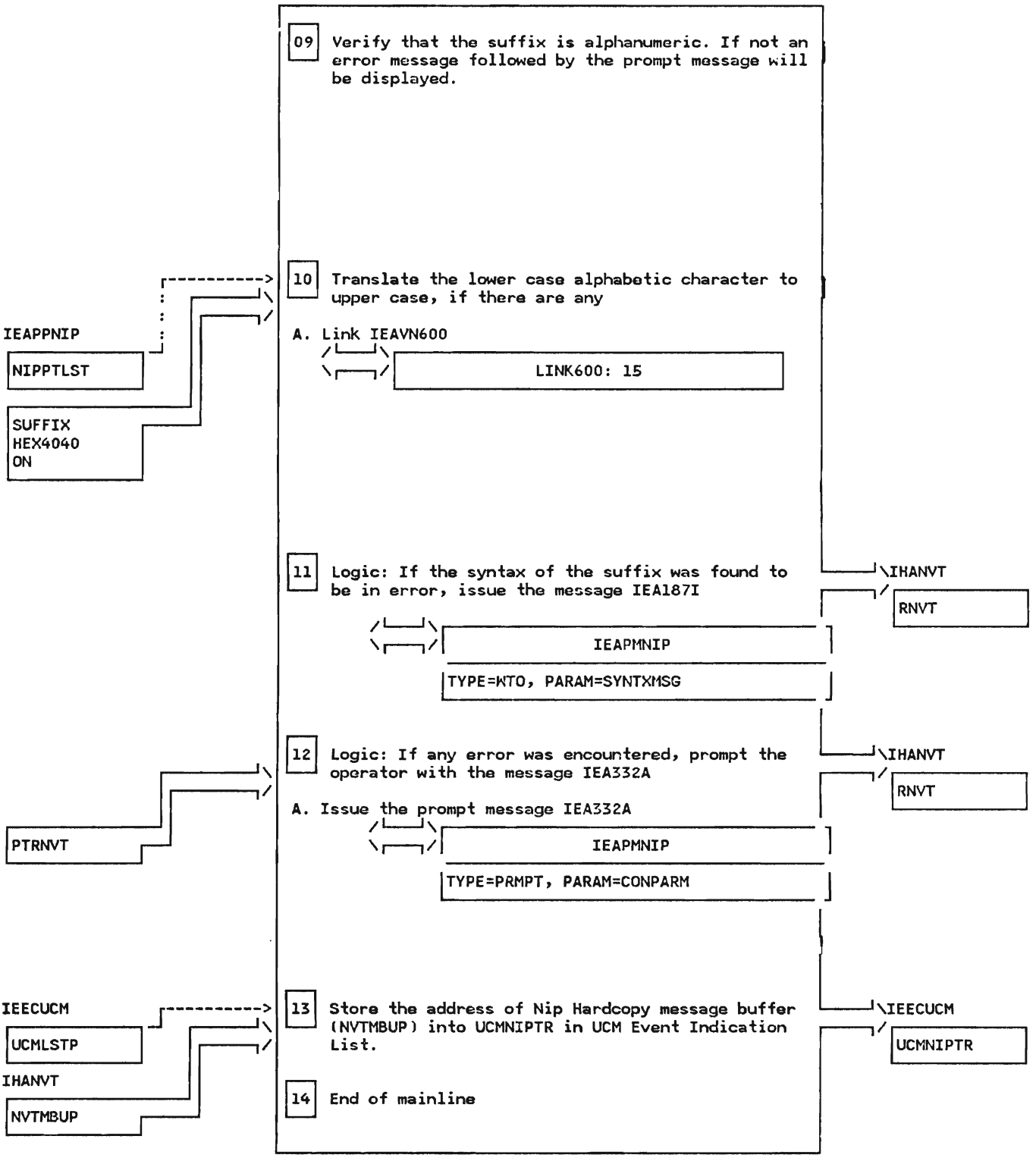


Diagram 54. Communications Task Initialization (IEAVNPA1) Part 7 of 8

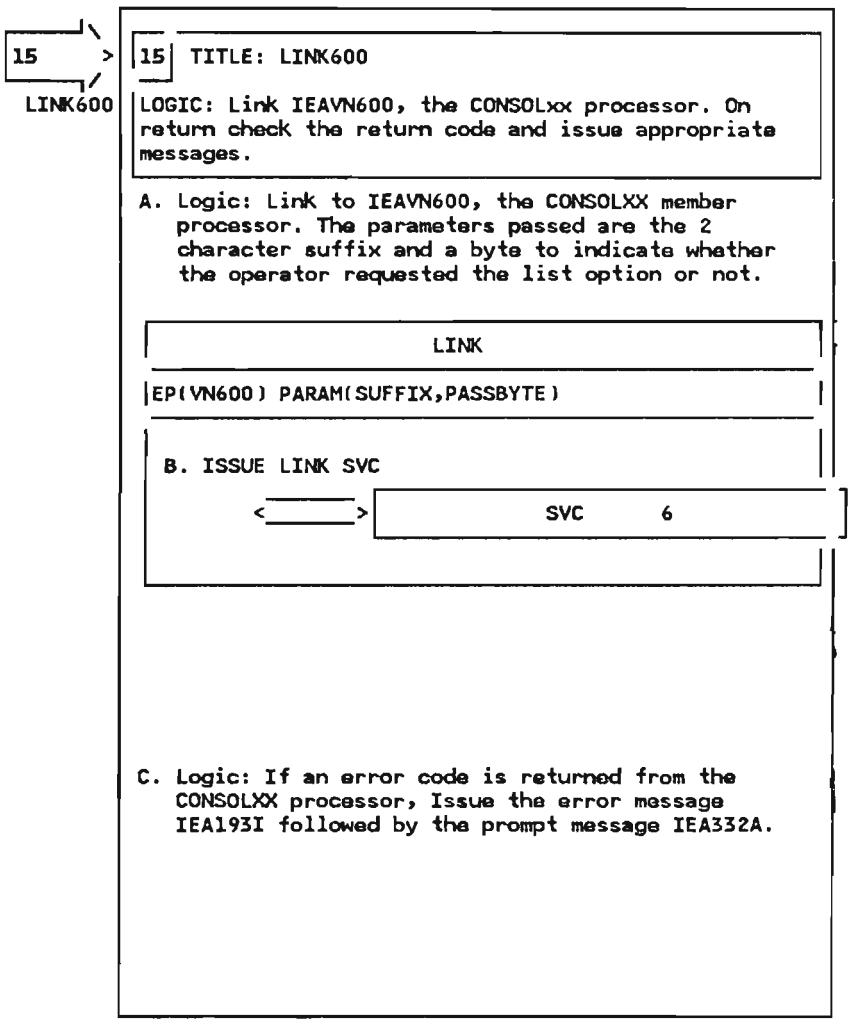


Diagram 54. Communications Task Initialization (IEAVNPA1) Part 8 of 8

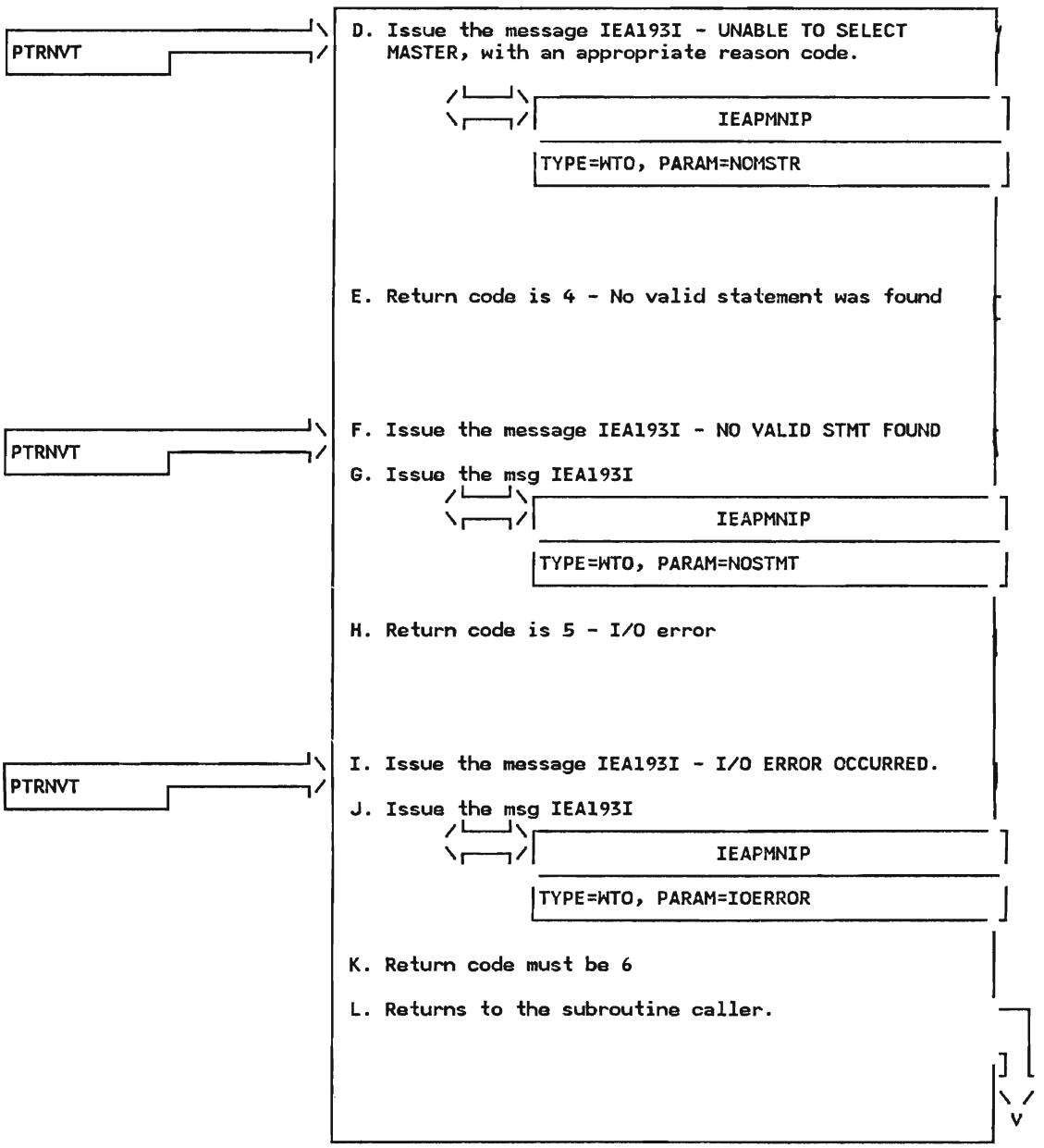


Diagram 55. CONSOLxx Member Processing (IEAVN600) Part 1 of 53

IEAVN600 - MODULE DESCRIPTION

DESCRIPTIVE NAME: CONSOLxx Member Front End Processor

FUNCTION:
Reads the records from the CONSOLxx member of SYS1.PARMLIB, calls IEEMB887 to syntax check the user specified input for the CONSOLE, HARDCOPY, DEFAULT and INIT statements, which will in turn call IEAVN601, IEAVN602, and IEAVN604 to build the Console Data Area. Also calls IEAVN613 to select the master console and to initialize the UCM control blocks.

ENTRY POINT: IEAVN600

PURPOSE: Process CONSOLxx member of SYS1.PARMLIB

LINKAGE: LINK

CALLERS: IEAVNPA1

INPUT:
Register 1 points to two 4 byte fields.
The first word contains the address of the suffix value of CONSOLxx member of SYS1.PARMLIB.
The second word contains the address of a byte which indicates whether the user wanted the list option.

OUTPUT: Return code passed back in register 15.

EXIT NORMAL: Return to the caller

EXIT ERROR: Return to the caller

EXTERNAL REFERENCES:

ROUTINES:

- IEEMB887 - Parses the CONSOLE, HARDCOPY, DEFAULT, and INIT statements
- IEEMB878 - Reads the records from CONSOLxx member
- IEAVN610 - When parsing is finished for a statement (CONSOLE, HARDCOPY, DEFAULT, or INIT), IEAVN610 is called to validate the data specified on the statement.
- IEAVN613 - To select the master console and initialize the UCM control blocks.
- IEAVM200 - To Queue messages in the message queue which is anchored off the Console Data Area.
- PTCONSOL - Start of parse definitions in module IEAVN603.

CONTROL BLOCKS:

Common Name	Macro Name	Usage
MSGS	IEAVM101	C,M,D
CDEF	IEAVN100	C,M,D
SCL	IEEZB815	C,M,D
IEEMB878 Parmlist	IEEZB822	C,M,D

KEY = C - Create, D - Delete, M - Modify, R - Read

Diagram 55. CONSOLxx Member Processing (IEAVN600) Part 2 of 53

IEAVN600 - MODULE OPERATION

Load the Parser module (IEEMB887) into virtual storage.
Load the Parmlib read module (IEEMB878) into virtual storage.
Load the Message service routine (IEAVM200) into virtual storage.

Initialize the Parameter List for the Parmlib read routine and the Parser Parameter List.
Call IEEMB878 to read the first record from the CONSOLxx member of SYS1.PARMLIB - the prefix "xx" was passed to this module from the caller (IEAVNPA1).
If the return code indicates that the specified member does not exist or there was an I/O error, set the return code in reg 15 indicating this condition and return to the caller.
If the user had specified the "L" option, list that record.

Call the parser passing the Scan Parameter List.
On return from the parser,
if the return code indicates that there was an I/O error, set the return code in reg 15 and return to the caller.
If the return code indicates that there was an error processing the comment text in the CONSOLxx member, issue a warning message to inform the user.
If the return code is anything but zero and none of the above mentioned, issue "SYSTEM ERROR" message with return and reason code that were obtained from the parser.
Return to the caller with a return code in register 15.

If the return code indicates that all the records have been parsed successfully,
call IEAVN610, the Console validation module to process the last statement that was parsed (IEAVN610 is called to process all the other statements from CONSOLxx member exit modules, IEAVN601 and IEAVN602).
Delete the modules IEEMB887 and IEEMB878 from virtual storage.

After all CONSOLXX records have been parsed,
call IEAVN613 to select the master console and to initialize the UCM control blocks.
Delete the message service routine from virtual storage.
Return to the invoker with return code.

IOEXIT -
This routine gets control whenever the module IEEMB887 wants to get a new record from the CONSOLxx member in SYS1.PARMLIB.
Set up the parameter list and call IEEMB878 to read a record from the CONSOLxx member in SYS1.PARMLIB.
On return, if the record is successfully read, update the record number, store the address of the newly read record in Scan Control Block and return back to IEEMB887.
If there was an I/O error, set the return code so that IEEMB887 (parser) stops.
If there are no more records to read, set the return code so that IEEMB887 (parser) recognizes this condition.

Diagram 55. CONSOLxx Member Processing (IEAVN600) Part 3 of 53

IEAVN600 - DIAGNOSTIC AIDS

ENTRY POINT NAME: IEAVN600

MESSAGES:

```
IEA195I - CONSOLxx  LINE yyyyy: SYSTEM ERROR xxxx-yyyy.
     xx - Suffix for CONSOLxx member
  yyyyy - Record number in CONSOLxx member

IEA196I - CONSOLxx  stmt-type: UNBALANCED COMMENT
          FOUND. DATA IGNORED.

        - CONSOLxx  stmt-type: PREMATURE END OF FILE
          DETECTED.
  stmt-type - Device number(s), "SUBSYS", "DEFAULT",
              or "INIT"
```

ABEND CODES: None

WAIT STATE CODES: None

RETURN CODES:

EXIT NORMAL:

Register 15 contains 0

EXIT ERROR:

Register 15 contains a non-zero value as described below.

1 - Unable to select master. No full capability Console defined.
2 - Unable to select master. No valid full capability Console found.
3 - Unable to select master. No online full capability Console found.
4 - No valid statements in CONSOLxx.
5 - I/O error while processing CONSOLxx.
6 - CONSOLxx is not in SYS1.PARMLIB or SYSTEM ERROR occurred.

REGISTER CONTENTS ON ENTRY:

```
Register        0 - Irrelevant
Register        1 - Address of a double word.
                  - The first word contains the
                    address of the suffix value
                    of the CONSOLxx member.
                  - The second word contains the
                    address of a byte which indicates
                    whether the user wanted the list
                    option.
Register     2-12 - Irrelevant
Register       13 - Address of 72 byte save area
Register       14 - Return address
Register       15 - Entry address of IEAVN600
```

REGISTER CONTENTS ON EXIT:

EXIT NORMAL:

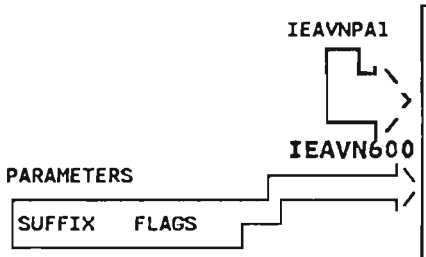
Diagram 55. CONSOLxx Member Processing (IEAVN600) Part 4 of 53

IEAVN600 - DIAGNOSTIC AIDS (Continued)

```
   Registers 0-14 restored to contents upon entry
EXIT ERROR:
   Registers 0-14 restored to contents upon entry
```

Diagram 55. CONSOLxx Member Processing (IEAVN600) Part 5 of 53

IEAVN600 - CONSOLxx Member Front End Processor **STEP 01**



Reads the records from the CONSOLxx member of SYS1.PARMLIB, calls IEEMB887 to syntax check the user specified input for the CONSOLE, HARDCOPY, DEFAULT and INIT statements, which will in turn call IEAVN601, IEAVN602, and IEAVN604 to build the Console Data Area. Also calls IEAVN613 to select the master console and to initialize the UCM control blocks.

01 MAINLINE

Read the first record from the CONSOLxx member of SYS1. PARMLIB by calling IEEMB878. Call IEEMB887 to parse the record according to the parse definitions in module IEAVN603. Call IEAVN610, the Console validation module to validate the last statement that was parsed but did not get validated (other statements were validated by IEAVN610 thru calls from the parser exits modules, IEAVN601 and IEAVN602). Freemain the area that was getmained. Return to the invoker with return code.

Diagram 55. CONSOLxx Member Processing (IEAVN600) Part 6 of 53

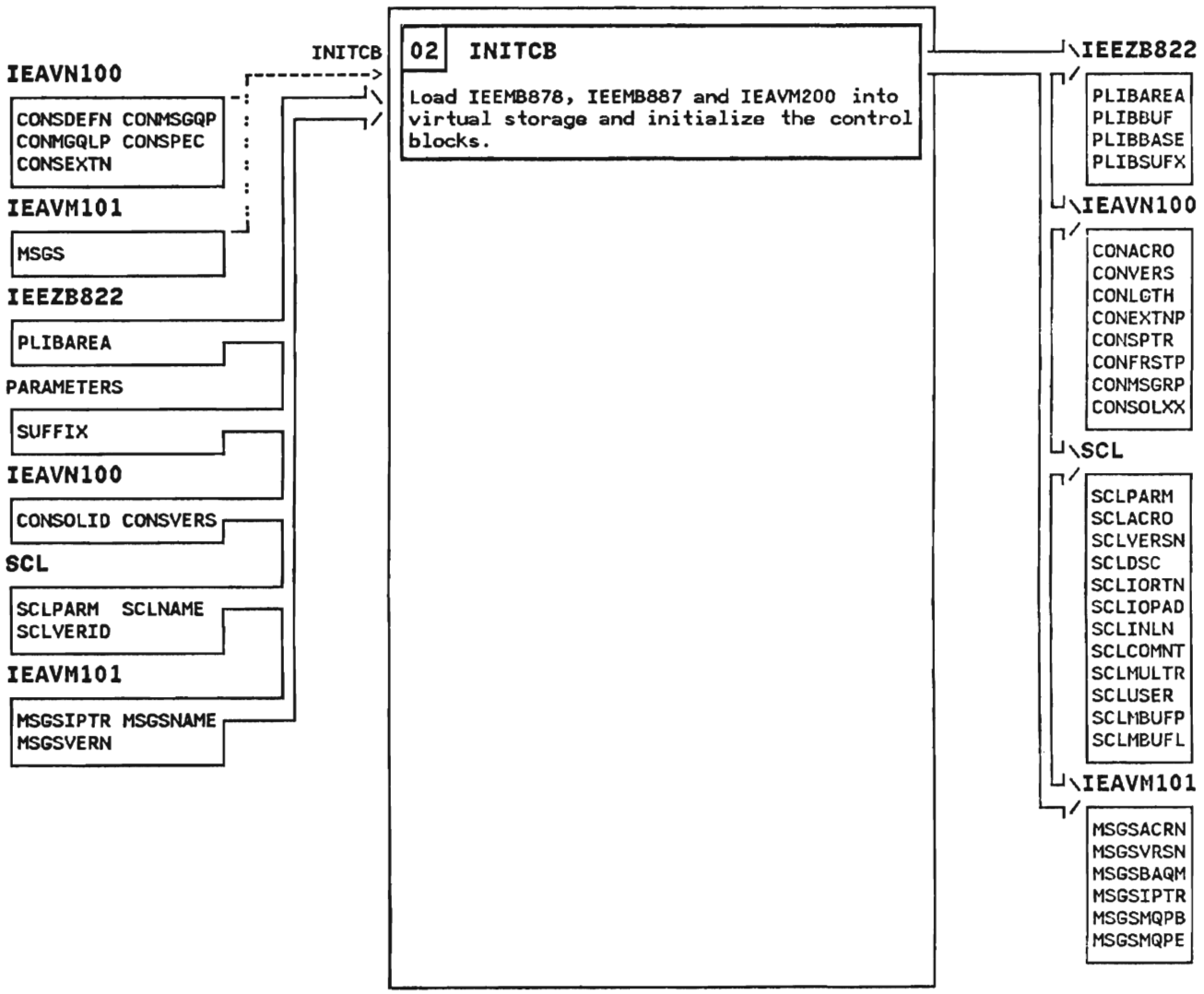


Diagram 55. CONSOLxx Member Processing (IEAVN600) Part 7 of 53

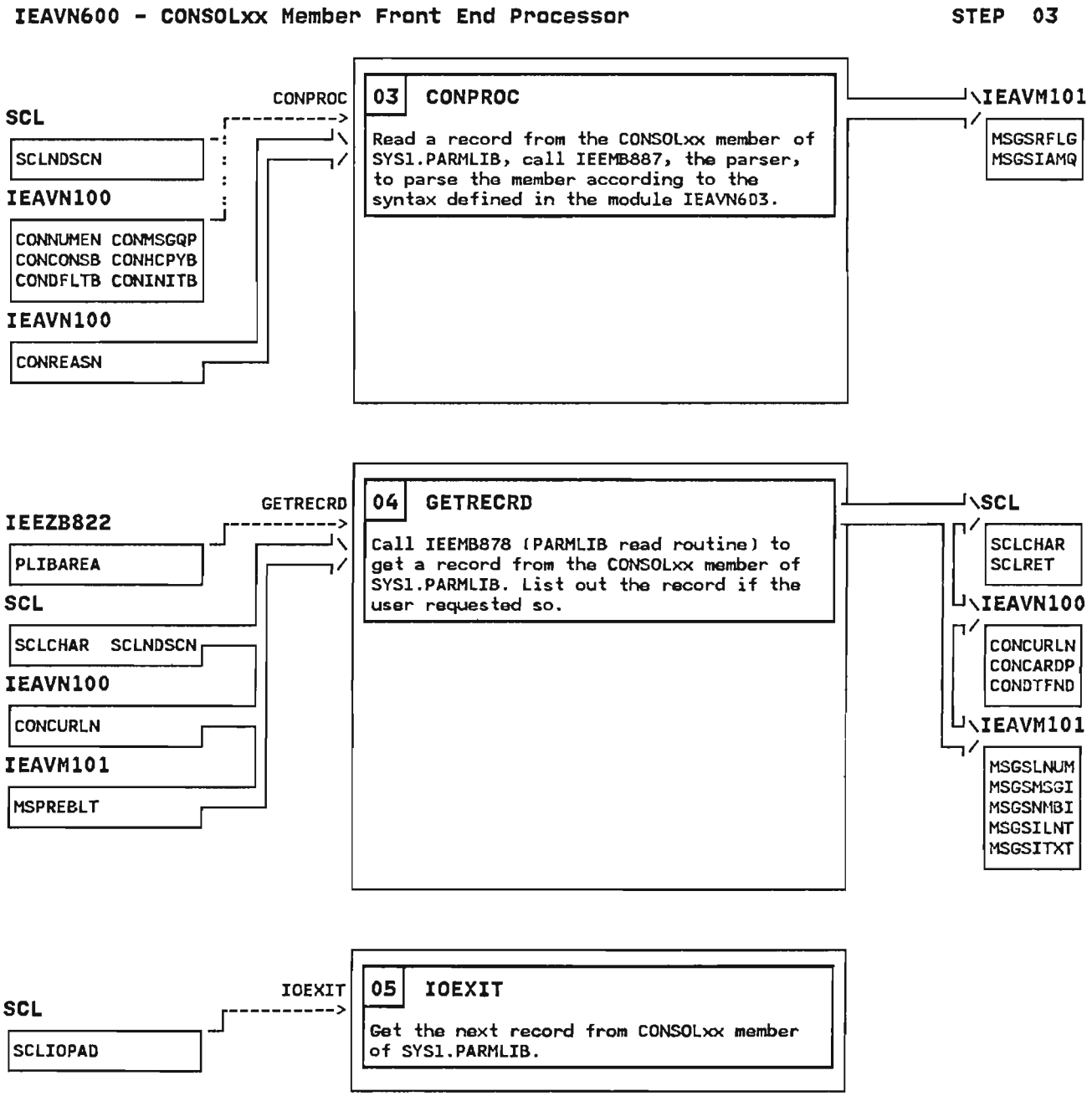


Diagram 55. CONSOLxx Member Processing (IEAVN600) Part 8 of 53

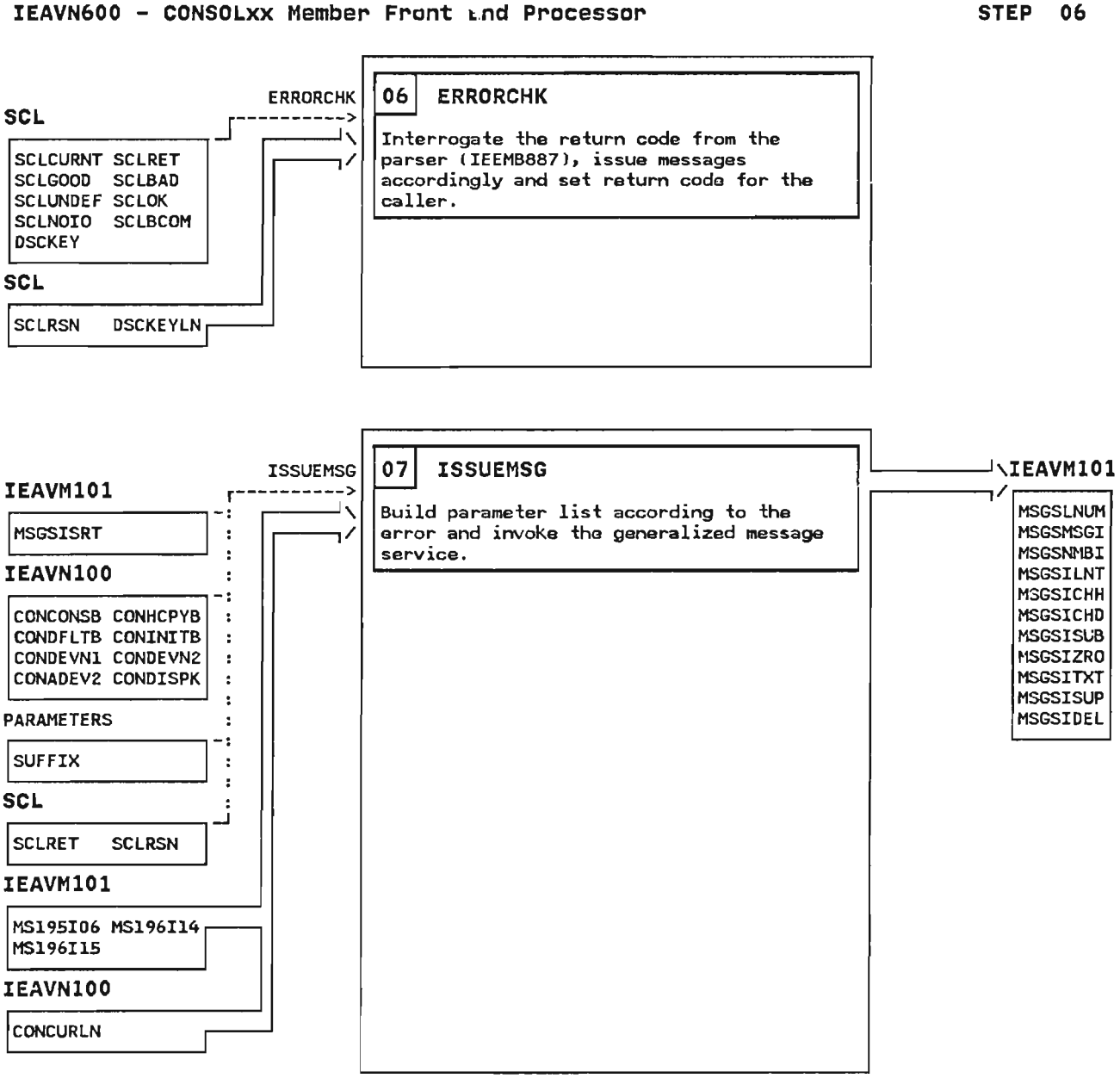


Diagram 55. CONSOLxx Member Processing (IEAVN600) Part 9 of 53

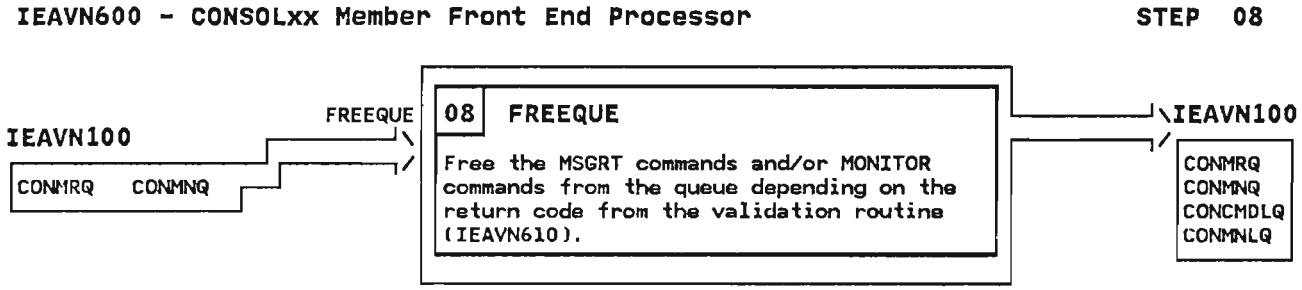


Diagram 55. IEAVN600/IEAVN601 Part 10 of 53

IEAVN601 - MODULE DESCRIPTION

DESCRIPTIVE NAME: CONSOLxx Member Exits - Part 1.

FUNCTION:
This module gets control from the parser (IEEMB887)
to set up the Console Data Area for the CONSOLE statement
key words whose value can be a list, and for ROUTCODE
key word on HARDCOPY and DEFAULT statements.
This module also gets control to process the errors found
while parsing all the statement types.

ENTRY POINT: IEAVN601

PURPOSE:
To store the user specified data in Console
Data Area for further processing.

LINKAGE: PL/AS CALL statement

CALLERS:
Parser Module - IEEMB887, via the parse
definitions in IEAVN603

INPUT:
Register 1 points to a word which contains address of the
Scan Parameter List.

OUTPUT:
Updated Console Data Area and/or
Updated Scan Parameter List

EXIT NORMAL: Returns to the caller.

EXTERNAL REFERENCES:

ROUTINES:
IEAVN610 - To validate the data that the user specified
via CONSOLE statement.

IEAVM200 - To queue the error messages.

CONTROL BLOCKS:

Common Name	Macro Name	Usage
CDEF	IEAVN100	R,M
SCL	IEEZB815	R,M
MSGS	IEAVM101	C,M,D
MGCR	IEZMGCR	C
Index Table	IEAVN102	C,D

KEY = C - Create, D - Delete, M - Modify, R - Read

Diagram 55. IEAVN600/IEAVN601 Part 11 of 53

IEAVN601 - MODULE OPERATION

When the parser finds a syntactically valid value for the keyword, it calls one of the exit routines to store the value in the Console Data Area. There are also routines to process several error situations. The operation of every major routine is explained below:

CONSOLXT -
This routine gets control whenever a CONSOLE statement is found in the CONSOLxx member.

Verify this is the first non-comment data on the current record.
If it is not queue the message "MISPLACED CONSOLE STATEMENT IGNORED" and ignore the data until the next statement type is found.

Verify the number of Console Statements specified.
If it is 99, the limit is reached.
Queue the message "LIMIT OF 99 CONSOLES EXCEEDED".
Ignore this statement and start looking for another statement type.
If it is less than 99, add 1 to the CONSOLE statement counter.

If a CONSOLE, HARDCOPY, DEFAULT, or INIT statement was Parsed just before, call the Console validation routine, IEAVN610, to validate the data that the user specified in that statement.
Update the entry number in the Console Data Area.
Set a flag to indicate that a CONSOLE statement is being processed.

KEYEXIT -
This routine gets control whenever a key word or a key word value is found.

Check the key flags in the Console Data Area to verify that this key word is not duplicately specified.
If duplicate, queue "DUPLICATE KEY WORD" message and ignore the data until the next right parenthesis.
Store the information that this key word is being processed.
Also set the code to indicate that this key word has been encountered.

DEVNUMXT -
This routine gets control when a value is found for DEVNUM keyword in CONSOLE statement.

If the value specified is "SUBSYSTEM", set the Console Data Area accordingly.
If there are two DEVNUM values specified, increment the CONSOLE statement counter by 1. If the counter exceeds 99, Set the error code to issue the message "LIMIT OF 99 CONSOLES EXCEEDED".
Ignore this statement and start looking for a statement type.
If the value is in hex, convert it to binary and verify that the value was not specified as DEVNUM value for a previous CONSOLE statement.
If the DEVNUM was specified in a previous CONSOLE statement, Issue the message "DEVNUM ALREADY DEFINED. STATEMENT IGNORED.". If the second device number is the same as the first device number when it is a composite console, issue the message "CONSOLE STATEMENT IGNORED. REASON=3".
Ignore this statement and start looking for a statement

Diagram 55. IEAVN600/IEAVN601 Part 12 of 53

IEAVN601 - MODULE OPERATION (Continued)

```
   type.

AUTHXT -
  This routine gets control when a value is found for
  the AUTH key word.

  If the same value is repeated, issue the message
  "AUTH VALUE IGNORED" and ignore the duplicate
  specification.
  Set appropriate value in the Console Data Area.
  If the value specified is mutually exclusive with one or
  more previously specified values,
  issue "INCONSISTENT AUTH VALUE IGNORED".

ALTXT -
  This routine gets control when a value is found for
  the ALTERNATE key word.

  Convert the value to binary.
  If it is the input part of a composite console, set
  the value in the Data Area.
  If it is the output part of a composite console,
  verify that it is not the same as input part of the
  composite console. If it is issue "ALTERNATE VALUE
  IGNORED. REASON=2".

ROUTXT -
  This routine gets control when a value is found for
  the ROUTCODE  key word on CONSOLE, HARDCOPY, or DEFAULT
  statements.

  If the value specified is mutually exclusive with one or
  more previously specified values, set the error code to
  issue "INCONSISTENT ROUTCODE VALUE IGNORED".
  If the specified value is decimal, convert it to binary.
  If the value specified is greater than 128, or if the
  range was specified, and it is in descending order,
  set the error code to issue the message "ROUTCODE VALUE
  IGNORED. REASON=1".
  If the same value is repeated, set the error code to
  issue the message "DUPLICATE ROUTCODE VALUE IGNORED" and
  ignore the duplicate specification.
  For any valid ROUTCODE value, set appropriate
  bits in the Console Data Area.

LEVELXT -
  This routine gets control when a value is found for
  the LEVEL key word.

  If the value specified is mutually exclusive with one or
  more previously specified values, set the error code to
  issue "INCONSISTENT LEVEL VALUE IGNORED".
  If the same value is repeated, set the error code to
  issue the message "DUPLICATE LEVEL VALUE IGNORED" and
  ignore the duplicate specification.
  Set appropriate value in the Console Data Area.

MFORMXT -
  This routine gets control when a value is found for
  the MFORM key word.

  If the value specified is mutually exclusive with one or
  more previously specified values, set the error code to
  issue the message "INCONSISTENT MFORM VALUE IGNORED".
  If the same value is repeated, set the error code to
  issue the message "DUPLICATE MFORM VALUE IGNORED"
  and ignore the latter specification.
  Set the appropriate value in the Console Data Area.
```

Diagram 55. IEAVN600/IEAVN601 Part 13 of 53

IEAVN601 - MODULE OPERATION (Continued)

AREAXT -
This routine gets control when a value is found for the AREA key word.

If the value specified is mutually exclusive with one or more previously specified values, set a bit to issue the message "AREA VALUE IGNORED. REASON = 1".
If specified value is anything but "none",
Convert the value to binary.
Store the value in the Console Data Area.
If more than 11 values are specified, set a bit to issue the message "AREA VALUE IGNORED. REASON = 1".

MRMNXT -
This routine gets control when a value is found for the MSGRT or MONITOR key word.

Getmain the storage for MSGRT or MONITOR commands

If the key word is MSGRT
Tag an "MR " in front of each command and queue it in the console command queue.
Store the address of the Console command queue in the Console Data Area

If the key word is MONITOR
If the same value is repeated, set error code to issue "DUPLICATE MONITOR VALUE IGNORED".
Build an MN command and queue it in the MONITOR command queue, off the Console Data Area.

FREEQUE -
This routine gets control when an error return code is received from IEAVN610, the CONSOLxx validation module.

Interrogate the return code and free MSGRT command queue and/or MONITOR command queue accordingly.

FINDPOS -
This routine gets control from the ROUTXT subroutine to find the position of the byte and the bit for a numeric ROUTCODE value.

BADDATA -
This routine gets control whenever an unrecognizable key word or statement type is found.

If this is the first non-comment data in this record, determine whether it is a possible statement type. If it is, issue a message "UNRECOGNIZED STATEMENT TYPE" and skip data until a valid statement type. If it is an erroneous key word, issue a message "UNRECOGNIZABLE KEY WORD" and skip data until the next right parenthesis.

If this is not the first non-comment data in this record, issue the message "UNRECOGNIZABLE KEY WORD" and skip data until the next right parenthesis.

CKSTTYPE -
This routine gets control when the parser has detected a bad value.

Check the data to see if it is a new statement type, if it is a message is displayed for a "MISSING RIGHT PARENTHESIS ASSUMED" and the parser is instructed to take the second successor route.

Diagram 55. IEAVN600/IEAVN601 Part 14 of 53

IEAVN601 - MODULE OPERATION (Continued)

ZEROOUT -
This routine gets control when the parser has detected an extra value for a key word that is only allowed one value.

A "single value key word" does not have a valid delimiter (')'). Reset the fields that were previously set.

CKVALUE -
This routine gets control when the parser has detected a bad value.

Check the data to see if it is the first value specified for the key word, if it is, the message "aaaaaaaa VALUE IGNORED. REASON=rc" is displayed.

Diagram 55. IEAVN600/IEAVN601 Part 15 of 53

IEAVN601 - DIAGNOSTIC AIDS

ENTRY POINT NAME: IEAVN601

MESSAGES:

IEA195I CONSOLxx LINE yyyyy: text

xx: Suffix for the member name.

yyyyy: Line number in CONSOLxx member.

text:

UNRECOGNIZED STATEMENT TYPE IGNORED.

CONSOLE STATEMENT IGNORED. REASON=rc
1 - The DEVNUM key word is not the first key word following the CONSOLE statement type.
2 - The DEVNUM key word value was not hexadecimal or in the range of 000 to FFF.
3 - A CONSOLE statement had the same device number specified for the input and output half of a composite console.

99 CONSOLES DEFINED. STATEMENT IGNORED.

MISPLACED xxxxxxxx STATEMENT IGNORED.

IEA196I CONSOLxx stmt-type: text

xx: Suffix for the member name.

stmt-type: Device Number(s), "SUBSYS", "HARDCOPY", "DEFAULT", or "INIT".

text:

UNRECOGNIZED KEYWORD aaaaaaaaaaaa IGNORED.

aaaaaaaa VALUE IGNORED. REASON=rc.
1 - The value does not conform to the syntax of the key word or the value is out of range for the key word or for the Console's device type.

2 - The input and the output part of the composite console were the same for the ALTERNATE key word on a CONSOLE statement or the DEVNUM key word on the HARDCOPY statement.

INCONSISTENT aaaaaaaa VALUE IGNORED.

MISSING RIGHT PARENTHESIS ASSUMED.

DEVNUM ALREADY DEFINED. STATEMENT IGNORED.

DUPLICATE aaaaaaaaa KEYWORD IGNORED.

DUPLICATE aaaaaaaaa VALUE IGNORED.

ABEND CODES: None

WAIT STATE CODES: None

Diagram 55. IEAVN600/IEAVN601 Part 16 of 53

IEAVN601 - DIAGNOSTIC AIDS (Continued)

RETURN CODES:

EXIT NORMAL:

Register 15 contains 0

REGISTER CONTENTS ON ENTRY:

Register	0	- Irrelevant
Register	1	- Address of a word which contains the address of the Scan Parameter List
Register	2-12	- Irrelevant
Register	13	- Address of 72 byte save area
Register	14	- Return address
Register	15	- Entry address of IEAVN601

REGISTER CONTENTS ON EXIT:

EXIT NORMAL:

Registers 0-14 restored to contents upon entry

Diagram 55. IEAVN600/IEAVN602 Part 17 of 53

IEAVN602 - MODULE DESCRIPTION

DESCRIPTIVE NAME: CONSOLxx Member Exits - PART 2.

FUNCTION:
This module gets control from IEEMB887, the parser to set up the Console Data Area with values that were specified for the HARDCOPY, DEFAULT, and INIT statement key words.

ENTRY POINT: IEAVN602

PURPOSE:
To store the user's data in the parameter list for further processing.

LINKAGE: PL/AS CALL statement

CALLERS:
Parser Module - IEEMB887, via the parse definitions in IEAVN603

INPUT:
Register 1 points to a word which contains address of the Scan Parameter List.

OUTPUT:
Updated Console Data Area and/or
Updated Scan Parameter List

EXIT NORMAL: Return to the caller with return code 0

EXTERNAL REFERENCES:

ROUTINES:
IEAVN610 - To validate the data supplied by the user in CONSOLE, HARDCOPY, DEFAULT, and INIT statement.

IEAVM200 - To queue the informational or error messages.

CONTROL BLOCKS:

Common Name	Macro Name	Usage
MSGS	IEAVM101	C,M,D
CDEF	IEAVN100	R,M
Index Table	IEAVN102	C,D
SCL	IEEZB815	R,M
MGCR	IEZMGCR	C,R

KEY = C - Create, D - Delete, M - Modify, R - Read

Diagram 55. IEAVN600/IEAVN602 Part 18 of 53

IEAVN602 - MODULE OPERATION

This module processes HARDCOPY, DEFAULT, and INIT statement types and all the key word values associated with them.

NOTE: Processing of the ROUTCODE key word on HARDCOPY and DEFAULT statements is done in CONSOLxx Member Exits - Part 1.

STMTXT -
This routine is invoked when the parser finds a HARDCOPY, DEFAULT, or INIT statement type.

Verify that the statement type is not duplicately specified.
If there was any statement that was parsed just prior to this, invoke the validation routine, IEAVN610 to validate the statement that was just parsed.

KEYVALXT -
This routine is invoked when the parser finds a value for a key word.

Determine the reason for the entry and route control to a routine to the appropriate routine to process the value.

DNCMDXT -
This routine is entered to process the values for the HARDCOPY statement key words, DEVNUM or CMDLEVEL.

DEVNUM value processing:
If the value specified is "SYSLOG", set the Console Data Area accordingly.
If the value is in hex, convert it to binary.
If the user specified a composite console, verify that the input and the output part of the console is not the same. Set the error code to issue the message "DEVNUM VALUE IGNORED. REASON=2"
Store the value in the data area, if no error found.

CMDLEVEL value processing:
Set the corresponding flag in the Console Data Area according to the value specified.

This routine is invoked when the parser finds a value for CMDLEVEL key word in HARDCOPY statement.

MLRLUXT-
This routine is entered to process the value specified for MLIM or RLIM key word on INIT statement.

Convert the value to binary.
Set the value in the Console Data Area

AMUXXT -
This routine is entered to process the value specified for AMRF, or UEXIT key word on INIT statement.

Store the value in Console Data Area depending on the user specified value.

MPFPFKXT -
This routine is entered to process the value specified for MPF, or PFK key word on INIT statement.

Store the value in Console Data Area depending on the user specified value.

Diagram 55. IEAVN600/IEAVN602 Part 19 of 53

IEAVN602 - MODULE OPERATION (Continued)

```
INMNXT -
  This routine is entered to process the value
  for  MONITOR key word on INIT statement.

  If the same value is repeated, queue appropriate message.
  Build an MN command and queue it in the system command
  queue.
  Store the address of the first MN command in the
  Console Data Area
```

Diagram 55. IEAVN600/IEAVN602 Part 20 of 53

IEAVN602 - DIAGNOSTIC AIDS

ENTRY POINT NAME: IEAVN602

MESSAGES:

```
    IEA195I  CONSOLxx LINE yyyyy: text

       xx: Member name suffix

     yyyyy: Line number in CONSOLxx member

      text:
         MISPLACED HARDCOPY STATEMENT IGNORED.
         MISPLACED DEFAULT STATEMENT IGNORED.
         MISPLACED INIT STATEMENT IGNORED.

    IEA196I  CONSOLxx stmt-type: text

       xx: Member name suffix

stmt-type: "HARDCOPY", "DEFAULT", or "INIT"

     text -
        aaaaaaaaa VALUE IGNORED. REASON=rc.
        DUPLICATE SPECIFICATION IGNORED.
        DUPLICATE aaaaaaaaa VALUE IGNORED.
```

ABEND CODES: None

WAIT STATE CODES: None

RETURN CODES:

```
EXIT NORMAL:

  Register 15 contains 0
```

REGISTER CONTENTS ON ENTRY:

```
Register        0 - Irrelevant
Register        1 - Address of a 4 byte field which
                    contains the address of the Scan
                    Parameter List
Register     2-12 - Irrelevant
Register       13 - Address of 72 byte save area
Register       14 - Return address
Register       15 - Entry address of IEAVN602
```

REGISTER CONTENTS ON EXIT:

```
EXIT NORMAL:

  Registers 0 - 14 restored to contents upon entry
```

Diagram 55. IEAVN600/IEAVN604 Part 21 of 53

IEAVN604 - MODULE DESCRIPTION

DESCRIPTIVE NAME: CONSOLxx Member Exits - Part 3.

FUNCTION:
This module gets control from the parser (IEEMB887) to set up the Console Data Area with values that were specified for UNIT, USE, CON, DEL, SEG, RNUM, RTME, UTME and PFKTAB key words on CONSOLE statement.

ENTRY POINT: IEAVN604

PURPOSE:
To store the user specified data in Console Data Area for further processing.

LINKAGE: PL/AS CALL statement

CALLERS:
Parser Module - IEEMB887, via the parse definitions in IEAVN603

INPUT:
Register 1 points to a word which contains address of the Scan Parameter List.

OUTPUT: Updated Console Data Area

EXIT NORMAL: Returns to the caller.

EXTERNAL REFERENCES:

ROUTINES: None

CONTROL BLOCKS:

Common Name	Macro Name	Usage
CDEF	IEAVN100	R,M
UCM	IEECUCM	R
SCL	IEEZB815	R,M
Index Table	IEAVN102	C,D

KEY = C - Create, D - Delete, M - Modify, R - Read

Diagram 55. IEAVN600/IEAVN604 Part 22 of 53

IEAVN604 - MODULE OPERATION

When the parser finds a syntactically valid value for the key words specified above, this module is entered to set appropriate values in the Console Data Area.

UNITXT -
This routine gets control when a value is found for the UNIT key word.

Set appropriate value in the Console Data Area according to the unit type specified.

USEXT -
This routine gets control when a value is found for the USE key word.

Set appropriate value in the Console Data Area.

CDUXT -
This routine gets control when a value is found for the USE, CON, or DEL key word.

Set appropriate value in the Console Data Area.

SGRNTPXT -
This routine gets control when a value is found for the SEG, RNUM, RTME, UTME, or PFKTAB key words.

If the key word is PFKTAB, store the value in Console Data area.
If the key word is SEG, RNUM, RTME, or UTME
  Convert the value to binary.
  Set the value in the Console Data Area

Diagram 55. IEAVN600/IEAVN604 Part 23 of 53

IEAVN604 - DIAGNOSTIC AIDS

ENTRY POINT NAME: IEAVN604

MESSAGES: None

ABEND CODES: None

WAIT STATE CODES: None

RETURN CODES:

EXIT NORMAL:

Register 15 contains 0

REGISTER CONTENTS ON ENTRY:

Register	0	- Irrelevant
Register	1	- Address of a word which contains the address of the Scan Parameter List
Register	2-12	- Irrelevant
Register	13	- Address of 72 byte save area
Register	14	- Return address
Register	15	- Entry address of IEAVN604

REGISTER CONTENTS ON EXIT:

EXIT NORMAL:

Registers 0-14 restored to contents upon entry

Diagram 55. IEAVN600/IEAVN610 Part 24 of 53

IEAVN610 - MODULE DESCRIPTION

DESCRIPTIVE NAME: CONSOLxx Statement Validation Module

FUNCTION:
To complete the validation of the CONSOLE, HARDCOPY, DEFAULT and INIT statements. This module receives control to process one statement at a time.

ENTRY POINT: IEAVN610

PURPOSE: See FUNCTION above.

LINKAGE:
IEAVN610 is CALLed using standard system linkage conventions.

CALLERS: IEAVN600, IEAVN601, IEAVN602, IEAVN613

INPUT: CONSOLxx Data Area (IEAVN100).

OUTPUT:
Updated CONSOLxx Data Area (IEAVN100)
Message Service Parameter List (IEAVM101).

EXIT NORMAL: Returns to the caller.

EXIT ERROR: Returns to the caller.

EXTERNAL REFERENCES:

ROUTINES:
IEAVN611 - CONSOLE Statement Semantic Module Part 1
IEAVM200 - Generalized Message Service Module

CONTROL BLOCKS:
The following control blocks are used, modified or created in this module:

NAME	MAPPING MACRO	REFERENCE
CVT	CVT	R
MSGS	IEAVM101	C,M,D
CONSDEFN	IEAVN100	M
IOCM	IECDIOCM	R
UCM	IEECUCM	R
UCB	IEFUCBOB	R
PSA	IHAPSA	R

KEY = R-read, C-create, M-modify, D-delete

SERIALIZATION:
None

SYSGEN =

LOAD MODULE - IEAVN600
DISTRIBUTION LIBRARY - AOSC5

Diagram 55. IEAVN600/IEAVN 610 Part 25 of 53

IEAVN610 - MODULE OPERATION

The major sub-routines in this module are listed:

1. MAINLINE
   - Determines Statement-Type being validated

2. CONSOLE Sub-routine
   Insures consistency between the Device Type
   in the UCB with the CONSOLxx specifications
   and applies defaults if necessary
   - Checks validity of Device Numbers
   - Calls UCB_FIND to validate Device Number
   against a matching UCB
   - Calls MS_MATCH to determine if the Device
   Type in the UCB matches that specified
   on the UNIT keyword in CONSOLxx
   - Calls CON_SUPP to determine if the Device
   Type in the UCB is supported as a Console
   - Calls CON_COMP to validate Device Type for
   a Composite Console
   - Applies defaults to the valid Console if
   necessary
   - Calls IEAVN611 to do semantic checking for
   the keywords on the CONSOLE statement

   Note: IEAVN611 calls IEAVN612 to continue
   the semantic checking.

3. HARDCOPY Sub-routine
   Applies defaults if necessary:
   - Applies default Routing Codes
   - Applies default Device Number for HARDCOPY LOG
   - Applies default CMDLEVEL

4. DEFAULT Sub-routine
   Applies default Routing Codes if necessary

5. INIT Sub-routine
   Validates the INIT statement and applies
   defaults if necessary:
   - Checks validity of MLIM (WTO buffer limit)
   - Checks validity of RLIM (WTOR reply limit)
   - Applies default for AMRF (Action Message
   Retention Facility status
   - Checks UEXIT (WTO user exit status)
   - Checks suffix for MPF (Message Processing
   Facility) parmlib member

Diagram 55. IEAVN600/IEAVN610 Part 26 of 53

IEAVN610 - DIAGNOSTIC AIDS

ENTRY POINT NAME: IEAVN610

MESSAGES:

IEAVN610 issues the following messages via IEAVM200:

IEA189I CONSOLxx: iii(,ooo) IGNORED. Text

Where:
xx is the suffix of the CONSOLxx SYS1.PARMLIB member being processed when the error was recognized.

iii(,ooo) is the Device Number(s) of the CONSOLE statement in error.

The Text inserts are:

UNIT NOT SUPPORTED.

NO IODEVICE STATEMENT FOR ddd.

Where ddd is the Device Type specified

iii NOT SUPPORTED AS INPUT DEVICE.

Where iii is the Device Number of the input device.

ooo NOT SUPPORTED AS OUTPUT DEVICE.

Where ooo is the Device Number of the output device.

IEA196I CONSOLxx stmt-type: text.

Where:
The stmt-types are:

iii(,ooo) or INIT

The text inserts are:

keyword VALUE IGNORED. REASON=rc

Where keyword is either:

MLIM or RLIM

The Reason Code identifies the situation:

1. The value does not conform to the syntax of the keyword or the value is out of range for the keyword or for the Console's Device Type.

IODEVICE STATEMENT UNIT APPLIED. REASON=rc

The Reason Code identifies the situation:

1. A second Device Number was specified on the DEVNUM keyword, but the UNIT

Diagram 55. IEAVN600/IEAVN610 Part 27 of 53

IEAVN610 - DIAGNOSTIC AIDS (Continued)

specification was not COMP.

2. The UNIT specification is COMP, but the DEVNUM keyword does not have two Device Numbers.

3. The Device Type specified on an IODEVICE statement in MVSCP is not consistent with the UNIT specification on the CONSOLE statement of CONSOLxx.

ABEND CODES: None

WAIT STATE CODES: None

RETURN CODES:

EXIT NORMAL:

The Return Code in Register 15 is:

0 Successful processing.

EXIT ERROR:

The Return Codes in Register 15 are:

4 Request is not valid.
8 Free MSGRT command queue.
12 Free all (MONITOR and MSGRT) command queues.

REGISTER CONTENTS ON ENTRY:

Register 0 - Irrelevant
Register 1 - Address of a 4-byte field that contains the address of the CONSOLxx Data Area (IEAVN100)
Registers 2-12 - Irrelevant
Register 13 - Address of a standard save area
Register 14 - Return address
Register 15 - Entry point address of IEAVN610

REGISTER CONTENTS ON EXIT:

EXIT NORMAL:

Registers 0 thru 14 are restored to contents at entry.

EXIT ERROR:

Registers 0 thru 14 are restored to contents at entry.

Diagram 55. IEAVN600/IEAVN611 Part 28 of 53

IEAVN611 - MODULE DESCRIPTION

DESCRIPTIVE NAME: CONSOLE Statement Semantic Module Part 1

FUNCTION:

To validate the keywords on the CONSOLE statement and apply defaults if necessary. The keywords ALTERNATE, UNIT, AUTH, USE, AREA, CON and DEL are validated in this module. This module receives control to process one statement at a time.

ENTRY POINT: IEAVN611

PURPOSE: See FUNCTION above.

LINKAGE:
IEAVN611 is CALLed using standard system linkage conventions.

CALLERS: IEAVN610

INPUT: CONSOLxx Data Area (IEAVN100)

OUTPUT:
Updated CONSOLxx Data Area (IEAVN100)
Message Service Parameter List (IEAVM101).

EXIT NORMAL: Returns to the caller.

EXIT ERROR: Returns to the caller.

EXTERNAL REFERENCES:

ROUTINES:
IEAVN612 - CONSOLE Statement Semantic Module Part 2
IEAVM200 - Generalized Message Service Module

DATA AREAS: Console Default Table (IEAVN800)

CONTROL BLOCKS:
The following control blocks are used, modified or created in this module:

NAME	MAPPING MACRO	REFERENCE
MSGS	IEAVM101	M
CONSDEFN	IEAVN100	M
DEFTMAP	IEAVN101	R
UCM	IEECUCM	R

KEY = R-read, C-create, M-modify, D-delete

SERIALIZATION:

None

SYSGEN =

LOAD MODULE - IEAVN600
DISTRIBUTION LIBRARY - AOSC5

Diagram 55. IEAVN600/IEAVN611 Part 29 of 53

IEAVN611 - MODULE OPERATION

The major sub-routines in this module are listed:

1. MAINLINE
   Invokes sub-routines to validate keywords for this module
   Calls IEAVN612 to continue validation of keywords for the CONSOLE statement

2. ALT_UNIT Sub-routine
   Validates the ALTERNATE Console specification and UNIT keyword
   - Checks validity for Subsystem Console

3. AUTH Sub-routine
   Validates the AUTHority keyword and applies defaults if necessary
   - Checks for duplicate Master Console
   - Checks validity for Subsystem and Printer Consoles

4. USE Sub-routine
   Validates the USE keyword and applies defaults if necessary
   - Checks validity for Subsystem, Composite, Master, 2740 and Printer Consoles

5. AREA Sub-routine
   Validates the AREA keyword and applies defaults if necessary
   - Checks validity for Subsystem, Composite, Printer and 2740 Consoles
   - Calls SRCH_TAB to Search Console Characteristics Default Table (IEAVN800) for match on Device Type
   - Calls AREA_SD to validate AREA for Status Display Console
   - Calls AREA_FC to validate AREA for Full Capability or Message Stream Console
   - Calls AREA_DFT to apply defaults if AREA keyword was not specified

6. CON_DEL Sub-routine
   Validates the CONversational and DELetion mode keywords and applies defaults if necessary
   - Checks validity for Subsystem, Composite, Printer, 2740 and Display Consoles
   - Calls DEL_MS to validate the DELetion mode for Message Stream Consoles and applies defaults if necessary

Diagram 55. IEAVN600/IEAVN611 Part 30 of 53

IEAVN611 - DIAGNOSTIC AIDS

ENTRY POINT NAME: IEAVN611

MESSAGES:

IEAVN611 issues the following messages via IEAVM200:

IEA195I CONSOLxx LINE yyyyy:
Keyword IGNORED FOR SUBSYSTEM CONSOLE.

Where:
xx is the suffix of the CONSOLxx SYS1.PARMLIB member being processed when the error was recognized.

yyyyy is the relative position of a line within the SYS1.PARMLIB member.

keyword is either:
ALTERNATE, AREA, AUTH(MASTER), CON, DEL, UNIT or USE

IEA196I CONSOLxx iii(,ooo): text

Where iii(,ooo) is the Device Number(s) of the CONSOLE statement in error.

The text inserts are:

AREA VALUE IGNORED. REASON=1

The Reason Code identifies the situation:

1. The value is out of range for the AREA specification. Each AREA length must be four or more lines.

Keyword IGNORED. REASON=rc

Where keyword is either:

AREA, AUTH, CON or DEL

The Reason Code identifies the situation:

1. AREA, CON or DEL was specified for a UNIT of COMP, PRT, or 2740. The AREA, CON, or DEL specification is ignored.

2. AUTH was specified for a UNIT of PRT. AUTH is ignored.

3. The AREA specification was not valid. The total out-of-line AREA specified can not exceed the screen size of the device.

USE(xx) FORCED. REASON=rc

Where xx indicates the value the USE parameter was changed to either:

FC - Full Capability
MS - Message Stream

Diagram 55. IEAVN600/IEAVN611 Part 31 of 53

IEAVN611 - DIAGNOSTIC AIDS (Continued)

The Reason Code identifies the situation:

1. A statement contained AUTH(MASTER) and USE(MS) or USE(SD). Since the Master Console must be full capability, USE(FC) is forced.

2. A definition contained UNIT(COMP) or UNIT(2740) but the USE specification was not full capability. USE(FC) is forced.

3. A USE of Full Capability or Status Display was specified for a Printer Console. USE(MS) is forced.

DUPLICATE MASTER. AUTH(INFO) FORCED.

DEL(RD) FORCED.

ABEND CODES: None

WAIT STATE CODES: None

RETURN CODES:

EXIT NORMAL:

The Return Code in Register 15 is:

0 Successful processing.

EXIT ERROR:

The Return Codes in Register 15 are:

8 Free MSGRT command queue.
12 Free all (MONITOR and MSGRT) command queues.

Note: Return Codes 8 and 12 are generated within IEAVN612 and passed back to the caller of this module.

REGISTER CONTENTS ON ENTRY:

Register 0 - Irrelevant
Register 1 - Address of a 4-byte area that contains the address of the CONSOLxx Data Area (IEAVN100)
Registers 2-12 - Irrelevant
Register 13 - Address of a standard save area
Register 14 - Return address
Register 15 - Entry point address of IEAVN611

REGISTER CONTENTS ON EXIT:

EXIT NORMAL:

Registers 0 thru 14 are restored to contents at entry.

Diagram 55. IEAVN600/IEAVN611 Part 32 of 53

IEAVN611 - DIAGNOSTIC AIDS (Continued)

EXIT ERROR:

Registers 0 thru 14 are restored
to contents at entry.

Diagram 55. IEAVN600/IEAVN612 Part 33 of 53

IEAVN612 - MODULE DESCRIPTION

DESCRIPTIVE NAME: CONSOLE Statement Semantic Module Part 2

FUNCTION:
To validate the keywords on the CONSOLE statement and apply defaults if necessary. The LEVEL, MFORM, MONITOR, MSGRT, PFKTAB, RNUM, RTME, UTME, SEG and ROUTCODE are validated in this module. This module receives control to process one statement at a time.

ENTRY POINT: IEAVN612

PURPOSE: See FUNCTION above.

LINKAGE:
IEAVN612 is CALLed using standard system linkage conventions.

CALLERS: IEAVN611

INPUT: CONSOLxx Data Area (IEAVN100)

OUTPUT:
Updated CONSOLxx Data Area (IEAVN100).
Message Service Parameter List (IEAVM101).

EXIT NORMAL: Returns to the caller.

EXIT ERROR: Returns to the caller.

EXTERNAL REFERENCES:

ROUTINES:
IEAVM200 - Generalized Message Service Module

DATA AREAS: Console Default Table (IEAVN800)

CONTROL BLOCKS:
The following control blocks are used, modified or created in this module:

NAME	MAPPING MACRO	REFERENCE
MSGS	IEAVM101	M
CONSDEFN	IEAVN100	M
DEFTMAP	IEAVN101	R
UCM	IEECUCM	R

KEY = R-read, C-create, M-modify, D-delete

SERIALIZATION:
None

SYSGEN =

LOAD MODULE - IEAVN600
DISTRIBUTION LIBRARY - AOSC5

Diagram 55. IEAVN600/IEAVN612 Part 34 of 53

IEAVN612 - MODULE OPERATION

The major sub-routines in this module are listed:

1. MAINLINE
   Invokes sub-routines to validate keywords for this module

2. LVL_MFRM Sub-routine
   Validates the LEVEL and MFORM keywords and applies defaults if necessary
   - Checks validity for Subsystem Console

3. MON_MGRT Sub-routine
   Validates the MONITOR keyword
   - Checks validity for Subsystem Console
   Validates the MSGRT keyword
   - Checks validity for Subsystem and Printer Consoles

4. PFKTABLE Sub-routine
   Validates the PFKTAB keyword
   - Checks validity for Subsystem, Composite, Printer and 2740 Consoles

5. RN_T_SEG Sub-routine
   Validates the RNUM, RTME, UTME and SEG keywords and applies defaults if necessary
   - Checks validity for Subsystem, Composite, Printer, 2740, Display and 3270-X Consoles

6. ROUTCODE Sub-routine
   Validates the ROUTCODE keyword and applies defaults if necessary
   - Checks validity for Subsystem and Master Consoles

Diagram 55. IEAVN600/IEAVN612 Part 35 of 53

IEAVN612 - DIAGNOSTIC AIDS

ENTRY POINT NAME: IEAVN612

MESSAGES:

IEAVN612 issues the following messages via IEAVM200:

```
IEA195I CONSOLxx LINE yyyyy:
  keyword IGNORED FOR SUBSYSTEM CONSOLE.

  Where:
    xx is the CONSOLxx SYS1.PARMLIB member
      being processed when the error was
      recognized.

    yyyyy is the relative position of a line
      within the SYS1.PARMLIB member.

    keyword is either:
      LEVEL, MFORM, MONITOR, MSGRT, PFKTAB,
      RNUM, ROUTCODE, RTME, UTME or SEG


IEA196I CONSOLxx iii(,ooo): text

  Where iii(,ooo) is the device number(s)
    of the CONSOLE statement in error.

  The text inserts are:

    keyword VALUE IGNORED. REASON=rc

      Where keyword is either:

        RNUM, RTME, UTME or SEG

      The Reason Code identifies the situation:

         1. The value does not conform to the
            syntax of the keyword or the value
            is out of range for the keyword or
            for the Console's Device Type.

    keyword IGNORED. REASON=rc

      Where keyword is either:

        MSGRT, PFKTAB, RNUM, RTME, UTME or SEG

      The Reason Code identifies the situation:

         1. PFKTAB, RNUM, RTME, UTME or SEG was
            specified for a UNIT of COMP, PRT,
            or 2740. The PFKTAB, RNUM, RTME, SEG OR
            UTME specification is ignored.

         2. MSGRT was specified for a UNIT of
            PRT. MSGRT is ignored.

         3. MSGRT was specified for a UNIT of
            COMP. MSGRT is ignored.
```

ABEND CODES: None

Diagram 55. IEAVN600/IEAVN612 Part 36 of 53

IEAVN612 - DIAGNOSTIC AIDS (Continued)

WAIT STATE CODES: None

RETURN CODES:

EXIT NORMAL:

The Return Code in Register 15 is:

0 Successful processing.

EXIT ERROR:

The Return Codes in Register 15 are:

8 Free MSGRT command queue.
12 Free all (MONITOR and MSGRT) command queues.

REGISTER CONTENTS ON ENTRY:

```
Register     0 - Irrelevant
Register     1 - Address of a 4-byte area that
                 contains the address of the
                 CONSOLxx Data Area (IEAVN100)
Registers 2-12 - Irrelevant
Register    13 - Address of a standard save area
Register    14 - Return address
Register    15 - Entry point address of IEAVN612
```

REGISTER CONTENTS ON EXIT:

EXIT NORMAL:

Registers 0 thru 14 are restored
to contents at entry.

EXIT ERROR:

Registers 0 thru 14 are restored
to contents at entry.

Diagram 55. IEAVN600/IEAVN613 Part 37 of 53

IEAVN613 - MODULE DESCRIPTION

FUNCTION:
This module receives control when all CONSOLxx records have been read. IEAVN613 calls IEAVN610 to apply defaults if no HARDCOPY, DEFAULT or INIT statement is specified in CONSOLxx. It selects a new master console if needed, validates all alternate consoles and calls IEAVN614 to initialize the UCM control blocks.

ENTRY POINT: IEAVN613

PURPOSE: See function

LINKAGE: PL/AS Call statement

CALLERS: IEAVN600

INPUT:
Register 1 points to a word that contains the address of the CONSOLxx data area (IEAVN100)

OUTPUT: The CONSOLxx data area (IEAVN100) is updated

EXIT NORMAL: Returns to the caller.

OUTPUT: The CONSOLxx data area is updated.

EXIT ERROR: Returns to the caller

OUTPUT: Reason code is set in the CONSOLxx data area.

EXTERNAL REFERENCES:

ROUTINES:
IEAVN610 - Console validation routine
IEAVN614 - UCM initialization routine
IEAVM200 - Generalized message service routine

CONTROL BLOCKS:

Common name	Macro Name	Usage
CDEF	IEAVN100	R,M
MSGS	IEAVM101	C,D,M
UCB	IEFUCBOB	R
UCM	IEECUCM	R

KEY = C-CREATE, D-DELETE, M-MODIFY, R-READ

TABLES: No tables used.

Diagram 55. IEAVN600/IEAVN613 Part 38 of 53

IEAVN613 - MODULE OPERATION

IEAVN613 receives control with register 1 pointing to a word that contains the address of the CONSOLxx data area. This module is called to validate the configuration of the consoles.

1. Validates HARDCOPY, DEFAULT and INIT statement.

   If no HARDCOPY statement is specified, calls IEAVN610 to apply the default values for the statemant.

   If no DEFAULT statement is specified, calls IEAVN610 to apply the default values for the statement.

   If no INIT statement is specified, calls IEAVN610 to apply the default values for the statement.

2. If the hardcopy log is not SYSLOG, then validates the hardcopy device.
   If the hardcopy console is not a console or a display console, or has no UCB or a bad UCB (note. throughout this module 'bad UCB' means that the device type indicated by the UCB is not a supported MCS console), issues the message IEA194I, and uses SYSLOG as the hardcopy log.

3. Selects a new master console if the master specified in the CONSOLxx is powered off, the master is not valid (i.e. no UCB or bad UCB) or no master is defined in CONSOLxx.

   If the master console is powered off, issues the message IEA190I to indicate the reason for not using the specified master console.

   If the master is not valid or the master was powered off, the first input-output capability (full capability) online console is picked from the master's alternate console chain and that console is used as a new master. ROUTCODE, LEVEL and AUTH of the old master console are merged into the new master console's ROUTCODE, LEVEL and AUTH. Issues the message IEA191I to indicate which console is chosen as the master.

   If no master is defined in CONSOLxx or no console is available to be used as a new master in the master's alternate console chain, the first full capability online console specified in CONSOLxx is selected. ROUTCODE, LEVEL and AUTH of the old master are merged into the new master console's ROUTCODE, LEVEL and AUTH. Issues the message IEA191I to indicate which console is chosen as the new master.

   If no online full capability console is available but there is at least one offline full capability console available which may be used as a new master, sets a reason code to indicate that there is no online full capability console and also for specification of a new CONSOLxx member or a re-ipl.

   If no valid full capability console is available to be used, set a reason code to indicate that no master could be selected and also request specification

Diagram 55. IEAVN600/IEAVN613 Part 39 of 53

IEAVN613 - MODULE OPERATION (Continued)

of a new CONSOLxx member.

4. Validates all consoles alternate.

   If the console is a subsystem console, the next console is processed.

   If no alternate console is defined for the console or a bad alternate console is assigned to the console (meaning a printer is used as the alternate for a full capability console or the alternate console is not valid), uses the user defined master as the alternate console if it is a valid online or offline console. Otherwise uses the new selected master as the alternate console. Processes the next console.

5. Calls IEAVN614 to build UCM fixed extension base, UCM fixed extension save area, UCM pageable extension base, UCME pageable extension and UCME fixed extension and initialize the UCM control blocks.

Diagram 55. IEAVN600/IEAVN613 Part 40 of 53

IEAVN613 - DIAGNOSTIC AIDS

ENTRY POINT NAME: IEAVN613

MESSAGES:

```
      IEAVN613 issues the following messages via IEAVM200.
IEA188I - CONSOLxx: NO MASTER CONSOLE SPECIFIED
  Where:
  xx - suffix for CONSOLxx member

IEA190I - CONSOLxx: MASTER CONSOLE xxx(,yyy) WAS OFFLINE DURING
          IPL
  Where:
  xx - suffix for CONSOLxx member
  xxx(,yyy) - console device number(s)

IEA191I - CONSOLE xxx(,yyy) text AS MASTER CONSOLE
  Where:
  xxx(,yyy) - console device number(s)
  text - SELECTED or DEFINED

IEA192I - CONSOLxx iii(,ooo): ALTERNATE DEFAULTED TO xxx(,yyy).
          REASON=rc
  Where:
  xx - suffix for CONSOLxx member
  iii(,ooo) - console device number(s)
  xxx(,yyy) - console device number(s)
  rc = 1    - the alternate console for the console iii(,ooo)
              does not have a CONSOLE statement in CONSOLxx.
              The master console xxx(,yyy) is used as the
              alternate console.
  rc = 2    - the alternate console for the console iii(,ooo)
              is not a valid console.
              The master console xxx(,yyy) is used as the
              alternate console.
  rc = 3    - the alternate console for the console iii(,ooo)
              has an invalid operating mode.
              The master console xxx(,yyy) is used as the
              alternate console.
  rc = 4    - no alternate console for the console iii(,ooo)
              is defined.
              The master console xxx(,yyy) is used as the
              alternate console.

IEA194I - CONSOLxx xxx(,yyy): HARDCOPY DEFAULTED TO SYSLOG.
          REASON=rc
  Where:
  xx - suffix for CONSOLxx member
  xxx(,yyy) - console device number(s)
  rc = 1    - xxx(,yyy) is not a console (ie. no console
              statement for xxx(,yyy))
  rc = 2    - xxx(,yyy) is not valid
  rc = 3    - xxx or yyy is a display console
```

ABEND CODES: None

WAIT STATE CODES: None

RETURN CODES:

```
EXIT NORMAL:

  0 in Register 15
```

Diagram 55. IEAVN600/IEAVN613 Part 41 of 53

IEAVN613 - DIAGNOSTIC AIDS (Continued)

EXIT ERROR:

4 in Register 15

REGISTER CONTENTS ON ENTRY:

```
Register     0 - Irrelevant
Register     1 - Address of a 4-byte field that contains
                 address of the CONSOLxx data area
Registers 2-12 - Irrelevant
Register    13 - Address of a 72-byte save area
Register    14 - Return address
Register    15 - Entry point address of IEAVN613
```

REGISTER CONTENTS ON EXIT:

EXIT NORMAL:

Registers 0-14 - Restored to contents upon entry

EXIT ERROR:

Registers 0-14 - Restored to contents upon entry

Diagram 55. IEAVN600/IEAVN613 Part 42 of 53

IEAVN613 - **STEP 01**

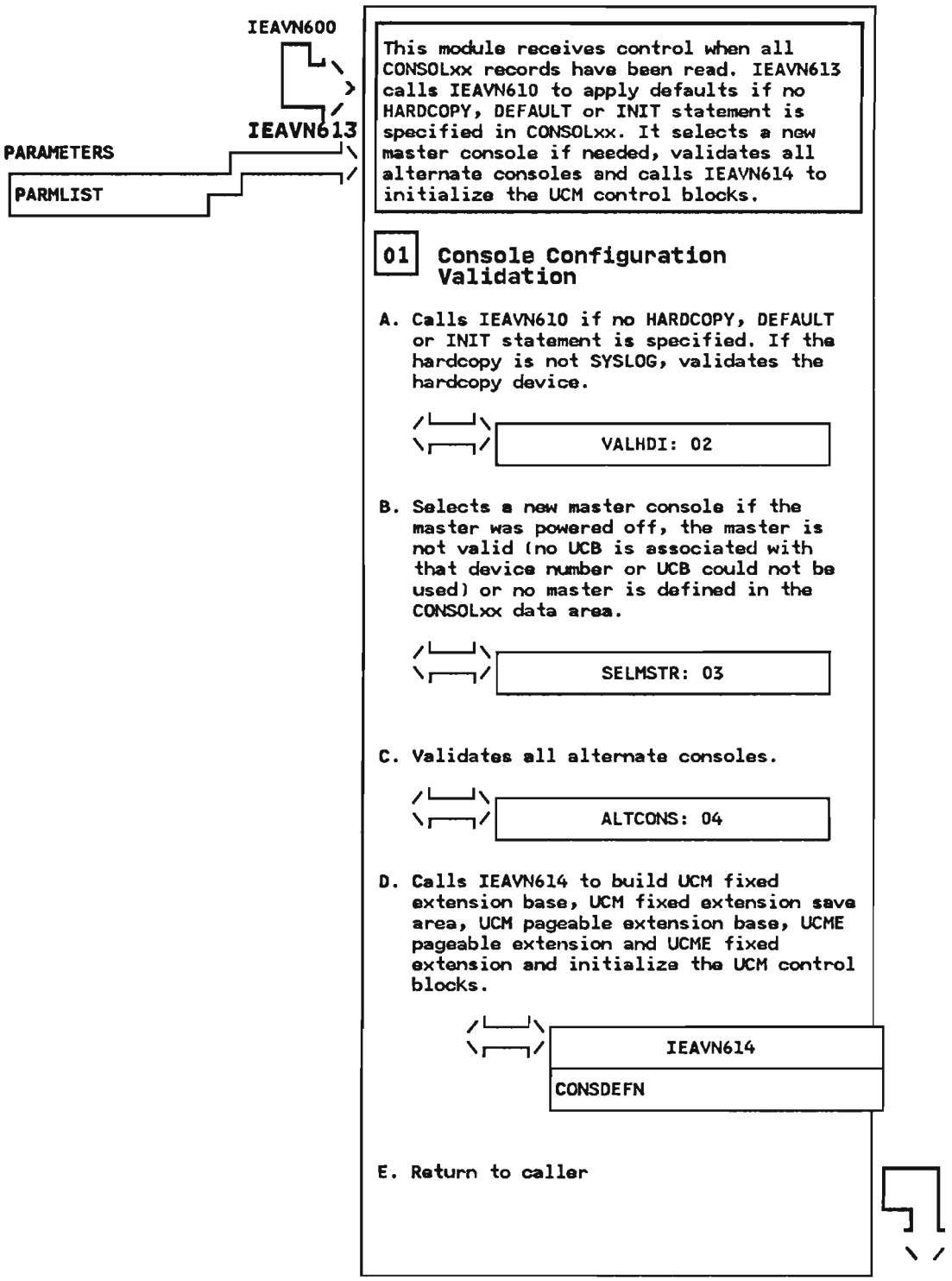


Diagram 55. IEAVN600/IEAVN613 Part 43 of 53

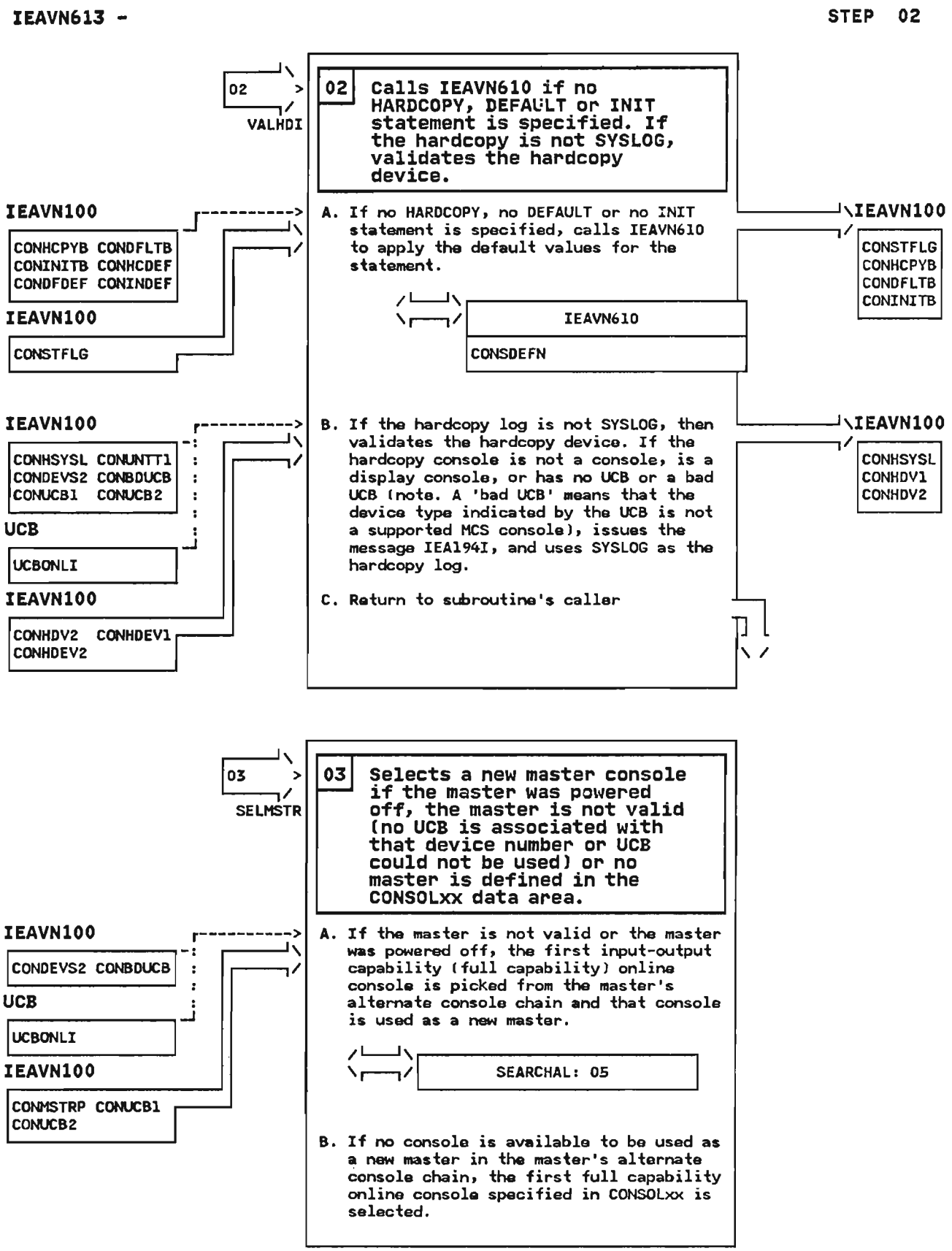


Diagram 55. IEAVN600/IEAVN613 Part 44 of 53

IEAVN613 - **STEP 03C**

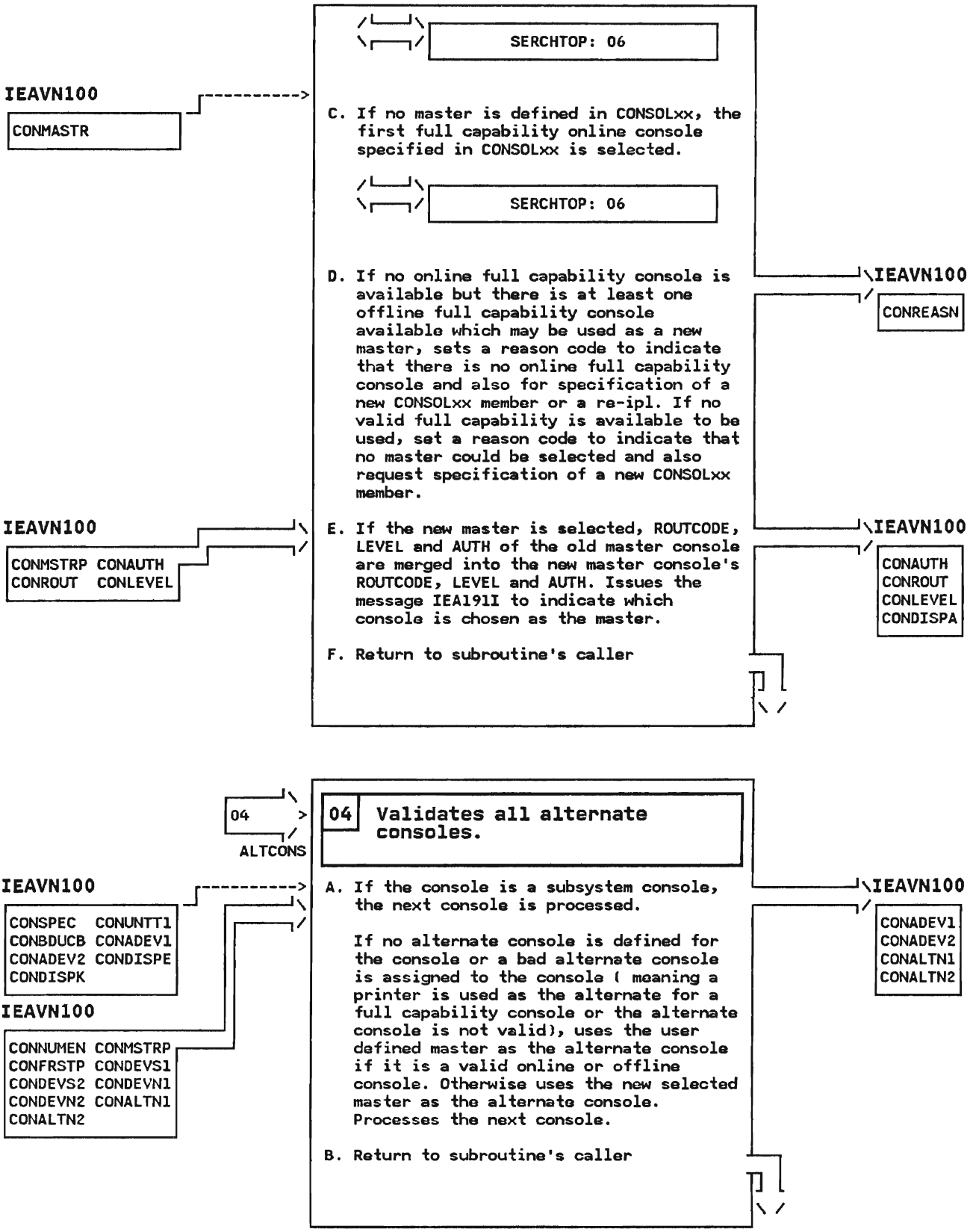


Diagram 55. IEAVN600/IEAVN613 Part 45 of 53

IEAVN613 -

STEP 05

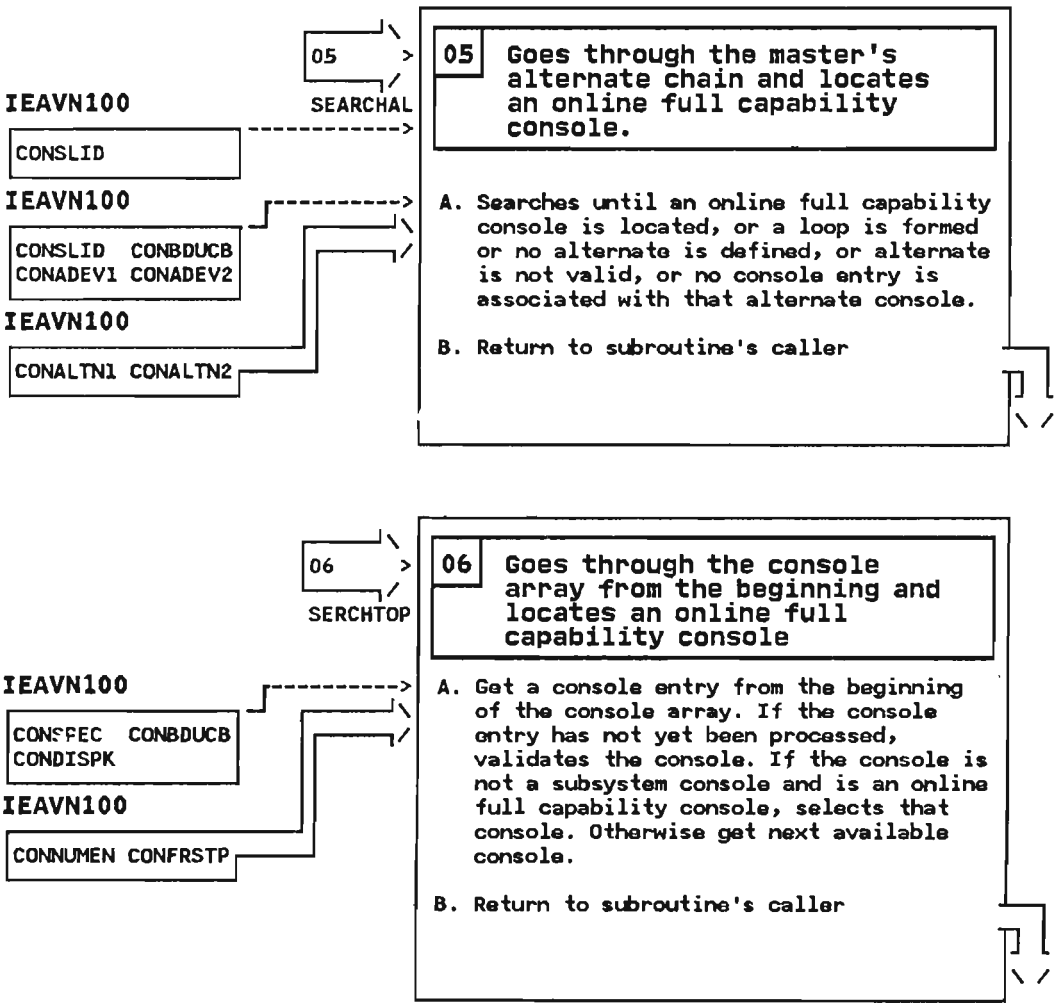


Diagram 55. IEAVN600/IEAVN614 Part 46 of 53

IEAVN614 - MODULE DESCRIPTION

FUNCTION:
Builds extensions to the UCM and initializes the UCM control blocks.

ENTRY POINT: IEAVN614

PURPOSE: See function

LINKAGE: PL/AS Call statement

CALLERS: IEAVN613

INPUT:
Register 1 points to a word that points to the address of the CONSOLxx data area (IEAVN100)

OUTPUT: The UCM control blocks are initialized.

EXIT NORMAL: Returns to the caller.

OUTPUT:
UCM base, MCS prefix, UCMEs, UCM fixed and pageable extension base and its extensions are initialized.

EXTERNAL REFERENCES:

ROUTINES: None

CONTROL BLOCKS:

Common name	Macro Name	Usage
CDEF	IEAVN100	R
BASEA	IEEBASEA	R
SNPL	IEEZB807	M
UCM	IEECUCM	C,M,R
UCB	IEFUCBOB	M,R
CVT	CVT	R
CMDELEM	IHACTM	R

KEY = C-CREATE, D-DELETE, M-MODIFY, R-READ

TABLES: No tables used.

Diagram 55. IEAVN600/IEAVN614 Part 47 of 53

IEAVN614 - MODULE OPERATION

IEAVN614 receives control with register 1 pointing to a word that contains the address of the CONSOLxx data area. IEAVN613 calls IEAVN614 to build and initialize the UCM control blocks.

1.Builds the UCM base and extensions

Recalculates the last UCME address and reinitializes the last UCME pointer (UCMVEL) in the UCM base.

Builds the UCM fixed extension base, UCME fixed extensions, and UCM fixed extension save area from storage obtained from Subpool 245. Initializes the storage with x'00'. Sets the address of the UCM fixed extension base in the UCM base. Puts the UCM fixed extension base's id, the address of the first UCME fixed extension, the address of the last UCME fixed extension, the length of a UCME fixed extension, and the address of the UCM fixed extension save area in the UCM fixed extension base.

Builds the UCM pageable extension base and the UCME pageable extension from storage obtained from Subpool 241. Initializes the storage with x'00'. Puts the address of the UCM pageable extension base in the UCM fixed extension base. Initializes the UCM pageable extension base's id, the address of the first UCME pageable extension, the address of the last UCME pageable extension, and the length of a UCME pageable extension in the UCM pageable extension base.

2.Initialize all UCMEs and their extensions

Initializes each UCME, UCME fixed extension and UCME Pageable extension with the data from the valid console entry in the CONSOLxx data area. If the console is a composite console, sets AUTH in the input half of the composite console. Sets ROUTCODE, MFORM and the command queue in the output of the composite console. Sets USE and LEVEL values in both the input and output halves of the composite console.

- If the console is online, turns on UCBSYSR to indicate that the device is being used as a console
- If the master console is being processed, sets the address of the master's UCME in the MCS Prefix (UCM#MCENT)
- If the console is a display console, turns on UCMSYSF to indicate that display consoles exist. If the console is an active display console, turns on UCMSYSN and UCMSYSB to indicate that an active display console exists and a hardcopy log is required. Increments console count.
- If the console is a non-display console but active, increment the active console count.

The UCME's chain is processed the second time to set the alternate consoles address. If the console is not a subsystem console, locates the UCME address of the alternate console. If the alternate console is a composite, calculates the other half of the alternate console's address.

Diagram 55. IEAVN600/IEAVN 614 Part 48 of 53

IEAVN614 - MODULE OPERATION (Continued)

Otherwise, sets the address of the alternate output UCME to zero.

3.Initializes the MCS Prefix, the UCM Fixed Extension Base and the UCM Pageable Extension Base

---

If the hardcopy is a device, invokes a routine to locate the address of the hardcopy console and initializes UCMHCUCM with that address. Initializes UCMSYSC (in byte 1 of system control flags), UCMSDS1A, UCMSDS1B (in SDS flag) with the CMDLEVEL in the CONSOLxx data area. If hardcopy support is required and NOCMDS is specified, forces the hardcopy to accept CMDS. Copies the hardcopy's routing codes into UCMFHCRT.

Copies the default routing codes into UCMOWTOR of the UCM Pageable Extension Base.

4.Initializes the UCM base, the UCM fixed extension base and the UCM pageable extension base with data from INIT statement.

Diagram 55. IEAVN600/IEAVN614 Part 49 of 53

IEAVN614 - DIAGNOSTIC AIDS

ENTRY POINT NAME: IEAVN614

MESSAGES: None

ABEND CODES: None

WAIT STATE CODES: None

RETURN CODES: None

REGISTER CONTENTS ON ENTRY:

```
Register     0 - Irrelevant
Register     1 - Address of a 4-byte field that contains
                 address of the CONSOLxx data area
Registers 2-12 - Irrelevant
Register    13 - Address of a 72-byte save area
Register    14 - Return address
Register    15 - Entry point address of IEAVN614
```

REGISTER CONTENTS ON EXIT:

```
EXIT NORMAL:

  Registers 0-14 - Restored to contents upon entry
```

Diagram 55. IEAVN600/IEAVN614 Part 50 of 53

IEAVN614 - **STEP 01**

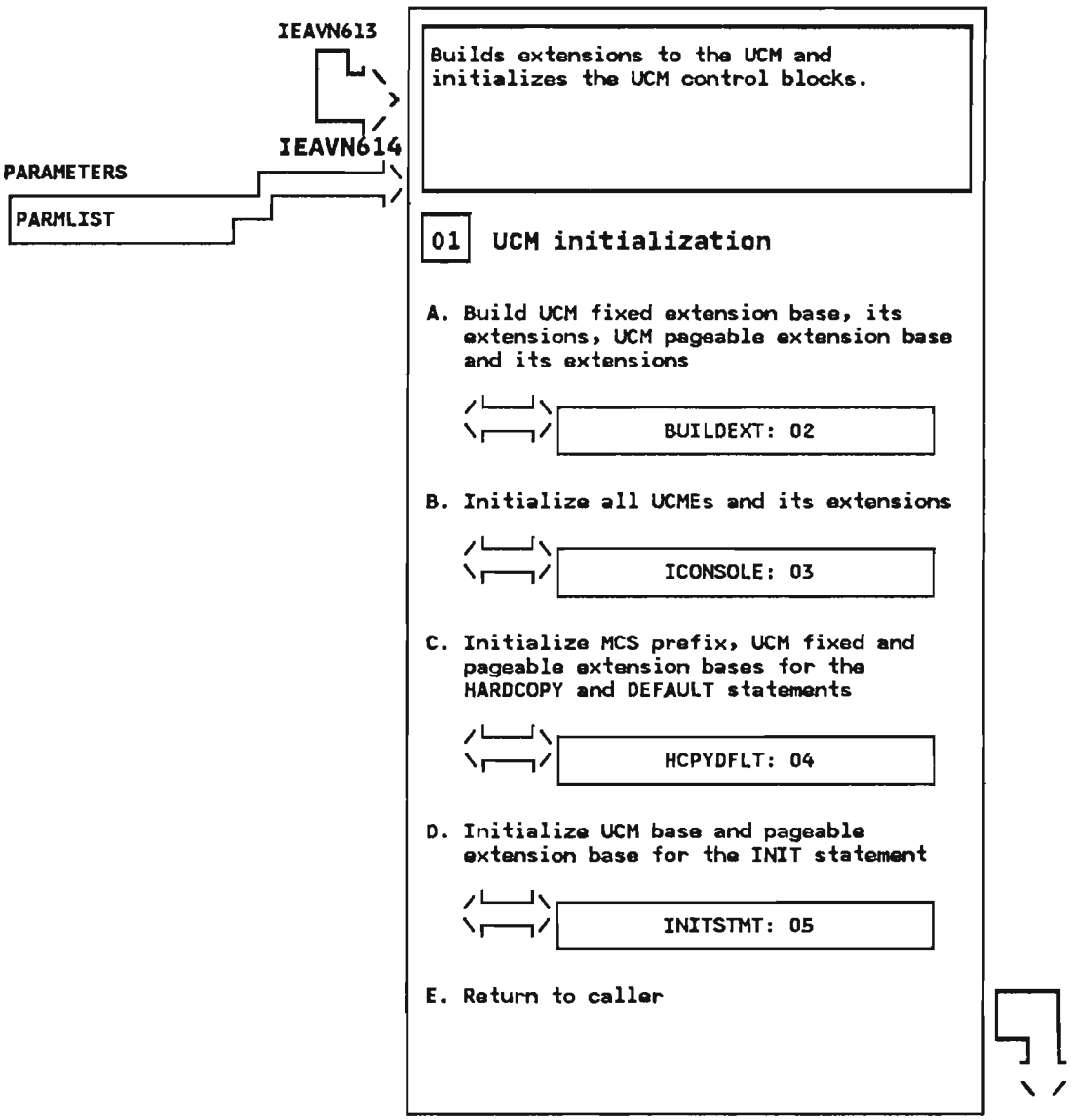


Diagram 55. IEAVN600/IEAVN614 Part 51 of 53

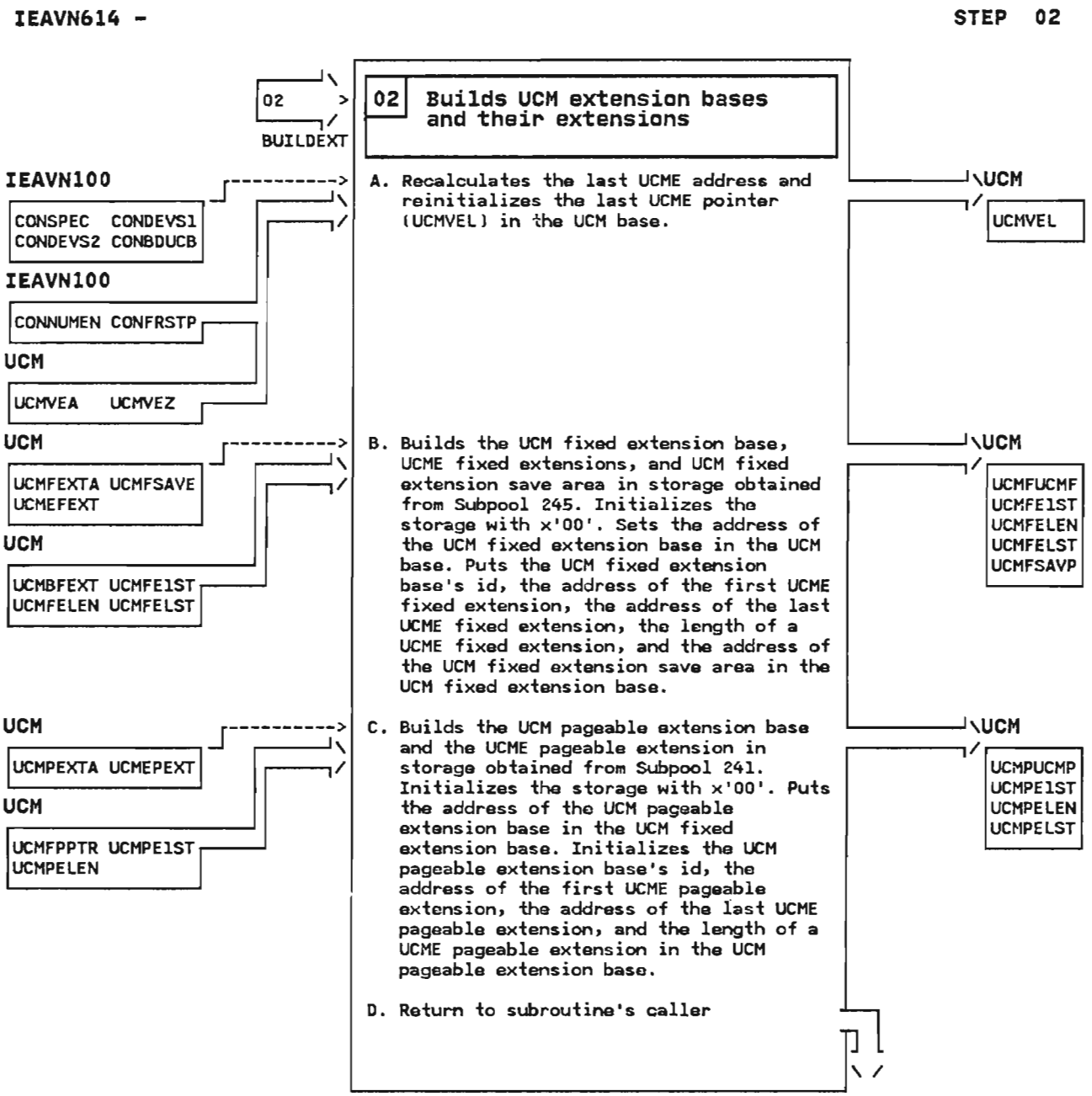


Diagram 55. IEAVN600/IEAVN614 Part 52 of 53

IEAVN614 - **STEP 03**

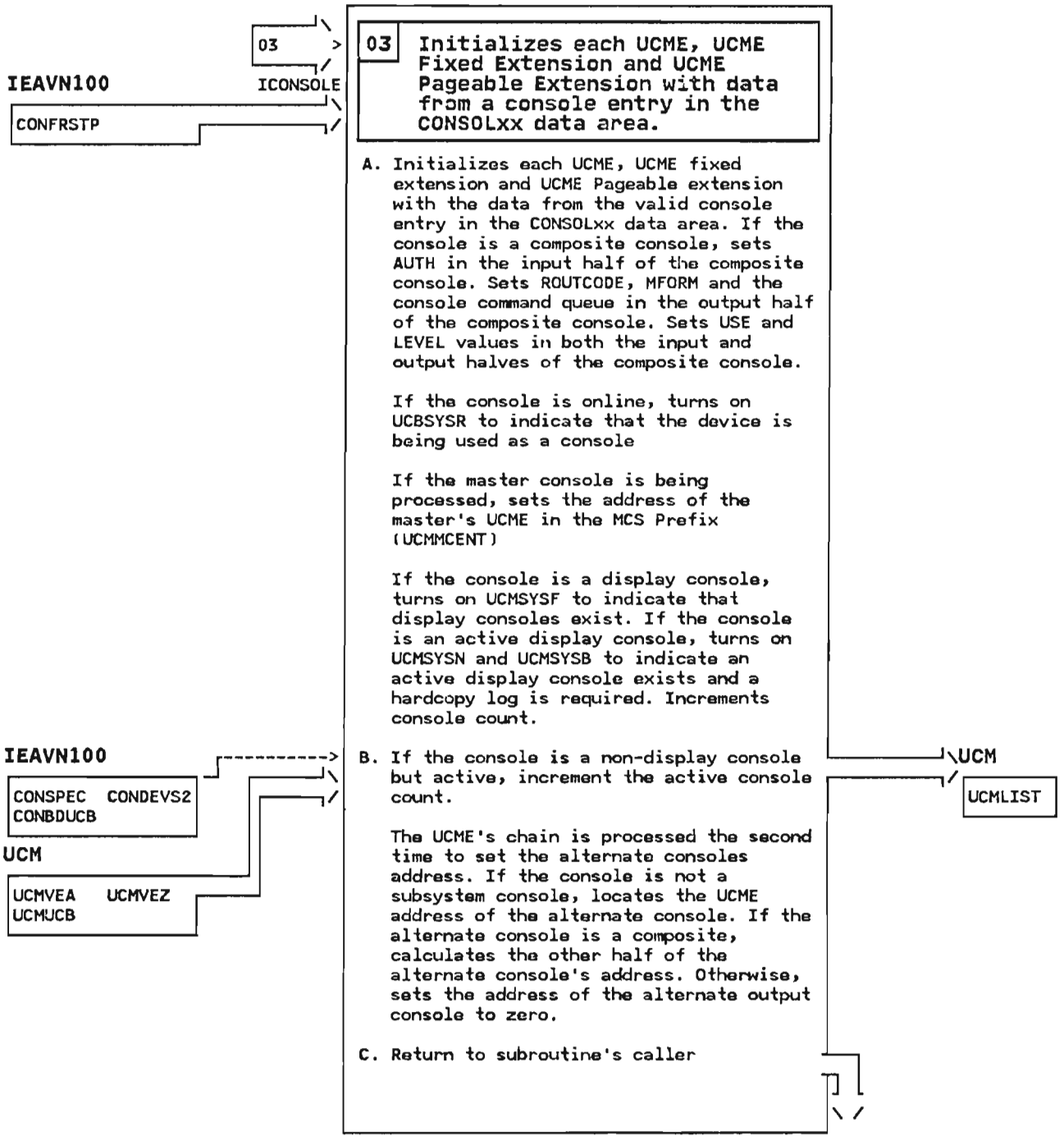


Diagram 55. IEAVN600/IEAVN614 Part 53 of 53

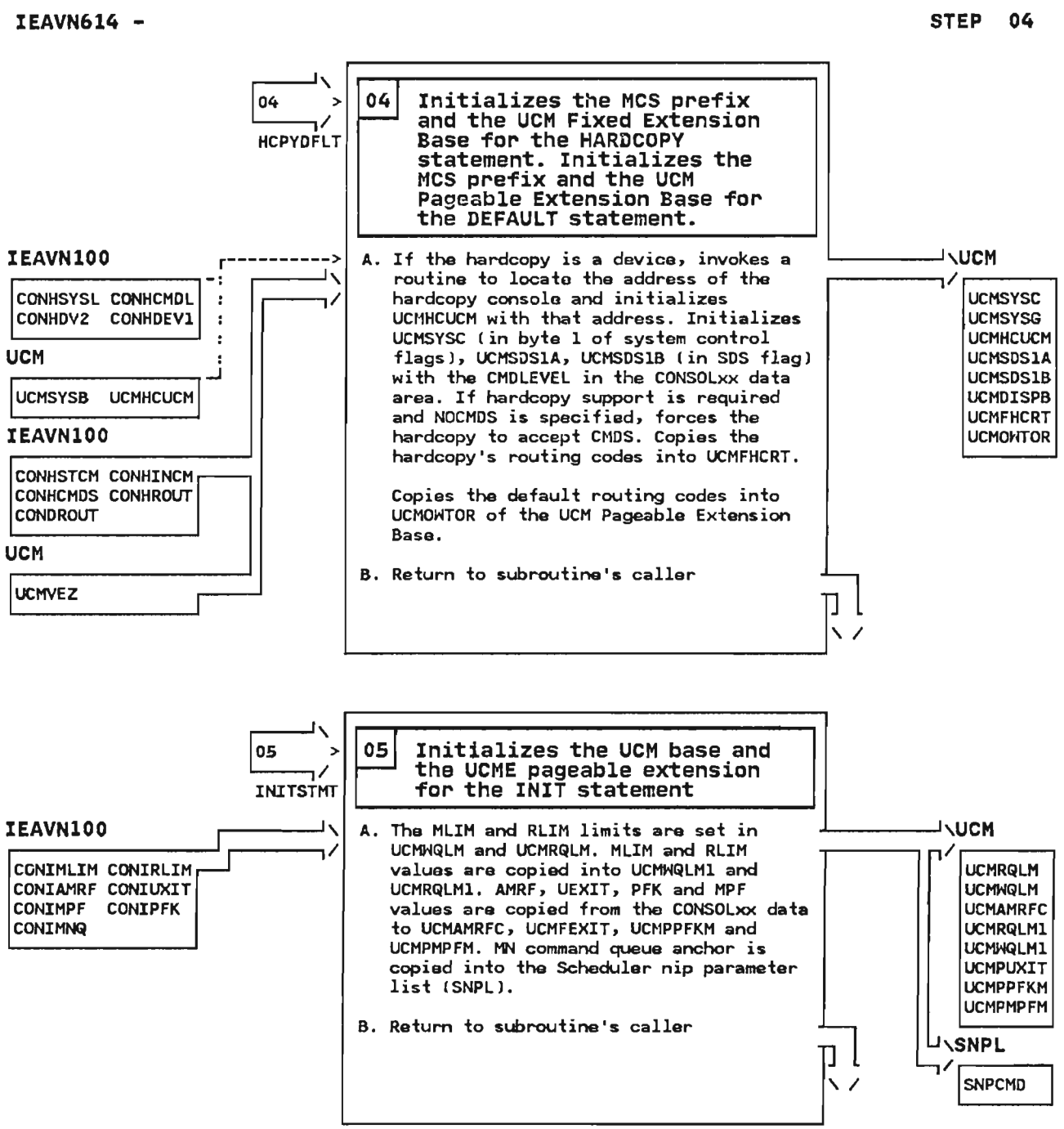


Diagram 56. Appendate Name Table Initialization (IEAVNP16) Part 1 of 2

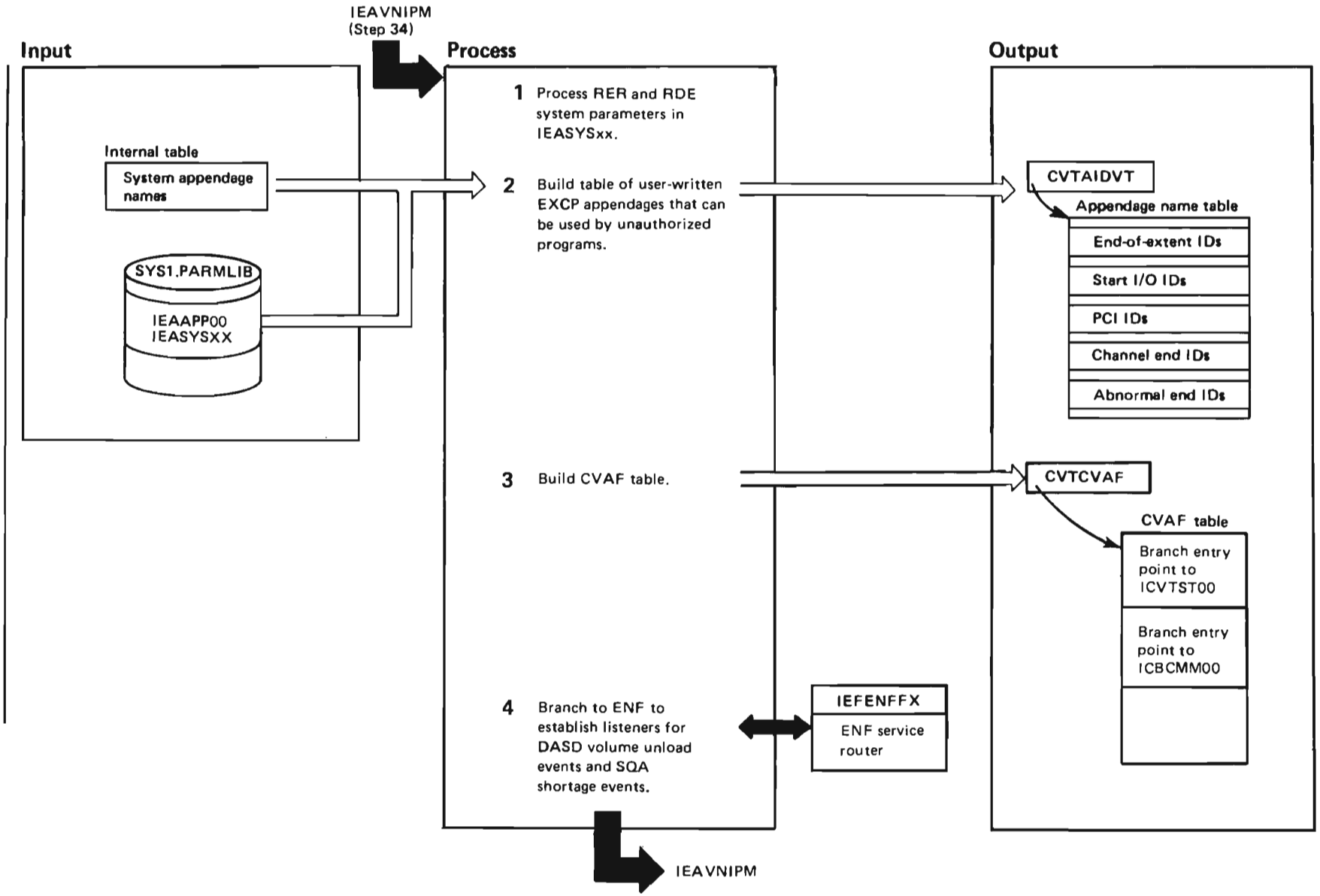


Diagram 56. Appendate Name Table Initialization (IEAVNP16) Part 2 of 2

Extended Description	Module	Label
IEAVNIPM calls the data management RIM, IEAVNPI6, to build a table of user-written EXCP appendages that can be used by unauthorized programs. In order for a program to use a user-written EXCP appendage (for example, a start I/O routine) during OPEN processing, one of the following conditions must exist: • The program is operating under system protect key 0-7. • The program is authorized under the authorized program facility (see Diagram IEAVNPA5). • The ID of the user-written EXCP appendage is in the appendage name table. The data management RIM builds this table as described in the next paragraph.	IEAVNP16	
1 The RIM processes the RER and RDE parameters in SYS1.PARMLIB member IEASYSxx.		
2 The data management RIM contains an internal table of EXCP appendage names used by system programs operating in a problem program state. (For the format of the EXCP appendage name table, refer to *Open/Close/EOV Logic*.) The RIM copies the system appendage IDs into the appendage name table. The installation specifies the names of the user-written appendages in member IEAAPP00 of SYS1.PARMLIB. The RIM reads records from IEAAPP00 and copies the user-written appendage IDs into the appendage name table. The RIM places the address of the appendage name table in field CVTAIDVT of the CVT. Data management's OPEN processing will check the appendage name table if it receives a request for an EXCP appendage from an unauthorized program.	IEAVNP16 IEAVNP16	CPYPRIME MOVEID
3 The RIM issues a GETMAIN for 4096 bytes of CSA and uses 28 bytes to build the common VTOC access facility (CVAF) table containing branch entry points to CVAF modules ICVTST00 and ICBCMM00. (Refer to CVAF Diagnosis Reference for additional information.) The RIM places the address of the CVAF table in field CVTCVAF of the CVT. The Rim uses the remaining 4068 bytes to construct VTOC information blocks.		
4 The RIM branches to ENF to: 1. establish CVAF module ICBLSN00 as a listener for DASD volume unload events. 2. establish media manager ICYELE as a listener for SQA shorage events. (Refer to MVS Media Manager Diagnosis Guide and Reference for additional information.) For information concerning the logic of IEFENFFX, the ENF service router refer to *System Logic Library*.	IEFENFFX	

Diagram 57. Master Scheduler RIM (IEAVNP13) Part 1 of 4

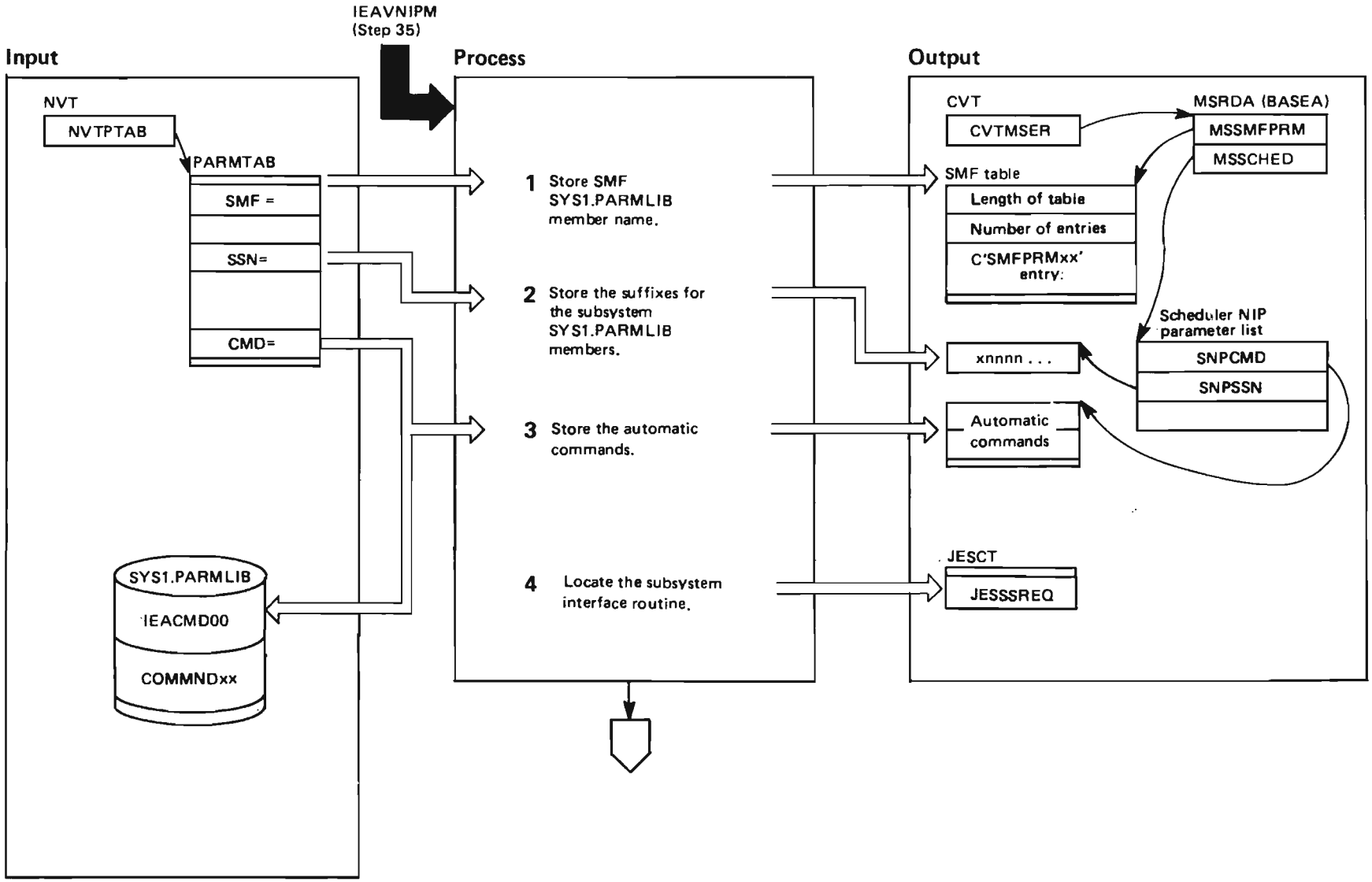


Diagram 57. Master Scheduler RIM (IEAVNP13) Part 2 of 4

Extended Description	Module	Label
Some information passed to NIP in the form of system parameters and SYS1.PARMLIB members will be used during master scheduler and timer initializations. To obtain this information and store it for later use NIP calls the master scheduler RIM, IEAVNP13.		
1 Master scheduler initialization uses SMF parameters specified in the SMFPRMxx member of SYS1.PARMLIB. Using the SMF system parameter, the installation can specify the suffix for the SMFPRMxx member. The master scheduler RIM obtains the suffix value from field SMF in PARMTAB. Then, the RIM builds a table that contains the SMFPRMxx member name specified. If the installation does not specify a suffix, the RIM uses the default member name SMFPRM00. The RIM places the address of the SMF parameter table in field MSSMFPRM of the master scheduler's resident data area (IEEBASEA). The master scheduler RIM obtains storage from subpool 241 for the scheduler NIP parameter list (IEEZB807) and stores its address in MSSCHED of the master scheduler resident data area (IEEBASEA).	IEAVNP13	SMFPROC
2 During master scheduler initialization, the subsystems specified in the IEFSSNxx members of SYS1.PARMLIB will be initialized. The installation uses the SSN system parameter to specify the suffixes for the IEFSSNxx members. The master scheduler RIM obtains the address of the suffix values from field SSN of PARMTAB and stores these values in SNPSSN. The suffix value is in the form 'xnnnn . . .', where x is the number of PARMLIB members and nnnn . . . are 2-character suffixes. The RIM stores the address of SNPSSN in the scheduler NIP parameter list. If the installation does not specify the SSN parameter, the RIM tries to locate the default member IEFSSN00. If the default exists, only the suffix '00' is stored. If the RIM cannot locate the default, it continues processing.		
3 Master scheduler initialization will process automatic commands contained in IEACMD00 and COMMNDxx members of SYS1.PARMLIB (see Step 18 of Diagram 63, "Master Scheduler Initialization"). NIP always uses member IEACMD00. Using the CMD system parameter, the installation can specify the suffixes for the COMMNDxx members of SYS1.PARMLIB that contain the automatic commands. The master scheduler RIM obtains the suffix values from field CMD of PARMTAB, then locates and reads the specified COMMNDxx members. If the installation does not specify the CMD parameter, the		SYNTXCHK
RIM uses the default member COMMND00. If the installation does not supply a COMMND00 member, the RIM uses only the automatic commands contained in IEACMD00. The RIM builds a queue of the commands contained in IEACMD00 and each COMMNDxx member. Each command element in the queue contains a pointer to the next command in the queue, the length of the current command, and the commands itself in the form the operator would enter it from the console.		READCMPS
4 The master scheduler RIM loads the subsystem interface routine, IEFJSREQ, stores its address in the JESCT, and then deletes the routine.		

Diagram 57. Master Scheduler RIM (IEAVNP13) Part 3 of 4

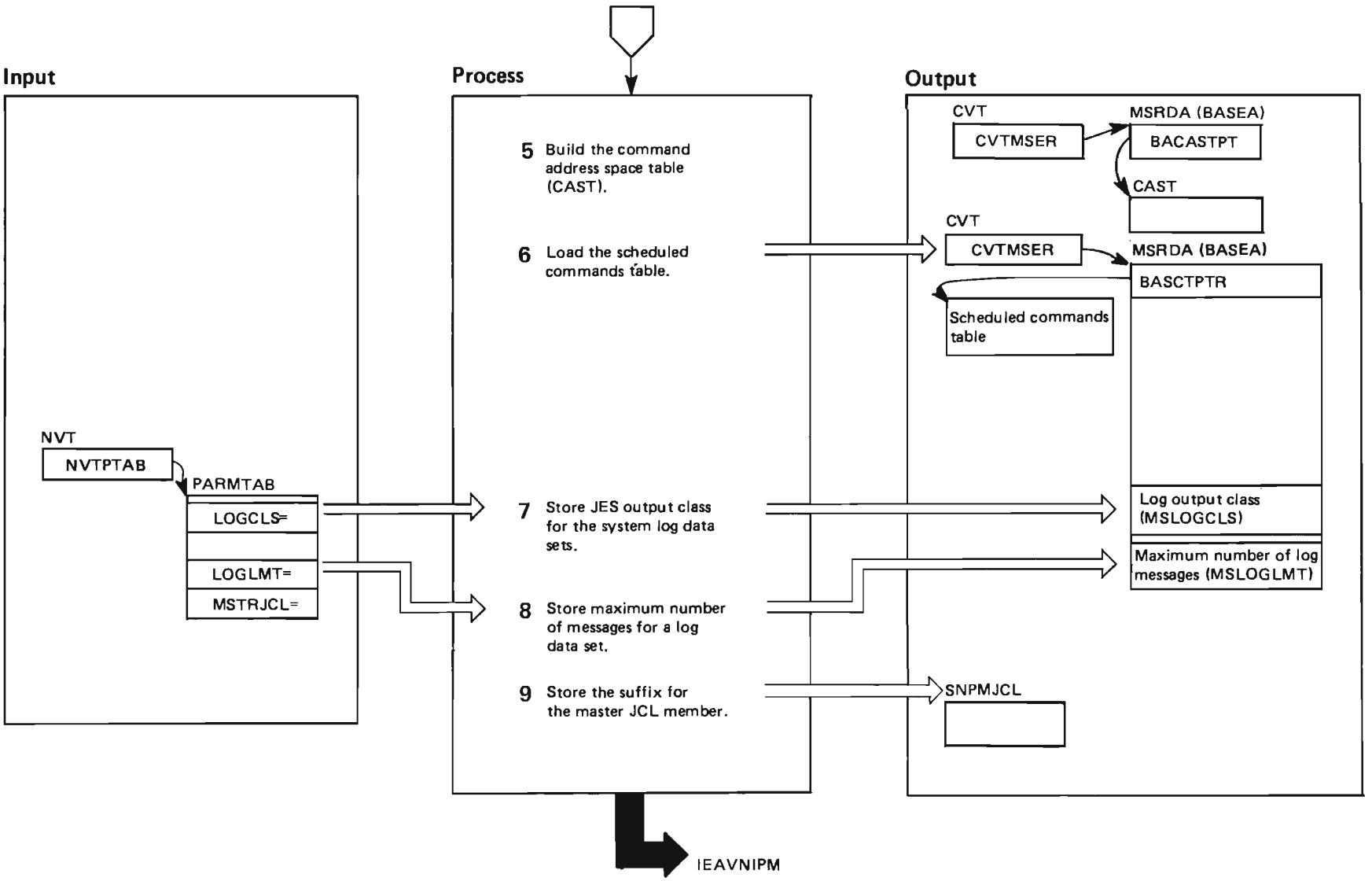


Diagram 57. Master Scheduler RIM (IEAVNP13) Part 4 of 4

Extended Description	Module	Label	
5 The master scheduler RIM gets storage for the command address space table (CAST) in common storage and initializes it.		BLDCAST	9 The master scheduler RIM uses the MSTRJCL system parameter to complete the name of the MSTJCLxx member located in SYS1.LINKLIB. The MSTRJCL parameter allows the installation to specify the suffix value for the MSTJCLxx member that the master scheduler RIM uses to start the master scheduler address space.
6 The RIM loads the scheduled commands table (module IEEMB884) into common storage and stores its address in BASEA.			
7 Using the LOGCLS system parameter, the installation can specify the JES output class for the system log data sets. Master scheduler initialization uses the LOGCLS value later during system initialization when it initializes the system log. The master scheduler RIM obtains the LOGCLS value from PARMTAB, validity checks the value, then places it in field MSLOGCLS of the master scheduler resident data area (IEEBASEA).		LOGRTN	The master scheduler RIM obtains the suffix value from the PARMTAB parameter list (field MSTRJCL) and stores it in SNPMJCL in the scheduler NIP parameter list. If the installation does not specify a suffix, the RIM uses the default member, MSTJCL00. The master scheduler RIM issues a LOAD macro instruction for the specified MSTJCLxx member to determine that it exists. If the RIM cannot find the member, it prompts the operator to respecify a new value, and this procedure will repeat until the specified member can be found or the operator enters an end-of-block (EOB). If an EOB occurs, the RIM uses MSTJCL00. If the RIM cannot find MSTJCL00, it prompts the operator to respecify. This procedure repeats until the operator issues an end-of-block.
8 The installation uses the LOGLMT system parameter to specify the maximum number of messages to be written to the log data set. Master scheduler initialization uses the LOGLMT value when it initializes the system log. The master scheduler RIM obtains the LOGLMT value from PARMTAB, validity checks the value, then places it in field MSLOGLMT of the master scheduler resident data area (IEEBASEA).			

Diagram 58. Generalized Trace Facility Initialization (IEAVNP17) Part 1 of 2

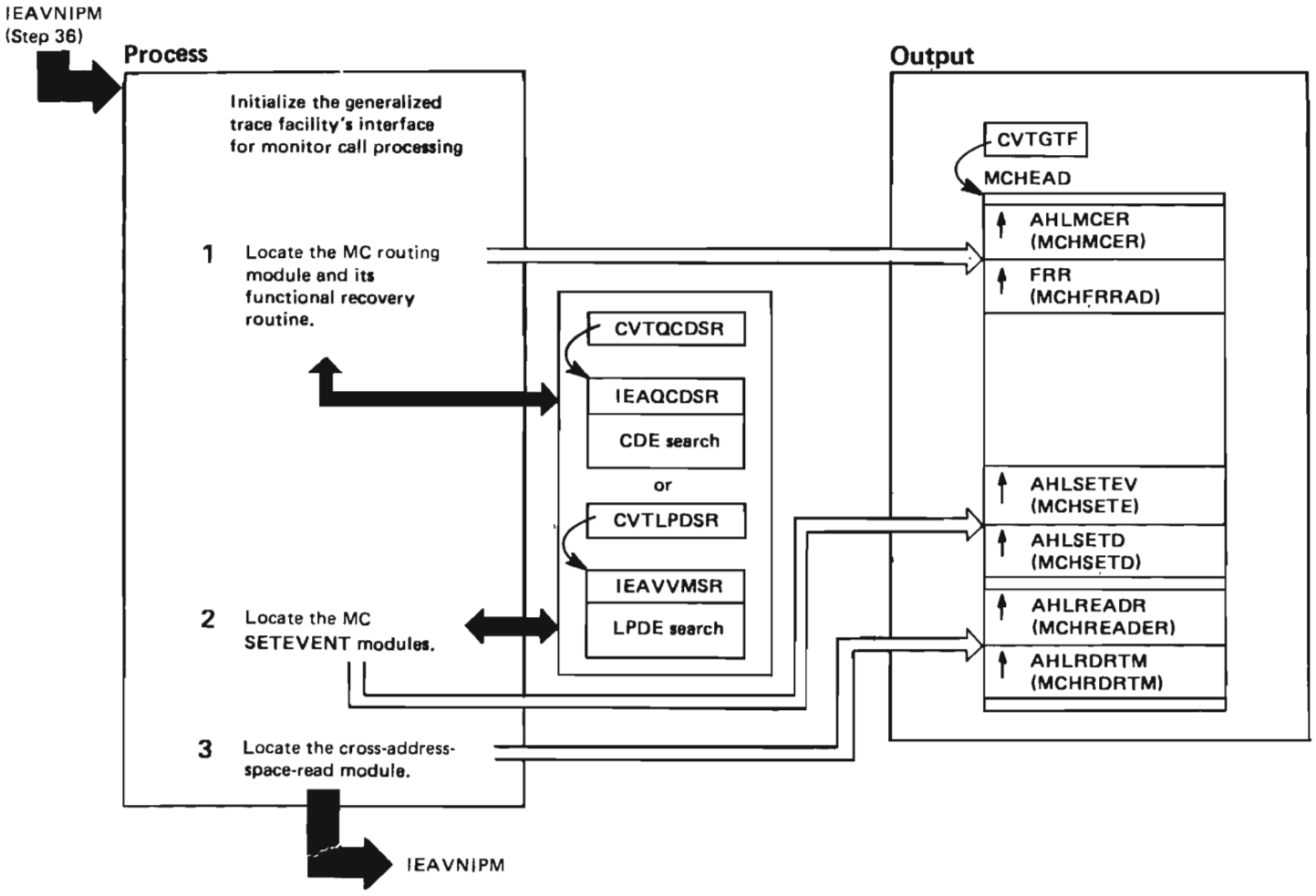


Diagram 58. Generalized Trace Facility Initialization (IEAVNP17) Part 2 of 2

Extended Description	Module	Label
The generalized trace facility (GTF) traces system and problem program events. A monitor call (MC) instruction causes an interruption to notify GTF to trace an event. Therefore, GTF sets up an interface with monitor call processing.		
1 Before tracing an event, GTF passes control to the MC routing module (AHLMCER); the routing module checks the SETEVENT tables to determine whether the event is one that GTF has selected for tracing. The GTF RIM locates AHLMCER and its functional recovery routine in order to save their addresses in the monitor call table head (MCHEAD). First, the RIM passes control to the nucleus-resident CDE search routine (IEAQCDSR). IEAQCDSR searches the CDE chain to determine whether AHLMCER is in the fixed link pack area or its modifications. If not, the RIM passes control to the nucleus-resident LPA directory search routine (IEAVVMSR). IEAVVMSR searches the LPDEs to determine whether AHLMCER is in the pageable link pack area. When either search routine finds AHLMCER, the RIM uses the indicated CDE or LPDE to locate the doubleword at the beginning of AHLMCER. This doubleword contains the entry point address of AHLMCER and its FRR. The RIM moves the addresses into fields MCHMCER and MCHFRRAD of MCHEAD. If neither search routine finds AHLMCER, the RIM issues a message to warn the operator that monitor call routing is inactive and restores MCHEAD to its original values. The original MCHEAD values are addresses of code that sets a return code indicating that MC routing is inactive.	IEAVNP17 IEAQCDSR IEAVVMSR IEAVNP17	IEAVNP17 IEAVNP17 ERROR
2 During processing, GTF issues the SETEVENT macro instruction to specify the events it will trace. Two MC SETEVENT modules (AHLSETEV and AHLSETD) build and manipulate MC tables that indicate the selected events. Using the two nucleus-resident search routines described in the preceding paragraph, the GTF RIM locates the MC SETEVENT modules and places their addresses into fields MCHSETE and MCHSETD of MCHEAD. If the search routines cannot find both modules, the GTF RIM issues a message to warn the operator that MC routing is inactive, and restores MCHEAD to its original values. The original MCHEAD values are addresses of code that sets a return code indicating that MC routing is inactive.		IEAVNP17 ERROR
3 GTF provides a cross-address-space-read function that allows ABDUMP, SNAP, and SVC dump to process GTF trace buffers. The GTF RIM locates the cross-address-space read module (AHLREADR) and places its entry point address and error routine address into fields MCREADR and MCRDRTM, respectively.		

Diagram 59. Scheduler Interface to the General Parmlib Scan Routine (IEAVNP18) Part 1 of 7

IEAVNP18 - MODULE DESCRIPTION

DESCRIPTIVE NAME: Scheduler Interface to the General Parmlib Scan Routine

FUNCTION:
This module issues LOADs for the statement processor routines associated with SCHEDxx, initializes a table (PSCNSTAB) to contain the statement types, statement processor routine names and statement processor routine addresses, and invokes the General Parmlib Scan Routine (IEEMB888) to process SCHEDxx.
Upon return from IEEMB888, this module deletes the statement processor routines.

ENTRY POINT: IEAVNP18

PURPOSE: See Function

LINKAGE: CALL

CALLERS: NIP Control/Services Routine (IEAVNIPM)

INPUT: NVT - NIP Vector Table

OUTPUT:
General Parmlib Scan Parameter List (IEEZB819):

FIELD	LENGTH/MASK	DESCRIPTION
PSCNP	40	General Parmlib Scan Parameter List
PSCNID	4	Identifier 'PSCN'
PSCNVERS	1	Version number
PSCNFLGS	1	Flags
PSCNLIST	X'80'	Write the SYS1.PARMLIB member records to the hardcopy log
PSCNEOB	X'40'	EOB was entered in response to a prompt
PSCNLEN	2	Length of parameter list
PSCNTABL	4	Pointer to the table of valid statement types and corresponding statement subfunction routines
PSCNMEM	6	SYS1.PARMLIB member name
PSCNSTMT	2	Number of statement types
PSCNENTY	4	The address of an area which contains the member suffixes to be appended to PSCNMEM.
PSCNUSEP	4	Pointer to a user parameter area
PSCNNVTP	4	Pointer to the NVT
PSCNRSV1	8	Reserved
PSCNSTAB		Table of statement types and statement subfunction routine names
PSCNELMT(*)	23	
PSCNTYPE	10	Statement type
PSCNNAME	8	Statement processor routine name
PSCNADDR	4	Statement processor routine address
PSCNFLAG	1	Reserved for use by IEEMB888

EXIT NORMAL: Return to caller

EXIT ERROR: Return to caller

EXTERNAL REFERENCES:

Diagram 59. Scheduler Interface to the General Parmlib Scan Routine (IEAVNP18) Part 2 of 7

IEAVNP18 - MODULE DESCRIPTION (Continued)

```
ROUTINES:
   IEEMB888 - Invoked to process the SYS1.PARMLIB
              member, SCHEDxx.

CONTROL BLOCKS:
   ASCB     -  Address Space Control Block
   CVT      -  Communication Vector Table
   ECB      -  Event Control Block
   IEAPMNIP -  NIP mapping for NIPWTO, NIPWTOR and NIPMOUNT
   IEAPPNIP -  NIP Parameter Area Mapping
   NVT      -  Nucleus Initialization Vector Table
   PSA      -  Prefix Save Area
```

Diagram 59. Scheduler Interface to the General Parmlib Scan Routine (IEAVNP18) Part 3 of 7

IEAVNP18 - MODULE OPERATION

This module does the following:

1. Issues LOADs for the statement processor routines.
2. Initializes the table of valid statement types (PSCNSTAB) with the following statement types for SYS1.PARMLIB member, SCHEDxx.
   - EDT
   - MT
   - NORESTART
   - PPT
   - RESTART
3. Stores the statement processor routine names and addresses in PSCNSTAB.
4. Sets PSCNSTMT to the number of statement types.
5. Sets PSCNTABL to the address of PSCNSTAB.
6. Initializes PSCNMEM to contain the characters SCHED, to which the two character combinations (which are specified via SCH=(xx)) are appended to form the SYS1.PARMLIB member, SCHEDxx.
7. Initializes PSCNENTY to the address of a word which points to an area containing suffix values for SCHEDxx.
8. Initializes the version number and identifier.
9. Stores the NVT pointer into PSCNNVTP.
10. If the SYS1.PARMLIB member is to be written to the hardcopy log, then sets bit PSCNLIST sithin IEEZB819 to indicate this.
11. Issues a LOAD for IEEMB888.
12. Invokes IEEMB888 to process the SCHEDxx SYS1.PARMLIB members.
13. If invalid suffix values were specified then IEAVNP18 issues prompt IEA906A to obtain valid SYS1.PARMLIB specifications.
    IEAVNP18 then invokes IEEMB888 with a parameter list and an initialized table of data.
    IEAVNP18 will invoke IEEMB888 until valid specifications are received.
14. If EOB is entered in response to the prompt, sets bit PSCNEOB within IEEZB819.
    This will inform IEEMB888 to perform end-of-processing (EOP) functions.
15. Issues DELETEs for the statement processor routines.
16. Issues a DELETE for IEEMB888.
17. Returns to the caller.

RECOVERY OPERATION:

The processing in this module is done during Nucleus Initialization (NIP) time and SVC functions are not yet available. Therefore, an ESTAE environment has no effect at this time.

Diagram 59. Scheduler Interface to the General Parmlib Scan Routine (IEAVNP18) Part 4 of 7

IEAVNP18 - DIAGNOSTIC AIDS

ENTRY POINT NAME: IEAVNP18

MESSAGES:

IEA906A RESPECIFY prm PARM OR ENTER EOB

This message appears when a NIP PROMPT is issued. IEAVNP18 issues this message when invalid suffix values are specified in response to SPECIFY SYSTEM PARAMETERS (for member SCH).

The parameter "prm" is "SCH" which is specified in response to SPECIFIY SYSTEM PARAMETERS for the SCHEDxx SYS1.PARMLIB member.

ABEND CODES: None

WAIT STATE CODES: None

RETURN CODES:

EXIT NORMAL:

Register 15 = 0

EXIT ERROR:

Register 15 = 0

REGISTER CONTENTS ON ENTRY:

```
Register  0    = Undefined
Register  1    = N/A
Register  2    = Address of the NIP Vector Table (NVT)
Register  3    = Address of the Communications Vector
                 Table (CVT)
Registers 4-12 = Undefined
Register  13   = Address of 18-word save area
Register  14   = Return address
Register  15   = Entry point address
```

REGISTER CONTENTS ON EXIT:

```
Register  0    = Restored
Register  1    = Address of a word which contains the
                 address of IEEZB821
Registers 2-12 = Restored
Register  13   = Address of 18-word save area
Register  14   = Return address
Register  15   = Return code
```

Diagram 59. Scheduler Interface to the General Parmlib Scan Routine (IEAVNP18) Part 5 of 7

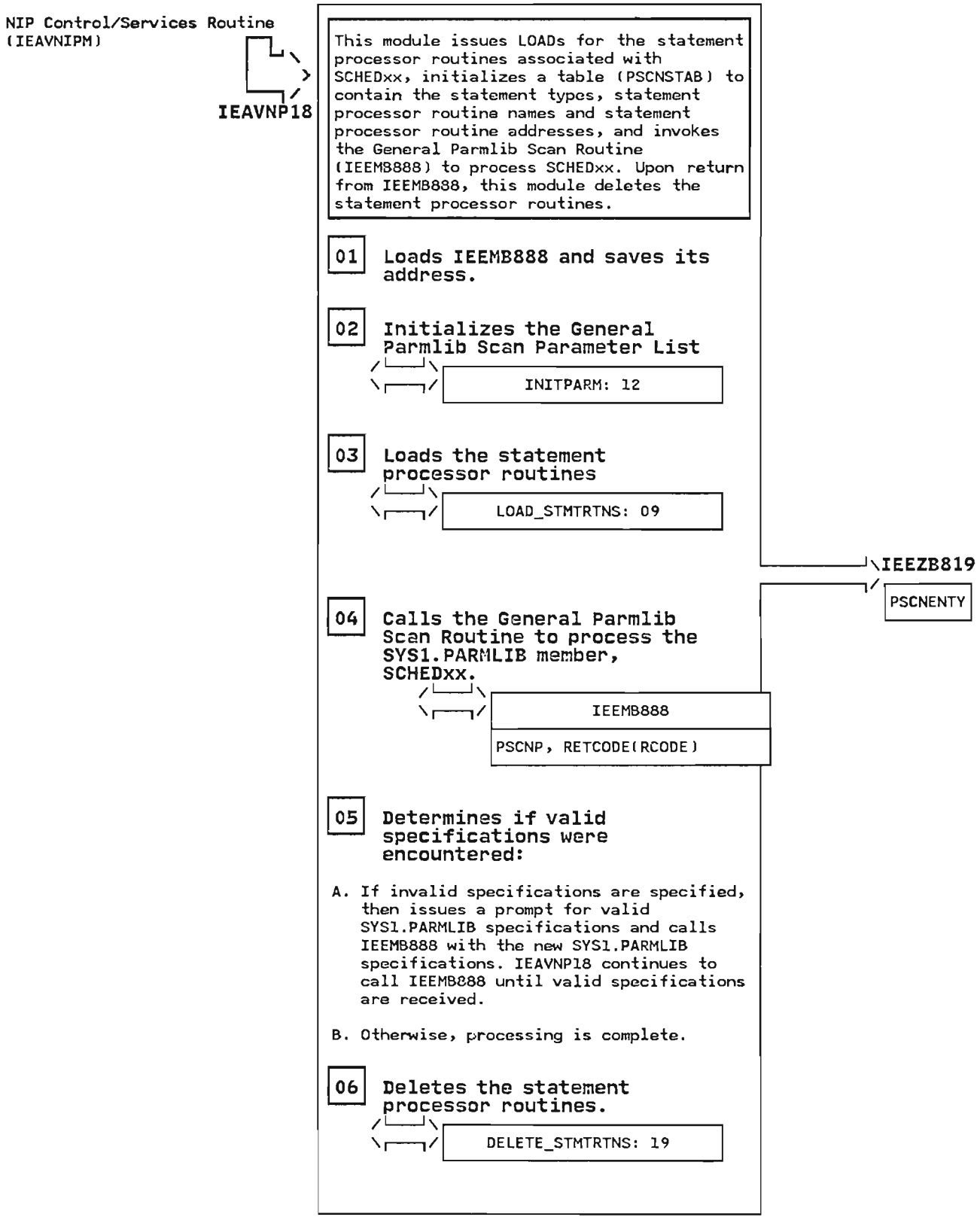


Diagram 59. Scheduler Interface to the General Parmlib Scan Routine (IEAVNP18) Part 6 of 7

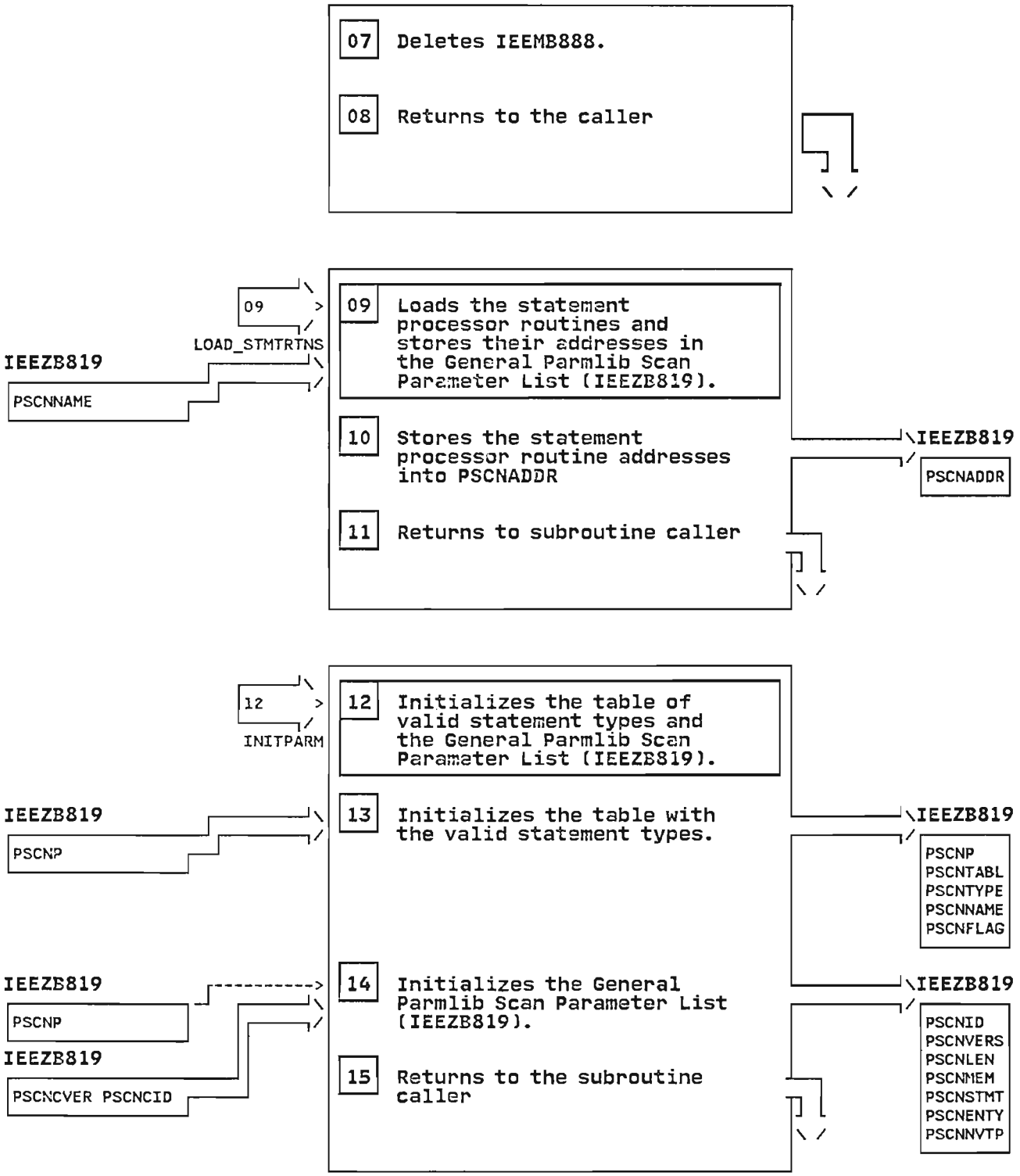


Diagram 59. Scheduler Interface to the General Parmlib Scan Routine (IEAVNP18) Part 7 of 7

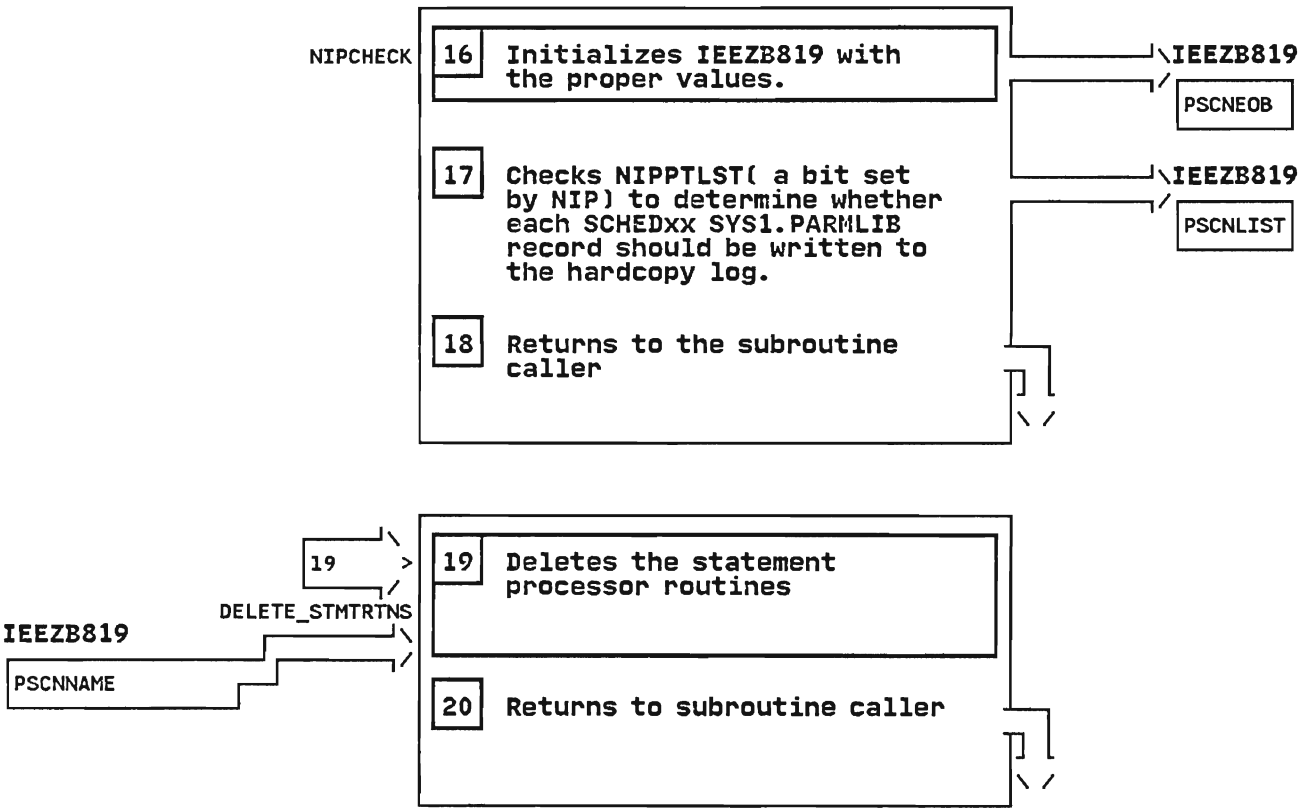


Diagram 60 General Parmlib Scan Routine (IEEMB888) Part 1 of 19

IEEMB888 - MODULE DESCRIPTION

DESCRIPTIVE NAME: General Parmlib Scan Routine

FUNCTION:
This module scans specified SYS1.PARMLIB member(s) for valid statement types, and invokes the proper statement processor routine to process these statement types.
This module may be invoked to process one or more of the same SYS1.PARMLIB members (e.g. SCHEDxx, CLOCKxx, etc.).

ENTRY POINT: IEEMB888

PURPOSE: See Function

LINKAGE: CALL

CALLERS:
- IEAVNP18 - Scheduler Interface to the General Parmlib Scan Routine
- IEAVNP20 - CLOCK processing RIM
- IEAVNP25 - Nucleus SVC table RIM

INPUT:
General Parmlib Scan Parameter List (IEEZB819)

FIELD	LENGTH/MASK	DESCRIPTION
PSCNP	40	General parmlib scan parameter list
PSCNID	4	Identifier 'PSCN'
PSCNVERS	1	Version number
PSCNFLGS	1	Flags
PSCNLIST	X'80'	Write the SYS1.PARMLIB member records to the hardcopy log
PSCNEOB	X'40'	EOB is entered in response to a prompt
PSCNLEN	2	Length of parameter list
PSCNTABL	4	Pointer to the table of valid statement types and the corresponding statement processor routine names
PSCNMEM	6	SYS1.PARMLIB member name
PSCNSTMT	2	Number of statement types
PSCNENTY	4	The address of an area which contains the member name suffixes to be appended to PSCNMEM.
PSCNUSEP	4	Pointer to a user parameter area
PSCNNVTP	4	Pointer to the NVT
PSCNRSV1	8	Reserved
PSCNSTAB		Table of valid statement types and
PSCNELMT(*)	23	statement processor routine names
PSCNTYPE	10	Valid statement type
PSCNNAME	8	Valid statement processor routine name
PSCNADDR	4	Address of a statement processor routine
PSCNFLAG	1	Reserved

OUTPUT:
Statement Processor Parameter List (IEEZB821)

FIELD	LENGTH/MASK	DESCRIPTION
STMTP	36	Statement processor parameter list
STMTID	4	Identifier 'STMT'
STMTVERS	1	Version number

Diagram 60 General Parmlib Scan Routine (IEEMB888) Part 2 of 19

IEEMB888 - MODULE DESCRIPTION (Continued)

STMTFLAG		1	Control Flags
STMTENT	X'80'		STMT type entry
STMTEOP	X'40'		EOP type entry
STMTERR	X'20'		An error has occurred while processing a SYS1.PARMLIB member
STMTLEN		2	Length of parameter list
STMTSUFX		2	Suffix value of the SYS1.PARMLIB member being processed
STMTRECL		2	Length of STMT (logical) record
STMTRECD		4	Address of a complete logical record
STMTUSEP		4	Pointer to a user parameter area
STMTNUMB		4	Physical record number of the record containing the statement type within the specified SYS1.PARMLIB member
STMTNVTP		4	Pointer to the NVT
STMTRSV1		8	Reserved

EXIT NORMAL: Return to caller

EXIT ERROR: Return to caller

EXTERNAL REFERENCES:

ROUTINES:
- IEAVM200 - routine containing the message texts for messages IEE180I, IEE181I, and IEE182I.
- IEEMB878 - routine which reads SYS1.PARMLIB records
- Statement processor routine names and addresses passed as input by the caller.

CONTROL BLOCKS:
- CVT - Communications Vector Table
- JESCT - Job Entry Subsystem Communication Table

TABLES:

PSCNSTAB - A table of valid statement types and statement processor routine names is passed by the caller as input. A pointer to that table is specified in the input parameter list (PSCNTABL).

Diagram 60 General Parmlib Scan Routine (IEEMB888) Part 3 of 19

IEEMB888 - MODULE OPERATION

IEEMB888 does the following:

1. Validates the input parameter list (i.e., acronym, the parameter list length, and the addresses must be non-zero).

   - If an invalid parameter list is detected, a return code of eight is returned to the caller of IEEMB888.

2. Examines the area containing the SYS1.PARMLIB specifications to determine if any members were specified.

   If no members were specified, or if end-of-block was specified in response to SPECIFY SYSTEM PARAMETERS then does the following:

   A. Invokes the statement processor routines specified with an EOP (end-of-processing) type entry. This will set defaults for the valid statement types.

3. Otherwise, SYS1.PARMLIB members were specified and the following are processed:

   A. Validates the SYS1.PARMLIB specification(s). Each SYS1.PARMLIB specification must consist of a two character alphanumeric combination. If invalid specifications are encountered, sets a return code of four and returns to the caller. The return code of four informs the caller to issue a prompt for valid SYS1.PARMLIB specifications.
   B. Left justifies the SYS1.PARMLIB prefix value and stores it into IEEMB878's parameter list (PLIBBASE).
   C. Invokes IEEMB878 to read a physical record from the specified SYS1.PARMLIB member.
   D. If a specified SYS1.PARMLIB member is not found, IEEMB878 issues a message indicating that the member is not found.
   E. If an I/O error occurs, IEEMB878 issues a message indicating this fact.
   F. For each physical record read, does the following:

      1. Removes any comments encountered.
      2. If the caller requests that each record of the SYS1.PARMLIB member should be written to the operator then, issues a WTO for that record.

   G. For each logical record (collection of all the operands for a particular statement type) does the following:
      1. If an unbalanced comment is detected, ignores data specified between the previous statement type and the next valid statement type. Issues message IEE181I indicating the unbalanced comment condition.
         - If a statement type has not been encountered before the unbalanced comment condition is detected, the data from the first physical record of the SYS1.PARMLIB member until the next valid statement type is ignored.
      2. Validates the statement type by searching the table of statement types and statement processor routine names that is passed by the caller.
         - If the statement type is not found in the table, checks if this is the first statement type encountered.
           - If it is the first statement type, issues message IEE180I to indicate an invalid

Diagram 60 General Parmlib Scan Routine (IEEMB888) Part 4 of 19

IEEMB888 - MODULE OPERATION (Continued)

statement type.
If it is not the first statement type, stores the data as operands and passes it to the statement processor routines when another valid statement type is detected or end-of-file is reached.
- Otherwise, deletes the statement type from the record and shifts the physical record to remove any leading blanks or commas.

3. Determines if operands are specified on the physical record containing the statement type.
   - If no operands are encountered, invokes the appropriate statement processor routine, passing to it a logical record length of zero.
   - If operands are specified, checks the 4K buffer to determine if enough room exists for this physical record. If there is enough space, moves the record into the buffer, and continues processing the logical record. Otherwise, issues message IEE182I indicating that the buffer is full. IEEMB888 then processes the next SYS1.PARMLIB member, if any exist.
4. Invokes the statement processor routine associated with the specified statement type.
5. Blanks out the buffer so it may be used again to collect logical records.
6. When an end-of-file is reached, does the following:
   - If more SYS1.PARMLIB members are to be read, sets up to read the next one.
   - Otherwise, invokes all of the statement processor routines for EOP processing.

H. Returns to the caller

RECOVERY OPERATION:
The processing in this module is done during Nucleus Initialization (NIP) time and SVC functions are not yet available. Therefore, an ESTAE environment has no effect. Diagnostic messages are issued for syntax or I/O errors.

Diagram 60 General Parmlib Scan Routine (IEEMB888) Part 5 of 19

IEEMB888 - DIAGNOSTIC AIDS

ENTRY POINT NAME: IEEMB888

MESSAGES:

```
IEE180I mem LINE xxxx: stmt STMT IGNORED. NOT VALID.

 Results when an invalid statement type is encountered in the
 SYS1.PARMLIB member specified in the input parameter list.
 Processing continues, although defaults may be set for
 those statement types which were not detected in the
 specified SYS1.PARMLIB member(s).

IEE181I mem: LINE xxxx-xxxx IGNORED. UNBALANCED COMMENT
        DETECTED.

 Results when a starting comment is found without a matching
 ending comment or an ending comment is found without a
 starting comment.  IEEMB888 searches for the next valid
 statement type and flags the physical records in error.
 The data located between the previous valid statement type
 and the next valid statement type is ignored.

IEE182I mem IGNORED. STMT STARTING LINE xxxx EXCEEDS 4096
  CHARS.

 If the buffer containing the logical record to be passed to
 the statement processor routine exceeds 4096 characters,
 this message is issued.  The next SYS1.PARMLIB member is
 processed, if any exist.
```

ABEND CODES: None

WAIT STATE CODES: None

RETURN CODES:

```
EXIT NORMAL:

   Register 15 = 0  - Request completed successfully
              = 4  - Invalid SYS1.PARMLIB specifications
              = 8  - Invalid parameter list
              = 20 - Error occurred in a statement
                     processor routine
```

REGISTER CONTENTS ON ENTRY:

```
Register  0    = Undefined
Register  1    = Address of a word which contains
                 the address of the input
                 parameter list.
Registers 2-12 = Undefined
Register  13   = Address of 18-word save area
Register  14   = Return address
Register  15   = Entry point address
```

REGISTER CONTENTS ON EXIT:

```
EXIT NORMAL:

  Register  0    = Restored
  Register  1    = Address of a word which contains the
```

Diagram 60 General Parmlib Scan Routine (IEEMB888) Part 6 of 19

IEEMB888 - DIAGNOSTIC AIDS (Continued)

```
                address of IEEZB821
  Registers 2-12 = Restored
  Register  13   = Address of 18-word save area
  Register  14   = Return address
  Register  15   = Return code

EXIT ERROR:

  See EXIT - NORMAL
```

Diagram 60 General Parmlib Scan Routine (IEEMB888) Part 7 of 19

IEEMB888 - General Parmlib Scan Routine STEP 01

IEAVNP18 - Scheduler Interface to the General Parmlib Scan Routine IEAVNP20 - CLOCK processing RIM IEAVNP25 - Nucleus SVC table RIM

IEEMB888

PARAMETERS

PSCNP

This module scans specified SYS1.PARMLIB member(s) for valid statement types, and invokes the proper statement processor routine to process these statement types. This module may be invoked to process one or more of the same SYS1.PARMLIB members (e.g. SCHEDxx, CLOCKxx, etc.).

01 Validates the caller's input parameter list, IEEZB819.

VALDTE_PARM: 05

02 IF no SYS1.PARMLIB members were specified, or end-of-block was specified in response to SPECIFY SYSTEM PARAMETERS, does the following:

A. Invokes the statement processor routines for EOP type processing.

EOPRTN: 07

B. If an error has occurred sets a return code of twenty indicating a severe error.

03 Otherwise, SYS1.PARMLIB members were specified:

A. Validates the SYS1.PARMLIB specifications.

SYNTXCHK: 22

B. Loads the parser routine.

LOAD
EP(IEEMB887)

C. Stores the address of the parser into the JESCT.

D. Loads the message service routine.

LOAD
EP(IEAVM200)

Diagram 60 General Parmlib Scan Routine (IEEMB888) Part 8 of 19

IEEMB888 - General Parmlib Scan Routine STEP 03E

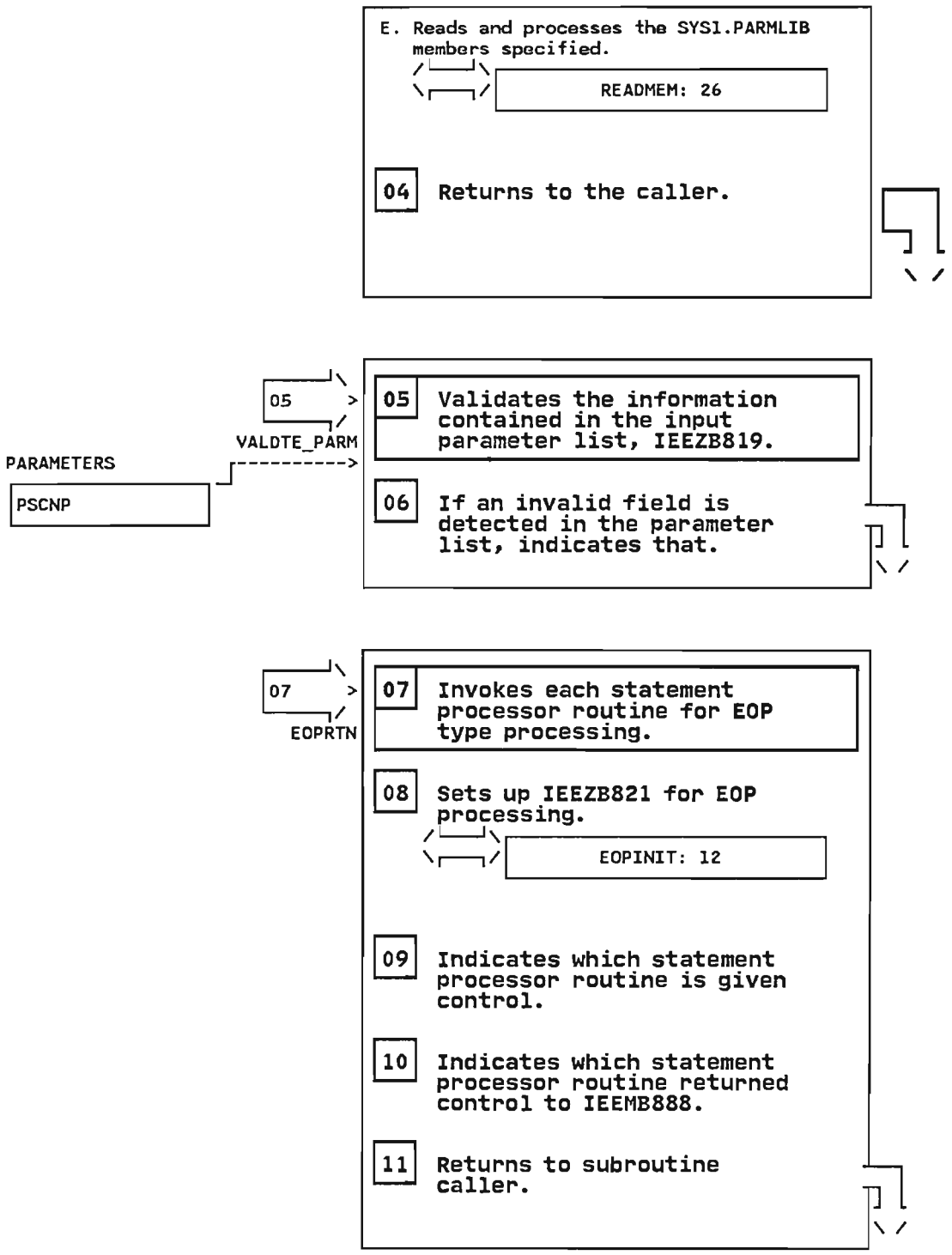


Diagram 60 General Parmlib Scan Routine (IEEMB888) Part 9 of 19

IEEMB888 - General Parmlib Scan Routine STEP 12

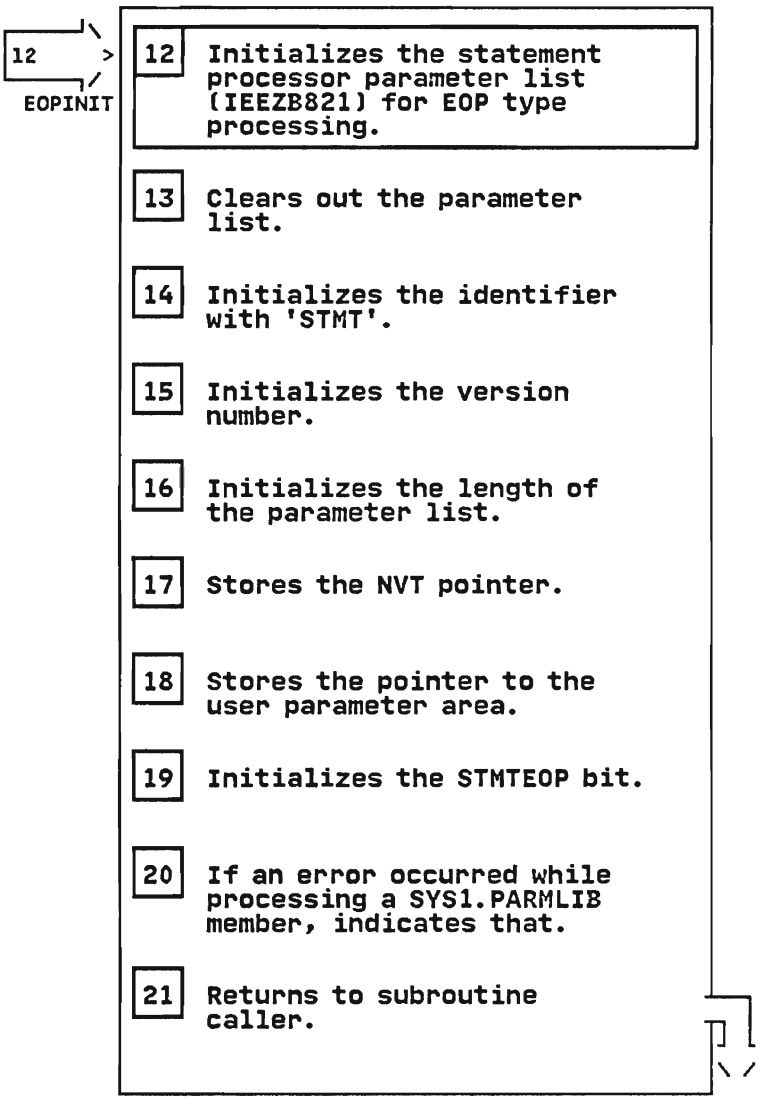


Diagram 60 General Parmlib Scan Routine (IEEMB888) Part 10 of 19

IEEMB888 - General Parmlib Scan Routine STEP 22

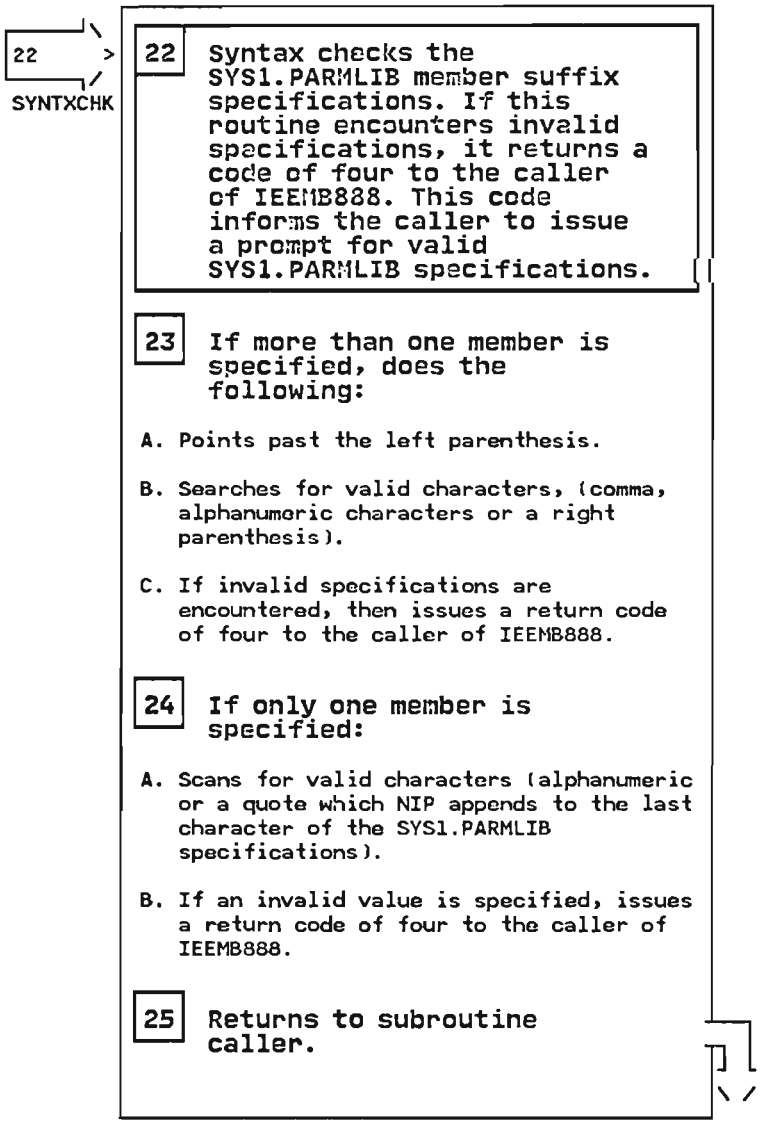


Diagram 60 General Parmlib Scan Routine (IEEMB888) Part 11 of 19

IEEMB888 - General Parmlib Scan Routine STEP 26

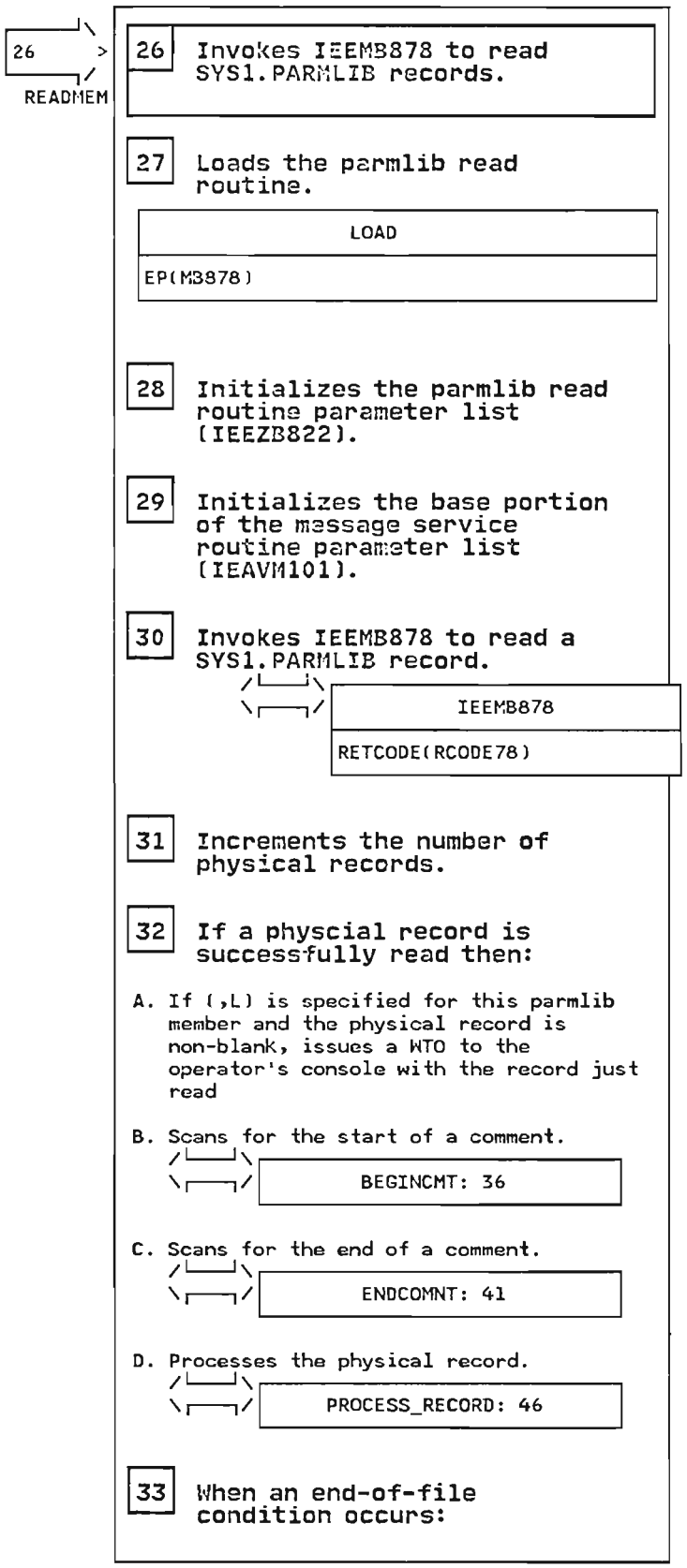
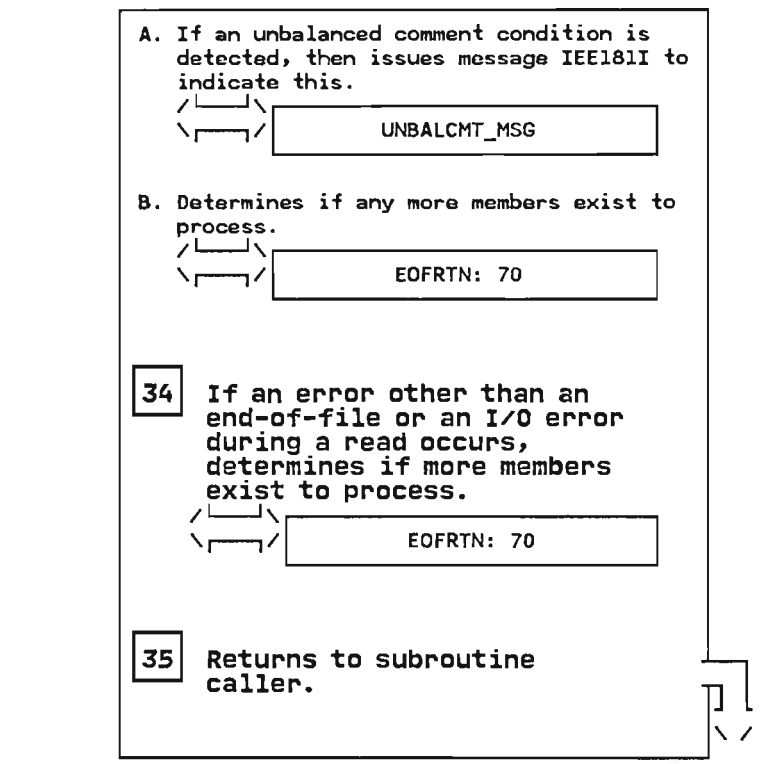


Diagram 60 General Parmlib Scan Routine (IEEMB888) Part 12 of 19

IEEMB888 - General Parmlib Scan Routine STEP 33A



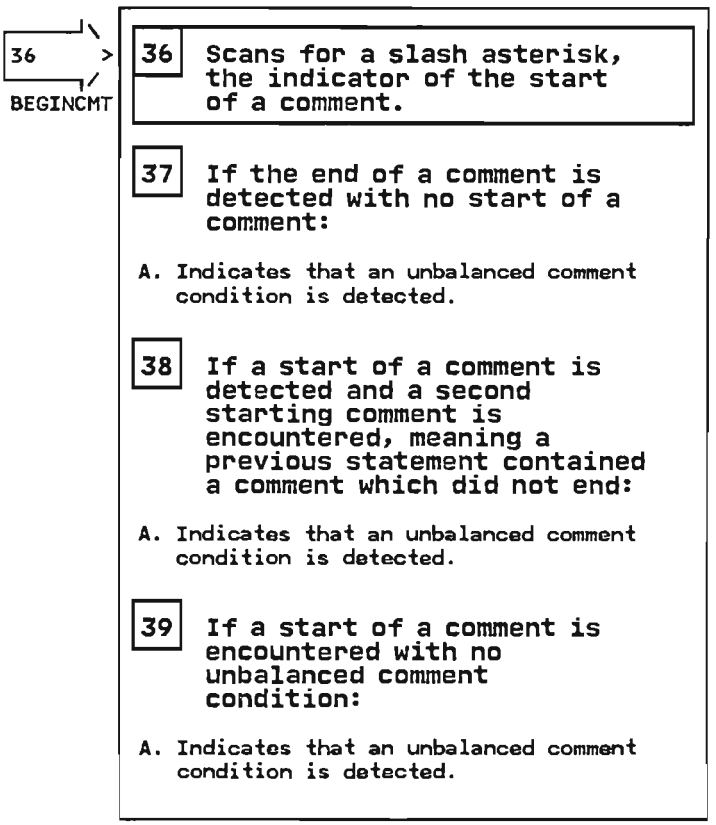


Diagram 60 General Parmlib Scan Routine (IEEMB888) Part 13 of 19

IEEMB888 - General Parmlib Scan Routine STEP 40

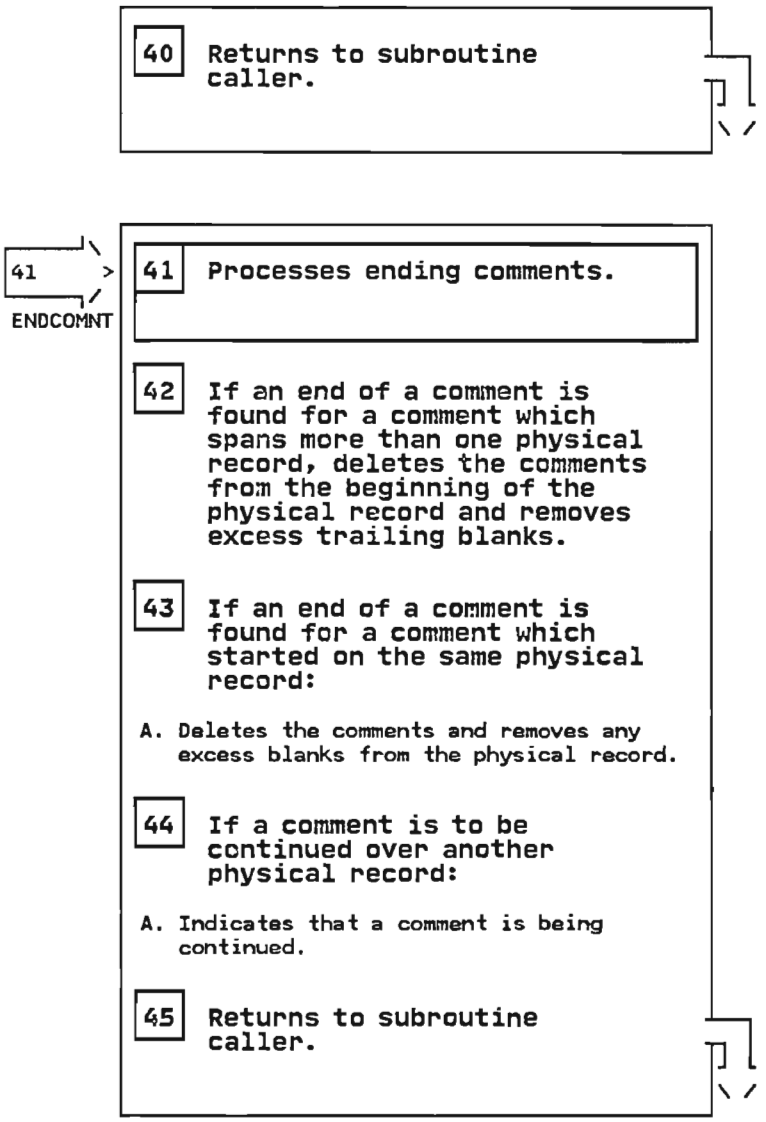


Diagram 60 General Parmlib Scan Routine (IEEMB888) Part 14 of 19

IEEMB888 - General Parmlib Scan Routine STEP 46

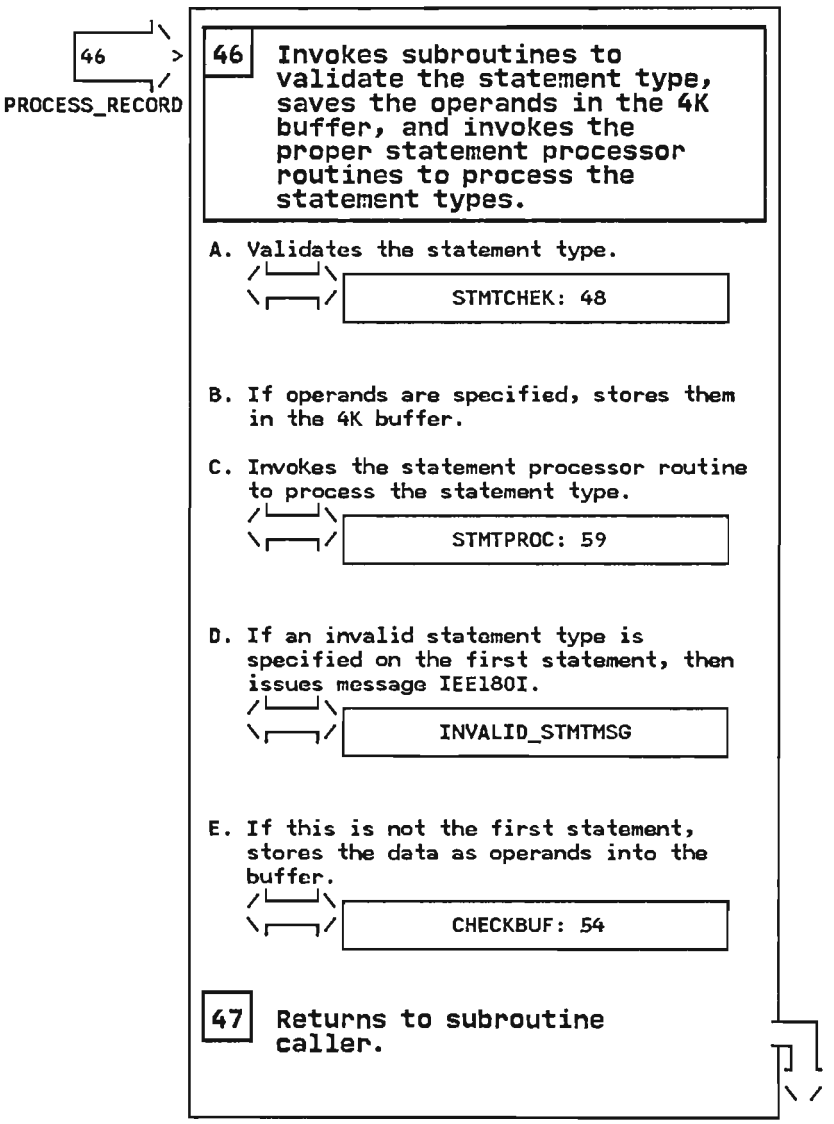


Diagram 60 General Parmlib Scan Routine (IEEMB888) Part 15 of 19

IEEMB888 - General Parmlib Scan Routine STEP 48

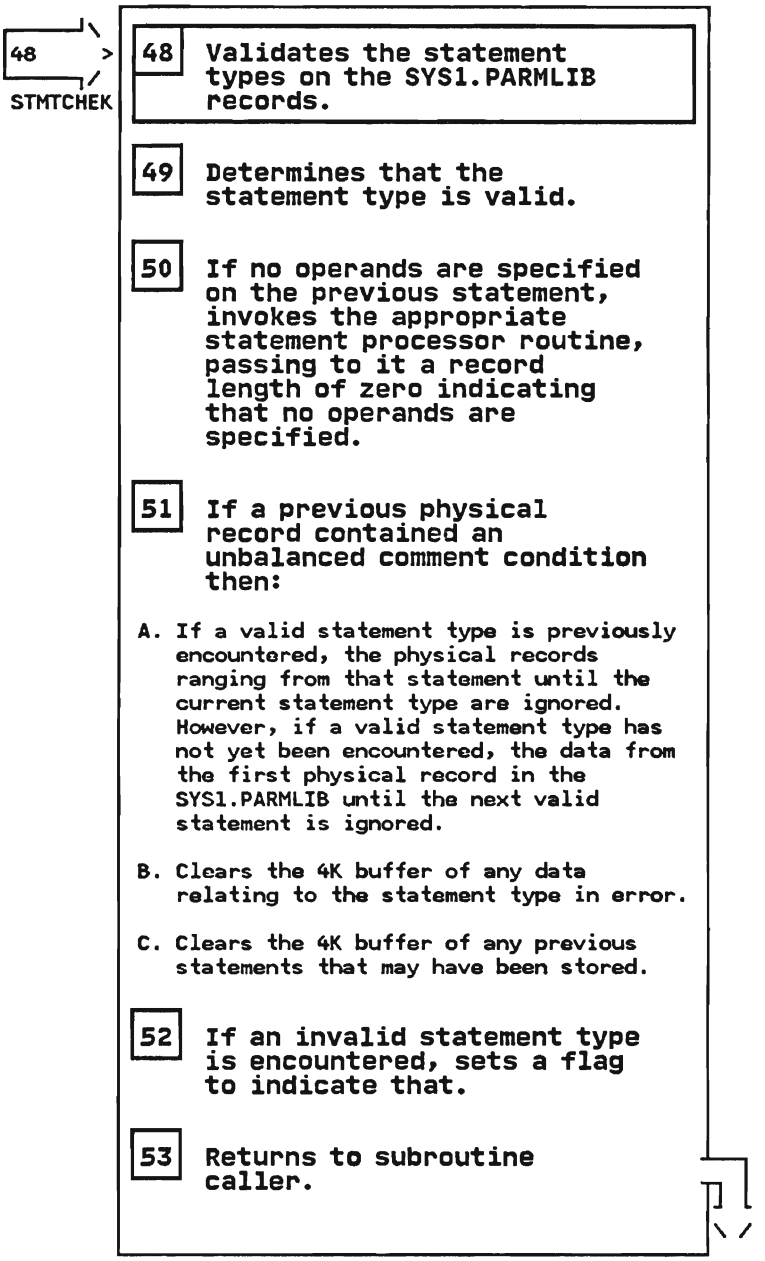


Diagram 60 General Parmlib Scan Routine (IEEMB888) Part 16 of 19

IEEMB888 - General Parmlib Scan Routine STEP 54

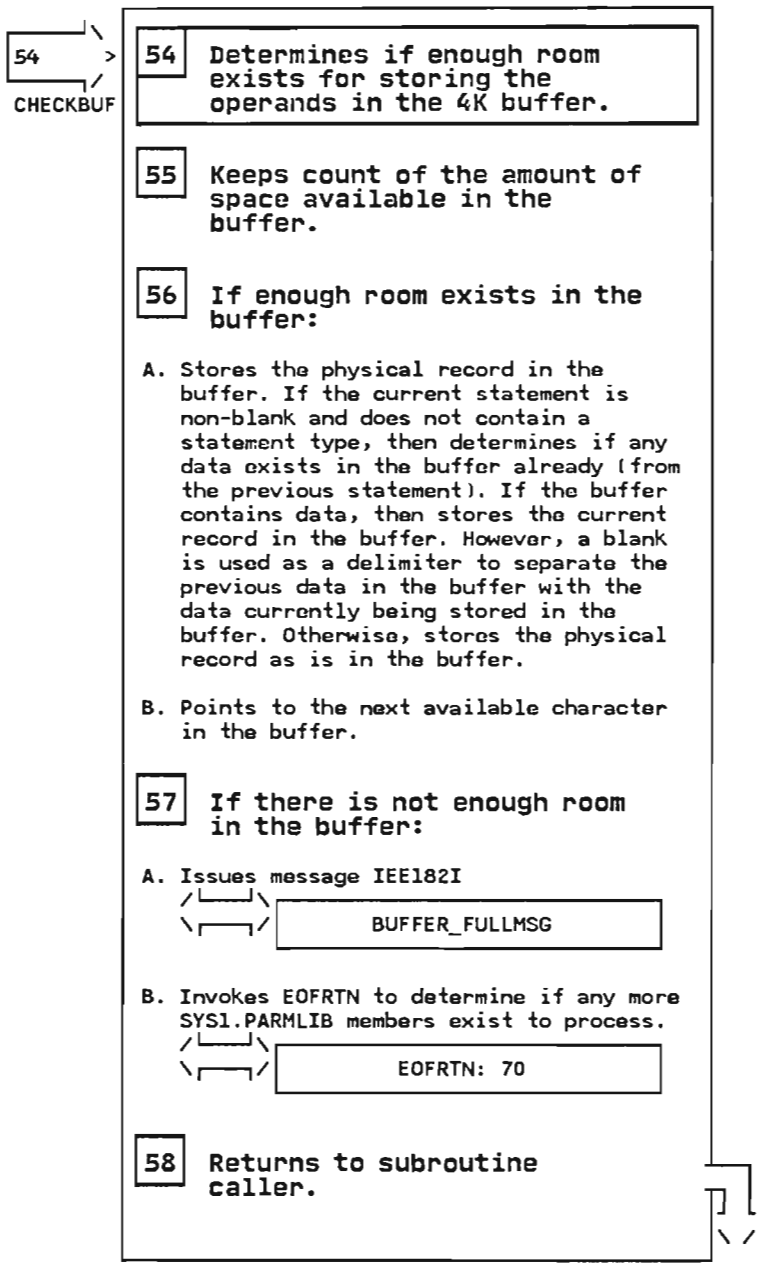


Diagram 60 General Parmlib Scan Routine (IEEMB888) Part 17 of 19

IEEMB888 - General Parmlib Scan Routine STEP 59

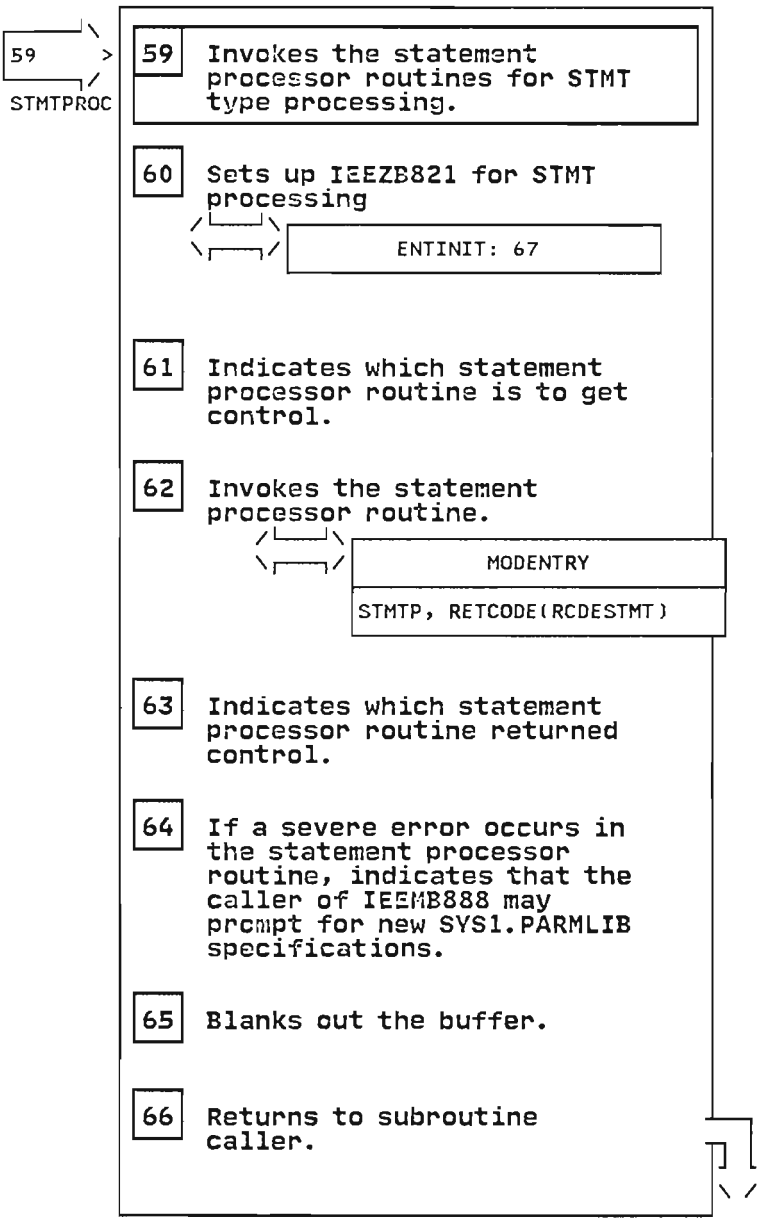
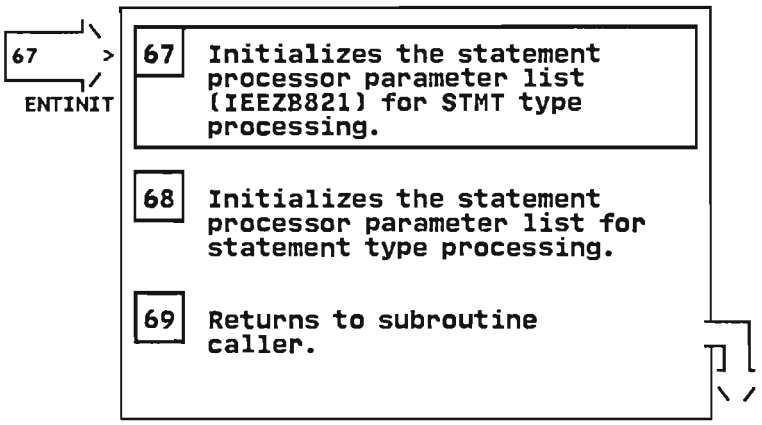


Diagram 60 General Parmlib Scan Routine (IEEMB888) Part 18 of 19

IEEMB888 - General Parmlib Scan Routine **STEP 67**



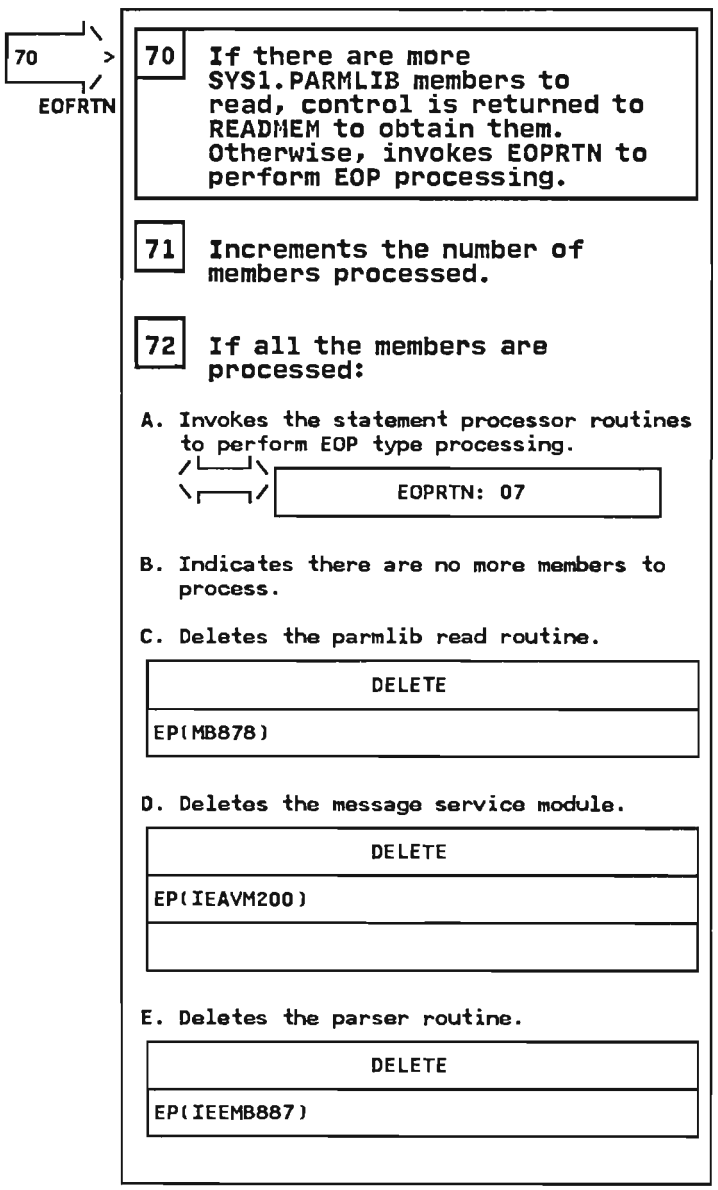


Diagram 60 General Parmlib Scan Routine (IEEMB888) Part 19 of 19

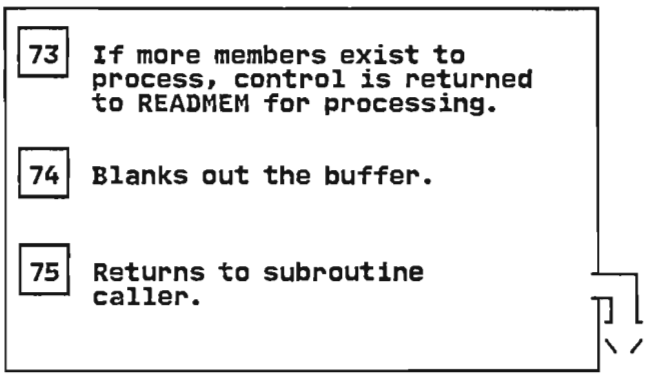


Diagram 61 Program Properties Statement Processor (IEFPPT) Part 1 of 18

IEFPPT - MODULE DESCRIPTION

DESCRIPTIVE NAME: PROGRAM PROPERTIES STATEMENT PROCESSOR

FUNCTION:
This module processes the PPT statements in SYS1.PARMLIB member SCHEDxx. The PPT statements in SYS1.PARMLIB allow an installation to specify a list of programs which require special attributes. IEFPPT validates each statement, and if the statement is correct, IEFPPT creates an entry in the Program Properties Table based on attributes specified in the PPT statement. The PPT will then consist of the IBM supplied entries plus any entries added as a result of valid PPT statements specified in SYS1.PARMLIB member SCHEDxx.

ENTRY POINT: IEFPPT

PURPOSE: See Function

LINKAGE: Call

CALLERS: IEEMB888

INPUT:
Statement Processor Parameter List (IEEZB821)

FIELD	LENGTH/MASK	DESCRIPTION
STMTP	36	Statement processor parameter list
STMTID	4	Identifier 'STMT'
STMTVERS	1	Version number
STMTFLAG	1	Control flags
STMTENT	X'80'	STMT type entry
STMTEOP	X'40'	EOP type entry
STMTERR	X'20'	Error in STMT
STMTLEN	2	Length of parameter list
STMTSUFX	2	Suffix value of the SYS1.PARMLIB member being processed
STMTRECL	2	Length of STMT record
STMTRECD	4	Address of a complete logical record
STMTUSEP	4	Pointer to a user parameter area
STMTNUMB	4	Record number within SCHEDxx being processed

OUTPUT: PPT is created

EXIT NORMAL: Return to caller

EXTERNAL REFERENCES:

ROUTINES:
IEEMB887 - Routine that parses the PPT statement for valid keywords

DATA AREAS:
IEFZB610 - Program Properties Table
IEEZB821 - Statement Processor Parameter List
IEEZB815 - Generalized Parser Input

CONTROL BLOCKS:
CVT - Communication Vector Table
IEFJESCT - Job Entry Subsystem Communication Table

TABLES:
PPT - Program Properties Table
PARSETAB - Parse descriptions for IEEZB815
NAMETABL - TRT for valid program name

Diagram 61 Program Properties Statement Processor (IEFPPT) Part 2 of 18

IEFPPT - MODULE DESCRIPTION (Continued)

```
ANYTABL  - TRT to skip over bad data
LPARTABL - TRT to skip over bad data until '(' reached
RPARTABL - TRT to skip over bad data until ')' reached
BLNKTABL - TRT to skip data until ')' reached
```

Diagram 61 Program Properties Statement Processor (IEFPPT) Part 3 of 18

IEFPPT - MODULE OPERATION

1. On the first call to IEFPPT, IEFPPT loads a data-only module IEFSDPPT, which contains the IBM supplied PPT entries, into storage. IEFPPT obtains storage from extended CSA to accommodate the default table, a header for the new PPT, plus an additional 50 entries. A copy of IEFSDPPT will be placed in the storage area obtained. IEFSDPPT is then deleted. IEFSDPPT will be used as a base and entries will be added based on any PPT statements that appear in SCHEDxx.

2. For each statement entry, IEFPPT calls IEEMB887 to parse the PPT statement for valid keywords. If the PPT statement is valid, those attributes specified are set in the next available entry in the PPT. If PPT statement is not valid, the statement is ignored and not added to the PPT.

3. On each subsequent STMT entry, IEFPPT determines if the program name specified is a duplicate of what was supplied in the IBM default PPT. If so, and if this is the first occurrence of a duplicate for the default entry, it overrides the IBM supplied entry. PPTDEFLT, in the PPT entry, will be set off so that any remaining duplicates for this entry will be ignored. If the program name specified on a PPT statement has not been specified in any previous entries, it is added to the PPT.

4. IEFPPT will check to make sure there is sufficient space in the table. If there is not, IEFPPT issues another GETMAIN for the current table size plus an additional 50 entries. The old table will be copied into the new storage area, and the previous storage area freed. The address of the new table is placed into the JESCT (JESPPT) and the PPT entry is added to the new table.

5. At the end of processing (EOP-type call), if no PPT statements have been specified, IEFPPT will obtain storage from extended CSA. IEFPPT loads the IBM supplied default table (IEFSDPPT) and copies it into the storage area obtained. IEFSDPPT is then deleted. If any valid PPT statements were processed before EOP call, IEFPPT will FREEMAIN the message area and return to IEEMB888.

RECOVERY OPERATION:
The processing for this module takes place during Nucleus Initialization (NIP) time. Although an ESTAE environment could be established at this point, if an ABEND did occur, control would not be received because RTM is not fully initialized yet. Therefore, an ESTAE environment is not established.

Diagram 61 Program Properties Statement Processor (IEFPPT) Part 4 of 18

IEFPPT - DIAGNOSTIC AIDS

ENTRY POINT NAME: IEFPPT

MESSAGES:

1.IEF731I mem LINE num:PPT STMT <FOR PGMNAME name>
IGNORED./
ACCEPTED. REASON=kwrc.

Where mem is the SCHEDxx SYS1.PARMLIB member which contains the bad record, name is the program name specified on the PPT statement in error and num is the line number in SCHEDXX where the PPT statement began.

The PPT statement will be accepted if the keyword that is missing a right parenthesis is the last keyword on the statement (reason code '40'). In this case, "ACCEPTED" will be substituted for "IGNORED" in the message text.

If the program name was not found at the time the error was encountered, then the phrase, "FOR PGMNAME name" will be suppressed.

kw denotes the keyword where the error occurred. The values for kw are as follows:

01 - NO KEYWORD CAN BE DETERMINED
02 - PGMNAME
03 - KEY
04 - CANCEL
05 - NOCANCEL
06 - SWAP
07 - NOSWAP
08 - PRIV
09 - NOPRIV
10 - DSI
11 - NODSI
12 - SYST
13 - NOSYST
14 - PASS
15 - NOPASS
16 - AFF
17 - SPREF
18 - LPREF
19 NOPREF

rc equals one of the following reason codes:

04 - Delimiter between keywords missing or not found where expected

08 - Keyword not valid

12 - Mutually exclusive keywords specified

16 - Parameter not valid

20 - Duplicate keyword specified

24 - Keyword list not valid

28 - Required program name not specified

32 - Program name not valid

36 - Duplicate keyword value specified

Diagram 61 Program Properties Statement Processor (IEFPPT) Part 5 of 18

IEFPPT - DIAGNOSTIC AIDS (Continued)

40 - Missing right parenthesis on last keyword (NOTE - in this case, the statement will be accepted)

2.IEF732I mem LINE num:DUPLICATE PPT STMT FOR PGMNAME name IGNORED.

Where mem is the SCHEDxx SYS1.PARMLIB member which contains the duplicate PPT statement, name is the program name specified on the PPT statement in error and num is the line number in SCHEDxx where the PPT statement began.

3.IEF098I mem LINE num:PPT STMT IGNORED. NO OPERANDS SPECIFIED.

Where mem is the SCHEDxx SYS1.PARMLIB member which contains the PPT statement with no operands, and num is the line number in SCHEDxx where the PPT statement began.

ABEND CODES: None

WAIT STATE CODES: None

RETURN CODES:

EXIT NORMAL:

0 - Request successfully completed

REGISTER CONTENTS ON ENTRY:

```
REGISTER  0     = Undefined
REGISTER  1     = Address of parameter list
REGISTERS 2-12  = Undefined
REGISTER  13    = Address of savearea
REGISTER  14    = return address
REGISTER  15    = Entry point address
```

REGISTER CONTENTS ON EXIT:

EXIT NORMAL:

```
REGISTERS 0-14  = Restored
REGISTER  15    = Return code
```

Diagram 61 Program Properties Statement Processor (IEFPPT) Part 6 of 18

IEFPPT - PROGRAM PROPERTIES STATEMENT PROCESSOR **STEP 01**

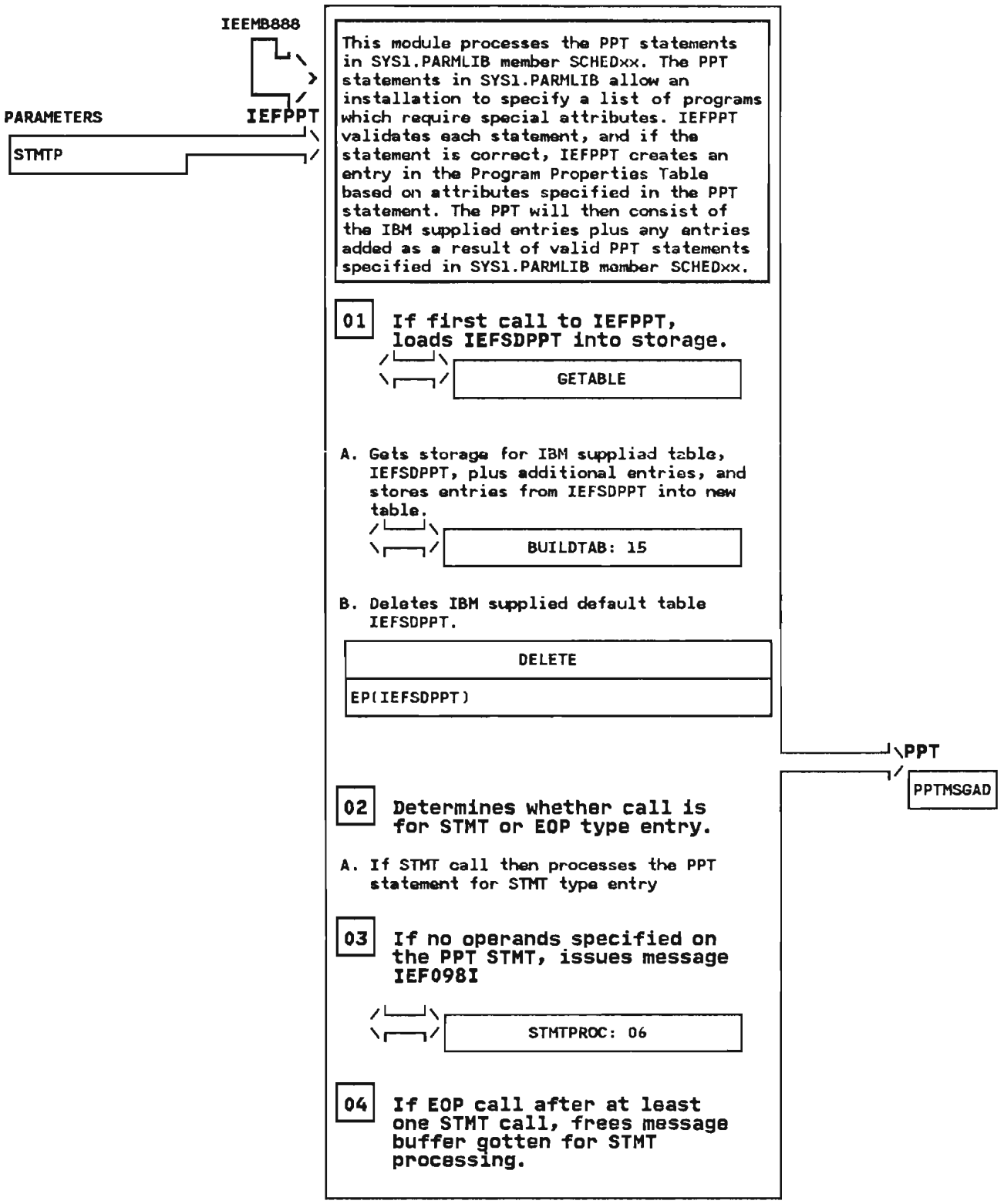


Diagram 61 Program Properties Statement Processor (IEFPPT) Part 7 of 18

IEFPPT - PROGRAM PROPERTIES STATEMENT PROCESSOR **STEP 05**

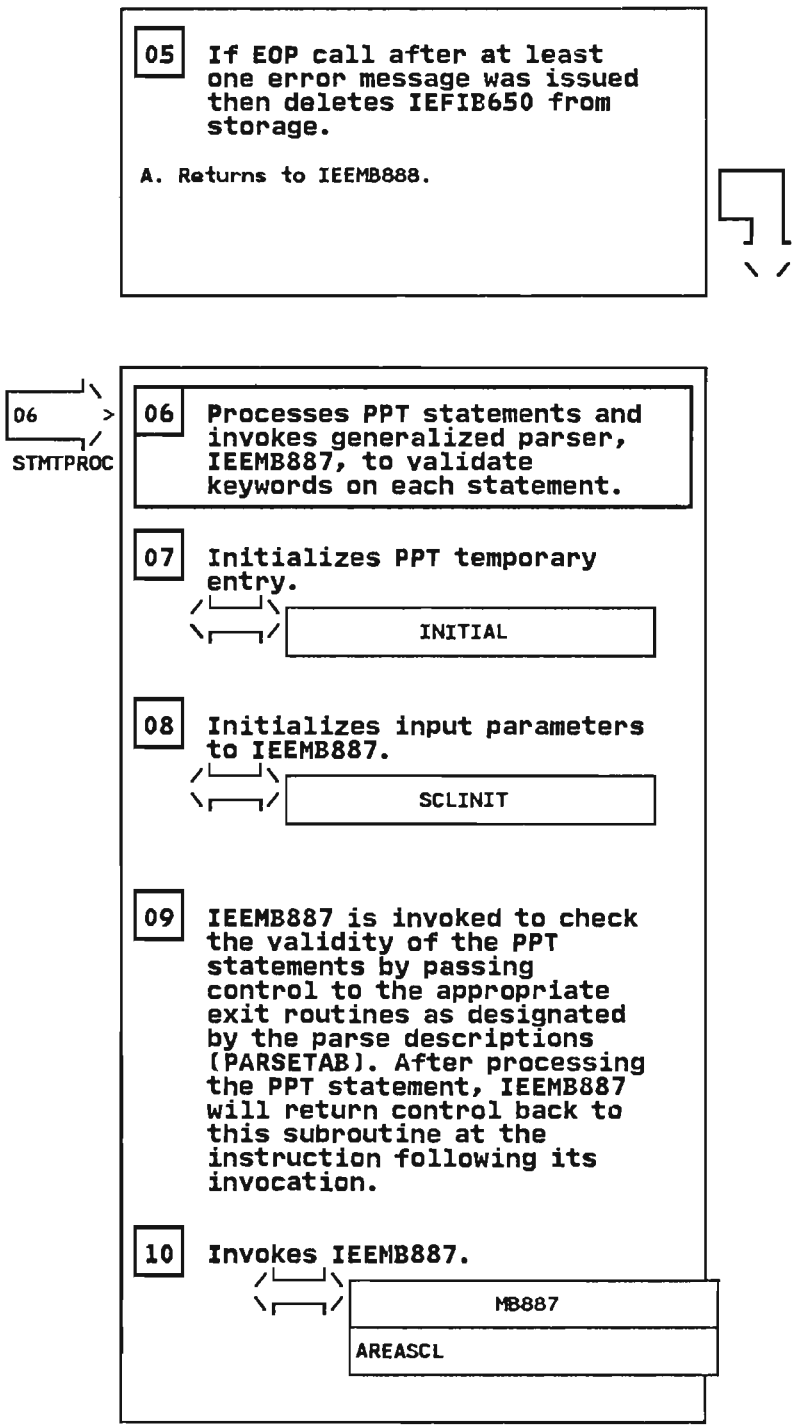


Diagram 61 Program Properties Statement Processor (IEFPPT) Part 8 of 18

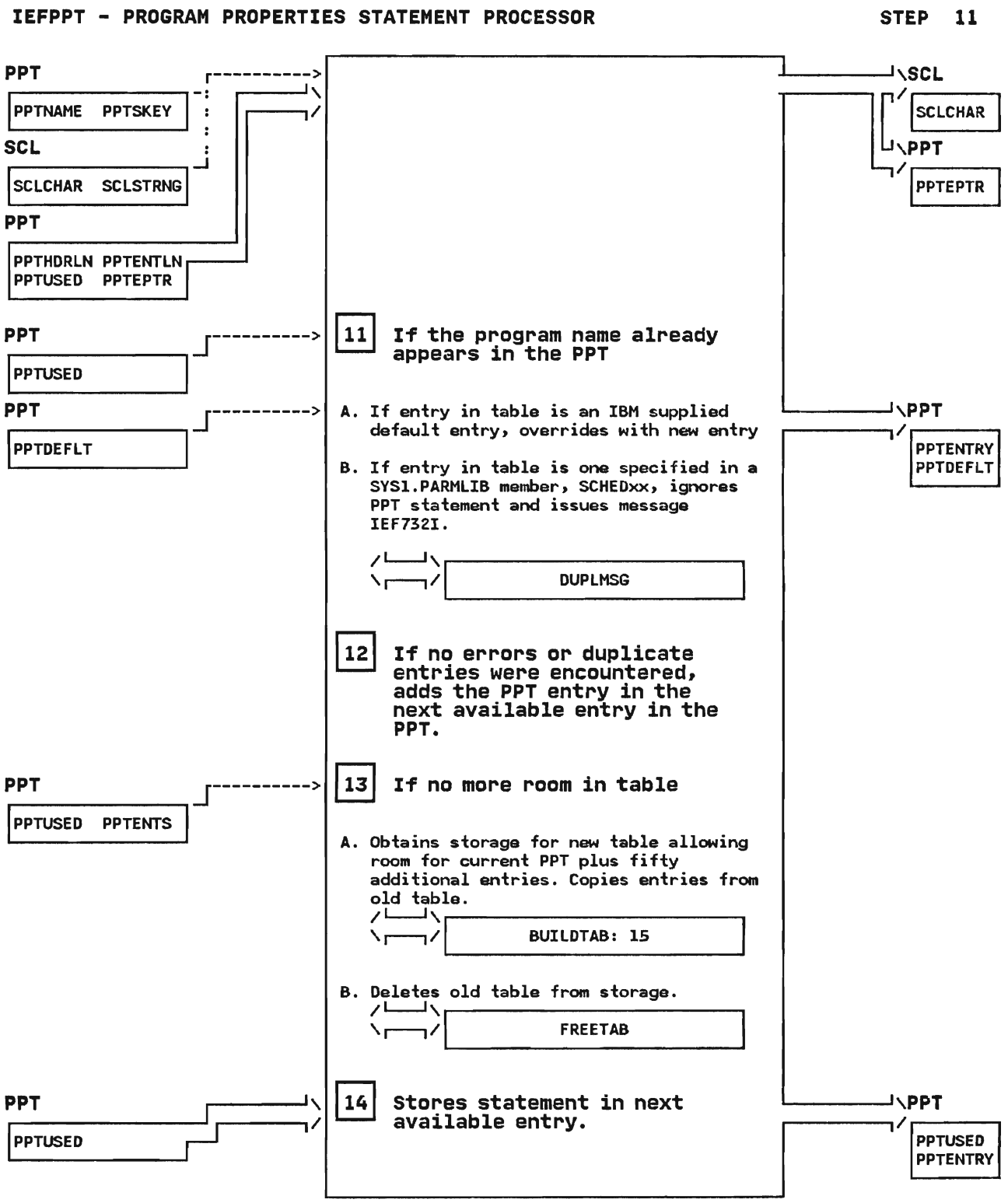


Diagram 61 Program Properties Statement Processor (IEFPPT) Part 9 of 18

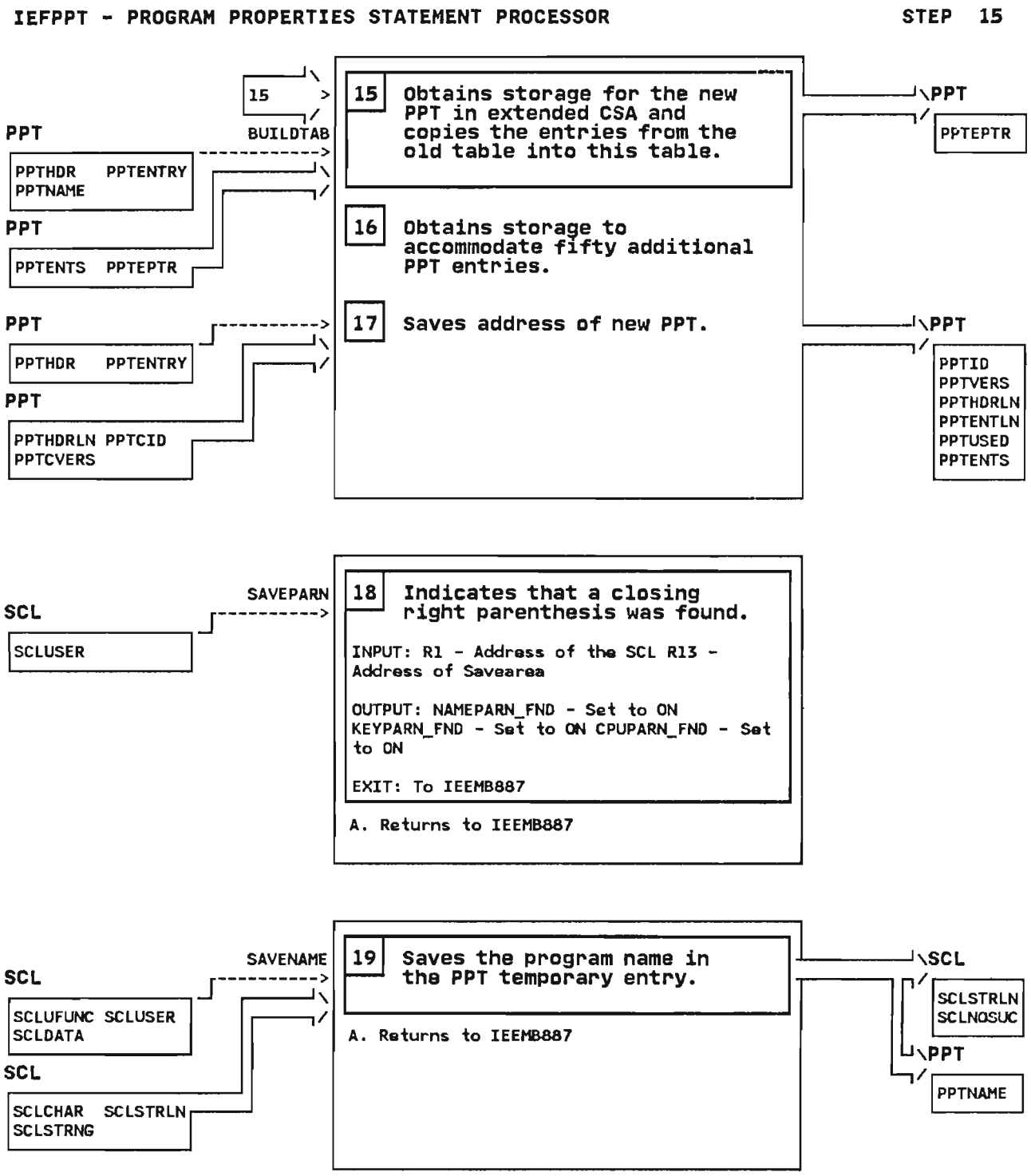


Diagram 61 Program Properties Statement Processor (IEFPPT) Part 10 of 18

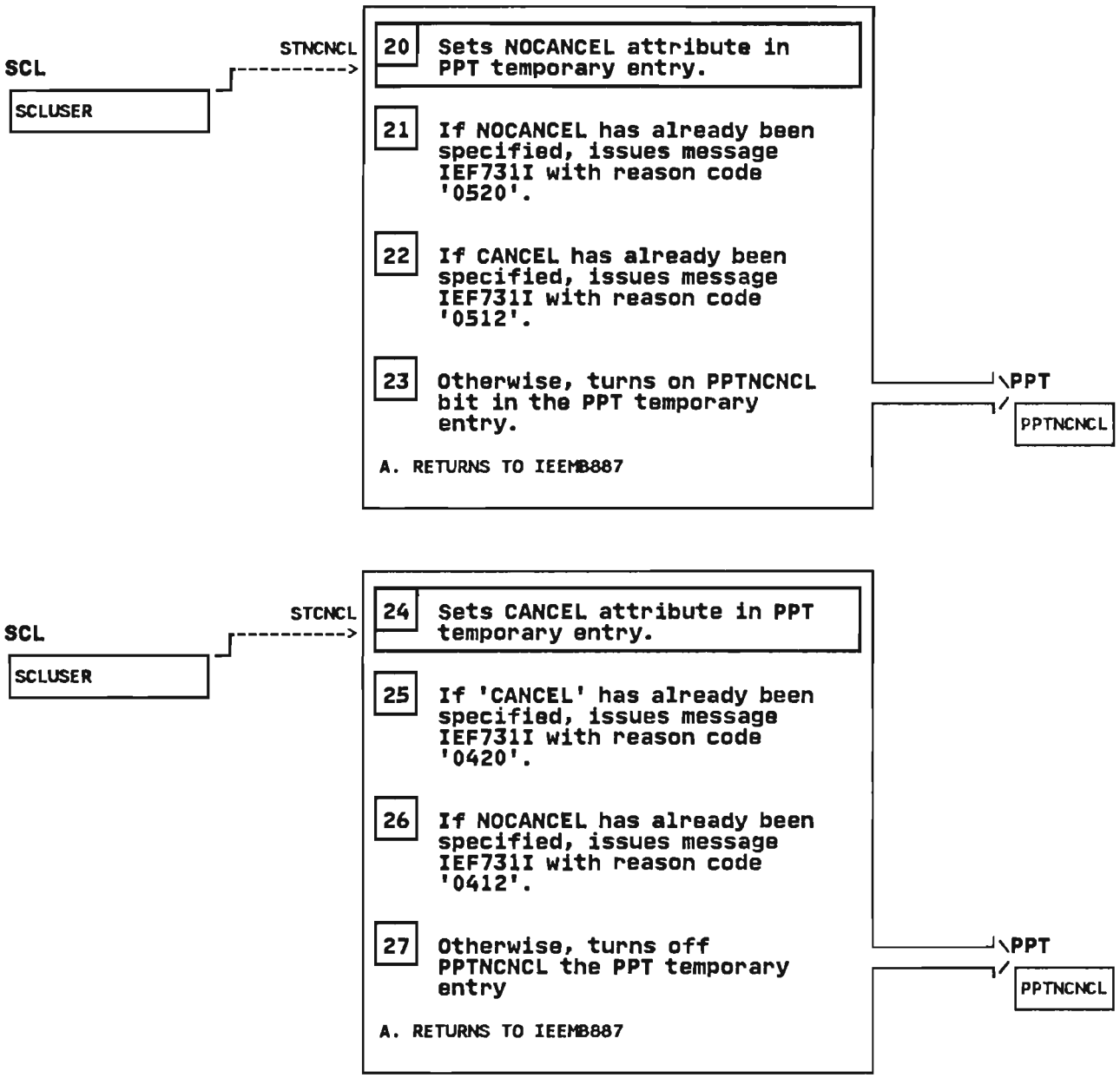


Diagram 61 Program Properties Statement Processor (IEFPPT) Part 11 of 18

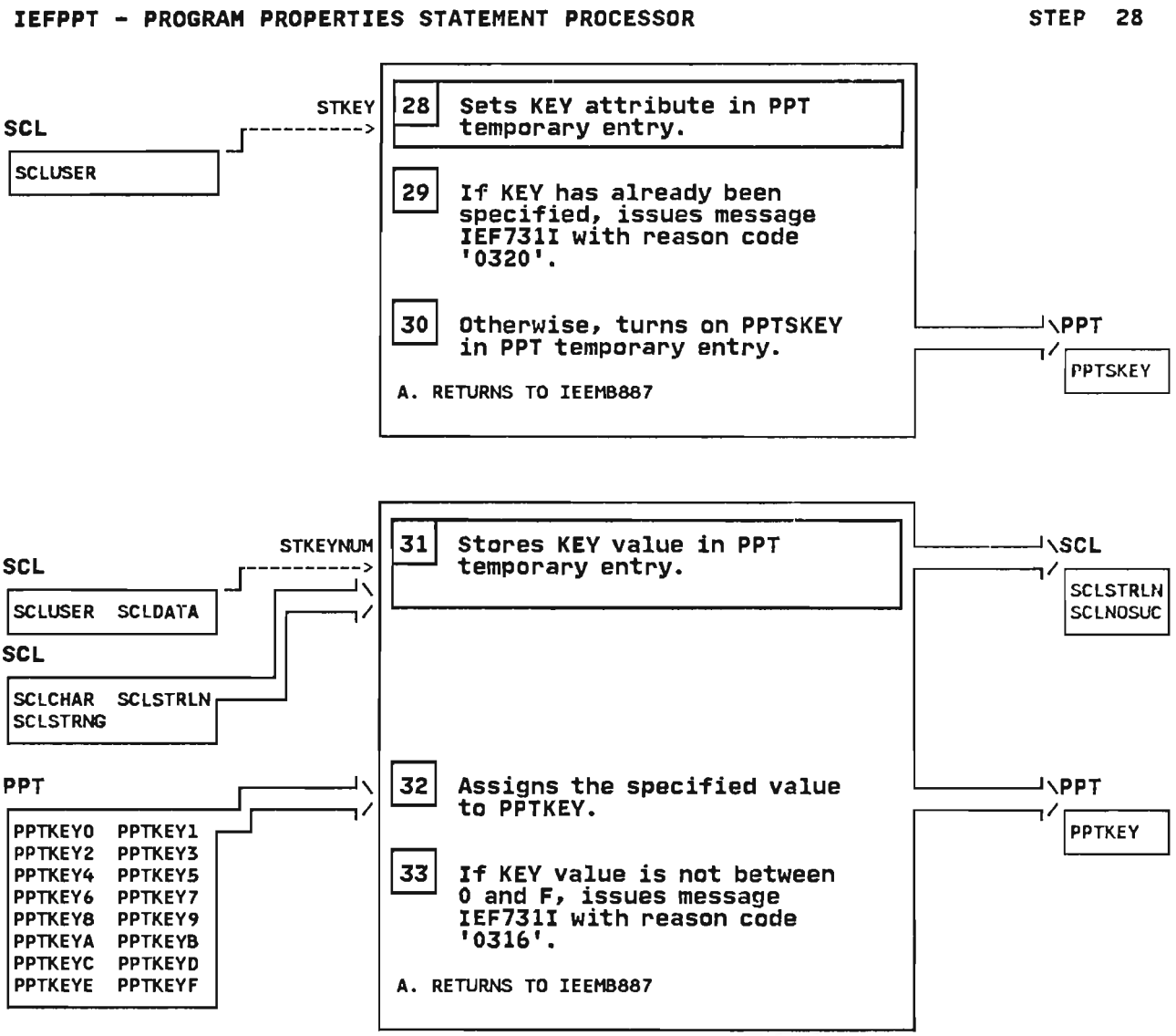


Diagram 61 Program Properties Statement Processor (IEFPPT) Part 12 of 18

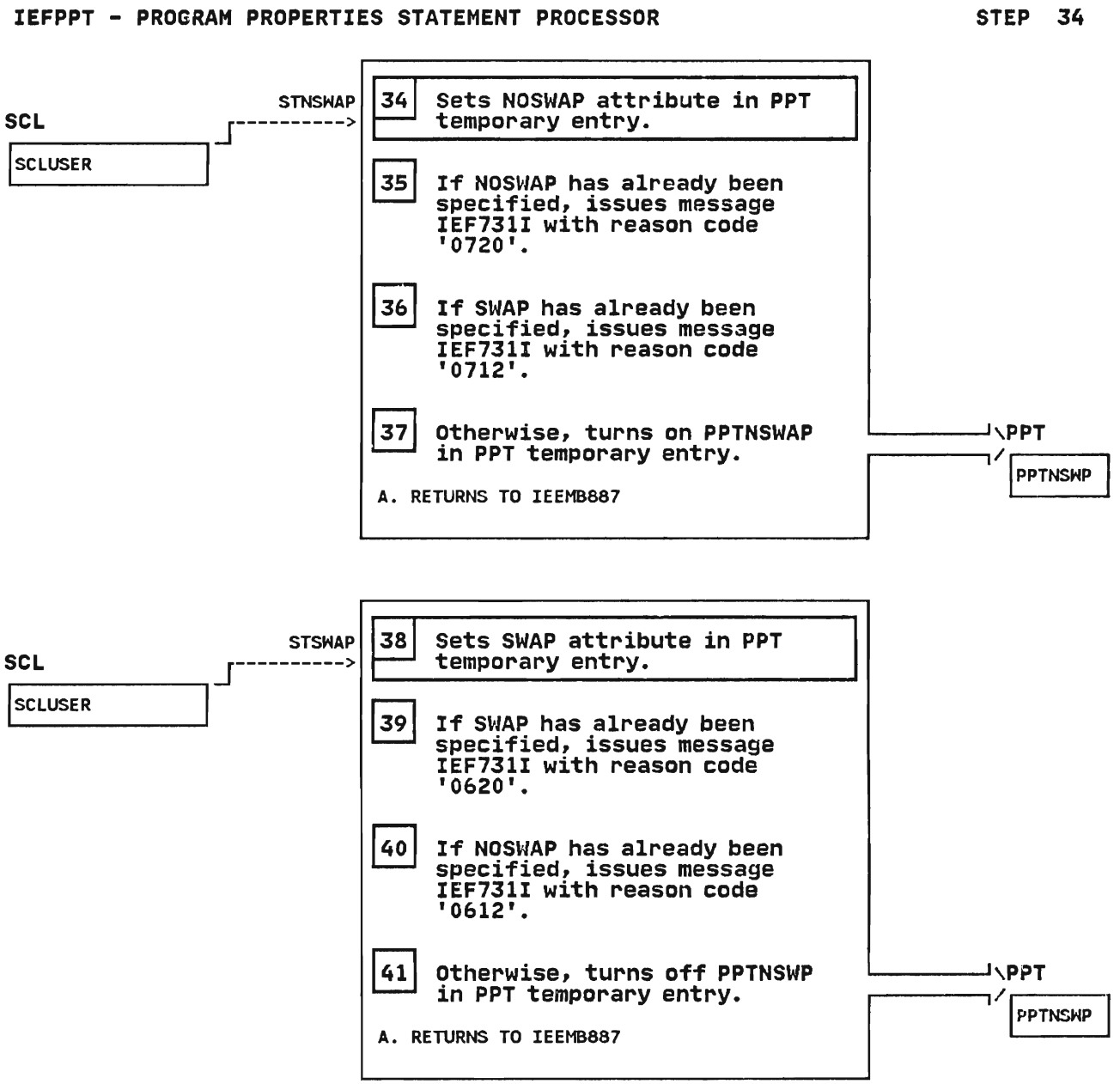


Diagram 61 Program Properties Statement Processor (IEFPPT) Part 13 of 18

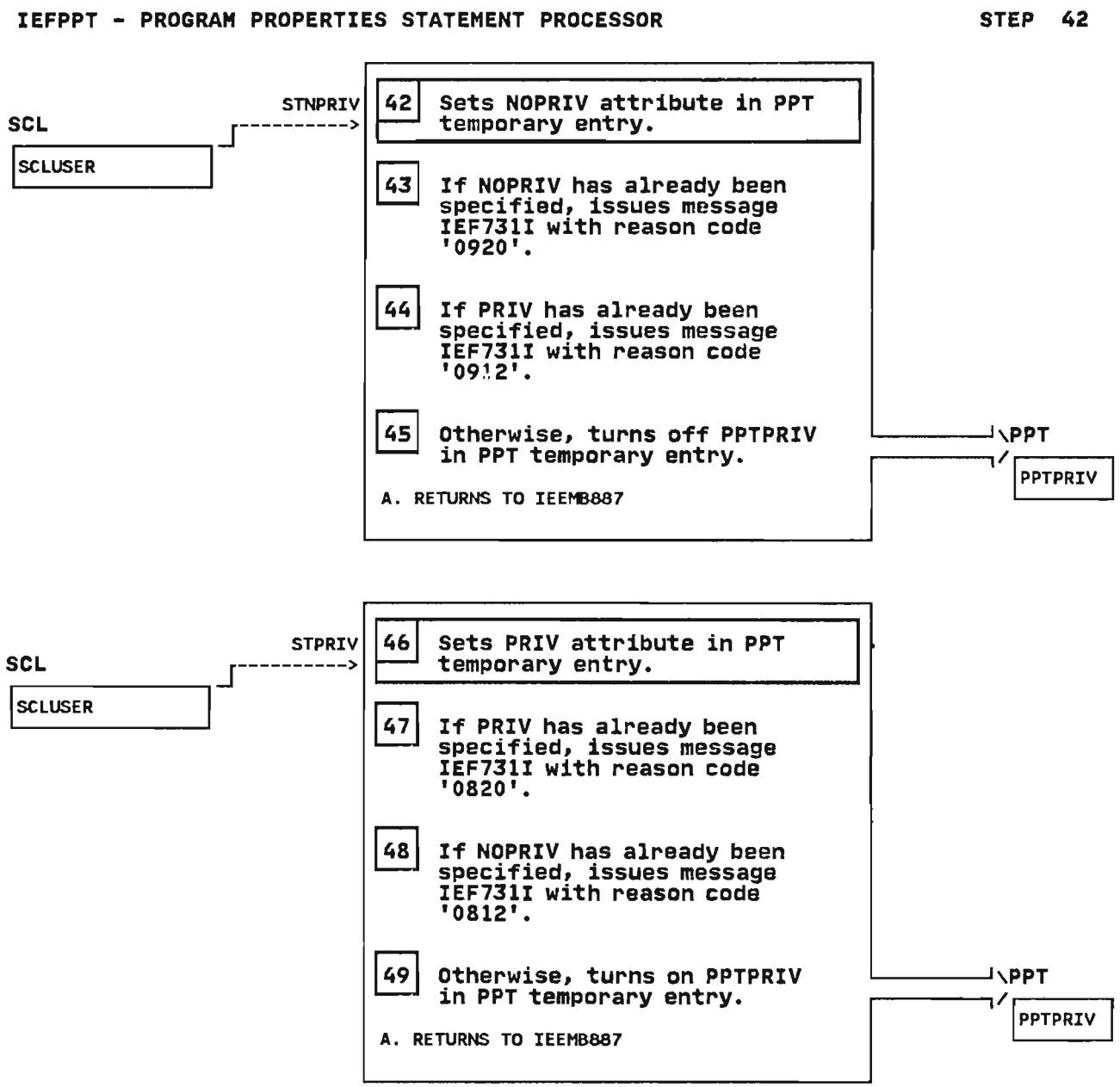


Diagram 61 Program Properties Statement Processor (IEFPPT) Part 14 of 18

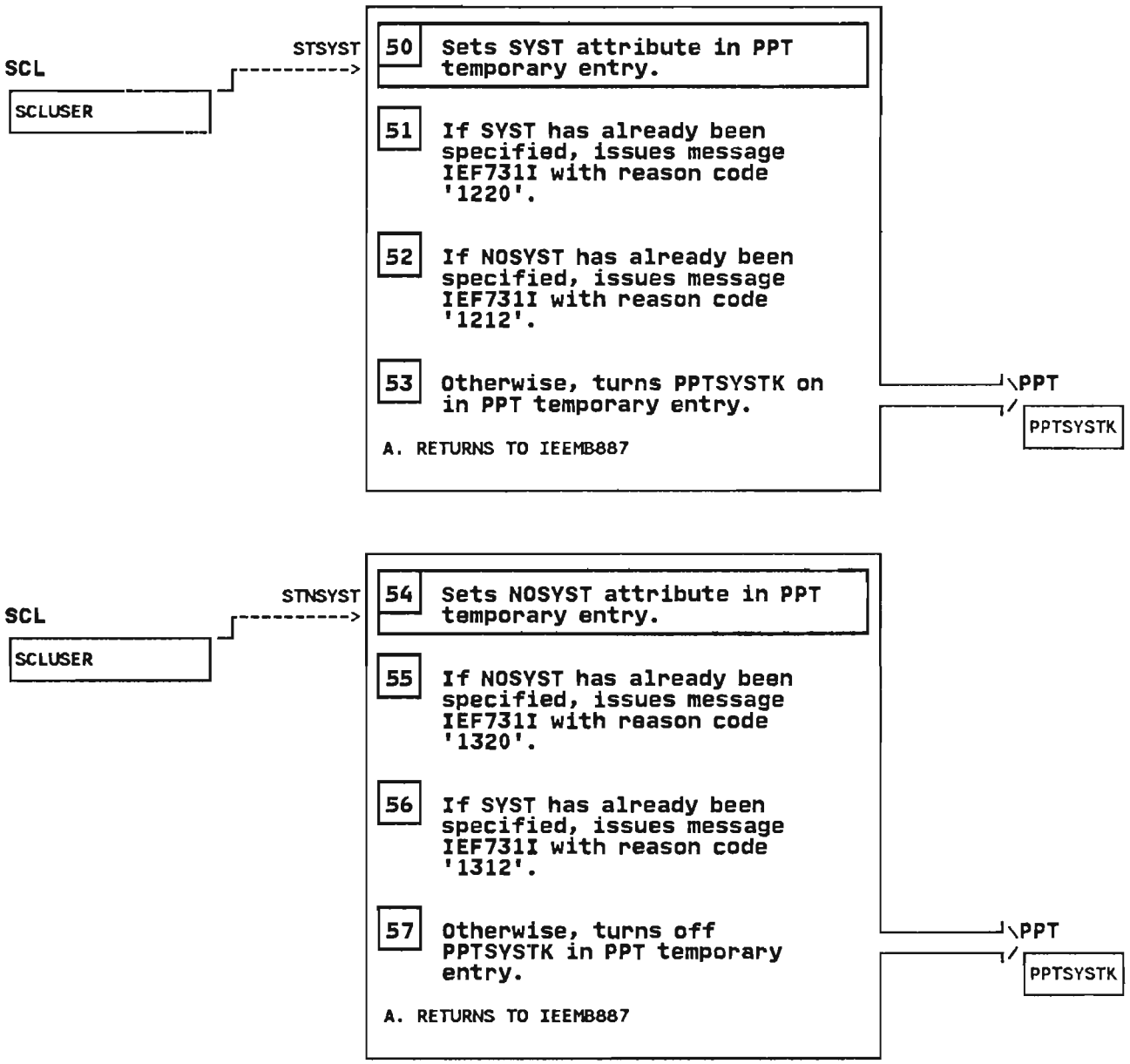


Diagram 61 Program Properties Statement Processor (IEFPPT) Part 15 of 18

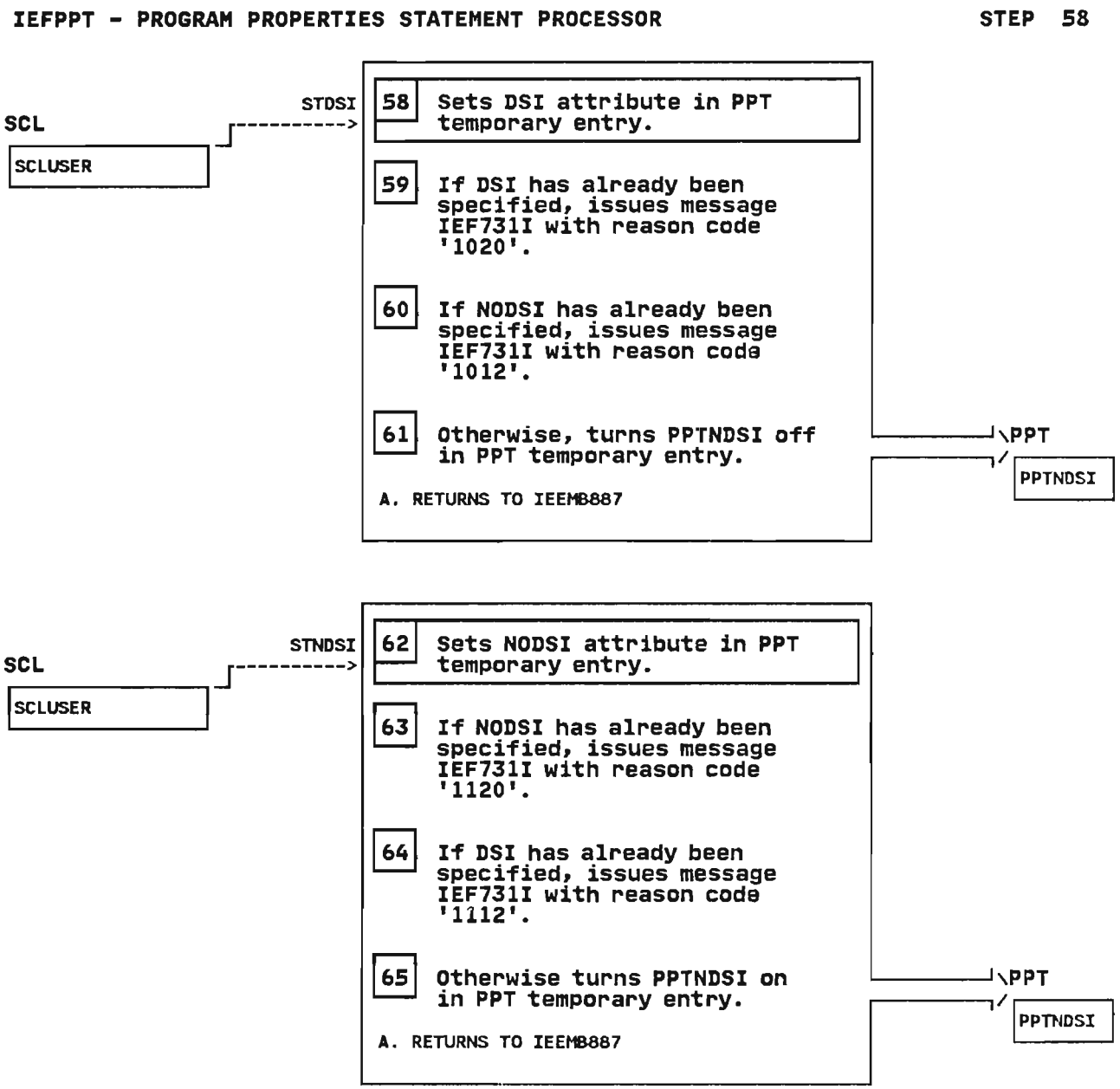


Diagram 61 Program Properties Statement Processor (IEFPPT) Part 16 of 18

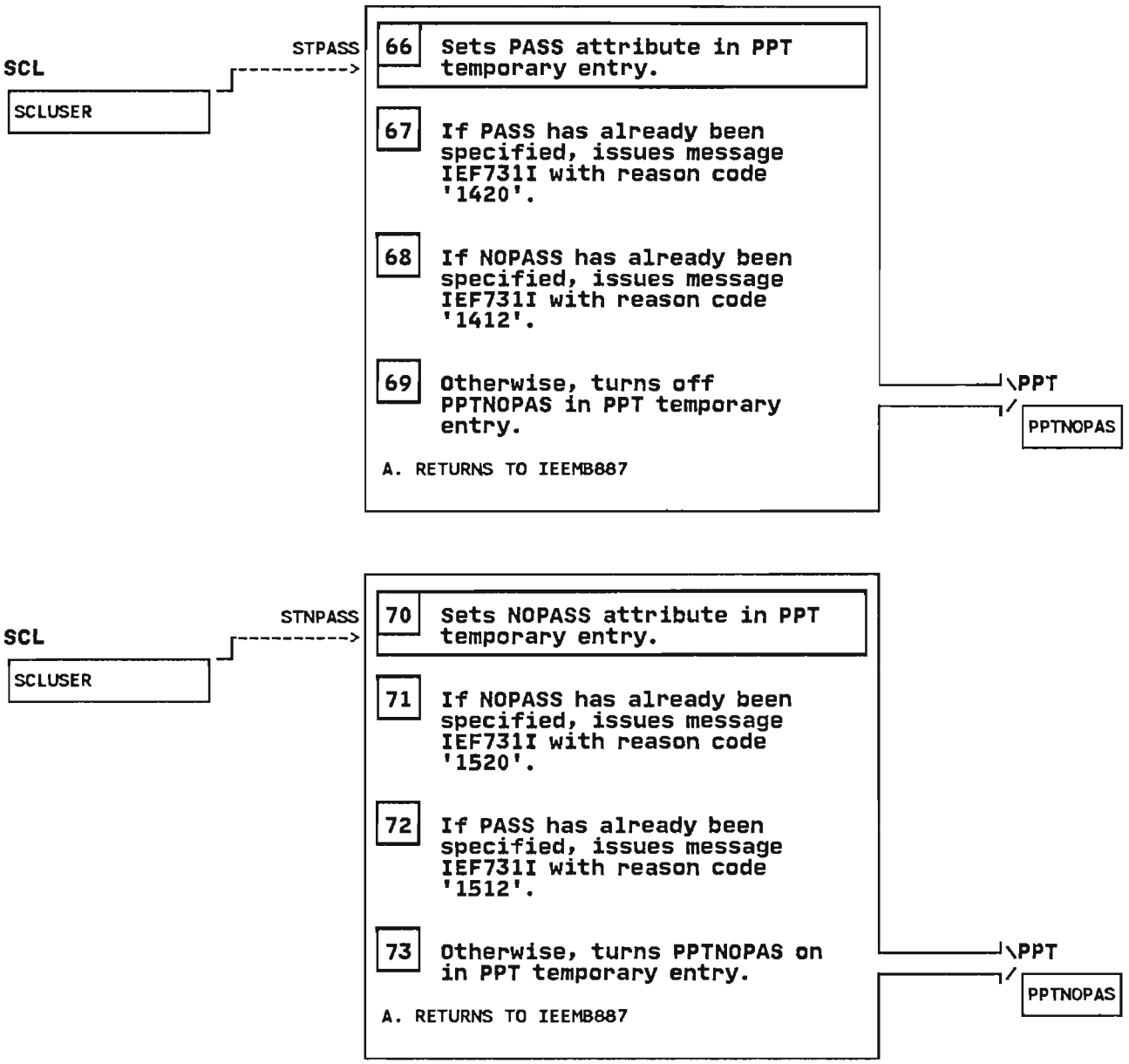


Diagram 61 Program Properties Statement Processor (IEFPPT) Part 17 of 18

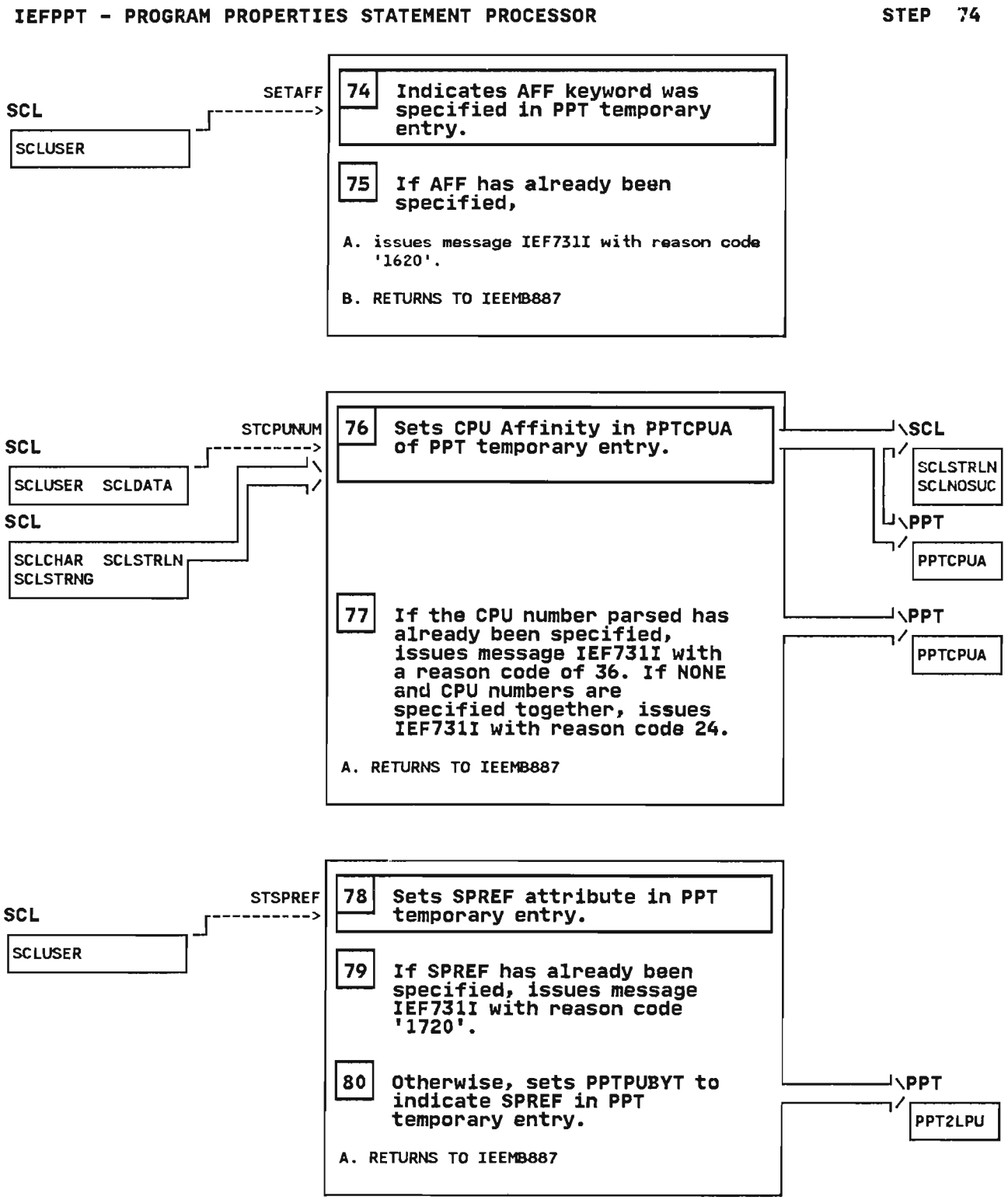


Diagram 61 Program Properties Statement Processor (IEFPPT) Part 18 of 18

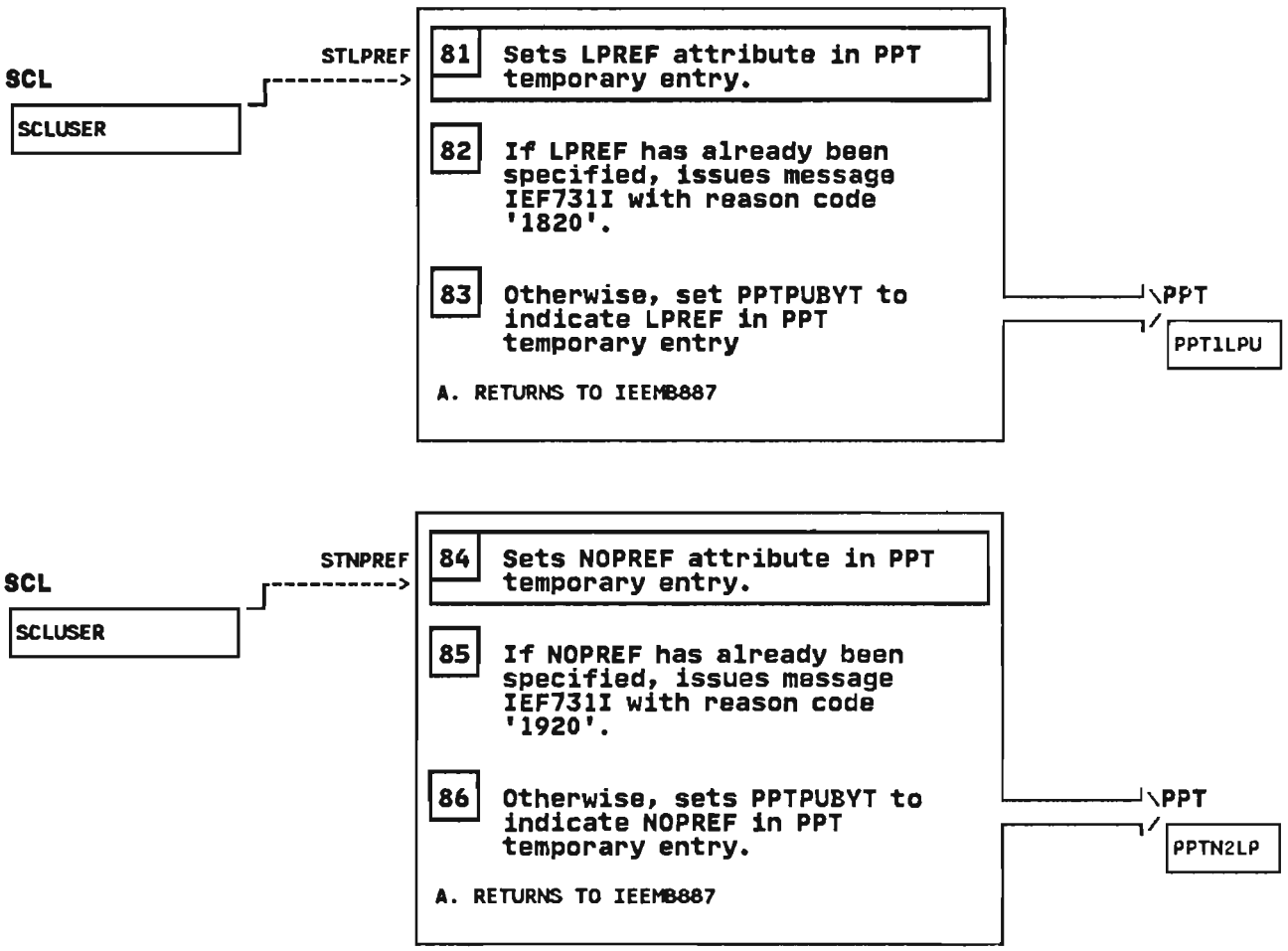


Diagram 62 Allocation Load EDT Routine (IEFAB4IE) Part 1 of 7

IEFAB4IE - MODULE DESCRIPTION

DESCRIPTIVE NAME: Allocation load EDT routine

FUNCTION:

IEFAB4IE processes the SYS1.PARMLIB EDT statement parameter to verify its correctness, and loads the specified ELIGIBLE DEVICE TABLE (EDT) member of SYS1.NUCLEUS. Any error detected in either loading the table or in verifying the parameter causes the parameter to be rejected.

ENTRY POINT: IEFAB4IE

PURPOSE: see Function

LINKAGE: CALL IEFAB4IE(STMTP)

CALLERS: IEEMB888

INPUT: STMTP - statement processor parameter list

OUTPUT:
JESEDT - this field will contain the address at which the EDT has been loaded

EXIT NORMAL:

EXTERNAL REFERENCES:

ROUTINES: IGFPTERM

DATA AREAS: IEASTMTP

CONTROL BLOCKS:
IEEZB821 - Statement Processor Parameter List
CVT - Communications Vector Table
IEFZB421 - Eligible Device Table
IEFJESCT - JES Communications Table
IHAECB - Event Control Block
IHALRB - LRB Mapping
IHANVT - NIP Vector Table
IHAPDS - PDS Directory Entry

Diagram 62 Allocation Load EDT Routine (IEFAB4IE) Part 2 of 7

IEFAB4IE - MODULE OPERATION

This module is called for two reasons: During parmlib processing of the SCHEDXX members when an EDT statement is found, and at the end of the SCHEDXX processing. It will always be invoked at the end of processing to ensure that the EDT gets loaded.

Module internals:

1) If an EDT is already loaded when this module gets control and this is not the end of SCHEDXX processing, a message is issued to the operator saying that the EDT ID parameter is rejected because it is a duplicate specification of the ID.
Control returns to the caller.

2) If the EDT hasn't been loaded yet, the EDT ID is syntax checked:
   a) For a statement-entry type of invocation , the statment must be as follow:

```
EDT   ID(   <VALUE>
```

   . ID(is the valid keyword
   . <VALUE> must be two alphanumeric characters
   . Spaces or commas are permitted between the above token

   following rules will be applied :
   . if no operands are specified or a keyword is not valid or missing, defaults will be applied at the end of statement.
   . if keyword is specified correctly, but value is an error, the default for that keyword is taken upon the detection of the errorneous value.
   . if KEYWORD(VALUE ERROR) is specified where:
     VALUE is the correctly specified value and
     ERROR is an incorrectly specified value
     the correct value for that keyword is rejected.
   . Once a valid value (including defaults) has been applied for a keyword, all other occurrences of that keyword are considered as duplicates.
   . statements with an unrecognized keyword containing a contiguous left parenthesis must scanned for an ending right parenthesis. When a right parenthesis is encountered, the data from the start of the unrecognized keyword until the right parenthesis is ignored.
   . if a blank or comma is located after the unrecognized keyword, (before a left parenthesis is located), the data from the unrecognized keyword until the blank or comma is ignored.
   . the end of a statement, which is passed to the statement, is a delimiter. That is, if a right parenthesis is not found, the data from the unrecognized keyword until the end of the statement is ignored.

   B) For an end-of-processing invocation, if EDT is not loaded, default value will be applied

   (default value is obtained from CVTIOCID in the CVT.)

3) If a valid ID is obtained, IEFAB4IE attempts to load the IEFEDTXX member. Note that if a default id is used, the operator will be prompted to respecify an EDT ID if the member cannot be loaded successfully.
This processing will be repeated until EDT is loaded.

Diagram 62 Allocation Load EDT Routine (IEFAB4IE) Part 3 of 7

IEFAB4IE - MODULE OPERATION (Continued)

A) A BLDL is issued for the EDT. If the BLDL fails
a message is sent to the operator and control exits.

B) An unconditional GETMAIN for ECSA storage is done
for the EDT.

C) A LOAD for the EDT is done to bring it in
at the address of the obtained storage.
If the load fails control goes to the load error
routine.
If the load worked, the EDT address is placed into
JESEDT. Also a check is made to ensure that the
EDT is the right version. If it is not, the
the system is put into a wait state.
(code 200, reason code 1).

4) Control returns to the caller.

LOADERR routine:
This is the error entry point if the LOAD fails.
Message IEF338I is issued to the operator.
The system is put into a wait state (code 200,
reason code 3).

Diagram 62 Allocation Load EDT Routine (IEFAB4IE) Part 4 of 7

IEFAB4IE - DIAGNOSTIC AIDS

ENTRY POINT NAME: IEFAB4IE

MESSAGES:

```
IEF338I
 |SCHEDXX LINE YYYYY: | EDT XXXXXXXX IGNORED. | UNRECOGNIZED
 |                    |                       |      KEYWORD.
 |DEFAULT EDT PARAM:  |                       | VALUE NOT VALID.
                      |                       | EXTRANEOUS DATA.

                      | IEFEDTXX NOT LOADED.  | MEMBER NOT FOUND.
                                              | DIRECTORY I/O
                                              |      ERROR.
                                              | BLDL RC
                                              |      XXXX-YYYY.
                                              | LOAD RETURN XXXX.
```

IEF338I SCHEDXX LINE YYYY: DUPLICATE EDT STATEMENT IGNORED.

IEF338I SCHEDXX LINE YYYY: DUPLICATE EDT ID KEYWORD IGNORED.

IEF338I SCHEDXX LINE YYYY: MISSING RIGHT PARENTHESIS ON EDT STMT.

IEF338I SCHEDXX LINE YYYY: DEFAULT EDT ID VALUE XX IS USED.

IEF338I SCHEDXX LINE YYYY: NO OPERANDS SPECIFIED ON EDT STMT

IEF339A RESPECIFY EDT CONFIGURATION ID

IEF927I WRONG LEVEL ELIGIBLE DEVICE TABLE: EDTID DATE TIME

ABEND CODES: None

WAIT STATE CODES:

'200'X, Reason code '003'X

'200'X, Reason code '001'X

RETURN CODES:

zero

REGISTER CONTENTS ON ENTRY:

Register 1 - statement processor parameter list
Register 13 - Address of register save area
Register 14 - Return address
Register 15 - Entry point address

REGISTER CONTENTS ON EXIT:

register 0-14: restored

Register 0: unpredictable
Register 1: address of parameter list for IGFPTERM
Register 2-14: unpredictable
Register 15: entry address of IGFPTERM

Diagram 62 Allocation Load EDT Routine (IEFAB4IE) Part 5 of 7

IEFAB4IE - DIAGNOSTIC AIDS (Continued)

```
Register 0: unpredictable
Register 1: address of parameter list for IGFPTERM
Register 2-14: unpredictable
Register 15: entry address of IGFPTERM
```

Diagram 62 Allocation Load EDT Routine (IEFAB4IE) Part 6 of 7

IEFAB4IE - Allocation load EDT routine **STEP 01**

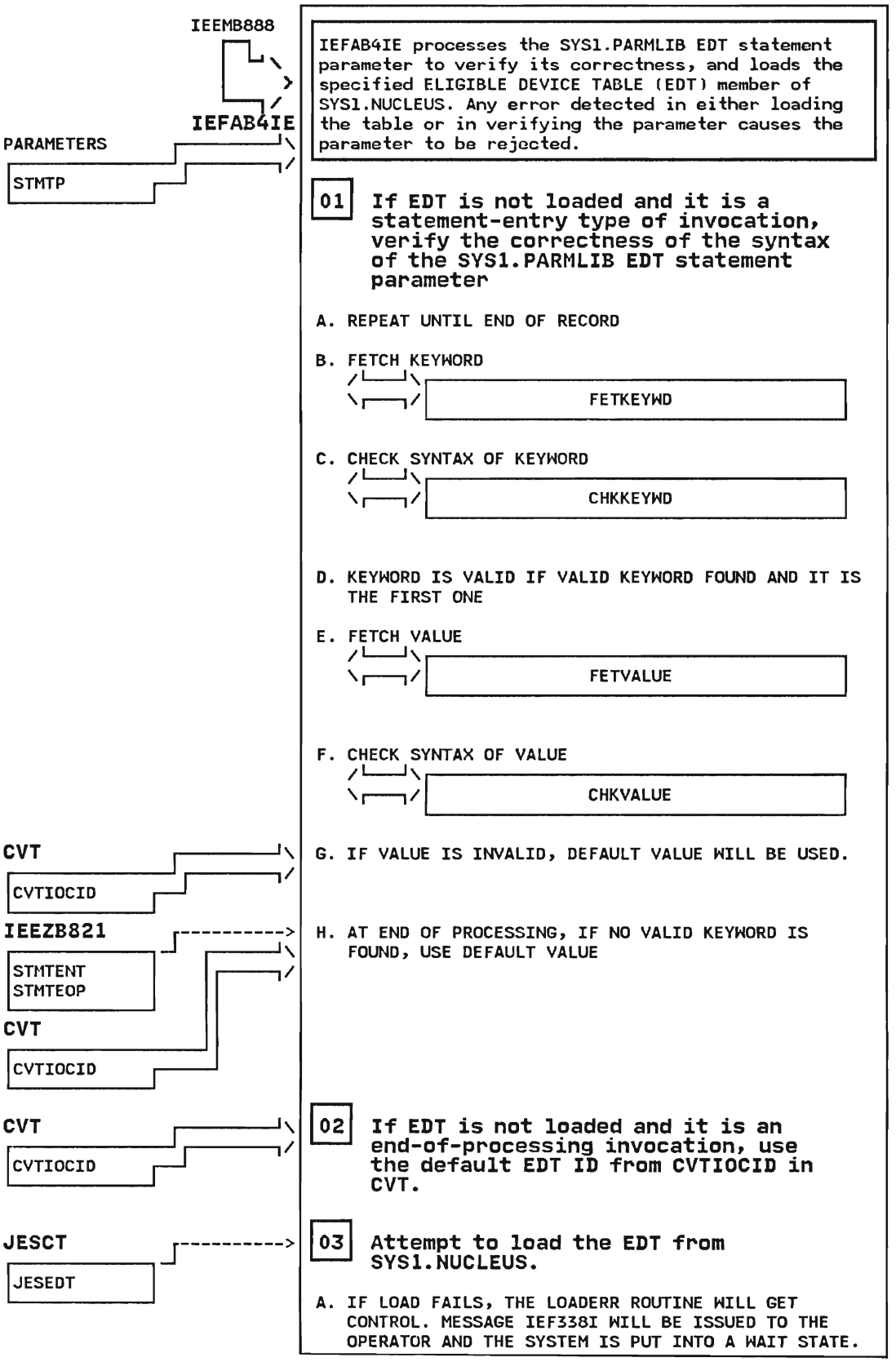
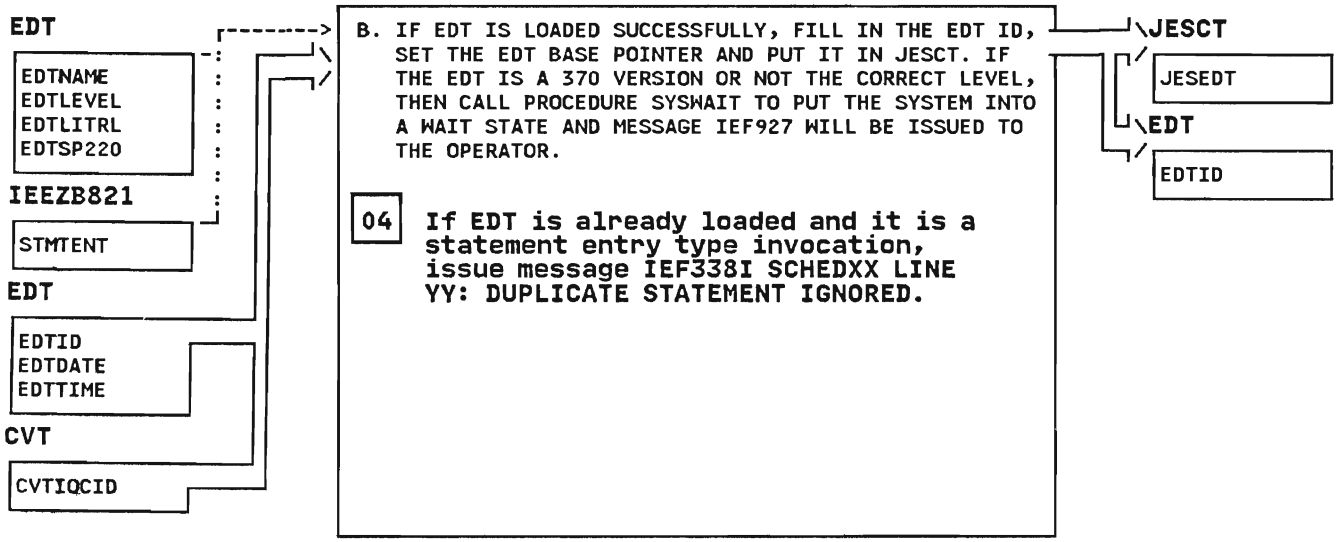


Diagram 62 Allocation Load EDT Routine (IEFAB4IE) Part 7 of 7

IEFAB4IE - Allocation load EDT routine STEP 03B



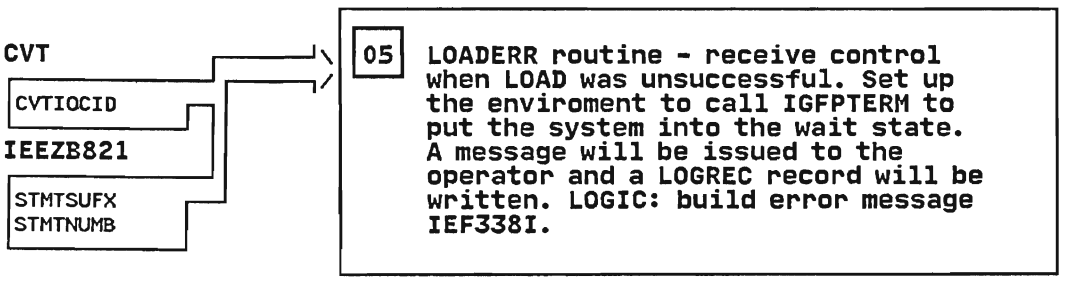


Diagram 63 RESTART Codes Statement Processor (IEFRCSTP) Part 1 of 16

IEFRCSTP - MODULE DESCRIPTION

DESCRIPTIVE NAME: Restart Codes Statement Processor

FUNCTION:
IEFRCSTP is used to customize the standard IBM supplied list of ABEND codes eligible for automatic restart. This list of codes is contained in IEFYRCDS, a data only module residing in SYS1.LINKLIB.
IEFRCSTP is invoked by the Generalized Parmlib Scan Routine (IEEMB888).
Two entry points have been defined within IEFRCSTP:
- Entry point IEFRCADD is responsible for processing RESTART type statements in the SYS1.PARMLIB member SCHEDxx. The RESTART statement enables an installation to expand the IBM supplied list of codes from IEFYRCDS to include user defined ABEND codes.
- Entry point IEFRCDEL is responsible for processing NORESTART type statements in the SYS1.PARMLIB member SCHEDxx. The NORESTART statement enables an installation to delete codes from the list of codes eligible for automatic restart. This list includes both the IBM supplied and the user defined codes.

ENTRY POINT: IEFRCADD

PURPOSE: See Function

LINKAGE: CALL

CALLERS: IEEMB888

INPUT:
Statement Processor Parameter List (IEEZB821)

FIELD	LENGTH/MASK	DESCRIPTION
STMTP	36	
STMTID	4	Identifier 'STMT'
STMTVERS	1	Version number
STMTFLAG	1	Control flags
STMTENT	x'80'	STMT type entry
STMTEOP	x'40'	EOP type entry
STMTERR	x'20'	Error in statement processor routine
STMTLEN	2	Length of parameter list
STMTSUFX	2	Suffix of the SYS1.PARMLIB member being processed
STMTRECL	2	Length of STMT record
STMTRECD	4	Address of a complete logical record
STMTUSEP	4	Pointer to a user parameter area
STMTNVTP	4	Pointer to the NVT
STMTNUMB	4	Record number within SCHEDxx being processed
STMTRSV1	8	Reserved

OUTPUT:
Updated version of the table of restart codes eligible for automatic restart

EXIT NORMAL: Return to caller

EXIT ERROR: None. IEFRCSTP will not return if ABBENDED.

ENTRY POINT: IEFRCDEL

Diagram 63 RESTART Codes Statement Processor (IEFRCSTP) Part 2 of 16

IEFRCSTP - MODULE DESCRIPTION (Continued)

PURPOSE: See Function

LINKAGE: CALL

CALLERS: IEEMB888

INPUT: same as for IEFRCADD

OUTPUT: same as for IEFRCADD

EXIT NORMAL: Return to caller

EXIT ERROR: None. IEFRCSTP will not return if ABBENDED.

EXTERNAL REFERENCES:

ROUTINES:
- IEEMB887 - Routine that parses the RESTART/NORESTART statement for valid keyword and abend codes

DATA AREAS:
- IEFYRCDS - Data only module containing the IBM supplied list of system completion codes eligible for automatic restart

CONTROL BLOCKS: None

TABLES:
- IEFRSTB - Table of codes eligible for automatic restart
- PARSETAB - Table of parse arguments to be passed to the generalized parser (IEEMB087)
- EBCDICX - Translate table used to convert EBCDIC to internal hex
- HEXTABLE - Translate table used to flag any non-hexadecimal character
- ALLTABLE - Translate table used to parse through an erroneous string so that all characters will pass

Diagram 63 RESTART Codes Statement Processor (IEFRCSTP) Part 3 of 16

IEFRCSTP - MODULE OPERATION

This module processes RESTART/NORESTART statements in the SYS1.PARMLIB member SCHEDxx. It does the following:

1. On the first statement (STMT) entry to IEFRCSTP, the IBM supplied list of system completion codes eligible for automatic restart will be loaded into storage. This list is contained in IEFYRCDS, a data only module residing in SYS1.LINKLIB. IEFRCSTP will obtain storage from extended CSA to accomodate the default list plus an additional 50 entries. A copy of the IBM supplied list of codes eligible for automatic restart will be placed in the storage area obtained and anchored in the JESRSTRT field of the JESCT. The IBM supplied list will be used as a base and entries will be added or deleted, based on any RESTART/NORESTART statements that appear in the SYS1.PARMLIB member SCHEDxx, to form the table of codes eligible for automatic restart (IEFRSTB).

2. For each STMT entry, IEFRCSTP calls the Generalized Parse Routine (IEEMB887) to parse the statement for valid syntax.
   If the statement is valid, the codes specified are either deleted from the table of codes eligible for automatic restart (IEFRSTB), or added to the end of the user defined portion of the table.
   IEFRCSTP will continue processing codes until the end of the statement or an erroneous code is encountered.
   If an invalid code is encountered, IEFRCSTP will issue an informational message and stop processing the statement. The remainder of the statement will be ignored.

3. On the end of processing (EOP) entry, if no valid RESTART/NORESTART statements were specified, IEFRCSTP will obtain storage from extended CSA. IEFRCSTP will then load the IBM supplied list of system completion codes eligible for automatic restart, found in data only module IEFYRCDS, and copy it into the storage area obtained. In this case the table of codes eligible for automatic restart (IEFRSTB) will consist only of the IBM supplied list of codes eligible for automatic restart.

4. If no operands are specified on the RESTART/NORESTART statement, IEFRCSTP will issue an informational message stating that no operands were specifed. The statement will be ignored.

IEFRCSTP is entered at one of two points, IEFRCADD or IEFRCDEL. The point of entry is determined by the general parmlib scan routine, IEEMB886.

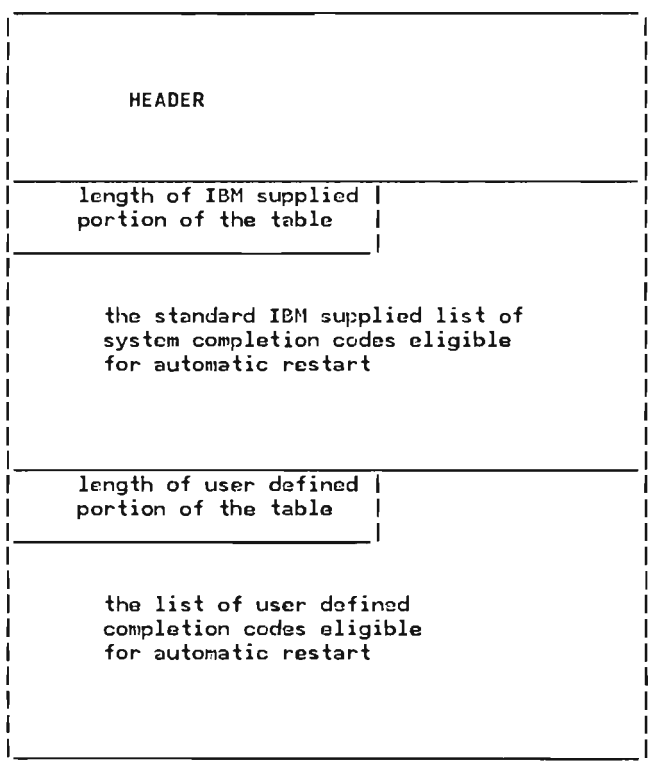
- When IEEMB886 invokes IEFRCSTP to process a RESTART statement type, control is passed to the RESTART codes statement processing entry point within IEFRCSTP (IEFRCADD). The restart statement processing will then search through the restart codes table (IEFRSTB) for a matching code. If a matching code is not found, the restart codes statement processing will add the valid code to the user defined portion of IEFRSTB.
If a matching code is found, the restart codes statement processing will issue an informational message stating that a duplicate code exists.

Diagram 63 RESTART Codes Statement Processor (IEFRCSTP) Part 4 of 16

IEFRCSTP - MODULE OPERATION (Continued)

- When (IEEMB883) invokes IEFRCSTP to process a NORESTART statement type, control is passed to the NORESTART statement processing entry point within IEFRCSTP (IEFRCDEL). The NORESTART statement processing looks for a matching code in the table of restart codes (IEFRSTB). If a matching code is found, the code will be deleted from the table. If a matching code is not found, the NORESTART statement processing will issue an informational message stating that the code was not found.

Diagram of IEFRSTB:
(the table of eligible restart codes)



RECOVERY OPERATION:
The processing for this module takes place during nucleus initialization (NIP) time. Although an ESTAE environment could be established at this point, if an ABEND did occur, control would not be received because RTM is not fully initialized yet, therefore an ESTAE environment is not established.

Diagram 63 RESTART Codes Statement Processor (IEFRCSTP) Part 9 of 16

IEFRCSTP - Restart Codes Statement Processor STEP 11

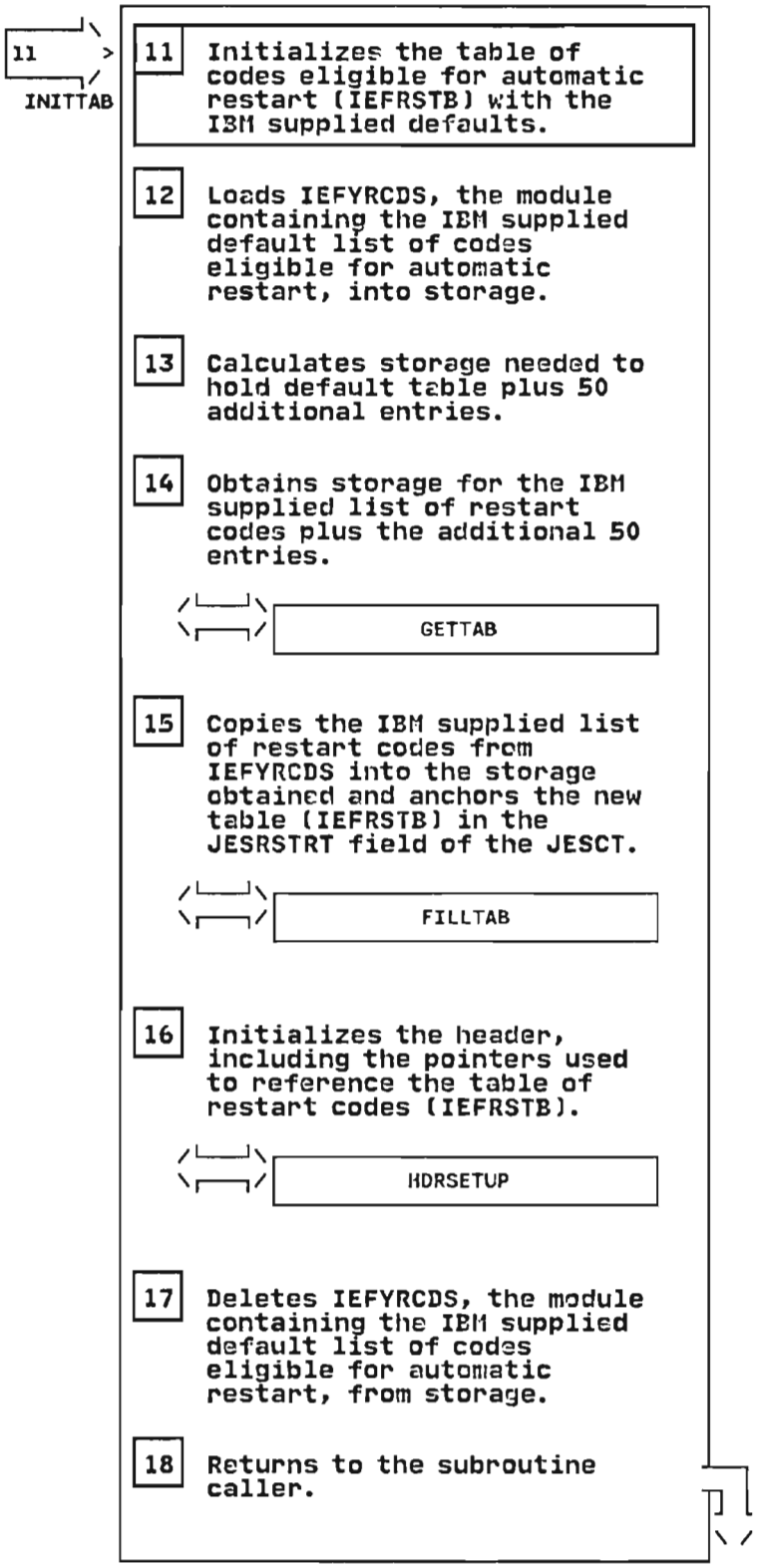
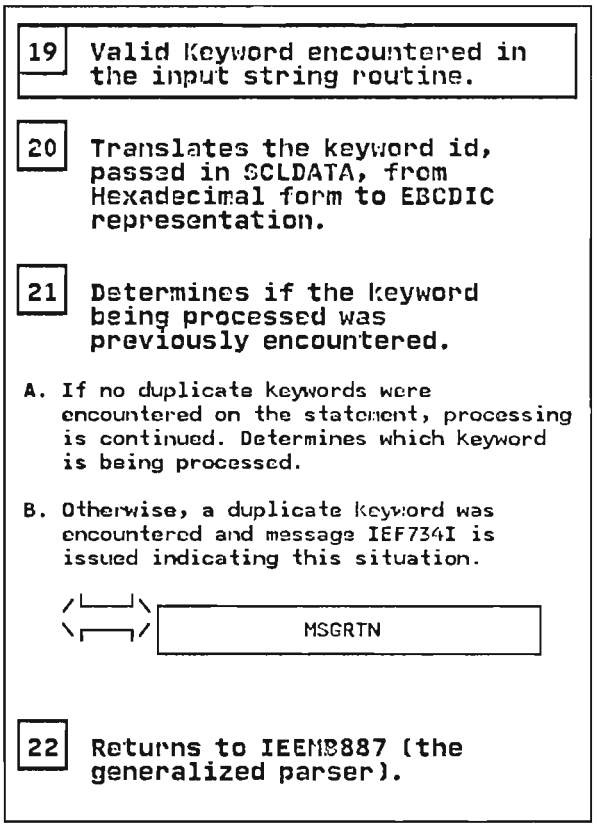


Diagram 63 RESTART Codes Statement Processor (IEFRCSTP) Part 10 of 16

IEFRCSTP - Restart Codes Statement Processor STEP 19



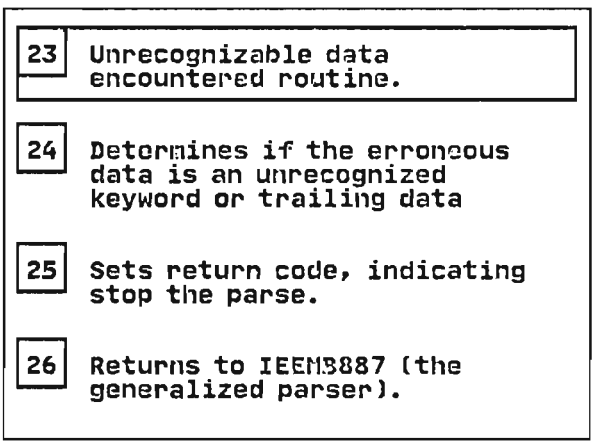


Diagram 63 RESTART Codes Statement Processor (IEFRCSTP) Part 7 of 16

IEFRCSTP - Restart Codes Statement Processor STEP 01

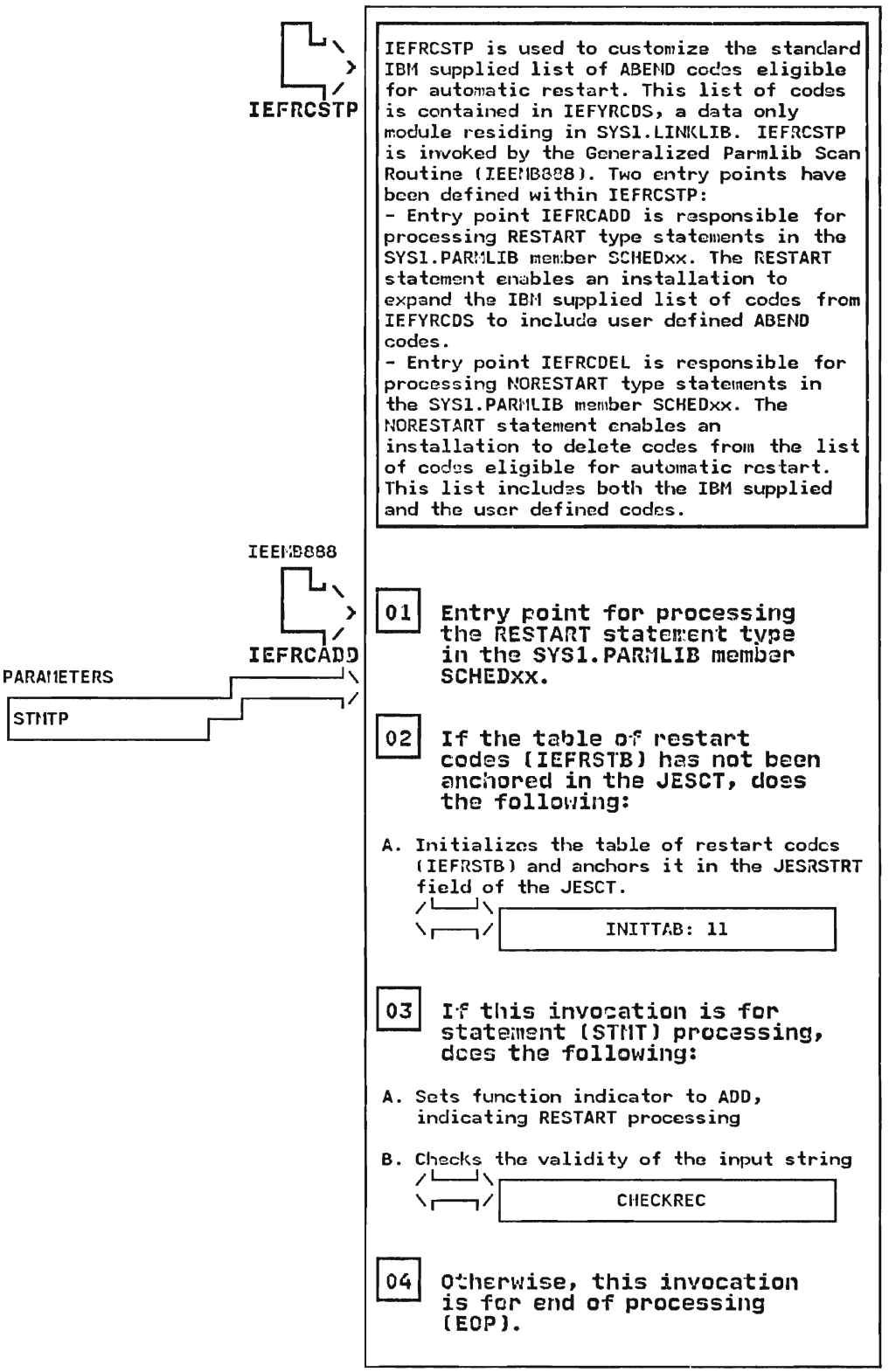


Diagram 63 RESTART Codes Statement Processor (IEFRCSTP) Part 8 of 16

IEFRCSTP - Restart Codes Statement Processor STEP 05

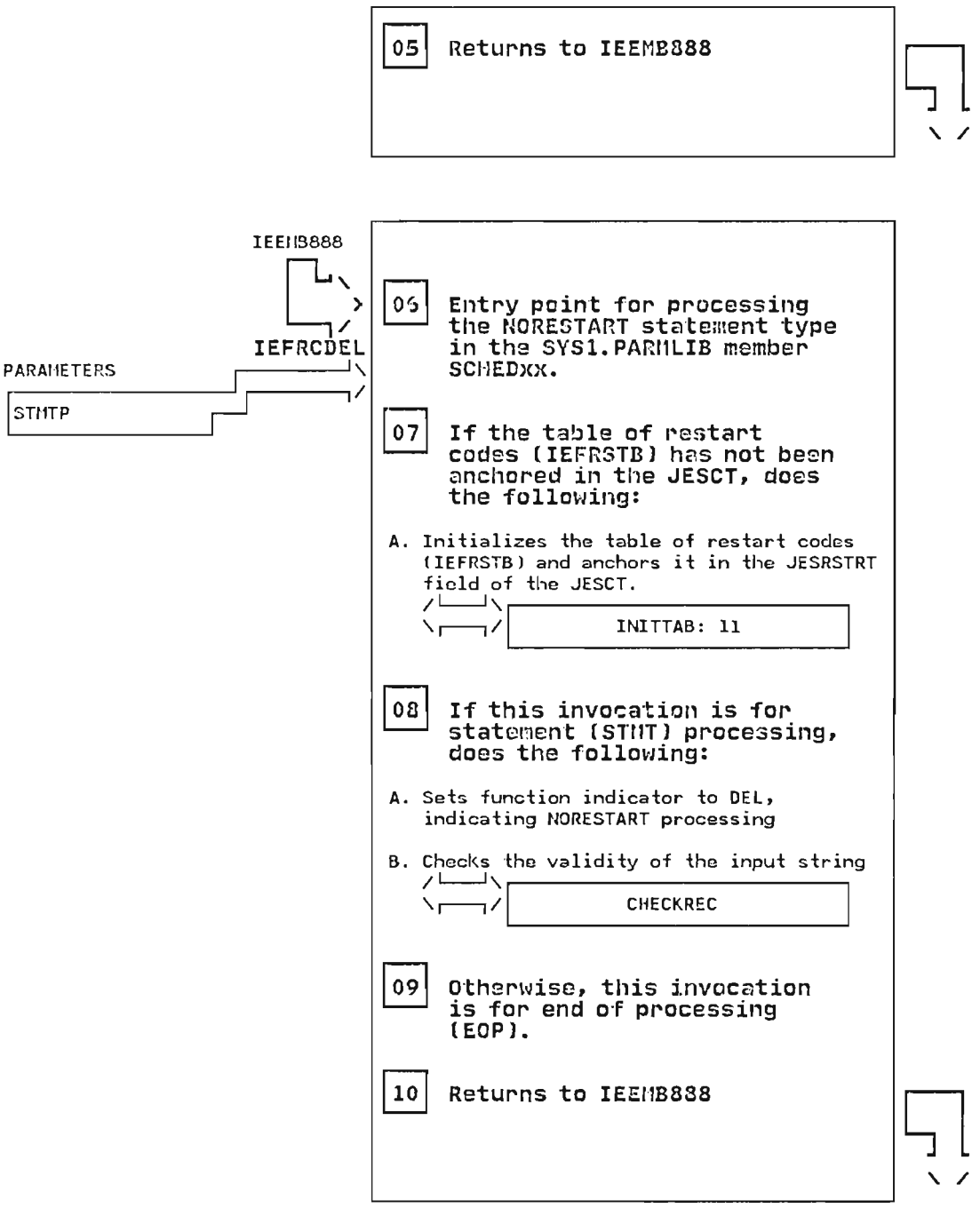


Diagram 63 RESTART Codes Statement Processor (IEFRCSTP) Part 5 of 16

IEFRCSTP - DIAGNOSTIC AIDS

ENTRY POINT NAMES: IEFRCADD
IEFRCDEL

MESSAGES:

```
  IEF738I mem LINE num: <RESTART,NORESTART> STMT IGNORED.
          NO OPERANDS SPECIFIED.

  IEF734I mem LINE num: <RESTART,NORESTART> <CODE code,
          STATEMENT> <IGNORED,ACCEPTED>. REASON= kyrc

 Where:
  mem  - indicates which SCHEDxx SYS1.PARMLIB member
         contains the bad statement

  num  - indicates which line in the SCHEDxx SYS1.PARMLIB
         member is being processed

  code - indicates the abend code (specified on the
         statement) in error

  kyrc - represents a keyword-reason code for the error
         encountered
    ky - keyword, the numerical representation of the
         keyword being processed  (decimal)
    rc - reason code, the numerical representation of the
         error encountered  (decimal)

  The following are valid keyword/reason code (kyrc)
  combinations:

     ky: 01 - Unexpected data encountered

     rc: 08 - Unrecognized keyword
         28 - Keyword list not valid
         44 - End delimiter in a keymord list (right
              parenthesis) missing
         48 - Data following keyword list ending delimiter
              (right parenthesis) encountered

     ky: 02 - CODES(

     rc: 16 - Value out of range
         52 - Code already exists
         56 - Code not found
         60 - Duplicate keyword encountered
```

ABEND CODES: None

WAIT STATE CODES: None

RETURN CODES:

```
 ENTRY POINT IEFRCADD:

   EXIT NORMAL:

     0 - Request successfully completed

 ENTRY POINT IEFRCDEL:

   EXIT NORMAL:
```

Diagram 63 RESTART Codes Statement Processor (IEFRCSTP) Part 6 of 16

IEFRCSTP - DIAGNOSTIC AIDS (Continued)

0 - Request successfully completed

REGISTER CONTENTS ON ENTRY:

ENTRY POINT IEFRCADD:

```
Register  0      = Undefined
Register  1      = Address of a word that points to
                   a parameter list
Registers 2-12   = Undefined
Register  13     = Address of savearea
Register  14     = Return address
Register  15     = Entry point address
```

ENTRY POINT IEFRCDEL:

same as for IEFRCADD

REGISTER CONTENTS ON EXIT:

ENTRY POINT IEFRCADD:

EXIT NORMAL:

```
Registers 0-14   = Restored
Register  15     = Return Code
```

ENTRY POINT IEFRCDEL:

EXIT NORMAL:

same as for IEFRCADD

Diagram 63 RESTART Codes Statement Processor (IEFRCSTP) Part 11 of 16

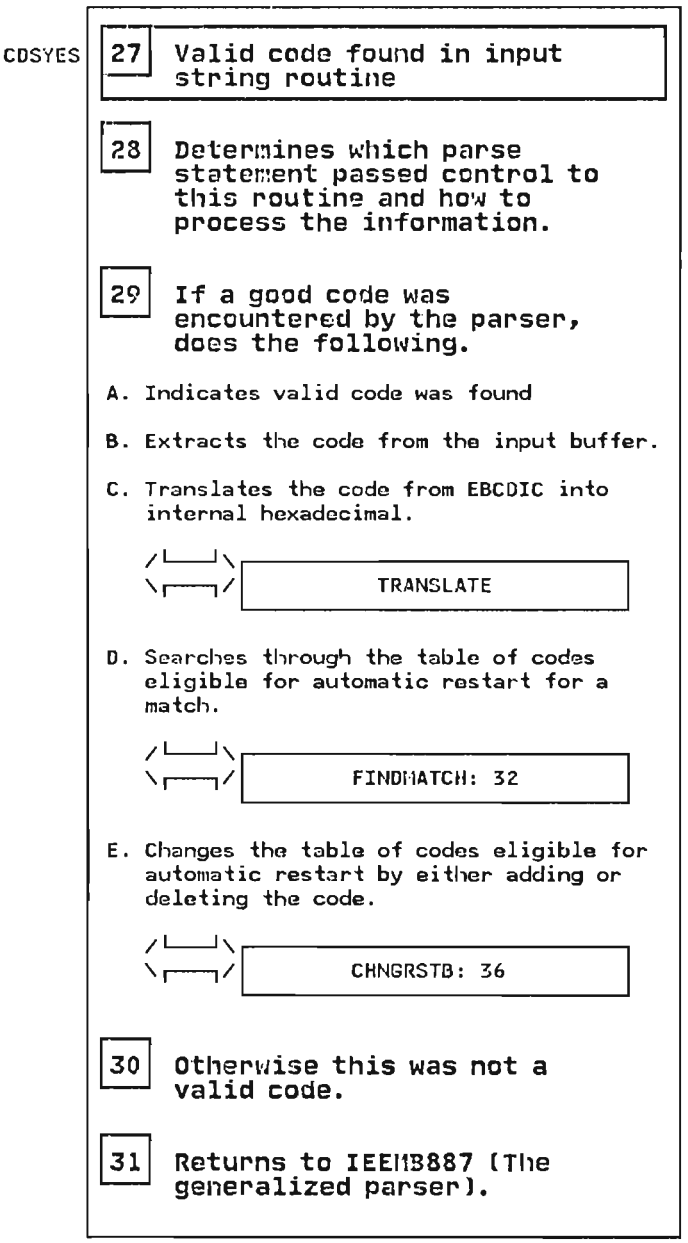
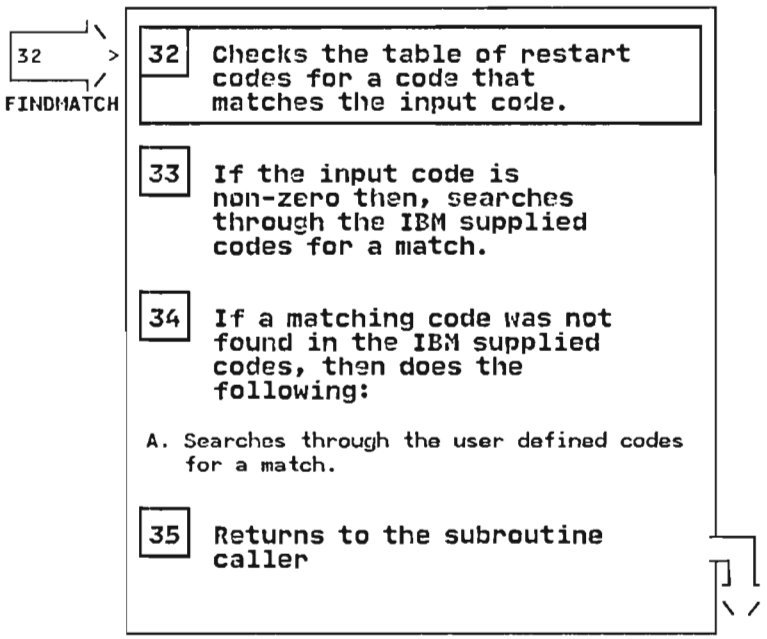


Diagram 63 RESTART Codes Statement Processor (IEFRCSTP) Part 12 of 16

IEFRCSTP - Restart Codes Statement Processor STEP 32



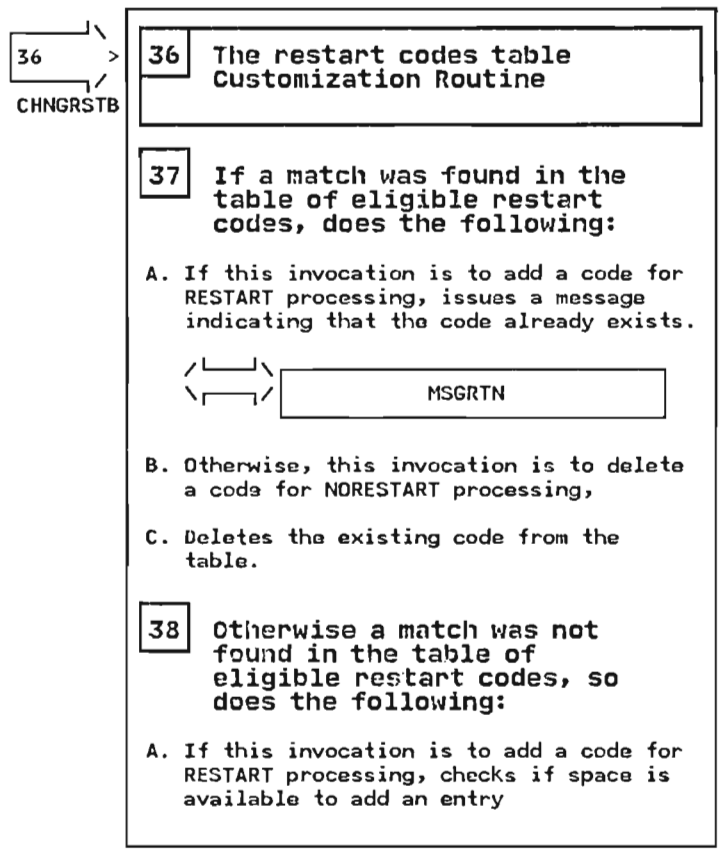


Diagram 63 RESTART Codes Statement Processor (IEFRCSTP) Part 13 of 16

IEFRCSTP - Restart Codes Statement Processor STEP 38B

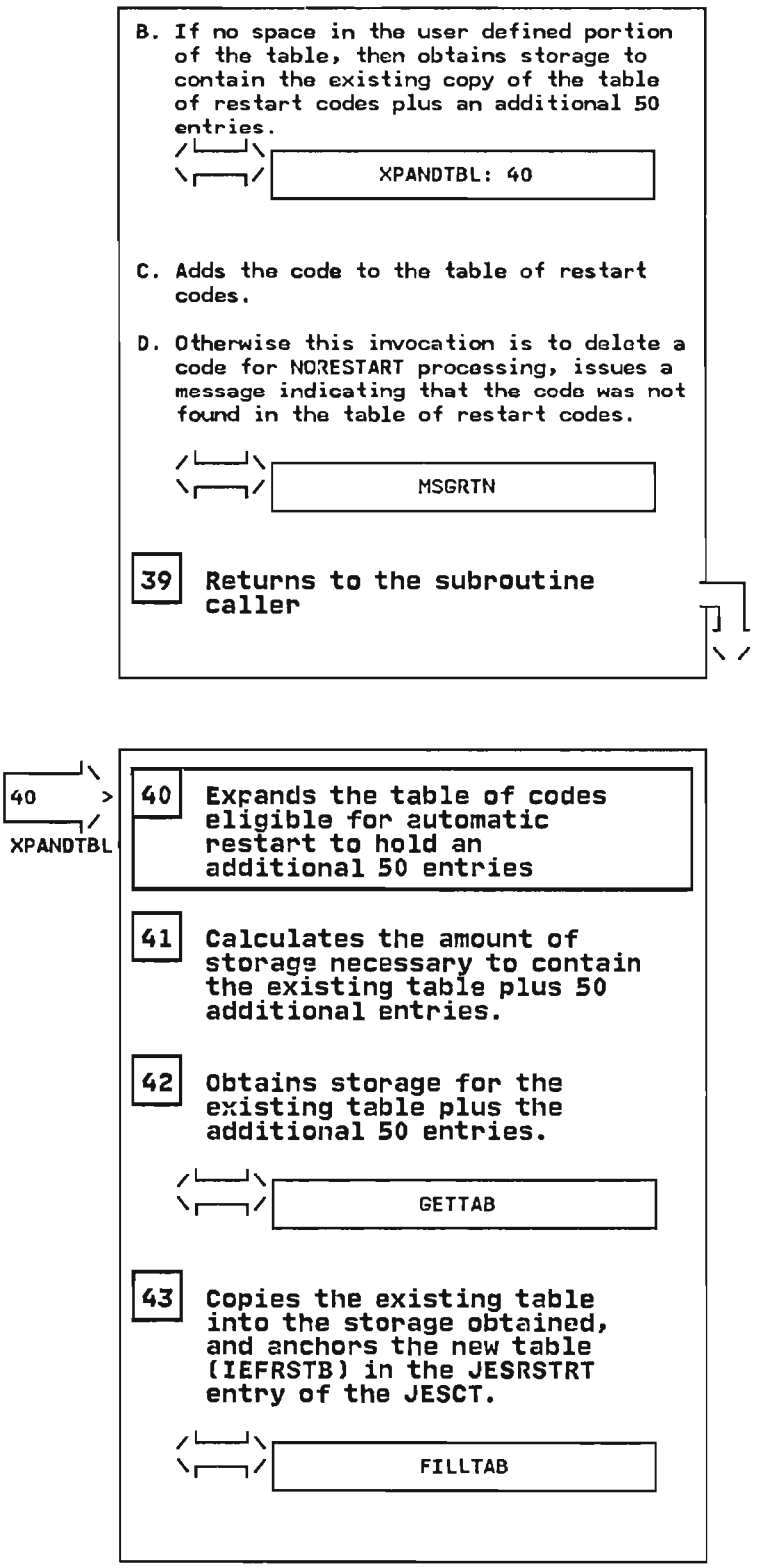
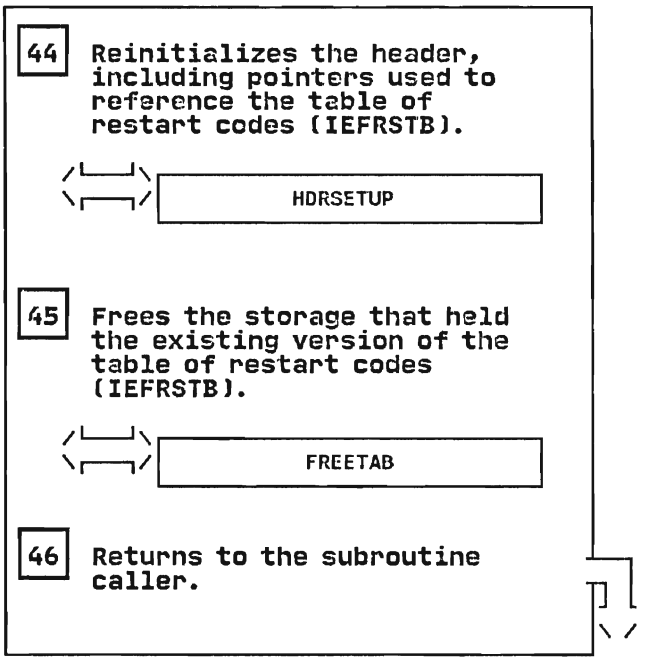
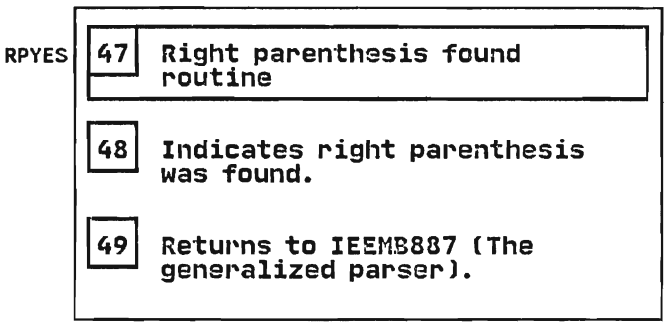


Diagram 63 RESTART Codes Statement Processor (IEFRCSTP) Part 14 of 16

IEFRCSTP - Restart Codes Statement Processor STEP 44





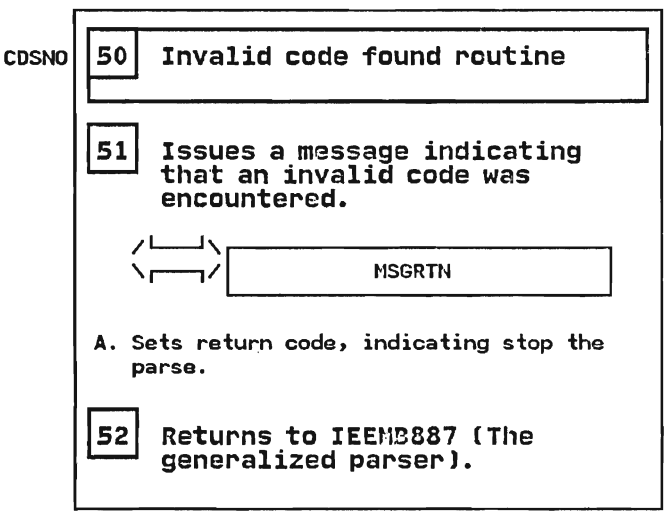
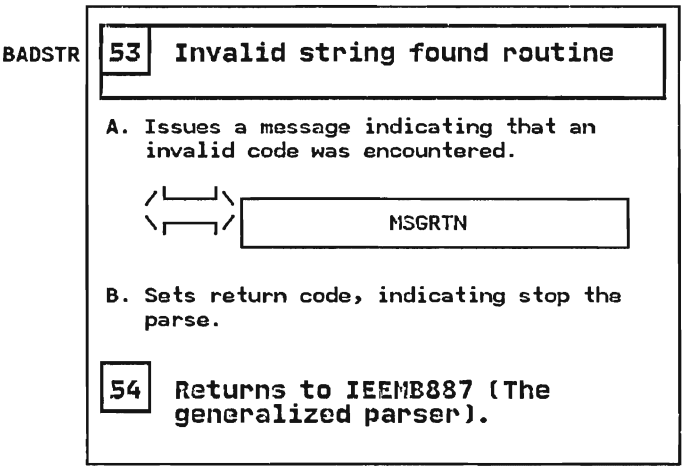


Diagram 63 RESTART Codes Statement Processor (IEFRCSTP) Part 15 of 16

IEFRCSTP - Restart Codes Statement Processor STEP 53



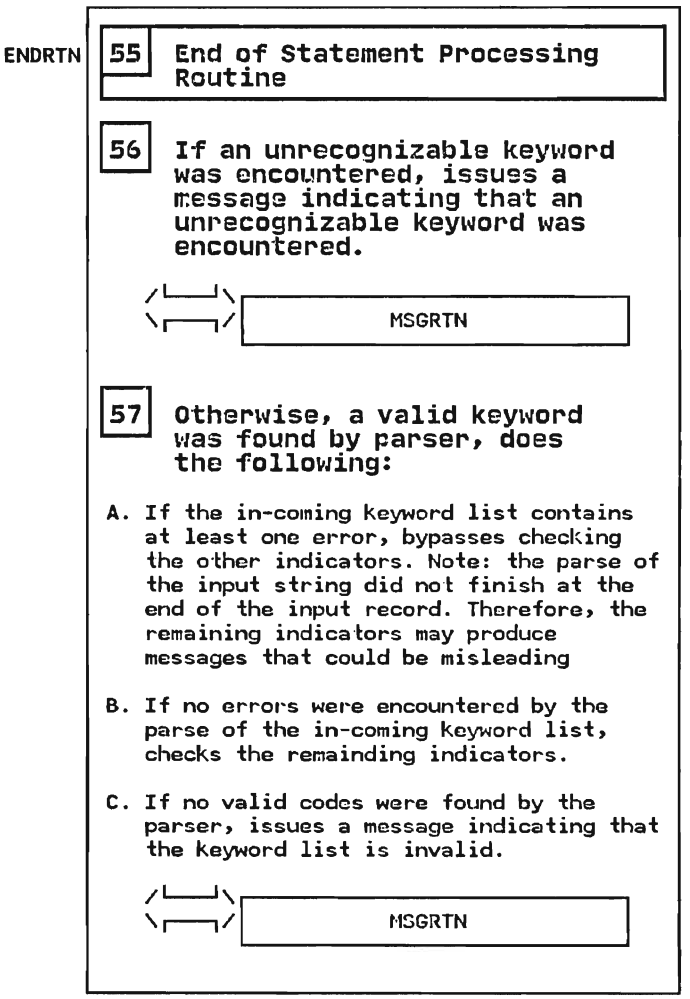
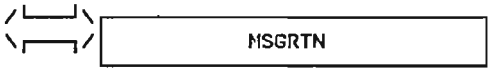


Diagram 63 RESTART Codes Statement Processor (IEFRCSTP) Part 16 of 16

IEFRCSTP - Restart Codes Statement Processor STEP 57D

D. Otherwise, the keyword list contains only valid codes, if the keyword list ending delimiter was not encountered by the parser, issues a message indicating that the keyword list ending delimiter is missing.



E. Otherwise, a right parenthesis was found. If extraneous data was encountered after the keyword list ending delimiter, then issues a message indicating that extraneous data was encountered.

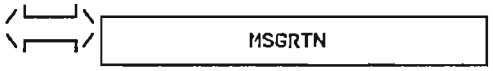


Diagram 64. MSSC Initialization (IEAVNP19) Part 1 of 2

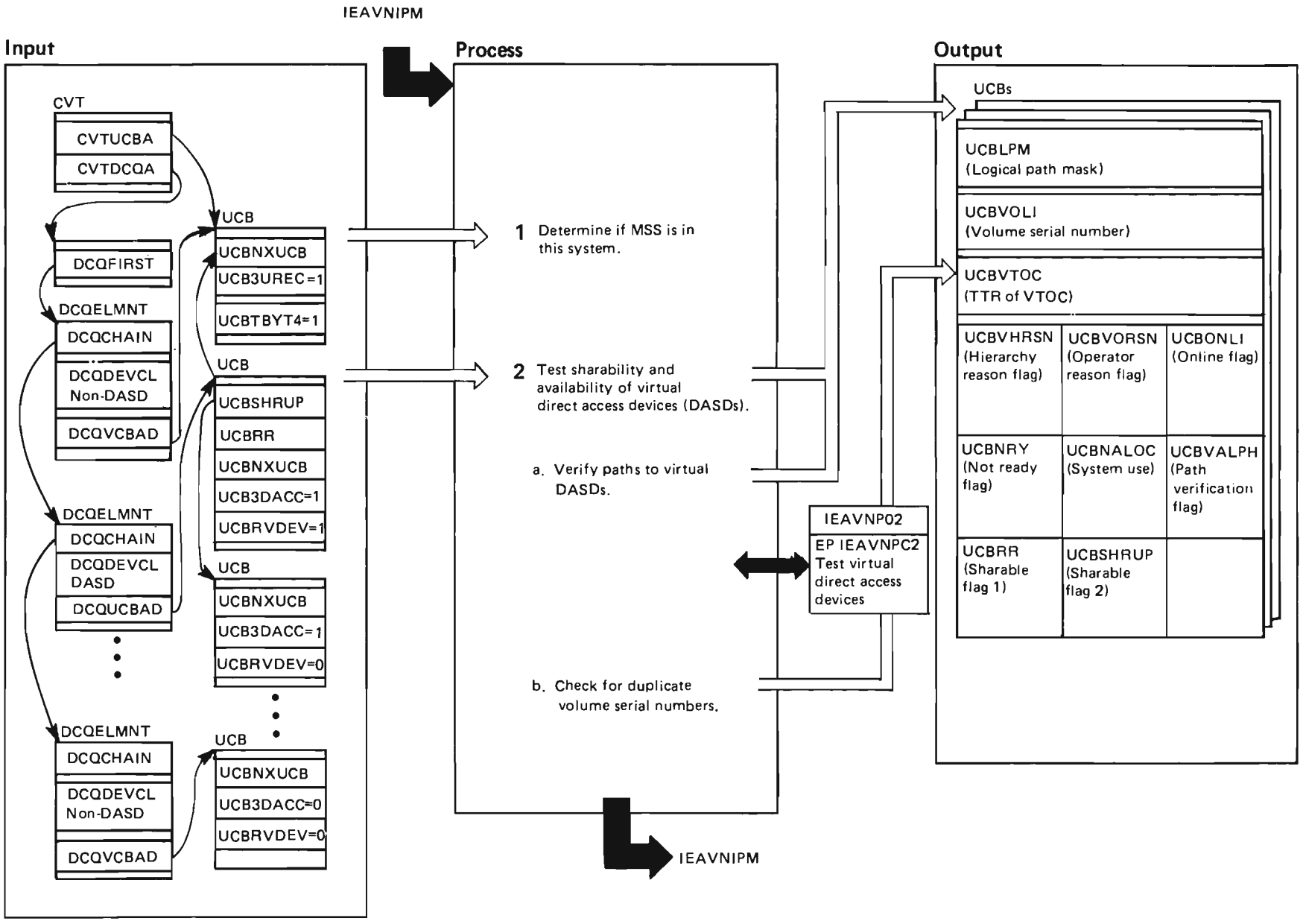


Diagram 64. MSSC Initialization (IEAVNP19) Part 2 of 2

Extended Description	Module	Label
IEAVNIPM calls the mass storage system communicator (MSSC) RIM to determine whether the mass storage system (MSS) is to be used in this system and, if so, to verify paths to the virtual direct access devices (DASDs).		
1 The MSSC RIM scans the unit record device class queue (DCQ) to locate all UCBs associated with mass storage controllers (MSCs) (UCBTBYT4=X'42'), and take appropriate action:	IEAVNP19	
• If the MSSC RIM finds no MSC UCBs, it performs no further initialization and returns control to IEAVNIPM. CVTICB remains zero, an indication to other system components that MSS is not in this system. • If the MSSC RIM finds online MSC UCBs, it sets the UCBALOC bits to show that the devices corresponding to the UCBs are allocated and continues processing.		
2 The MSSC RIM calls the IOS RIM at entry point IEAVNPC2 to test the MSS devices for shareability and availability and initialize the UCBs. The IOS RIM uses the DCQ to select the UCBs for each MSS device. The fields UCB3DACC and UCBRVDEV indicate whether devices are virtual direct access storage devices.	IEAVNP02	
a. The IOS RIM calls the service routine IOSVNPTH to test the availability of each of these devices. IOSVNPTH issues a STARTIO macro instruction along each installed online path to determine if the device is physically available on that path.		
When a device is marked sharable at SYSGEN (UCBRR=1), IOSVNPTH verifies its sharability using a RELEASE CCW command. If the device doesn't have this capability, it rejects the RELEASE command. IOSVNPTH sets the flags UCBSHRUP=0 and UCBRR=0 to indicate that the device is not sharable.		NPTHIO
If the device was designated as not sharable, IOSVNPTH issues an I/O NOP CCW command.		
IOSVNPTH times all I/O operations. If an operation does not complete within 1.5 seconds, IOSVNPTH issues SVC 16 to purge the operation and resets UCBLPM to indicate that the device is offline.		NPTHSTIO
If any path to the device verifies successfully, IOSVNPTH sets the following UCB fields:		
Field initialized — *Initialization value* UCBONLI — '1'—device is online UCBLPM — Indication of which paths are available to the device UCBVALPH — '0'—indicates that a path is verified for this device		
If no paths verify successfully, IOSVNPTH marks the device offline, sets UCBLPM to the initial state (the same value as the path installed mask), and sets fields UCBVHRSN and UCBVORSN to indicate why the device is offline.		
If, after the above processing, the MSS device is to be brought online and a device initialization exit exists for the device, then IOSVNPTH issues a sense-id command. The device initialization exit then initializes any feature fields unique to this device. If the exit fails, the device is marked offline.		DEVINIT
b. To check for duplicate volume serial numbers for online direct access device, the IOS RIM scans all UCBs. If it finds a duplicate serial number, it issues message IEA212A to request that the operator demount one of the volumes. After the operator responds with the device number, the IOS RIM issues the MSSC SVC instruction to demount the volume.	IEAVNP02	NP2LDUP
On return from the MSSC SVC, the IOS RIM places zeroes in the volume serial number and VTOC fields of the UCB for the device that had the volume demounted and marks the device not read.		
For additional information, see *Mass Storage System Extensions Logic: MSS Communicator (MSSC).*		

Diagram 65 CLOCK Processing Resource Initialization (IEAVNP20) Part 1 of 11

IEAVNP20 - MODULE DESCRIPTION

DESCRIPTIVE NAME: CLOCK Processing Resource Initialization Module

FUNCTION:
IEAVNP20 processes SYS1.PARMLIB member CLOCKxx.

ENTRY POINT: IEAVNP20

PURPOSE:
IEAVNP20 is the RIM that processes SYS1.PARMLIB member CLOCKxx.

LINKAGE: CALL

CALLERS: NIP Control/Services Routine (IEAVNIPM)

INPUT: NIP Parmtab (pointed to out of the NIP Vector Table - NVT)

OUTPUT:
TPCPRMPT in the TPC set to one if the TOD Clock is in the not set state. Also see Statement Subfunction Routine entry points.

EXIT NORMAL: Return to IEAVNIPM

ENTRY POINT: IEAVOPER

PURPOSE:
IEAVOPER is the OPERATOR Statement Subfunction Routine. IEAVOPER processes the OPERATOR statements from SYS1.PARMLIB member CLOCKxx. IEAVOPER is invoked once for each OPERATOR statement in each SYS1.PARMLIB CLOCKxx member specified, and once for end-of-processing (after all OPERATOR statements specified, if any, have been processed).

LINKAGE: CALL

CALLERS: General Parmlib Scan Routine (IEEMB888)

INPUT: Statement Processor Parameter List (IEEZB821)

OUTPUT:
TPCPRMPT in the TPC updated according to the first OPERATOR statement specification, as long as no errors were detected on the OPERATOR statements.

EXIT NORMAL: Return to caller

ENTRY POINT: IEAVTZ

PURPOSE:
IEAVTZ is the TIMEZONE Statement Subfunction Routine. IEAVTZ processes the TIMEZONE statements from SYS1.PARMLIB member CLOCKxx. IEAVTZ is invoked once for each TIMEZONE statement in each SYS1.PARMLIB CLOCKxx member specified, and once for end-of-processing (after all TIMEZONE statements specified, if any, have been processed).

LINKAGE: CALL

CALLERS: General Parmlib Scan Routine (IEEMB888)

INPUT: Statement Processor Parameter List (IEEZB821)

OUTPUT:
CVTTZ in the CVT updated according to the first TIMEZONE statement specification, as long as no errors

Diagram 65 CLOCK Processing Resource Initialization (IEAVNP20) Part 2 of 11

IEAVNP20 - MODULE DESCRIPTION (Continued)

were detected on the TIMEZONE statement.

EXIT NORMAL: Return to caller

EXTERNAL REFERENCES:

ROUTINES: IEEMB888 - General Parmlib Scan Routine

CONTROL BLOCKS:
CVT - Communications Vector Table (R,W)
IEAPPNIP - NIP Parmtab (R)
NVT - NIP Vector Table (R)
TPC - Timer Supervision Work Area (R,W)

TABLES:
IEEZB819 - General Parmlib Scanner Parameter List (W)
IEEZB821 - Statement Processor Parameter List (R)

SERIALIZATION: Serialized by IEAVNIPM

Diagram 65 CLOCK Processing Resource Initialization (IEAVNP20) Part 3 of 11

IEAVNP20 - MODULE OPERATION

IEAVNP20 processes SYS1.PARMLIB member CLOCKxx. Main entry point IEAVNP20 invokes IEEMB888 (General PARMLIB Scan Routine) to scan the NIP Parameter Table for a CLOCK specification, read the contents of the CLOCKxx members specified, and call the statement subfunction routines, IEAVOPER and IEAVTZ. IEAVOPER processes the OPERATOR statements found by IEEMB888 and entry point IEAVTZ processes the TIMEZONE statements found by IEEMB888.

The OPERATOR statement from SYS1.PARMLIB member CLOCKxx has the following format:

OPERATOR PROMPT or
OPERATOR NOPROMPT

The TIMEZONE statement from SYS1.PARMLIB member CLOCKxx has the following format:

TIMEZONE E.hh.mm.ss or
TIMEZONE W.hh.mm.ss

where hh.mm.ss represent hours, minutes, and seconds, with minutes and seconds optional.

RECOVERY OPERATION:
Within NIP, IEAVNP20 operates as a RIM under IEAVNIPM. IEAVNP20 does not establish a recovery environment since RTM services have not yet been established and NIP traps abnormal conditions (e.g. ABENDs and program checks) such that a permanent system wait state results.

Diagram 65 CLOCK Processing Resource Initialization (IEAVNP20) Part 4 of 11

IEAVNP20 - DIAGNOSTIC AIDS

ENTRY POINT NAMES: IEAVNP20
IEAVOPER
IEAVTZ

MESSAGES:

The following message is issued by IEAVNP20, using the NIPWTO service. It is issued unconditionally when CLOCK member processing is complete.

IEA598I TIME ZONE = E/W.hh.mm.ss

Using the NIPPRMPT service causes the following existing message to be issued by IEAVNP20. It is issued when an error is detected in a statement or in the Parmtab specification of CLOCKxx members.

IEA906A RESPECIFY CLOCK PARM OR ENTER EOB

The following messages are issued by the Statement Subfunction Routines, using the NIPWTO service. They are issued upon detection of the particular error condition described.

IEA599I CLOCKxx LINE nnnn: &&&&&&&& STMT IGNORED. UNRECOGNIZED PARM.
IEA600I CLOCKxx LINE nnnn: TIMEZONE STMT IGNORED. VALUE NOT VALID.
IEA601I CLOCKxx LINE nnnn: DUPLICATE &&&&&&&& STMT IGNORED.

nnnn - indicates line number containing the error
&&&&&&&& - indicates the statement type in error, either OPERATOR or TIMEZONE

ABEND CODES: None

WAIT STATE CODES: None

RETURN CODES:

ENTRY POINT IEAVNP20:

EXIT NORMAL:

Register 15 = 0 - Processing completed

ENTRY POINT IEAVOPER:

EXIT NORMAL:

Register 15 = 0 - Processing completed successfully
20 - Error detected in an OPERATOR statement

ENTRY POINT IEAVTZ:

EXIT NORMAL:

Register 15 = 0 - Processing completed successfully
20 - Error detected in a TIMEZONE statement

REGISTER CONTENTS ON ENTRY:

ENTRY POINT IEAVNP20:

Diagram 65 CLOCK Processing Resource Initialization (IEAVNP20) Part 5 of 11

IEAVNP20 - DIAGNOSTIC AIDS (Continued)

```
   Register 0     = Undefined
   Register 1     = Undefined
   Register 2     = Address of the NIP Vector Table (NVT)
   Register 3     = Address of the CVT
   Register 4-12  = Undefined
   Register 13    = Address of an 18-word save area
   Register 14    = Return address
   Register 15    = Entry point address

 ENTRY POINT IEAVOPER:

   Register 0     = Undefined
   Register 1     = Address of a word which contains the
                    address of IEEZB821
   Register 2-12  = Undefined
   Register 13    = Address of 18-word save area
   Register 14    = Return address
   Register 15    = Entry address

 ENTRY POINT IEAVTZ:

   Register 0     = Undefined
   Register 1     = Address of a word which contains the
                    address of IEEZB821
   Register 2-12  = Undefined
   Register 13    = Address of 18-word save area
   Register 14    = Return address
   Register 15    = Entry address
```

REGISTER CONTENTS ON EXIT:

```
 ENTRY POINT IEAVNP20:

   EXIT NORMAL:

     Register  0-12 = Restored
     Register  13   = Address of an 18-word save area
     Register  14   = Return address
     Register  15   = Return code

 ENTRY POINT IEAVOPER:

   EXIT NORMAL:

     Register  0-12 = Restored
     Register  13   = Address of 18-word save area
     Register  14   = Return address
     Register  15   = Return code

 ENTRY POINT IEAVTZ:

   EXIT NORMAL:

     Register  0-12 = Restored
     Register  13   = Address of 18-word save area
     Register  14   = Return address
     Register  15   = Return code
```

Diagram 65 CLOCK Processing Resource Initialization (IEAVNP20) Part 6 of 11

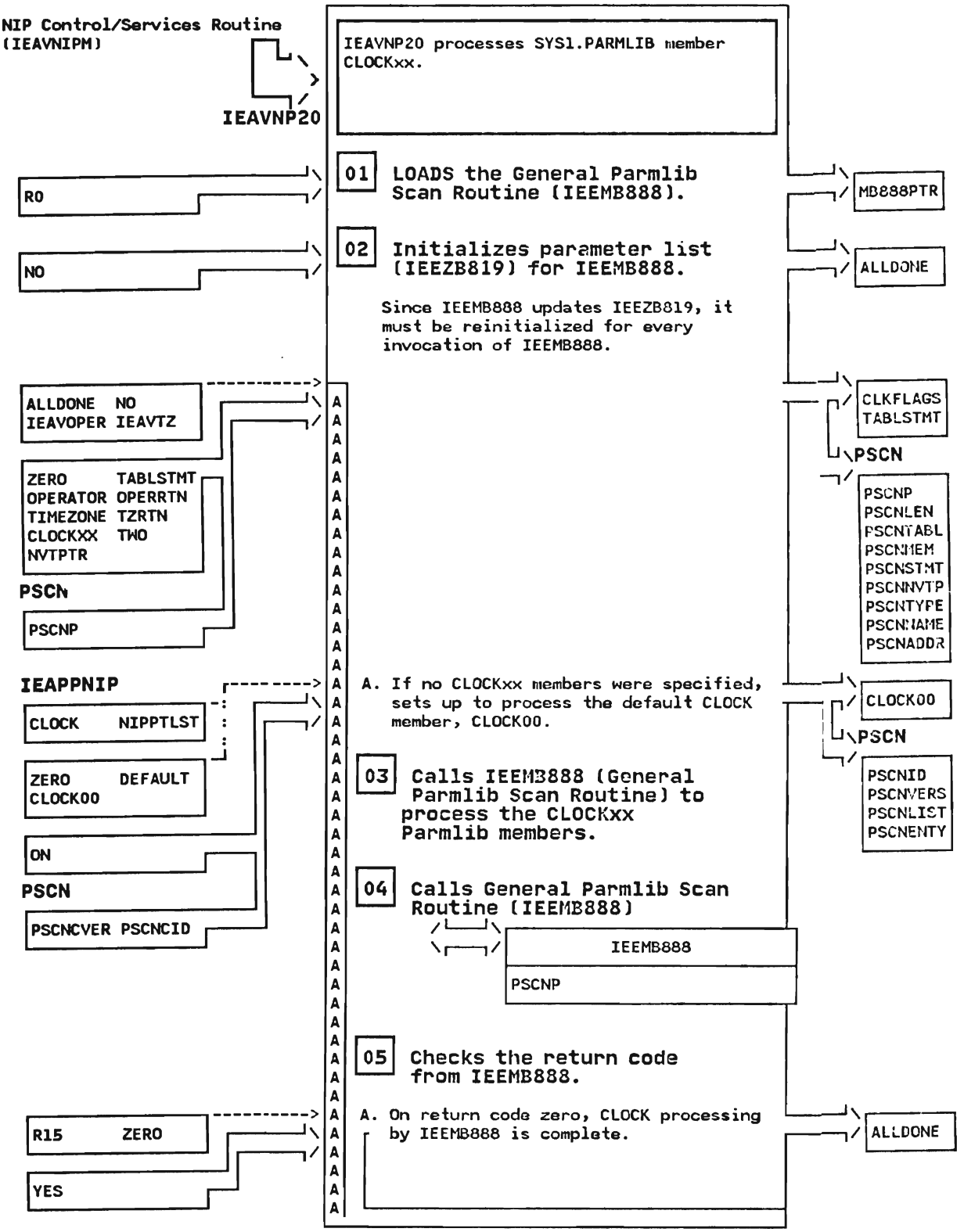


Diagram 65 CLOCK Processing Resource Initialization (IEAVNP20) Part 7 of 11

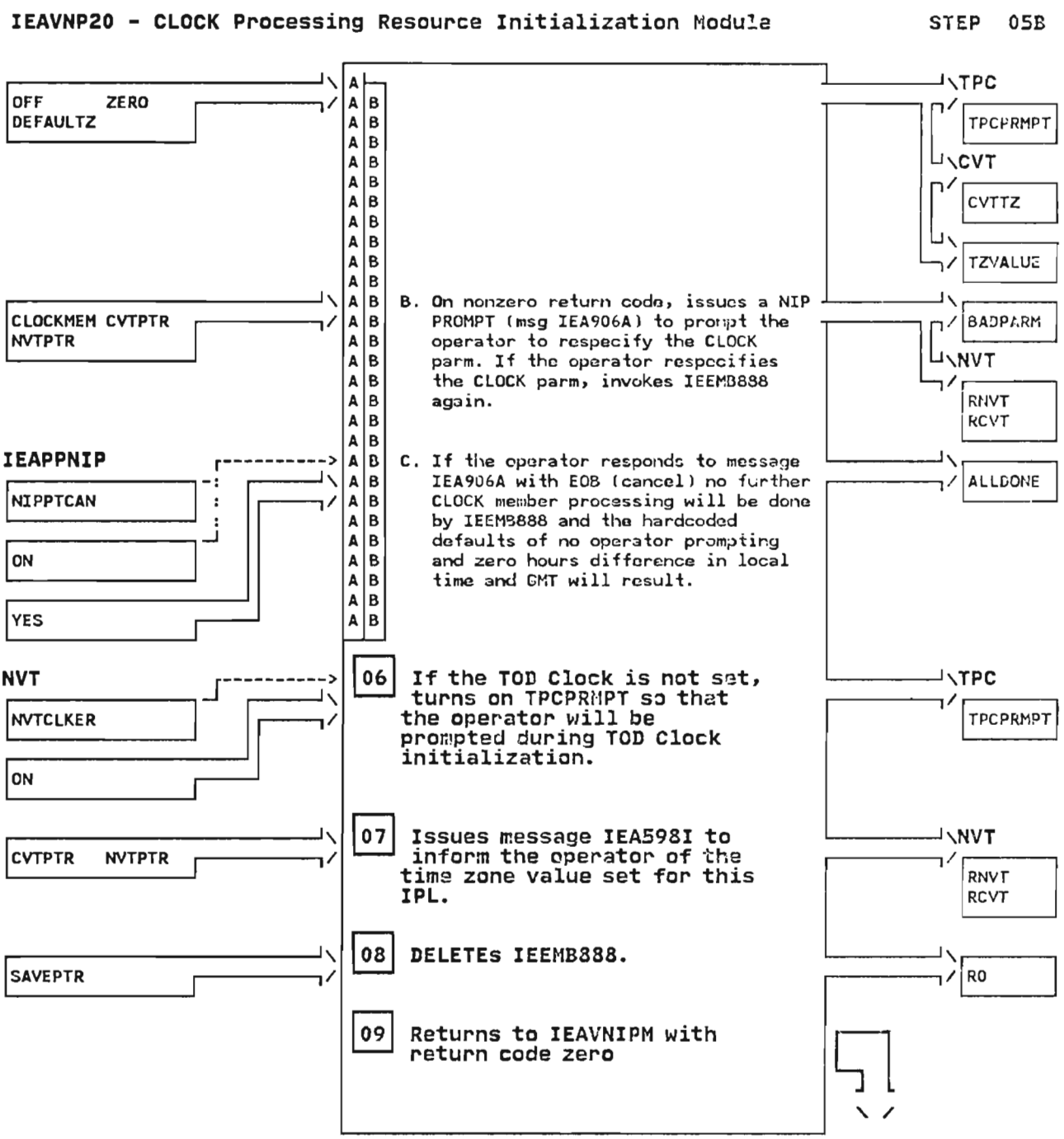


Diagram 65 CLOCK Processing Resource Initialization (IEAVNP20) Part 8 of 11

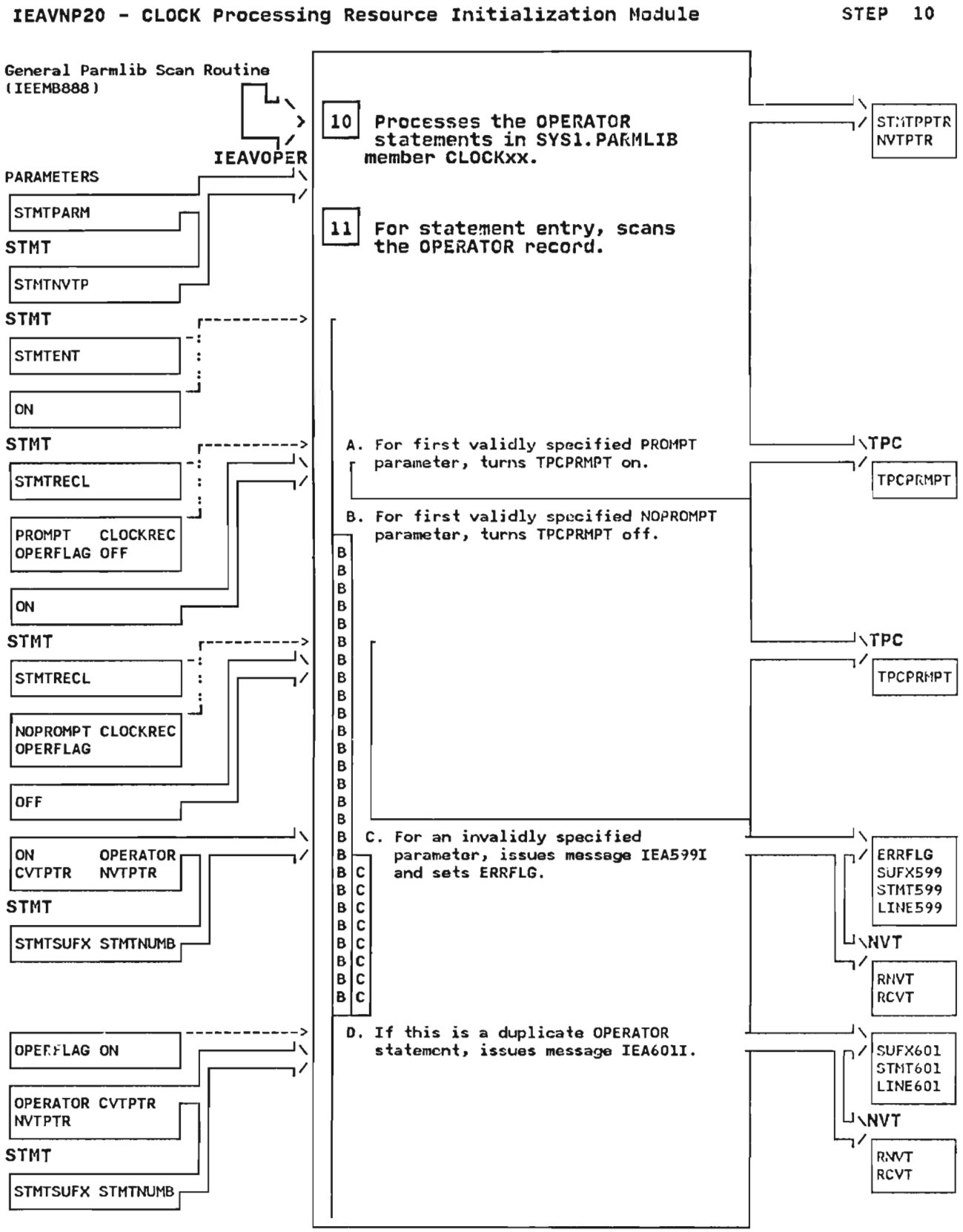


Diagram 65 CLOCK Processing Resource Initialization (IEAVNP20) Part 9 of 11

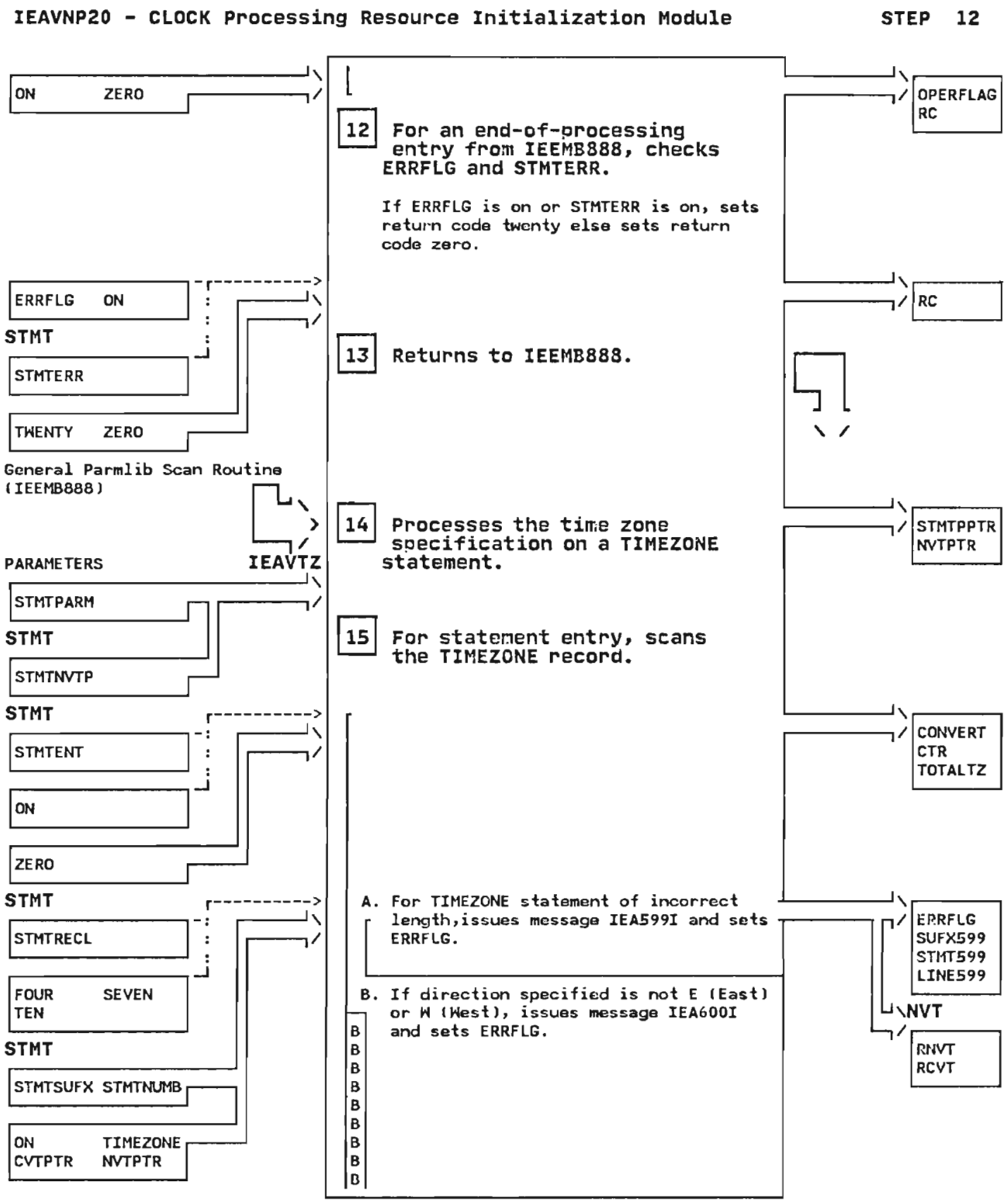


Diagram 65 CLOCK Processing Resource Initialization (IEAVNP20) Part 10 of 11

IEAVNP20 - CLOCK Processing Resource Initialization Module STEP 16

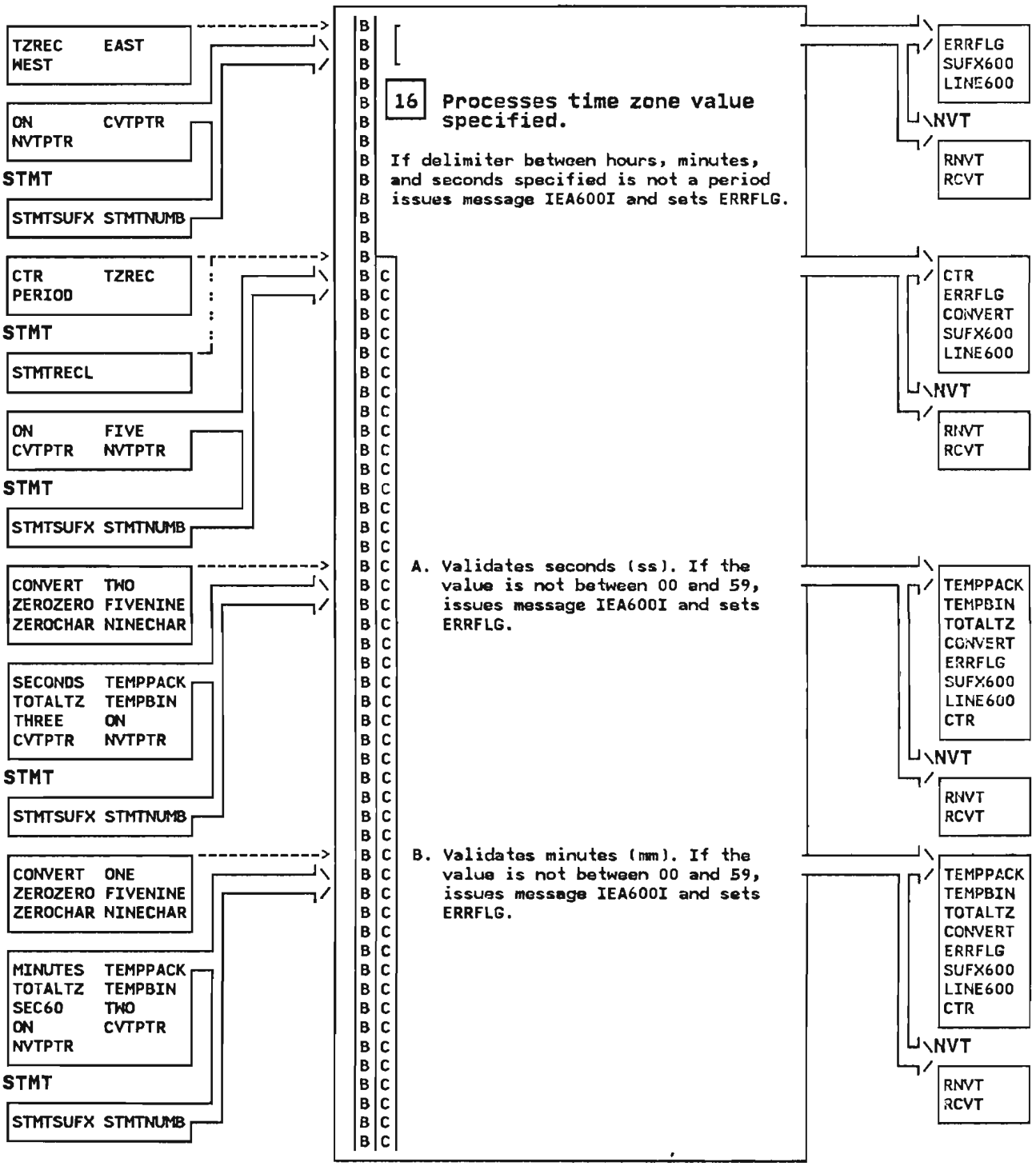


Diagram 65 CLOCK Processing Resource Initialization (IEAVNP20) Part 11 of 11

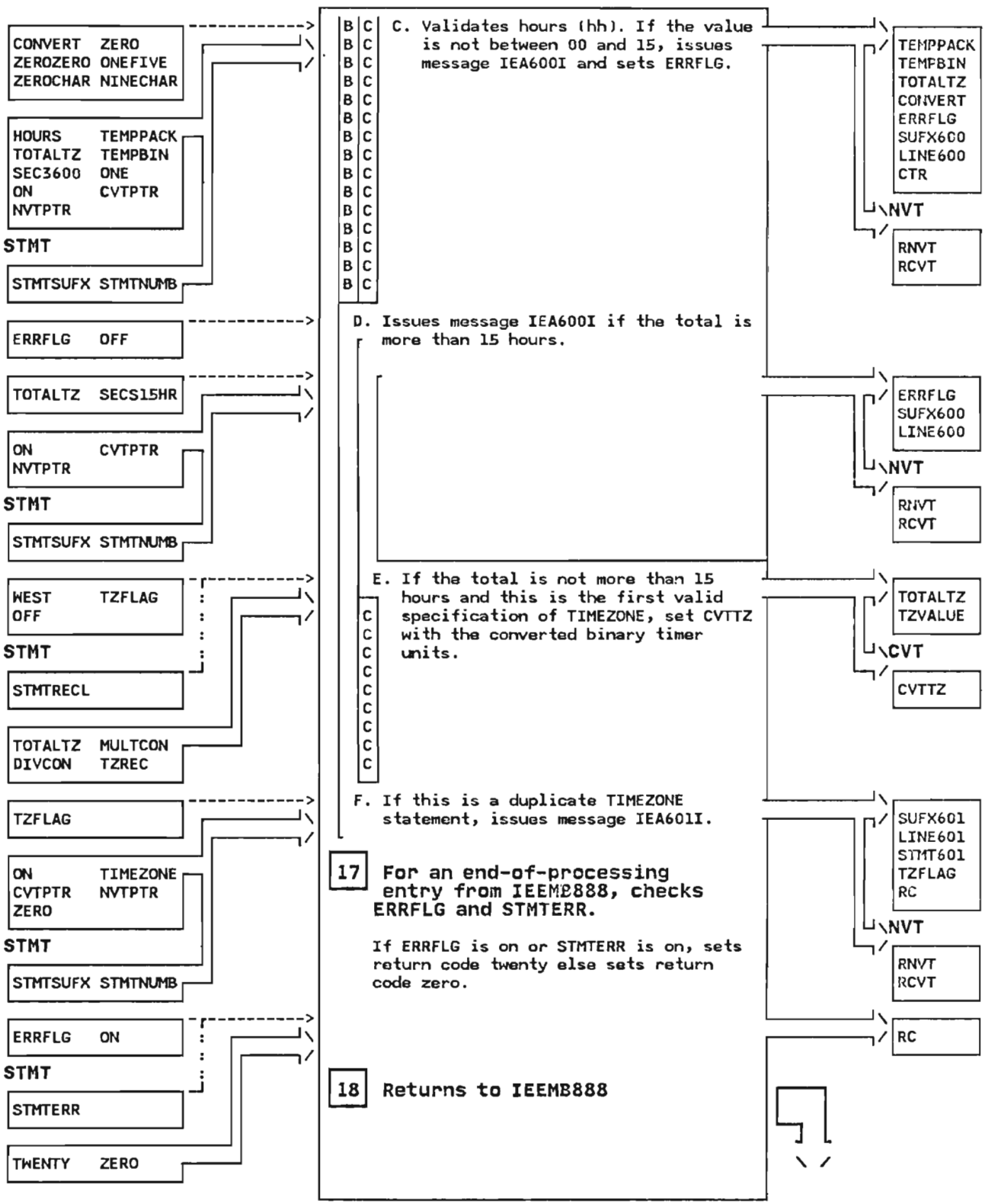


Diagram 66. IOS Parameter Initialization (IEAVNPF2) Part 1 of 2

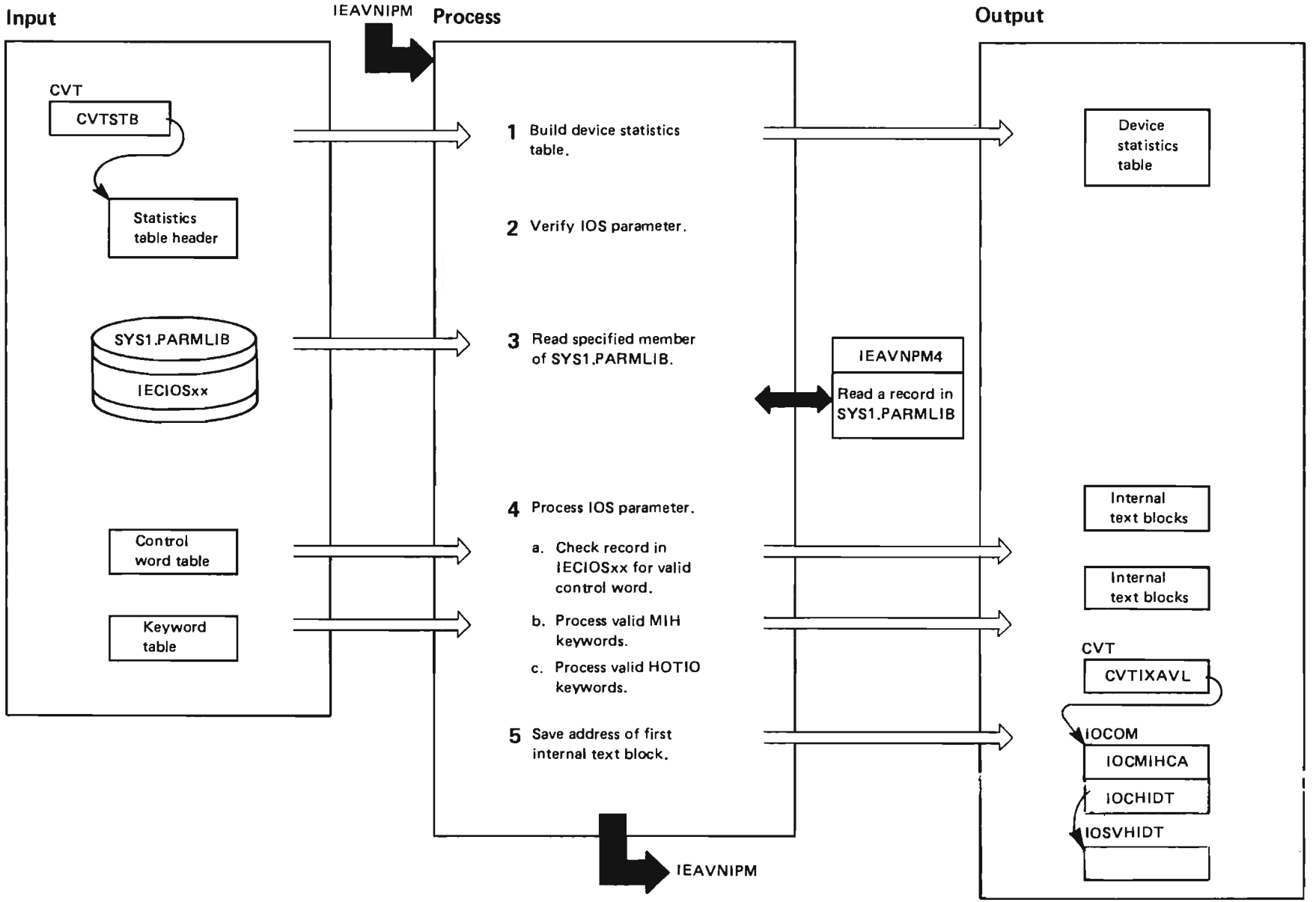


Diagram 66. IOS Parameter Initialization (IEAVNPF2) Part 2 of 2

Extended Description	Module	Label
IEAVNIPM calls the input/output supervisor (IOS) RIM, IEAVNPF2, to process the IOS parameter. IOS options are specified at initialization by the IOS=xx system parameter. The IOS RIM reads member IECIOSxx from SYS1.PARMLIB and scans the control statements in this member for valid syntax and for command type. The IOS RIM also obtains storage for the device statistics table.		
1 The IOS RIM issues a GETMAIN (storage form) for common pageable storage (subpool 241) and builds the device statistics table. This table is pointed to by the statistics table header, which was built during system generation. The table consists of pointers, one entry for every 2560 bytes of storage.	IEAVNPF2	DSTPROC
2 The IOS RIM verifies the IOS parameter. If the IOS parameter was not specified, the RIM returns control to IEAVNIPM. Otherwise, the RIM checks the 'xx' value. If it finds a syntax error, it prompts the operator for a correct value.		VLDLOOP
3 The IOS RIM calls IEAVNPM4 to read one record at a time from member IECIOSxx of SYS1.PARMLIB.	IEAVNPM4	
4 The IOS RIM checks the return code from IEAVNPM4. If IEAVNPM4 has successfully read a record from IEACIOSxx, the IOS RIM processes the record.		
a. It uses its internal control word table to determine if the record contains a valid control word 'MIH' or 'HOTIO'. If it does not, the RIM writes an error message, then reads the next record from IECIOSxx. Otherwise, the RIM checks the syntax of the keywords in the operand field. If the RIM finds syntax errors, it issues message IEA419I containing a description of the error.	IEAVNPF2	BLDBLK1 CARDRTN
b. For valid MIH keywords, the RIM builds an internal text block for each keyword. Valid MIH keywords are: INTERVAL=mm:ss DASD=mm:ss TIME=mm:ss DEV=devlist devtype=mm:ss where mm indicates minutes, ss indicates seconds, devlist is a list of one or more devices, and devtype is either 3330V or 3851. The TIME and DEV keywords must be specified together. The 3330V and 3851 keywords are not really recognized by this RIM, but are simply passed along in an internal text block for processing by the missing interrupt handler (MIH). An invalid keyword on a record does not cause the entire record to be considered invalid. For example, if the record contains: MIH TIME=05:00, DASD=FF:25, DEV=3C0, the IOS RIM issues two error messages: one message indicates the invalid operand following the DASD keyword and the other message indicates that the characters from the beginning of the invalid operand field (FF:25) to the next valid delimiter are skipped. It will also build an internal text block that describes the valid TIME and DEV keyword values. If, however, a TIME or DEV keyword is invalid, both keywords are considered invalid and the IOS RIM issues an error message. Processing for all internal text blocks will be done by MIH initialization in IOSRMIHT. For a description of IOSRMIHT see the IOS section of *System Logic Library*.		BLDBLK1 MIHCRDNO
c. For valid HOTIO keywords, the RIM updates the HIDT (hot I/O detection table) to reflect the information specified. Valid keywords are: • DVTHRSH=nnn (DVTHRSH has a maximum value of 32, 767) • DFLT110=(non-recursive, recursive action) • DFLT111=(non-recursive, recursive action) • DFLT112=(non-recursive, recursive action) Valid values for recursive and non-recursive recovery action are: BOX – box the device CHPK – perform channel path recovery CHPF – perform channel path offline OPER – obtain recovery action from the operator An invalid keyword causes only that keyword to be considered in error. Any other valid keyword on the record will be processed and the HIDT updated. The only exception occurs when an invalid delimiter is placed in the variable length operand field of DVTHRSH and preceded by a valid DFLT11x keyword. In this case, everything from the error on in the control card is ignored.		VLDLOOP BLDMSGBK
5 When the return code from IEAVNPM4 indicates that it has reached the end of IECIOSxx, the IOS RIM puts the address of the first internal text block in IOCMIHCA and returns control to IEAVNIPM.		

Diagram 67. Volume Attribute List Processor (IEAVNP15/IEAVAP00) Part 1 of 2

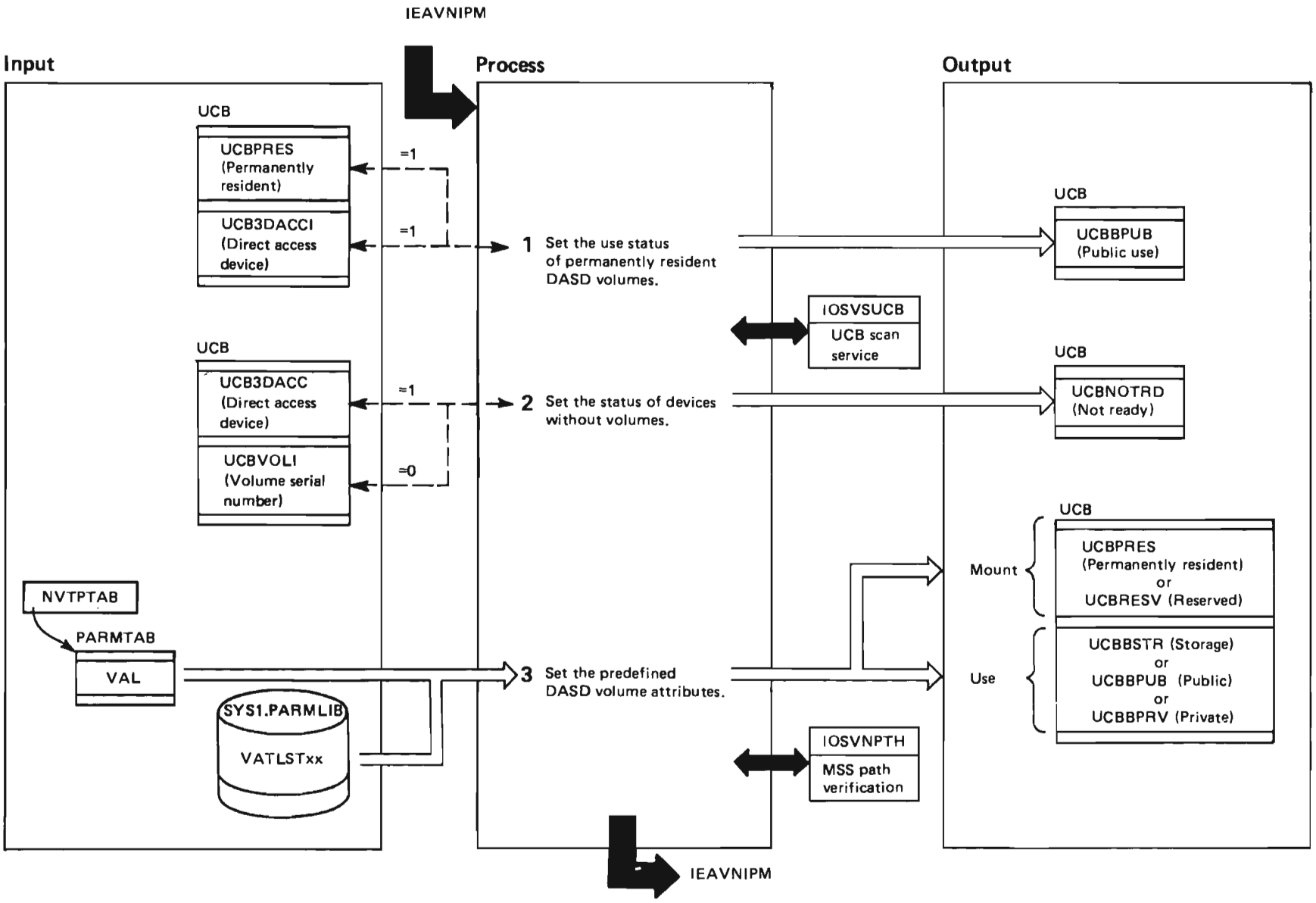


Diagram 67. Volume Attribute List Processor (IEAVNP15/IEAVAP00) Part 2 of 2

Extended Description	Module	Label
IEAVNIPM calls the volume attribute RIM, IEAVAP00, to initialize UCBs with mount and use status information about direct access data set volumes. Note that IEAVNIPM knows this module as IEAVNP15. IEAVAP00 is an entry point in IEAVNP15.		
1 The RIM invokes the UCB scan service (IOSVSUCB) in IOS to obtain the addresses of all UCBs for direct address devices. If the RIM finds a UCB that represents a direct access device that contains a permanently resident volume, it sets the use status to public (bit UCBBPUB).	IEAVAP00	INITUCBS
2 If the RIM finds a UCB that represents a direct access device with no volume mounted (field UCBVOLI is zero), it sets the not ready flag on (UCBNOTRD).		INITUCBS
3 The installation can specify mount and use attributes for direct access volumes by entering the attributes in member VATLSTxx of SYS1.PARMLIB. Using the VAL system parameter, the installation specifies the suffix for VATLSTxx. The RIM checks field VAL in PARMTAB to determine whether a suffix is specified.		SYSPARMS
If no suffix is specified, the RIM uses default member VATLST00. The RIM reads the entries in the specified VATLSTxx member. If the RIM finds an entry for a mounted volume, the RIM sets the appropriate use and mount status bits in the associated UCB. If the volume is not mounted, the RIM issues a mount request to the operator unless the volume is a mass storage system (MSS) volume. For an MSS volume, the device must be online. If it is not, the RIM calls IOSVNPTH to verify a path to the device. If a path exists, the RIM issues a MSSC MOUNT macro to mount the volume. After the operator mounts the volume, the RIM sets the mount and use status in the associated UCB when the device is ready. If the operator cannot find a volume or if a device is not available, the operator replies to the RIM with a request to bypass attribute processing for the device. For a description of member VATLST00, see *Initialization and Tuning Guide*.		RDENTRY SETSTAT MOUNTLST VIRTMONT CKPATH

Diagram 68. Time-of-Day Clock Initialization (IEAVRTOD) Part 1 of 6

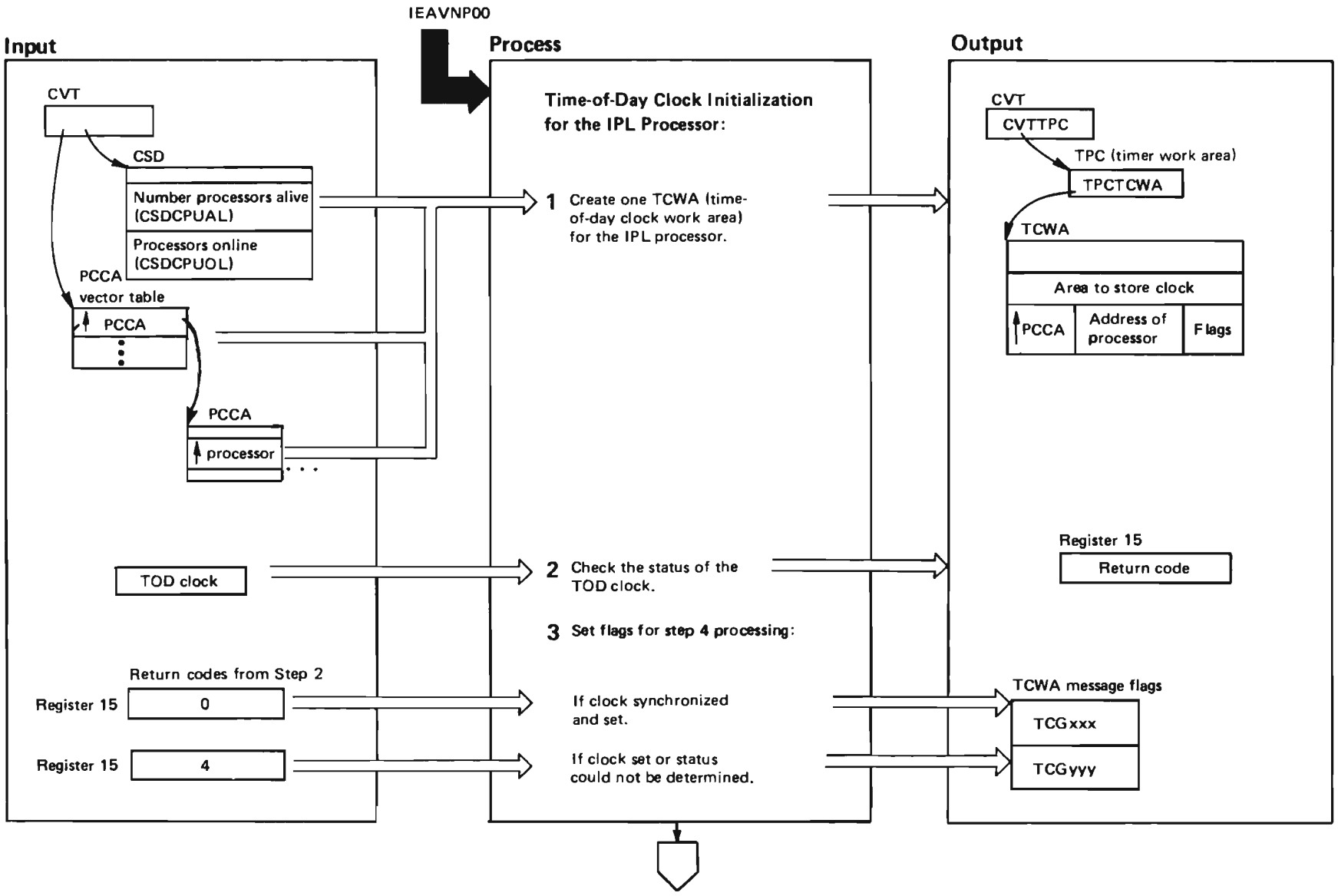


Diagram 68. Time-of-Day Clock Initialization (IEAVRTOD) Part 2 of 6

Extended Description	Module	Label
Time-of-day clock initialization, invoked by IEAVNP00, sets the TOD clock on the IPL processor to the correct GMT (Greenwich Mean Time) and initializes the local time and date in the CVT. The TOD clock, a 64-bit counter, holds the time elapsed at Greenwich since 0000 hours, January 1, 1900. Bit position 51 of the TOD clock is incremented every microsecond. The time zone constant in the CVT contains the difference between the time of day at Greenwich and local time.	IEAVRTOD	IEAVRINT
Note: Detailed descriptions of routines called by IEAVRTOD appear in *System Logic Library*.		
IEAVRINT does not require operator input if the clock referenced by the IPL processor is set. To ensure the operator does not change a clock known to be correct, an installation can specify in SYS1.PARMLIB (using the OPERATOR statement in a CLOCKxx member) that during initialization the operator not be allowed to enter a reply when the clock is set (OPERATOR=NOPROMPT). However, the operator will always be prompted in any of the following cases: • The TOD clock is not set. • The installation allows operator intervention. When prompted, the operator can change the value of the TOD clock as well as the local time and/or date.		
1 IEAVRTOD builds and initializes the TOD clock workarea (TCWA) for the IPL processor. This workarea will be used to communicate between various routines called during TOD clock initialization.	IEAVRTOD	IEAVRINT
2 IEAVRTOD calls subroutine IEAVRTST to check the status of the TOD clock. The return code in register 15 from IEAVRTST indicates the status of the TOD clock. 0 The clock is set. 4 The clock is not set.	IEAVRTOD	IEAVRTST
3 IEAVRTOD checks the return code to initialize the message indicator flags in the TCWA. If the return code is 0, IEAVRTOD sets the flag for message IEA888A. If the return code is 4, it sets the flag for message IEA886A.	IEAVRTOD	IEAVRINT

Diagram 68. Time-of-Day Clock Initialization (IEAVRTOD) Part 3 of 6

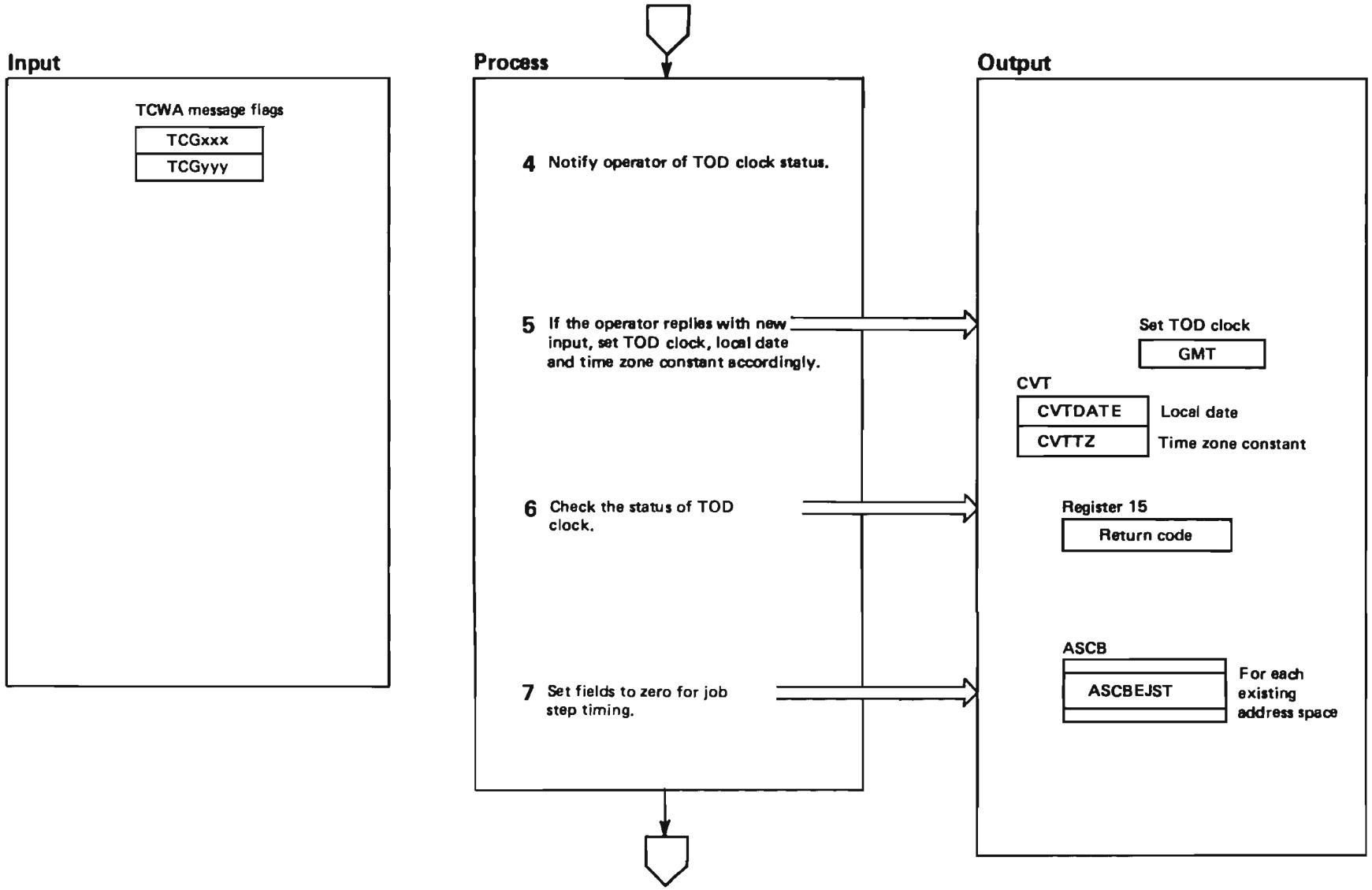


Diagram 68. Time-of-Day Clock Initialization (IEAVRTOD) Part 4 of 6

Extended Description	Module	Label
4 IEAVRCOM issues one of two messages: • Message IEA888A displays the Greenwich Mean Time and local times and dates and allows the operator to change the time or date on the IPL processor. • Message IEA886A requests the time and date from the operator.	IEAVRTOD	IEAVRCOM
5 After the operator responds to the message, IEAVRCOM issues an internal SET command. The SET command processor sets a TOD clock on the IPL processor if the operator's reply requires it. Message IEA903A requests that the operator enable the clock security switch at the exact moment the entered time occurs. If the operator's reply does not require the TOD clock to be set, SET command processing changes the local date and/or time zone constant. For a detailed description of SET command processing, refer to *System Logic Library.*	IEE0603D	
IEAVRCOM then issues message IEA888A for approval of the current values or for new operator input (a new GMT or new local time and/or date). This process of setting the clock and issuing the message repeats until the operator finally approves message IEA888A by not entering a change.	IEAVRTOD	IEAVRCOM
6 IEAVRTST again tests the clock to ensure successful operation so far. The return code from IEAVRTST is eventually passed back to IEAVNP00. If the clock was set successfully, IEAVRTOD determines if a sync check is outstanding. If so, IEAVRTOD sets the return code to 24 and resets TPCSYNC to zero.	IEAVRTOD	IEAVRTST
7 IEAVRTOD updates the elapsed job step timing field in the ASCB. A list of pointers to the ASCBs for all existing address spaces is contained in the ASVT pointed to by CVTASVT.	IEAVRTOD	IEAVRINT

Diagram 68. Time-of-Day Clock Initialization (IEAVRTOD) Part 5 of 6

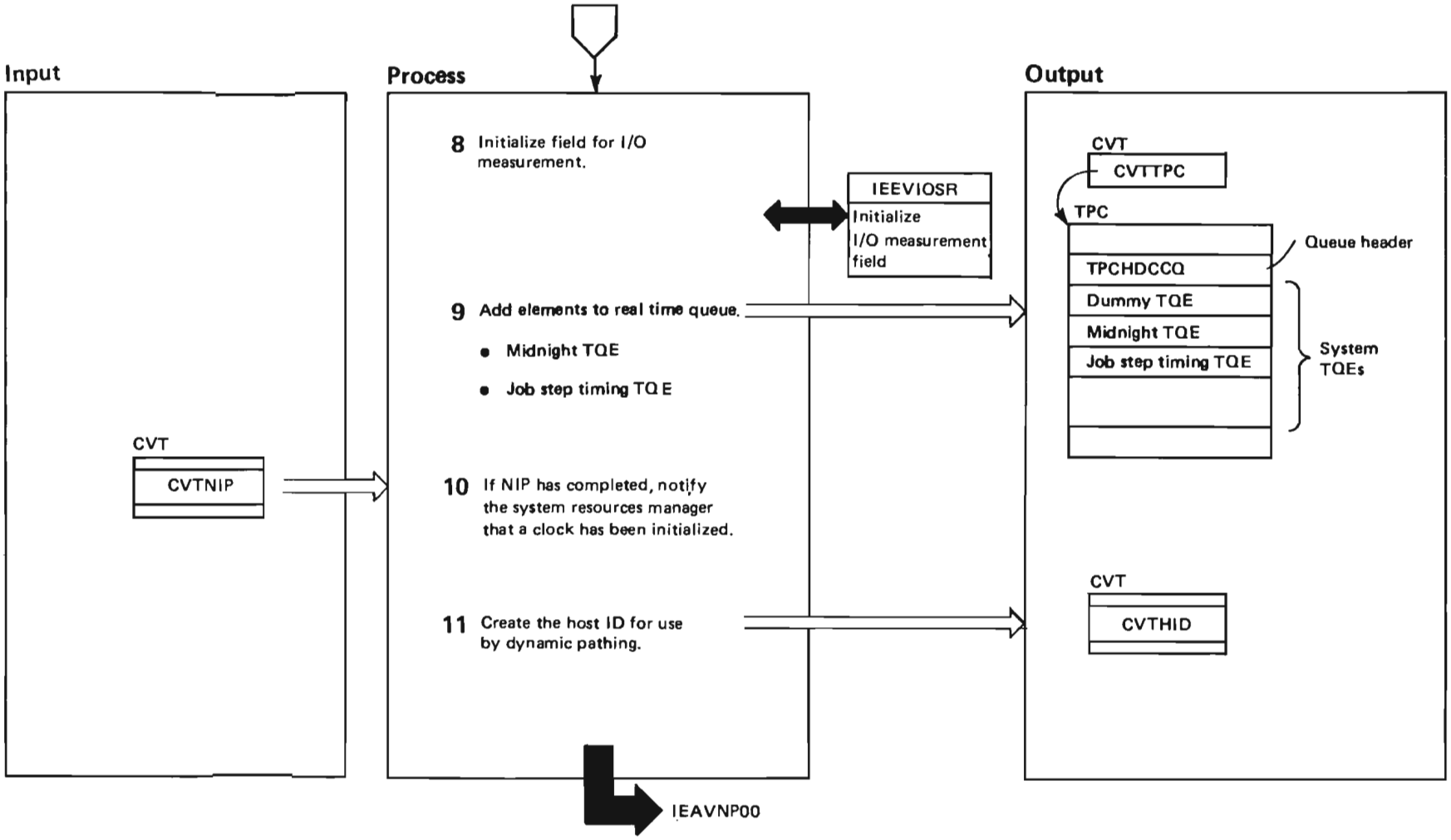


Diagram 68. Time-of-Day Clock Initialization (IEAVRTOD) Part 6 of 6

Extended Description	Module	Label
8 IEAVRTOD invokes IEEVIOSR to initialize fields for I/O measurement. These fields indicate whether SRM is receiving data based on an incorrect clock.	IEEVIOSR	
9 The real time queue contains TQEs (timer queue elements) that expire at a particular time of day. The order in which the elements expire determines the order of the elements in the queue. The first element of the queue is the one that expires next. Certain TQEs created for system functions are placed directly in the TPC (timer work area)—for example, the midnight TQE which expires every midnight and the job step timing TQE which expires every 10 seconds. Other TQEs created for problem programs are chained in the real time queue but are not located in the TPC itself.	IEAVRTOD	TQEINIT
10 If CVTNIP=0, indicating NIP has completed, IEAVRTOD issues a SYSEVENT macro instruction when a clock has been initialized. IEAVNP00 issues an initial SYSEVENT for TIMER when all processors are online and the TOD clock(s) are initialized.	IEAVRTOD	TQEINIT
11 IEAVRTOD initializes the host ID in the CVT. The host ID contains the current CPU address, CPU serial number, model number, and left half of TOD clock.		

Diagram 69. Reconfiguration Processor Initialization (IEAVNP00) Part 1 of 8

IEAVNP00 - MODULE DESCRIPTION

DESCRIPTIVE NAME: Reconfiguration Processor Initialization

FUNCTION:
Initialize the IPL processor's clock and console
RESTART text, and bring online and initialize the
non-IPL processors in the hardware configuration.

ENTRY POINT: IEAVNP00

PURPOSE: See FUNCTION.

LINKAGE: BALR R14,R15

CALLERS: IEAVNIPM

INPUT: SCP INFO data from the service processor.

OUTPUT: All processors online and ready for work.

EXIT NORMAL: Return to IEAVNIPM via BR 14

EXTERNAL REFERENCES:

ROUTINES:
IEAVRTOD - Timer Initialization Routine
IEEVCPR - CPU Reconfiguration Routine

DATA AREAS: SCP INFO pointed to by CVTSCPIN field

CONTROL BLOCKS:

Name	Macro	Description
CPRPARMS	IEEMCPR	IEEVCPR Parameter List
CSD	IHACSD	Common System Data Area
CVT	CVT	Communications Vector Table
MSFDBLK	IHAMSF	Service processor data block
MSFCMDWD	IHAMSF	Service processor command word
MSFSVCPL	IHAMSF	SVC 122 parameter list
NVT	IHANVT	IPL/NIP Vector Table
PCCA	IHAPCCA	Processor Control Communications Area
PSA	IHAPSA	Prefixed Save Area
SCCB	IHASCCB	Service Call Communications Block
SCPINFO	IHAMSF	SCP INFO data mapping for MSSF
SPCS	IEEMSPCS	Service processor Call SVC Interface
SVT	IHASVT	Supervisor Vector Table

SERIALIZATION: IEAVNP00 ENQ's on the SYSZVARY,CPU resource exclusively.

Diagram 69. Reconfiguration Processor Initialization (IEAVNP00) Part 2 of 8

IEAVNP00 - MODULE OPERATION

1. Call IEAVRTOD to initialize the IPL processor's TOD clock.
2. If processing is not under VM, invoke the service processor to write the console restart text to the console screen of this CPU.
3. Set up VF parameters in the CSD.
4. If the service processor architecture is supported, and the TOD clocks are out of sync, take all other CPUs physically offline.
5. Call IEEVCPR to bring each of the non-IPL processors in the system online.
6. Issue a SYSEVENT to inform SRM that clocks are now synchronized.
7. Return to the caller.

RECOVERY OPERATION:
There is no recovery for this module because this is a NIP RIM.

Diagram 69. Reconfiguration Processor Initialization (IEAVNP00) Part 3 of 8

IEAVNP00 - DIAGNOSTIC AIDS

ENTRY POINT NAME: IEAVNP00

MESSAGES:

```
IEA846I SYSTEM CONSOLE INTERFACE UNSUCCESSFUL.
 RESTART OPTIONS MAY NOT BE INITIALIZED ON CPU x (yyyy)
   The service processor supports the WRITE RESTART TEXT
   command but was unable to write the text to the console.

IEA953I UNABLE TO INITIALIZE CPU x, reason text
   Module IEEVCPR could not bring the target CPU online with
   one of the following reason texts:
   ESTAE COULD NOT BE SET UP
   CALL TO SERVICE PROCESSOR FAILED
   INVALID CPU ID
   TARGET CPU IS IN ANOTHER PARTITION
   CPU IS IN AN S/M/R CONFIGURATION
   SERVICE PROCESSOR FAILED
   UNABLE TO OBTAIN WORK AREA STORAGE
   MAX NUMBER OF CPUS ALREADY ONLINE
   INITIAL CPU RESET FAILED
   SET PREFIX SIGP FAILED
   RESTART SIGP FAILED
   WAKEUP ROUTINE DID NOT GET CONTROL
   WAKEUP ROUTINE DID NOT COMPLETE
   CLOCKS COULD NOT BE SYNCHRONIZED
```

ABEND CODES: None

WAIT STATE CODES: None

RETURN CODES: None

REGISTER CONTENTS ON ENTRY:

```
Registers 0-1  - Irrelevant
Register   2   - Address of the NVT
Register   3   - Address of the CVT
Registers 4-12 - Irrelevant
Register  13   - Address of standard register save area
Register  14   - Return address to IEAVNIPM
Register  15   - Entry address
```

REGISTER CONTENTS ON EXIT:

```
EXIT NORMAL:

  Registers 0-15 - Unchanged
```

Diagram 69. Reconfiguration Processor Initialization (IEAVNP00) Part 4 of 8

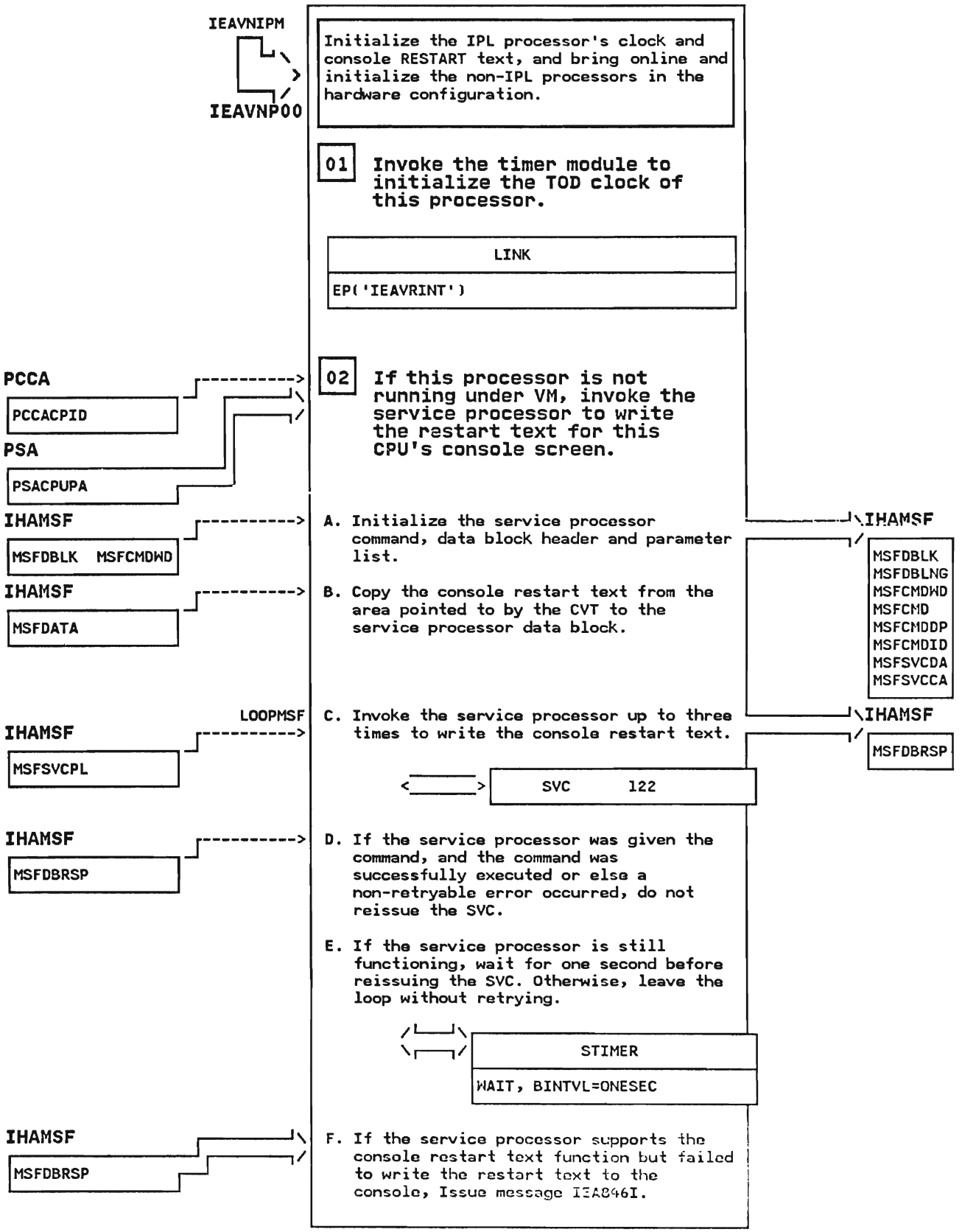


Diagram 69. Reconfiguration Processor Initialization (IEAVNP00) Part 5 of 8

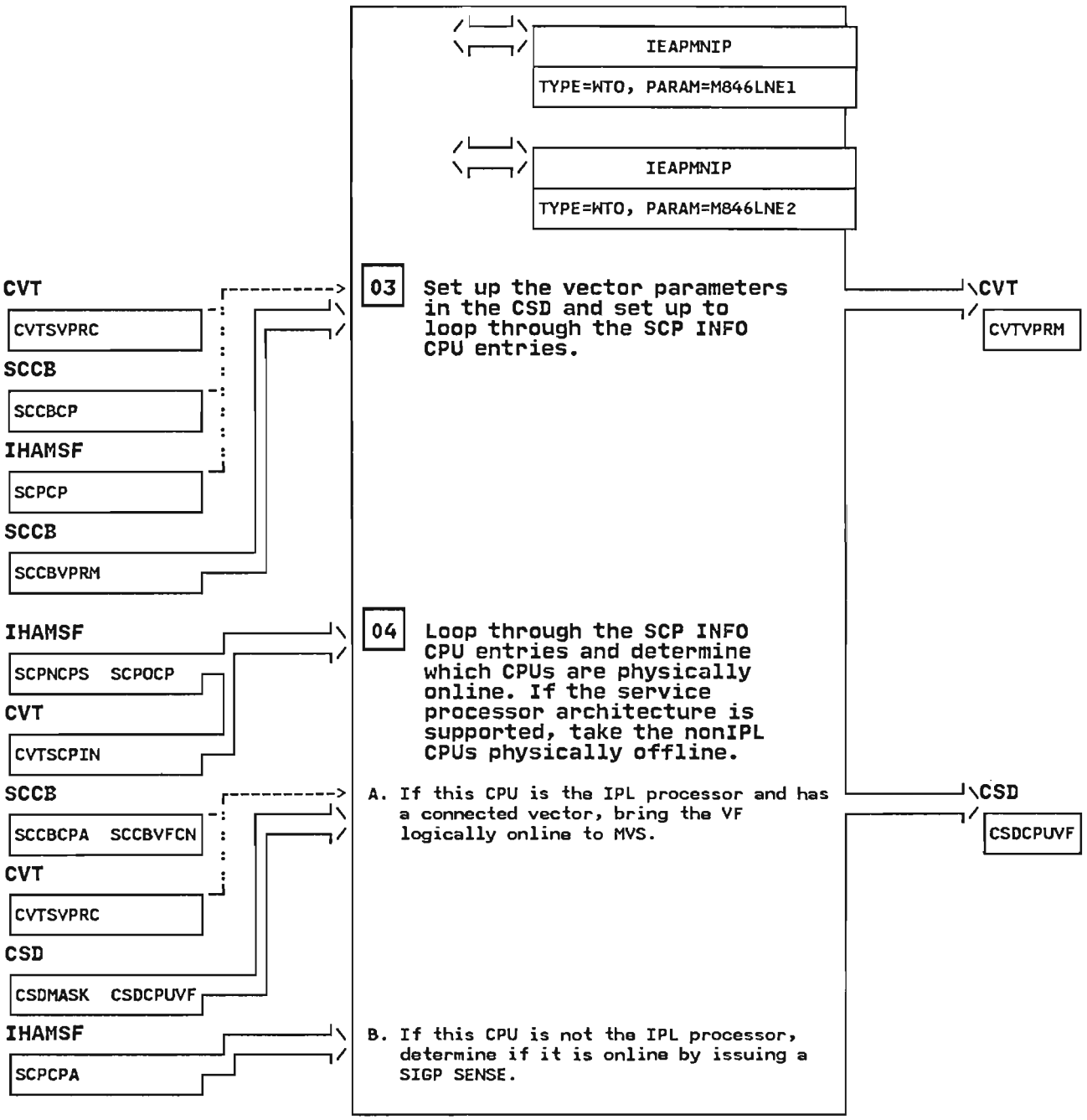


Diagram 69. Reconfiguration Processor Initialization (IEAVNP00) Part 6 of 8

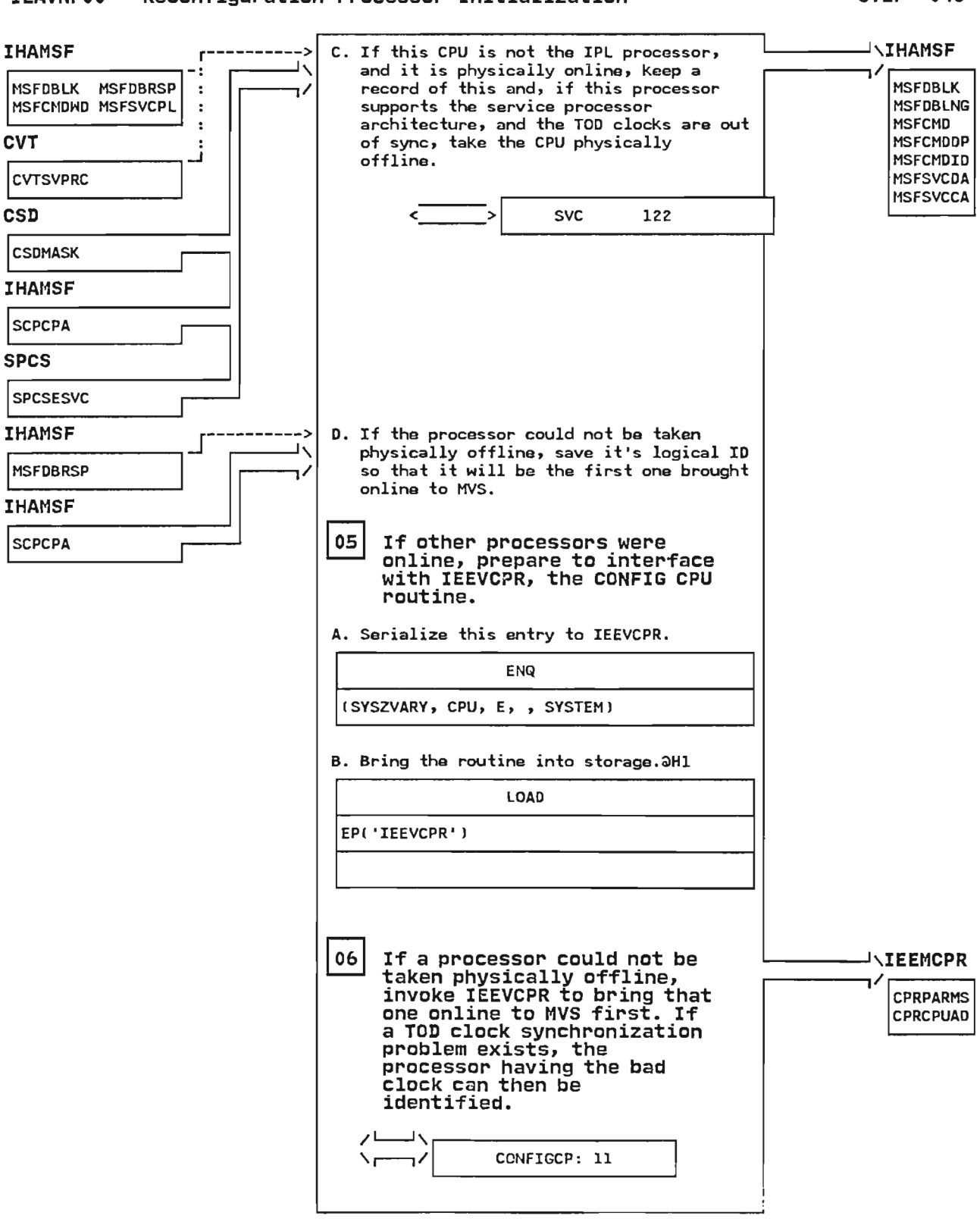


Diagram 69. Reconfiguration Processor Initialization (IEAVNP00) Part 7 of 8

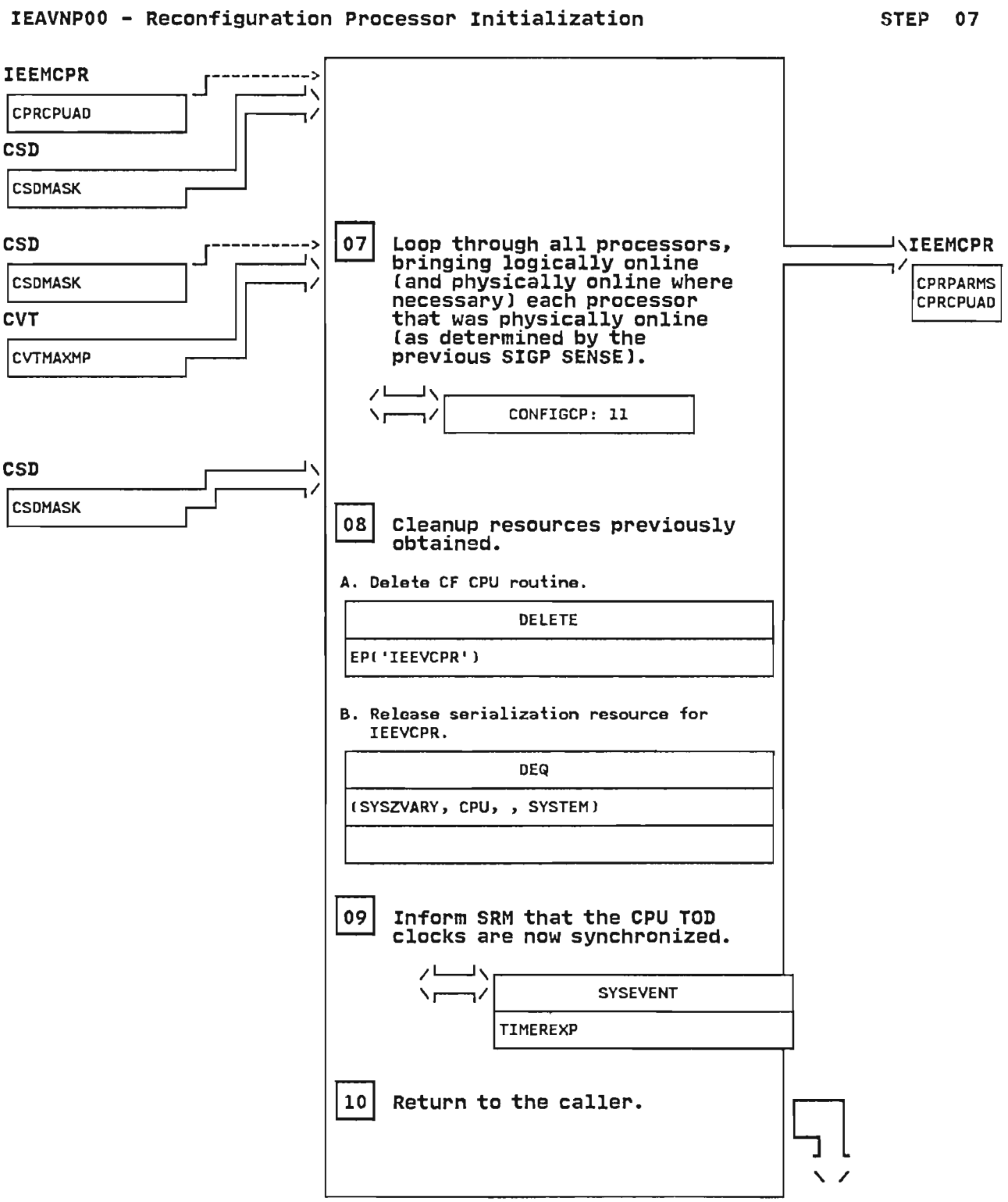


Diagram 69. Reconfiguration Processor Initialization (IEAVNP00) Part 8 of 8

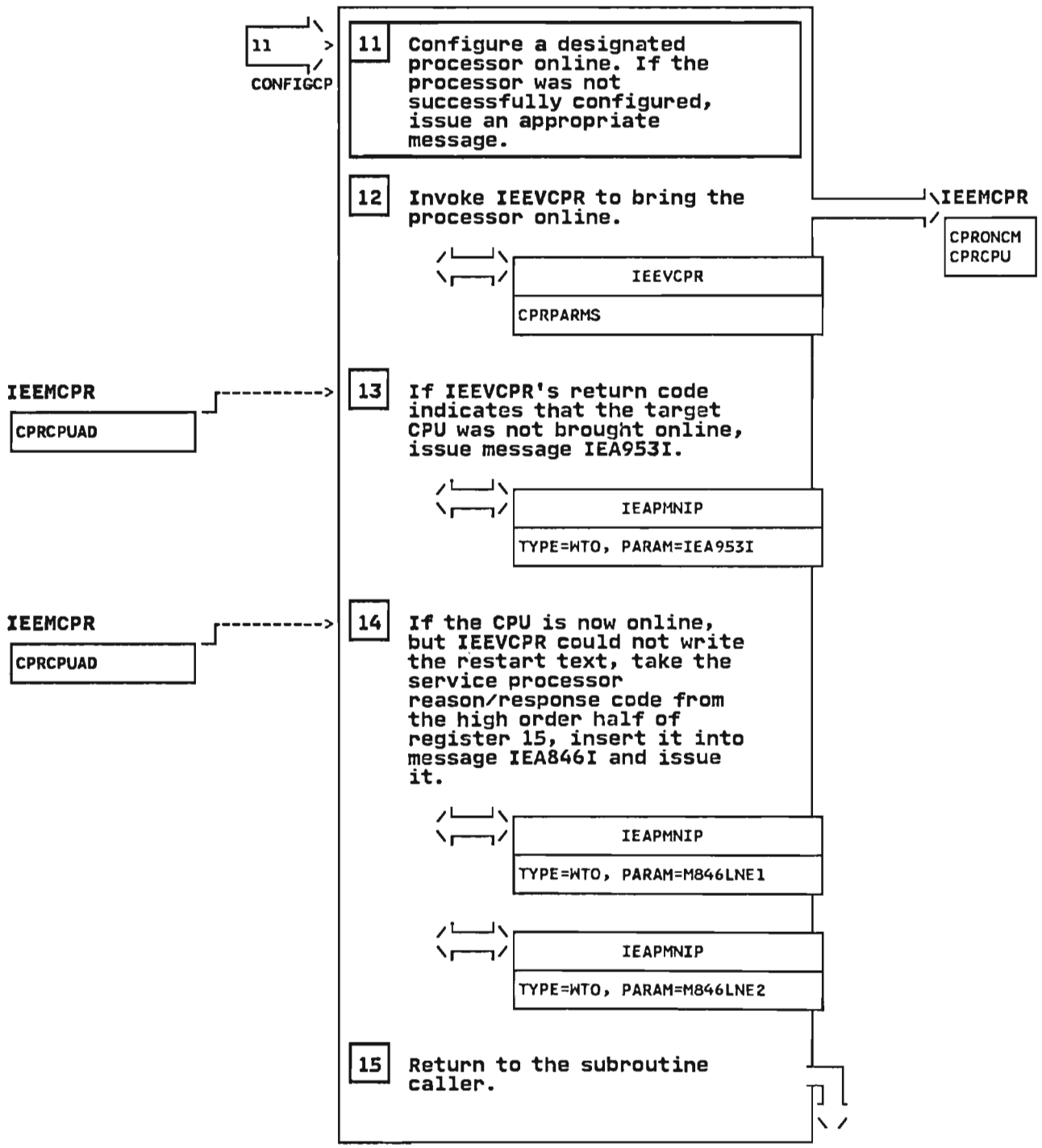


Diagram 70. NIP Exit Processing (IEAVNIPX) Part 1 of 4

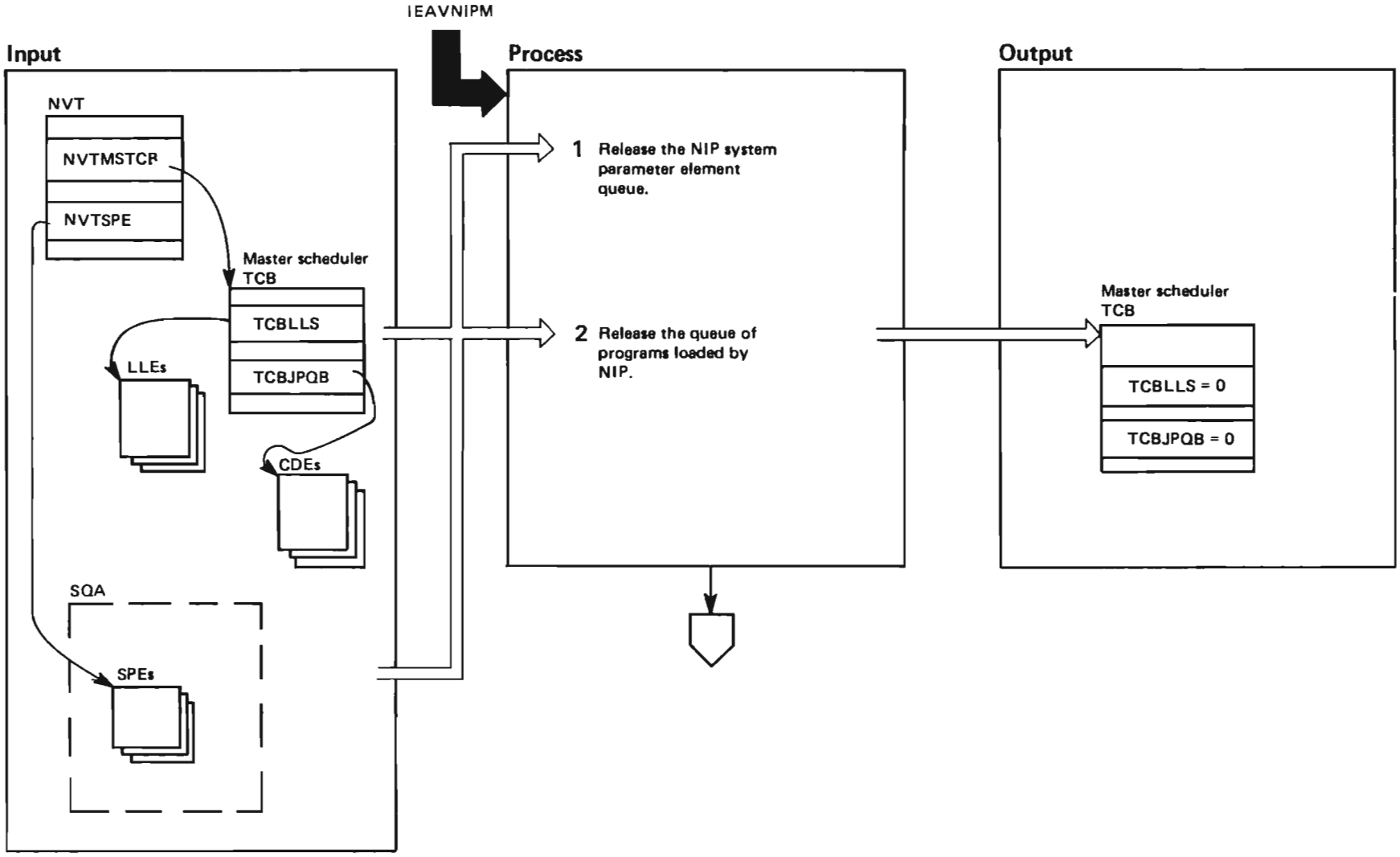


Diagram 70. NIP Exit Processing (IEAVNIPX) Part 2 of 4

Extended Description	Module	Label
The NIP RIM, IEAVNIPX, performs final exit and clean up processing before passing control to master scheduler initialization. NIP removes itself from the system, including its control blocks, buffers, and error traps.	IEAVNIPX	NPXFBUF
1 NIP obtains the origin of the NIP system parameter element (SPE) queue from field NVTSPE. NIP frees each SPE on the queue except the SPE contained in the NVT. NIP frees the SYS1.NUCLEUS DCB and DEB that IEAVNIP0 obtained from the non-extended SQA (subpool 245). NIP also generates the IEAPMNIP interface to invoke NIPWTOR2 to free the master console image buffer. For a map of virtual storage an exit from IEAVNIPX, see Figure 1-6.	IEAVNPM2	NIPWTOR2
2 After acquiring the local locks, NIP dequeues the LLEs (load list elements) and CDEs (contents directory entries) representing programs that NIP has loaded that remain on its TCB job pack queue (TCBJPQB). NIP then releases the storage that the LLEs and CDEs occupy.	IEAVNIPX	NPXFJPQ
When dequeueing LLEs, NIP first dequeues each LLE from the TCB under which NIP executes (the master scheduler TCB), starting with the LLE pointed to by field TCBLLS. Before it releases the storage occupied by each LLE, NIP updates the TCBLLS pointer to point to the next LLE. Processing is complete when the TCBLLS pointer is zero.		
NIP then dequeues the CDEs from the master scheduler TCB. NIP dequeues each CDE pointed to by the TCBJPQB field. Before dequeueing each CDE, NIP tests the extent list created flag in the CDE to determine whether an extent list is associated with the CDE. If an extent list exists, NIP also releases the storage containing the extent list. If no extent list exists, NIP continues with the next CDE. NIP updates the TCBJPQB pointer with the address of the next CDE, dequeues the CDE, and releases the storage containing the dequeued CDE. Processing is complete when the TCBJPQB pointer is zero.		
NIP then releases the local locks.		

Diagram 70. NIP Exit Processing (IEAVNIPX) Part 3 of 4

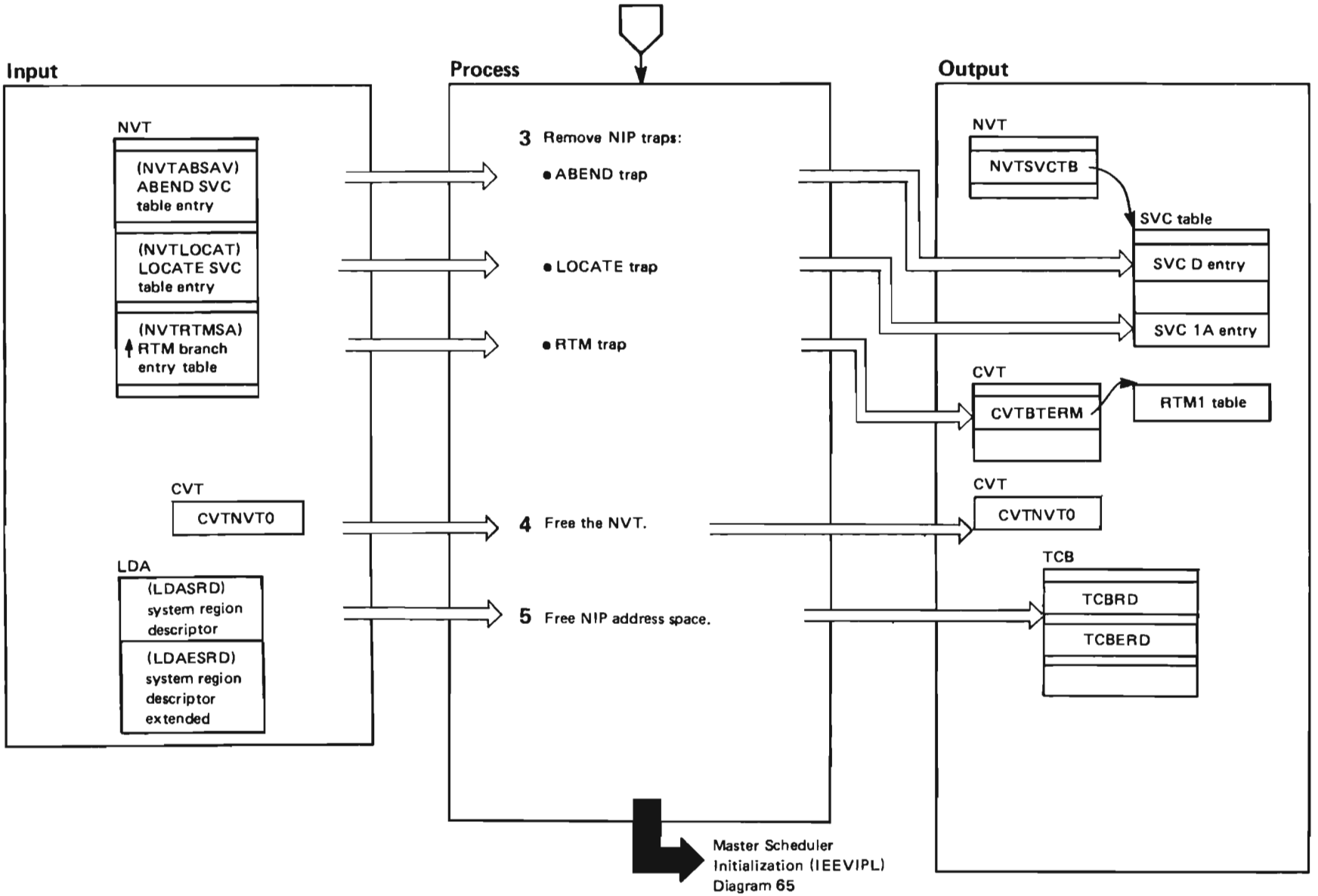


Diagram 70. NIP Exit Processing (IEAVNIPX) Part 4 of 4

Extended Description	Module	Label
3 NIP removes traps that were set by NIP in IEAVNIPM.	IEAVNIPX	NPXRTRAP

- NIP removes the ABEND SVC trap by moving the ABEND SVC table entry from the NVT save area (NVTABSAV) into the SVC 13 entry in the SVC table.
- NIP removes the LOCATE SVC trap by moving the LOCATE SVC table entry from the NVT save area (NVTLOCAT) into the SVC 26 entry in the SVC table.
- NIP removes the recovery/termination management trap by moving the address of the RTM1 table from the NVT save area (NVTRTMSA) into field CVTBTERM in the CVT.

NIP frees the global storage (SQA) that was used for the RTM traps.

4 NIP releases the storage in the non-extended SQA that contained the NIP vector table (NVT) and resets the pointer (CVTNVTO=0).

5 NIP moves the remaining IEAVNIPX code to the non-extended SQA and continues to execute. NIP uses the VSM routine, IGVGVRGN, to free the region in which it has been processing. It then sets fields in the TCBs to point to system regions. It loops through each TCB in storage, placing the contents of LDASRD into TCBRD, and LDAESRD into TCBERD.

Finally, NIP passes control (via the LINK macro instruction) to master scheduler initialization (IEEVIPL).

Diagram 71. System Address Space Create Routine (IEEMB881) Part 1 of 10

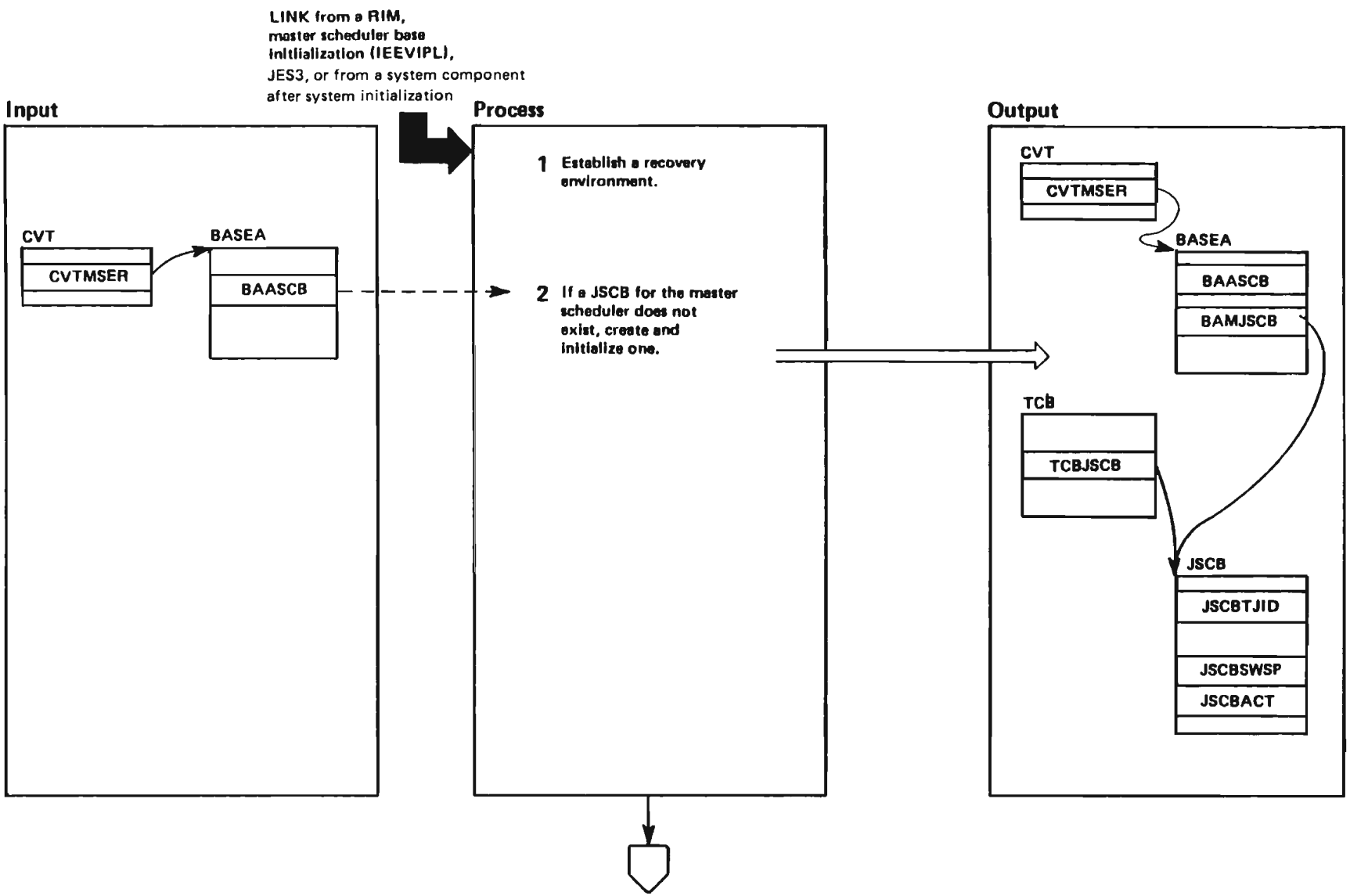


Diagram 71. System Address Space Create Routine (IEEMB881) Part 2 of 10

Extended Description	Module	Label

The system component address space routine (IEEMB881) can be invoked from a RIM, the master base initialization routine (IEEVIPL), or any module in supervisor mode to create an address space for a system component. This processing can occur during master scheduler initialization, during JES initialization, or after system initialization is complete. (See Section 1, "System Component Address Space Initialization" for additional information on system address spaces.)

The input parameters consist of either zero (meaning the parameter is not supplied) or pointers (USRATTR, USRPGMN, and USRPARMS) to:

- A list of system address space attributes
- The name of the address space initialization routine
- The START parameters

USRATTR and USRPGMN are optional parameters, but USRPARMS must be supplied. If USRATTR is not supplied, IEEMB881 provides default values for the address space attributes. If USRPGMN is not supplied, IEEMB881 can create an address space but the address space will not be initialized in any special manner. If USRPARMS is not supplied, IEEMB881 does not create the address space and returns to the caller with X'04' in both registers 15 and 0 to indicate the error. (See step 10 for a description of the return and reason codes.)

The address space attributes supply the attributes normally obtained from the program properties table (PPT) and the JCL. EATTRSMF corresponds to the PPT preferred storage usage flag; EATTRTYP is a description of the address space type; and EATTRACT contains swapping information.

The initialization routines provided by the components that use IEEMB881 are listed in Section 1 under "System Component Address Space Initialization."

Extended Description	Module	Label

The START parameters consist of a halfword length field followed by a variable length character string. This character string is used as the procedure name by the DISPLAY A command when displaying the new address space. (The procedure name must consist of 1 through 7 alphameric or national characters, the first of which is alphabetic or national.)

1 IEEMB881 establishes an ESTAE environment using the ESTAE recovery routine EAESTAE, which is an entry point in IEEMB881. This ESTAE routine, which is operative after NIP, receives control if an ABEND occurs during system component address space create processing. See recovery processing for details. — IEEMB881

2 IEEMB881 checks the BAASCB field of the master scheduler's resident data area (BASEA). If this field is zero, the master scheduler's JSCB does not exist and IEEMB881 issues a GETMAIN to obtain storage for it from subpool 237; places the address of the master scheduler's ASCB into BAASCB; chains the JSCB to the master scheduler's TCB; and initializes the JSCB as follows: — IEEMB881

- Sets JSCBSWSP to indicate that storage for the JSCB came from the SWA
- Places the address of the JSCB in JSCBACT to indicate that the JSCB is active
- Places the master scheduler's ASID in JSCBTJID to indicate that this JSCB was created for the master scheduler
- Sets all other fields of the JSCB to zero.

Diagram 71. System Address Space Create Routine (IEEMB881) Part 3 of 10

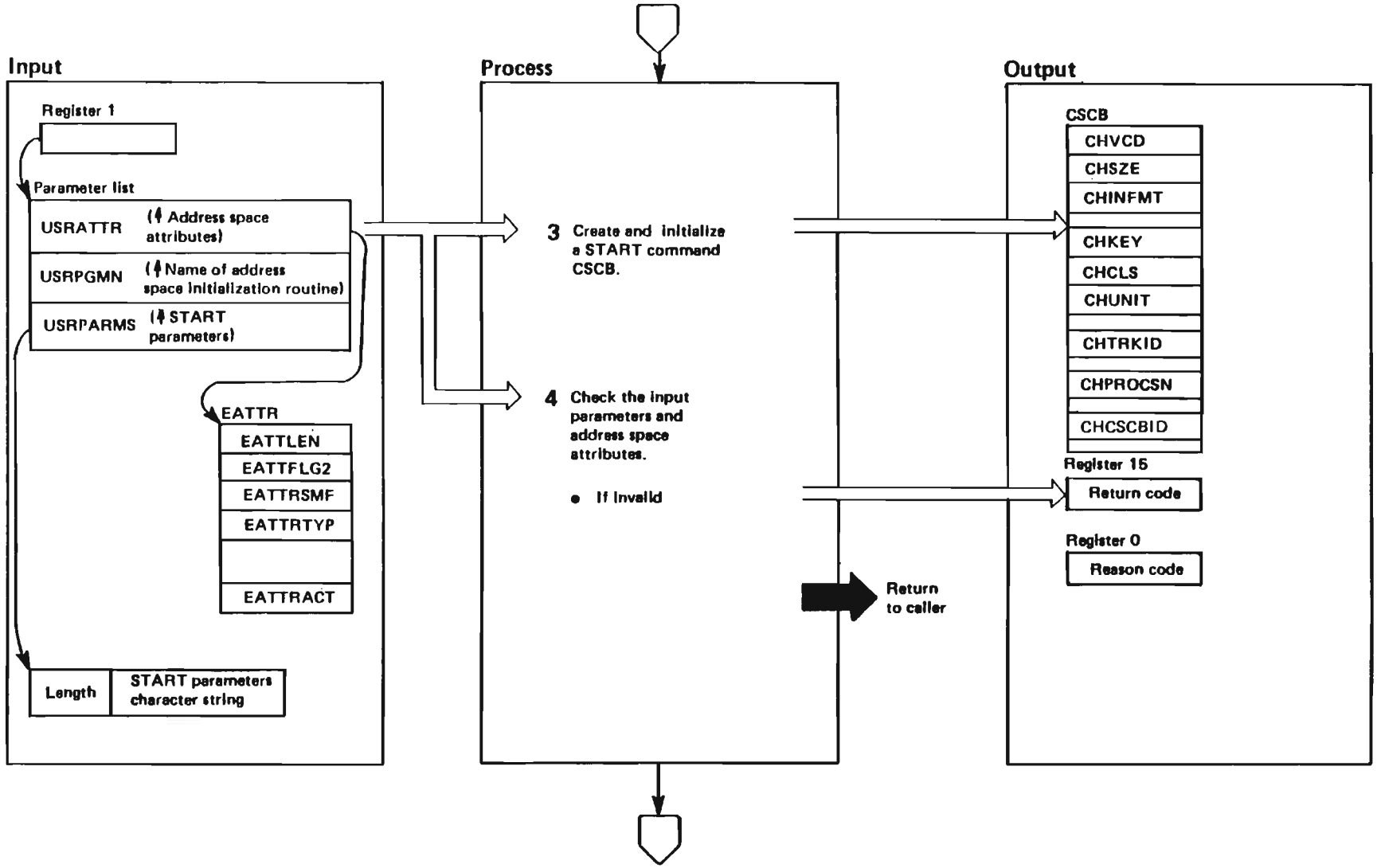


Diagram 71. System Address Space Create Routine (IEEMB881) Part 4 of 10

Extended Description	Module	Label
3 IEEMB881 issues a GETMAIN to obtain storage for a CSCB from subpool 245 (SQA) and then it initializes the CSCB as follows: • Sets the CSCB to zero • Stores X'04' in the verb code (CHVCD) to indicate a START command is being processed • Stores the length of the CSCB in CHSZE • If supplied, places the name of the address space initialization routine in CHKEY • Stores the START parameter character string in the procedure name that can be used by the DISPLAY A command (CHCLS) for a limited function address space • If the procedure name is IEESYSAS, place the step-name (the name of the address space) in CHCLS. This permits the address space to be treated as if it had its own procedure. • Stores blanks in the start unit address field (CHUNIT) and the procedure step name field (CHPROCSN) • Stores X'04' in the DISPLAY/TRACK identifier (CHTRKID) to indicate that this is a system component address space • Sets the CHINFMT bit to one to indicate that the CSCB is in input format • Places the acronym 'CSCB' in the CHCSCBID field	IEEMB881	
4 IEEMB881 checks to determine if the address space attributes (EATTR) and the START parameters are valid. If an invalid condition exists, IEEMB881 sets a return code in register 15, a reason code in register 0, and returns to the caller. See step 10 for a list of these codes.	IEEMB881	

Diagram 71. System Address Space Create Routine (IEEMB881) Part 5 of 10

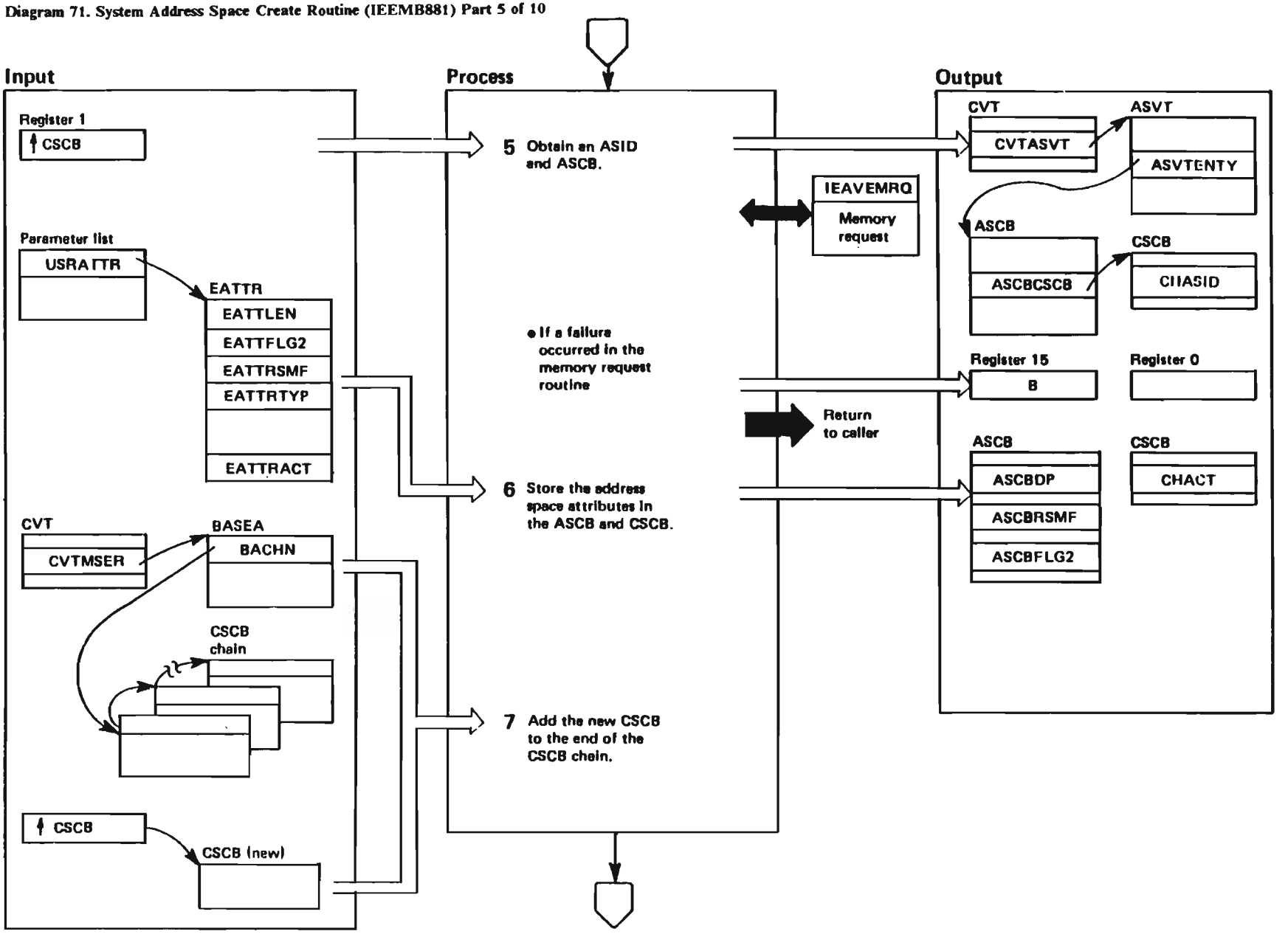


Diagram 71. System Address Space Create Routine (IEEMB881) Part 6 of 10

Extended Description	Module	Label
5 Before obtaining an ASID and ASCB for the new address space, IEEMB881 enqueues exclusively on the CSCB chain using SYSIEFSD and Q10 as the major and minor queue names. This serialization of the CSCB chain prevents the master scheduler WAIT routine (IEEVWAIT) from updating the ASCB before the address space is created. IEEMB881 provides the address of the CSCB and calls memory request (IEAVEMRQ) to obtain the ASID and ASCB. IEAVEMRQ stores a pointer to the ASCB in the ASVT (ASVTENTY), a pointer to the CSCB in the ASCB (ASCBCSCB), and the ASID in the CSCB (CHASID). If IEAVEMRQ could not obtain the ASID and ASCB, IEEMB881 sets the return code to 8, the reason code to 4, 8, or 12, frees the storage for the new CSCB, and returns to the caller. (See step 10 for an explanation of these codes.)	IEEMB881 IEAVEMRQ IEEMB881	
6 IEEMB881 updates the ASCB with the caller-supplied address space attributes (USRATTR) or the default attributes. The attributes that IEEMB881 stores in the ASCB include: • Address space function attributes (ASCBFLG2=EATTFLG2) • RSM storage allocation attributes (ASCBRSMF=EATTRSMF) • Address space dispatching priority (SRM sets ASCBDP using SYSEVENT EASINIT with EATTRTYP as the caller's address space type.) This module also stores the caller-supplied job scheduler attributes in the CSCB (CHACT=EATTRACT).	IEEMB881	
7 This module adds the newly-created CSCB to the end of the CSCB chain enchored from BASEA.	IEEMB881	

Diagram 71. System Address Space Create Routine (IEEMB881) Part 7 of 10

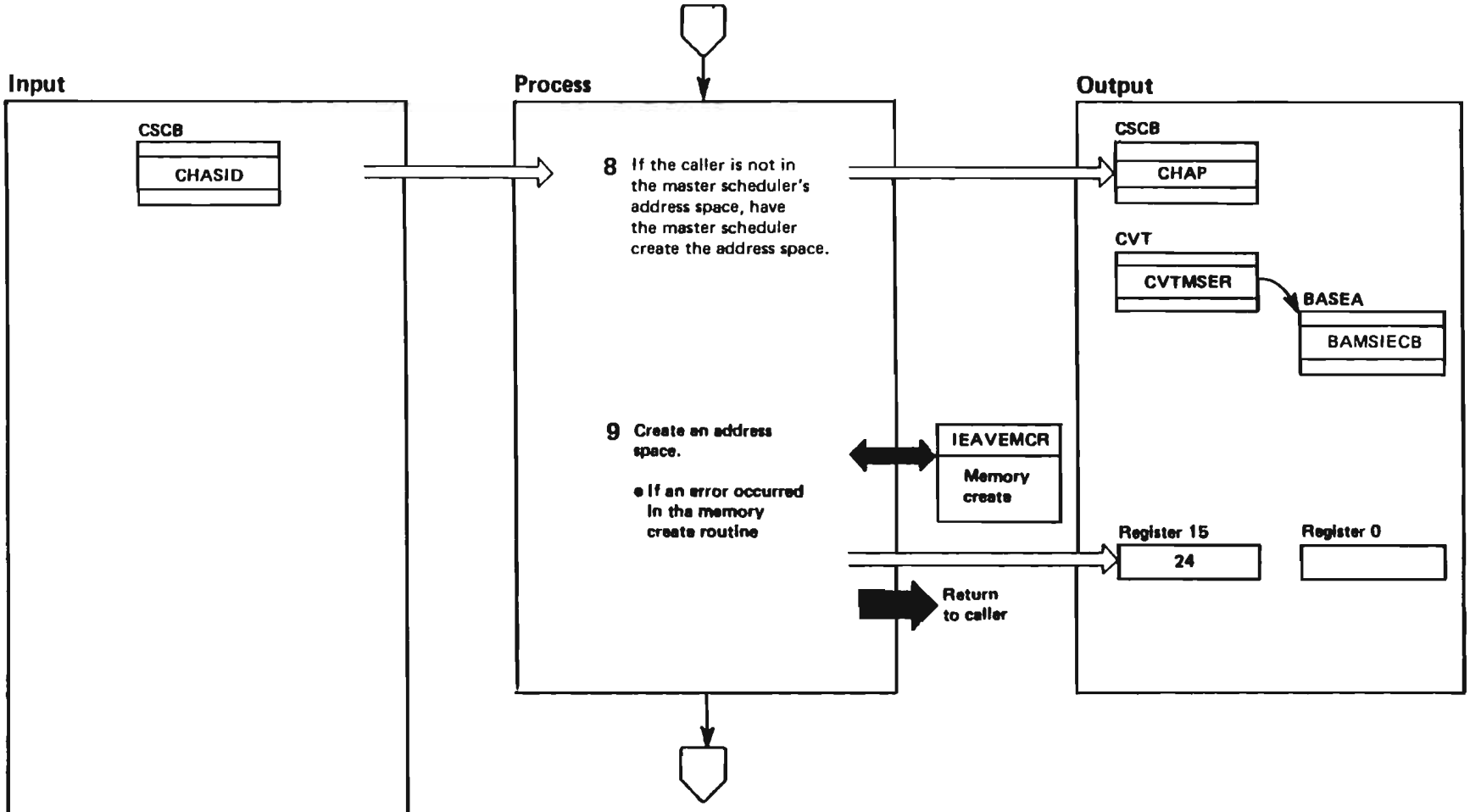


Diagram 71. System Address Space Create Routine (IEEMB881) Part 8 of 10

Extended Description	Module	Label
8 A user must be in the master scheduler's address space (ASID=1) to create an address space by invoking memory create (IEAVEMCR). If IEEMB881 is not running in the master scheduler's address space, it sets the CHAP field of the CSCB to request IEEVWAIT (which is running in the master scheduler's address space) to create an address space. IEEMB881 then posts the ECB BAMSIECB, which IEEVWAIT is waiting on. IEEVWAIT then attaches IEAVEMCR.	IEEMB881	
9 If IEEMB881 is running in the master scheduler's address space, it attaches IEAVEMCR to create the system component address space. If the address space is not created successfully (IEAVEMCR returned a nonzero completion code), IEEMB881 sets the reason code equal to the completion code and the return code equal to 24 and returns to the caller.	IEAVEMCR IEEMB881	

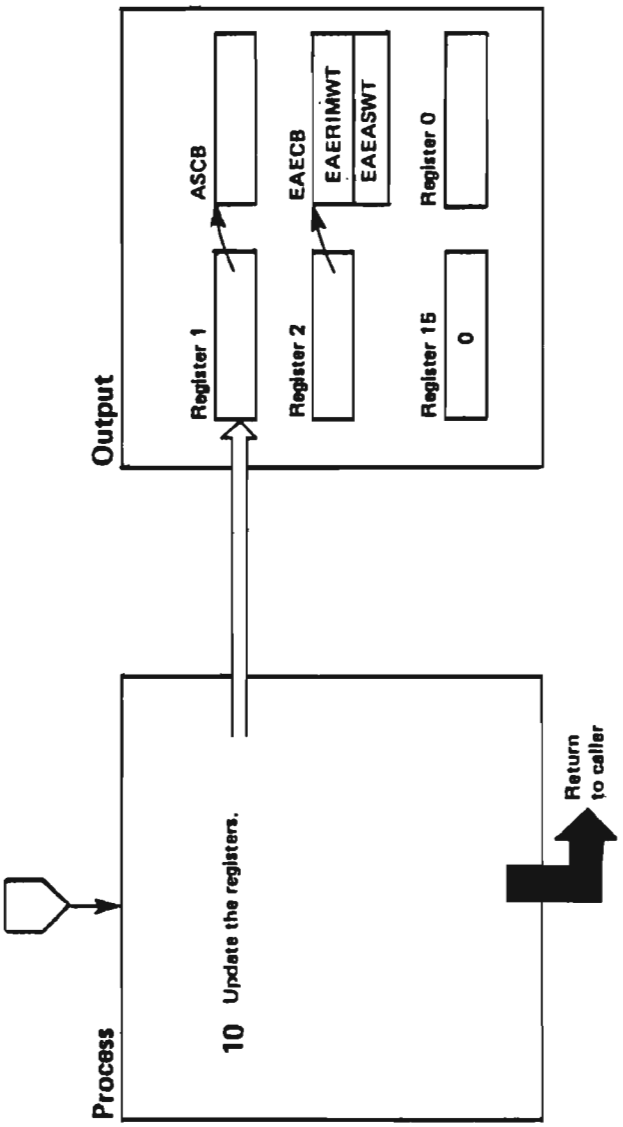


Diagram 71. System Address Space Create Routine (IEEMB881) Part 9 of 10

Diagram 71. System Address Space Create Routine (IEEMB881) Part 10 of 10

Extended Description	Module	Label

10 If the address space was successfully created, IEEMB881 stores the following values in the specified registers:

- Register 1: the address of the new ASCB
- Register 2: a pointer to the ECBs mapped from the CSCB (EAERIMWT=CHRIMWT and EASASWT=CHASWT)
- Register 15: X'00'
- Register 0: X'00' if IEEMB881 attached IEAVEMCR or X'04' if IEEVWAIT attached IEAVEMCR

A summary of the decimal return codes and the corresponding reason codes returned by IEEMB881 follows. If the return code is X'00', an address space was created and register 1 contains the address of the ASCB; if the return code is not X'00', an address space was not created.

Register 15 Return code	*Register 0 Reason Code*	*Explanation*
0	0	The address space was successfully started.
0	4	IEEVWAIT was invoked to create the address space.
4	4	The caller failed to specify USRPARMS.
4	8	An invalid DISPLAY A procedure name was supplied.
8	4	SRM would not allow memory request to create an address space.
8	8	Memory request indicated that the system was overloaded.
8	12	Memory request experienced an unexpected system failure.
12	4	Unsupported flag bits are on in the EATTFLG2 field.
12	8	Unsupported flag bits are on in the EATTRSMF field.
12	12	Unsupported flag bits are on in the EATTRACT field.
12	16	The user specified an invalid address space type.
16	0	An ABEND took place during the execution of IEEMB881.
24	Return code from IEAVEMCR	An error occurred in IEAVEMCR.
28	0	There was a failure during the ESTAE setup.

Recovery Processing:

If an ABEND occurs during address space create processing, subroutine EAESTAE receives control. If an SDWA exists, this subroutine performs the following functions: — Module: IEEMB881 — Label: EAESTAE

- Fills in the appropriate fields and issues the SETRP macro specifying retry and record as options
- Takes an SVC dump
- Performs a retry

If an SDWA does not exist, this subroutine:

- Sets the registers according to standard subroutine linkage conventions
- Takes an SVC dump
- Performs a retry

EARETRY, the retry routine, which receives control from RTM after EAESTAE executes, performs these functions: — Label: EARETRY

- Sets return and reason codes
- Frees the storage for the new CSCB — Label: CLEANUP
- Terminates the new address space (if it was created)
- Releases the serialization of the CSCB chain

Diagram 72. System Address Space Initialization WAIT/POST Routine (IEEMB883) Part 1 of 2

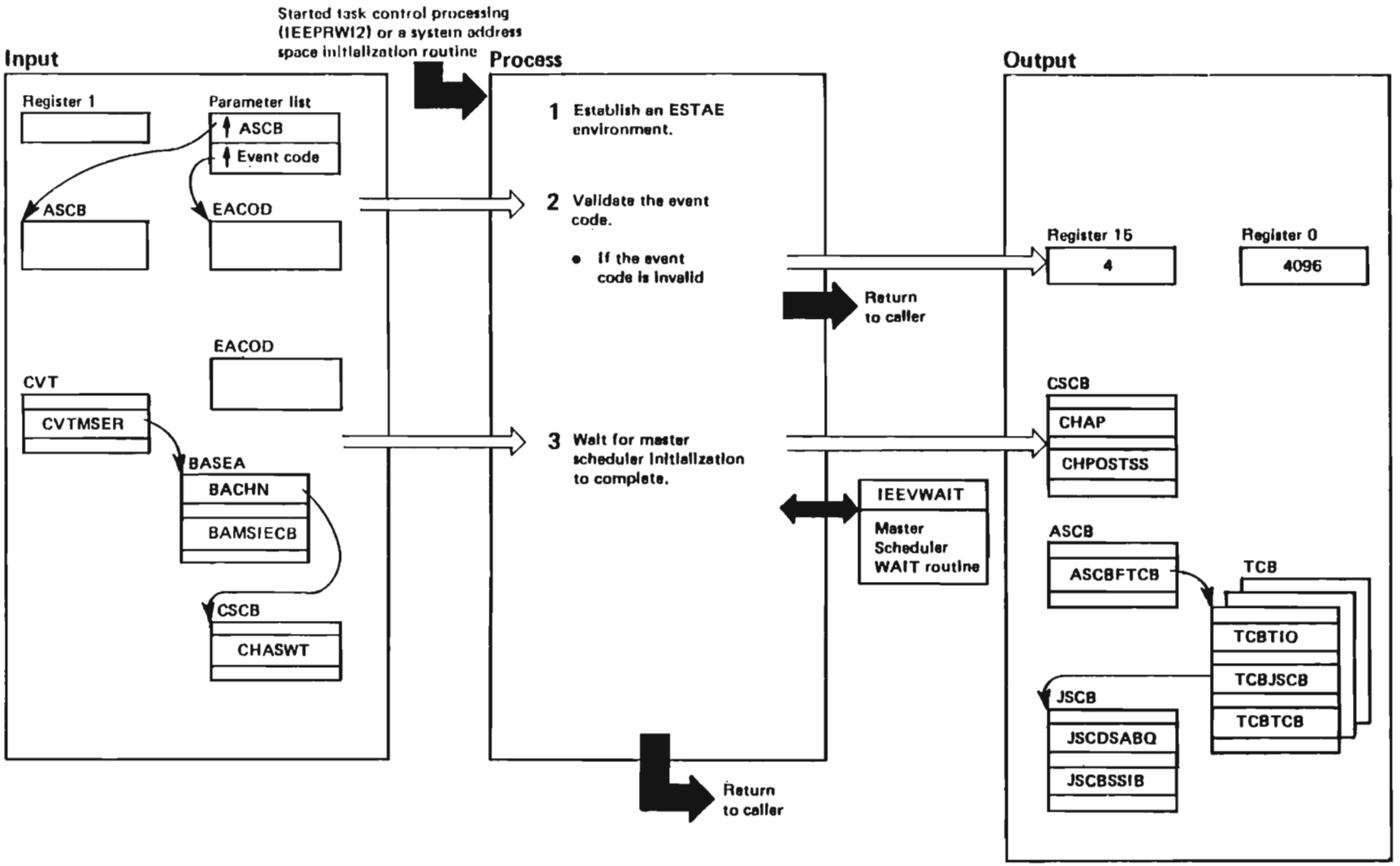


Diagram 72. System Address Space Initialization WAIT/POST Routine (IEEMB883) Part 2 of 2

Extended Description	Module	Label
Started task control (STC) processing (IEEPRWI2) or a system address space initialization routine calls the system address space initialization WAIT/POST routine (IEEMB883) to wait for master scheduler initialization to complete. The input to the routine consists of the ASCB of the new system address space and an event code (EACOD) of 1, which identifies the event (master scheduler initialization) being waited on.		
1 The system address space initialization WAIT/POST routine (IEEMB883) establishes an ESTAE environment with WPESTAE as the recovery routine and WPRETRY as the retry routine. See "Recovery Processing" for details.	IEEMB883	
2 IEEMB883 checks the event code. If it is invalid (not equal to 1), IEEMB883 sets a return code of X'04' and a reason code of X'1000' and returns to the caller.	IEEMB883	
3 If the event code (EACOD) is 1, IEEMB883 waits for master scheduler initialization to complete.	IEEMB883	
IEEMB883 does this by posting the ECB BAMSIECB to activate the master scheduler wait routine (IEEVWAIT) and setting a bit in the CSCB (CHPOSTSS) to inform IEEVWAIT that it should post the ECB CHASWT when master scheduler initialization is complete. IEEMB883 also turns on the action-pending-bit in the CSCB (CHAP) to indicate to IEEVWAIT that there is work to do. After	IEEVWAIT	
IEEVWAIT has posted CHASWT, IEEMB883 regains control and does the following:	IEEMB883	
• Initializes the job step control blocks (JSCBs) for the TCBs in the new system address space with information copied from the master scheduler's JSCB. • Sets the address of the task I/O table (TCBTIOT) in all of the TCBs in the new system address space to point to the master scheduler's TIOT. Dumps of this address space can now be taken if required.		

Extended Description	Module	Label
Recovery Processing:		
If an ABEND occurs during WAIT/POST processing in IEEMB883, RTM passes control to the recovery routine, WPESTAE. WPESTAE is operative only after NIP. If an SDWA exists, WPESTAE updates it and issues a SETRP specifying the record and retry options and takes an SVC DUMP. If there is no SDWA. WPESTAE sets registers and attempts a retry. The retry routine, WPRETRY, receives control from RTM after the recovery routine, WPESTAE, executes, WPRETRY sets the return and reason codes and returns to the caller of IEEMB883.	IEEMB883	WPESTAE WPRETRY

Diagram 73. Model System Address Space Initialization Part 1 of 2

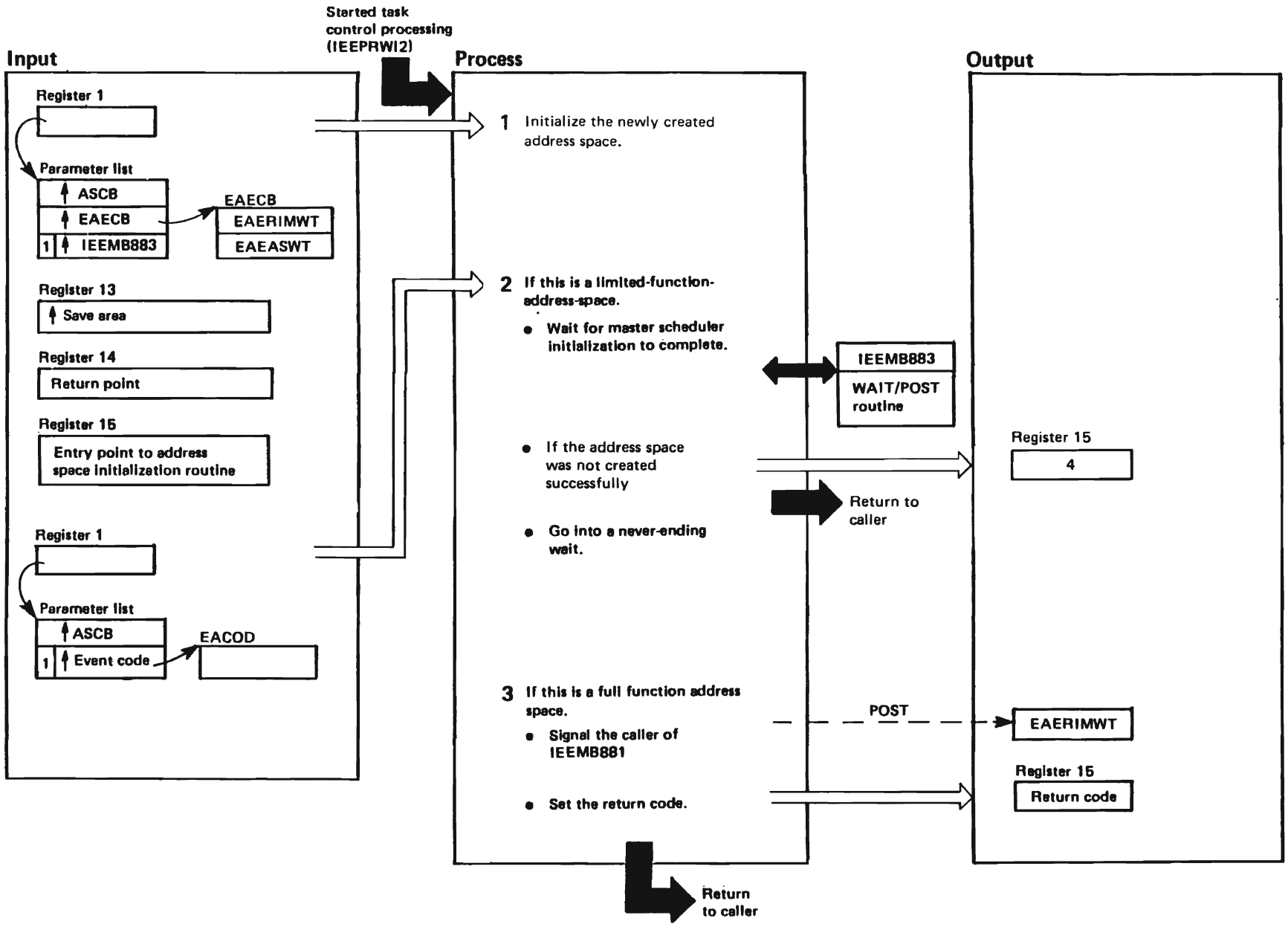


Diagram 73. Model System Address Space Initialization Part 2 of 2

Extended Description	Module	Label
Each of the system component address spaces created via the system component address space create routine (IEEMB881) has its own system component address space initialization routine, which is specified as input to IEEMB881. These routines are listed in Section 1 under "System Component Address Space Initialization." This diagram describes the basic functions that these system address space initialization routines have in common. The input to system address space initialization routines consists of pointers to the ASCB of the system address space, a list of ECBs, and the system address space WAIT/POST routine (IEEMB883). The list of ECBs contains two ECBs, EAERIMWT and EAEASWT. (The caller of IEEMB881 waits on ECB EAERIMWT, which is posted by the system address space initialization routine only if the address space is a full function address space. The system address space initialization routine waits on EAEASWT, which is posted by the caller of IEEMB881.) **1** Each initialization routine initializes its address space according to its own requirements. This includes creating and initializing control blocks, loading code, and performing any necessary functions. (These control blocks can be accessed by other address spaces via cross memory instructions.)		
2 If a system address space is created as a limited-function-address-space, the initialization routine calls the system address space WAIT/POST routine IEEMB883 to ensure that master scheduler initialization is complete. As input, to IEEMB883, the initialization routine supplies the event code (EACOD) (set equal to one to indicate a wait for master scheduler initialization to complete) and a pointer to the ASCB of the address space bing initialized. IEEMB883 posts ECB BAMSIECB to activate the master scheduler wait routine (IEEVWAIT) and then waits on ECB CHASWT, which is posted by IEEVWAIT when master scheduler initialization is complete. After the system address space initialization routine receives control again, it performs one of the following functions: • If the address space was not created successfully, the initialization routine returns to IEEPRWI2 with a return code of X'04' to indicate that the system address space being initialized is to be terminated. • If the address space was created successfully, the system address space initialization routine goes into a never-ending-wait, by waiting on an ECB that is never posted. Either before or after calling IEEMB883, the initialization routine posts ECB EAERIMWT, which the caller of IEEMB881 is waiting on.	IEEVWAIT	
3 If the system address space is a full-function-address space, the initialization routine posts ECB EAERIMWT, which the caller of IEEMB881 is waiting on, and sets the return code. If the address space was created successfully, the system address space initialization routine sets a return code of zero and returns to the started task control processing routine (IEEPRWI2), which then invokes the START command syntax routine (IEEVSTAR) using the START parameters supplied as input to the system address space create routine (IEEMB881). If the address space was not created successfully, the initialization routine returns to IEEPRWI2 with a return code of X'04' to indicate that the system address space being initialized is to be terminated.	System address space initialization routine	

Diagram 74. Started Task Control Processing Part 1 of 10

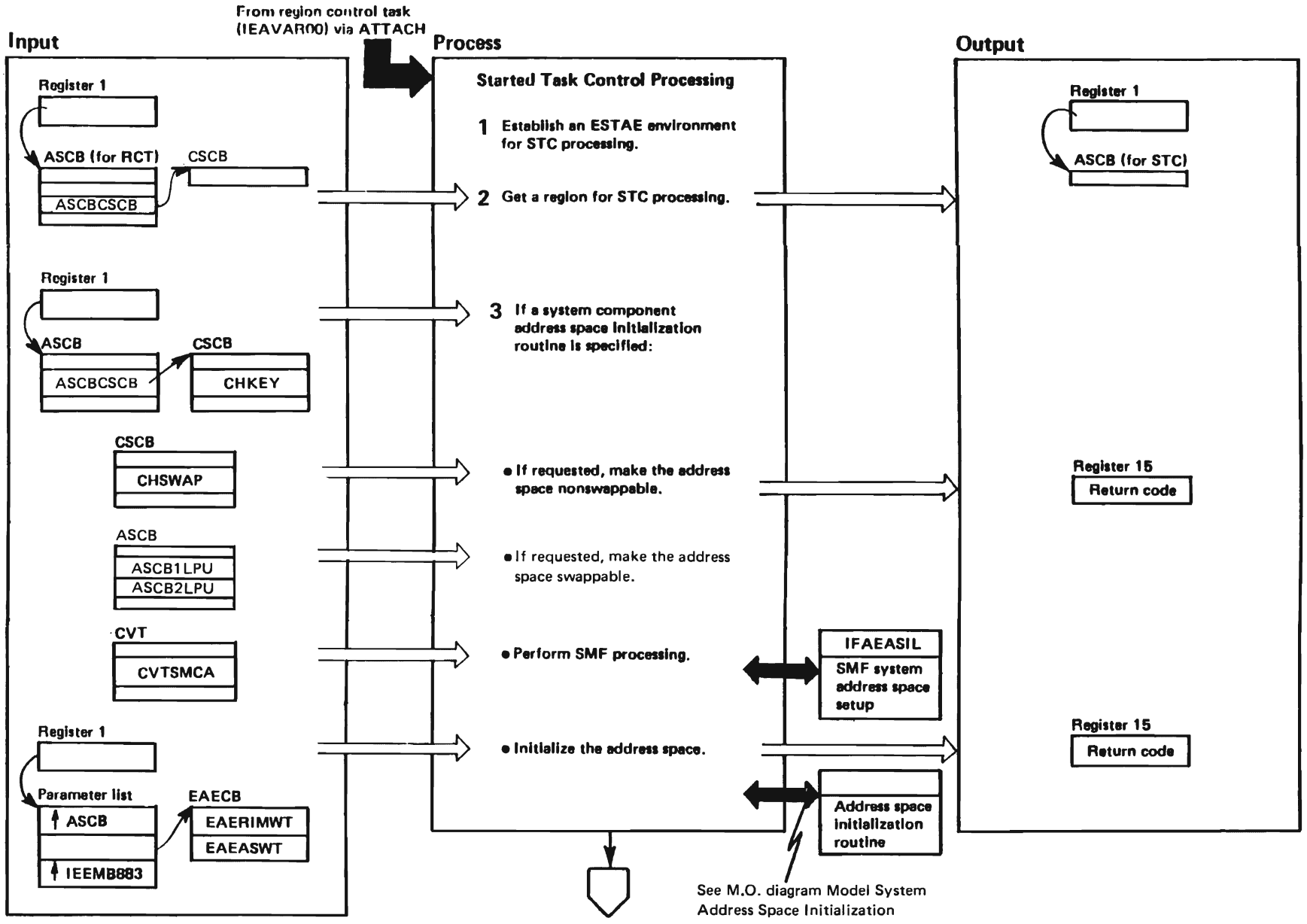


Diagram 74. Started Task Control Processing Part 2 of 10

Extended Description	Module	Label
The started task control (STC) routines oversee both the initialization of system component address spaces and the processing of START, MOUNT, and LOGON commands. In both situations, STC processing includes steps 1 and 2. If a system component address space is to be initialized, processing continues at step 3, where STC links to the system component address space initialization routine. If the system component address space is initialized as a limited function address space, the initialization routine goes into a never-ending-wait and control does not return to STC. If the system component address space is initialized as a full function address space, control returns to step 4 and processing continues at step 5 as for a START command. If a START, LOGON, or a MOUNT command is being processed, control proceeds directly from step 2 to step 5.		
1 The first STC module that receives control is IEEPRWI2. This routine establishes an ESTAE environment for STC by creating an ESTAE parameter list and loading module IEESB665, the STC recovery exit routine.	IEEPRWI2	
The ESTAE environment ensures that the recovery termination management (RTM) routines will receive control in the event of an STC error. RTM will, in turn, invoke IEESB665, which will then attempt to recover.		
If RTM provides a system diagnostic work area (SDWA) containing the necessary data, IEESB665 will schedule a retry by terminating the current STC processing and issuing a SETRP macro instruction.		
If no SDWA exists, IEESB665 will simply continue ABEND processing.		
In either case, the recovery routine will record the error in the SYS1.LOGREC data set and then return to RTM.		
2 Once the ESTAE environment is complete, IEEPRWI2 calls IGVGVRGN to obtain its own region.	IEEPRWI2	
3 If a system component address space is being created, the caller specified a system address space initialization routine (CHKEY contains the name of the routine) and IEEPRWI2 performs the following functions:	IEEPRWI2	
• If the system address space is to be nonswappable (CHSWAP=0), IEEPRWI2 issues a TRANSWAP SYSEVENT. A return code of X'20' indicates a TRANSWAP error, in which case an ABEND occurs.		
• If the system address space is to be swappable (ASCB1LPU=on or ASCB2LPU=on), IEEPRWI2 issues a REQSWAP SYSEVENT.		
• If SMF has been initialized (CVTSMCA≠0), IEEPRWI2 links to the SMF routine, IFAEASIL, to schedule an SRB for data gathering purposes. The SRB, when dispatched, will gather SMF accounting data, timer information, and processor statistics for limited-function-start address spaces that do not go through START processing. (The system address space initialization routine controls whether or not the address space being created goes through a limited or a full-function start.)	IFAEASIL	
• In order to perform the necessary initialization, IEEPRWI2 links to the caller-supplied address space initialization routine, whose name is located in CHKEY. (See M.O. diagram Modal System Address Space Initialization.) If one is not supplied, the address space will not be initialized in any special manner.	System address space initialization routine	
For the module flow of started task control, see Figure 4-11.		

Diagram 74. Started Task Control Processing Part 3 of 10

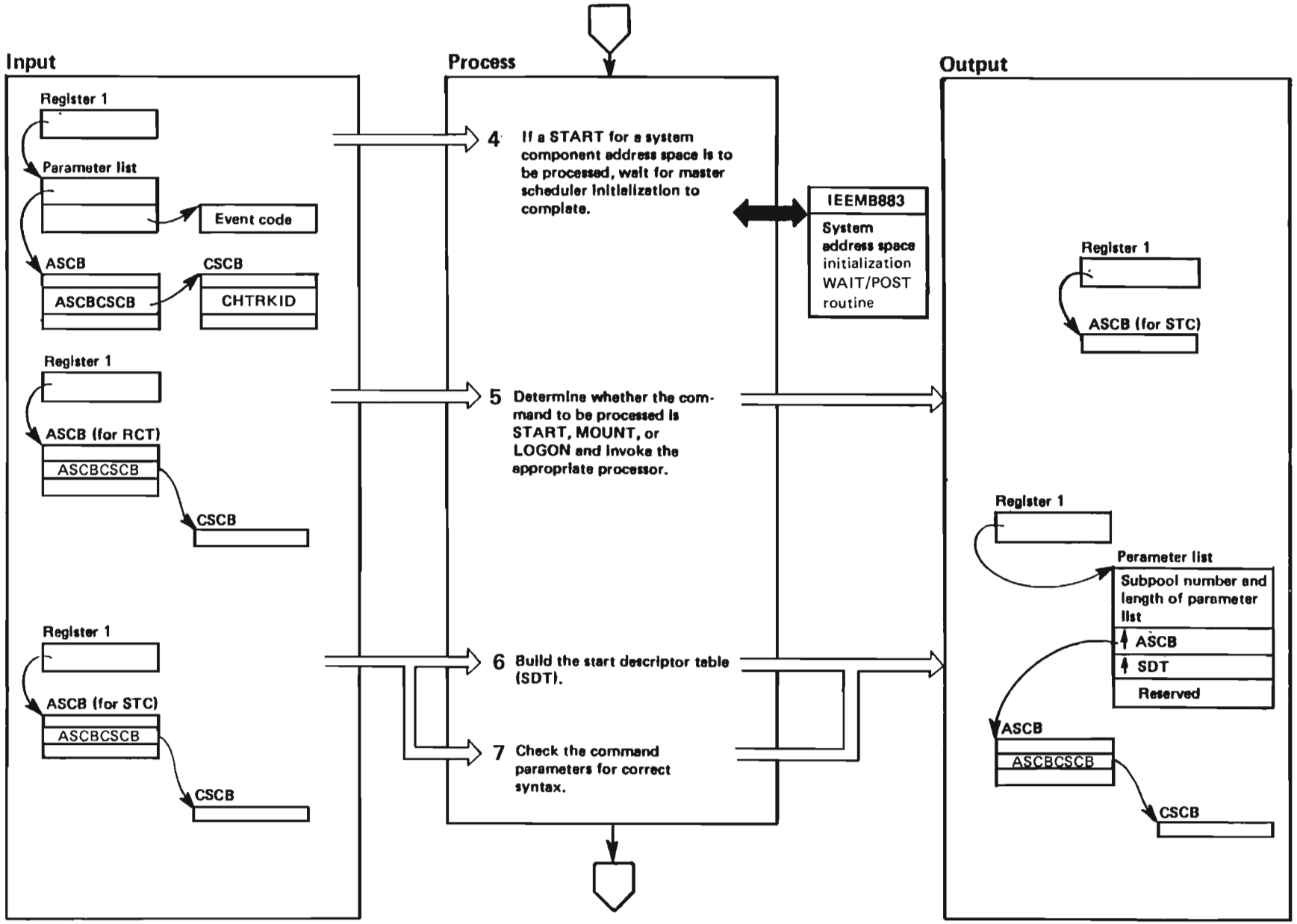


Diagram 74. Started Task Control Processing Part 4 of 10

Extended Description	Module	Label
4 If a START for a system component address space is to be processed (CHTRKID=X'04'), which means that the address space is a full function address space, IEEPRWI2 initializes the event code to 1 and calls the system address space initialization WAIT/POST routine (IEEMB883) to wait for master scheduler initialization to complete.	IEEPRWI2 IEEMB883	
5 IEEPRWI2 checks an indicator in the CSCB to determine which command, START, MOUNT, or LOGON, is to be processed.		
If the command is START, IEEPRWI2 issues an XCTL macro instruction to IEEVSTAR.	IEEVSTAR	
If the command is MOUNT, the XCTL macro instruction invokes IEEVMNT1, and for a LOGON, IKJEFLA.	IEEVMNT1 IKJEFLA	
Both IEEVSTAR and IEEVMNT1 are part of STC; IKJEFLA is part of LOGON processing.	IEEVSTAR and IEEVMNT1	
6 Both routines, IEEVSTAR and IEEVNMT1 begin processing by creating a start descriptor table (SDT) and initializing it with blanks and zeroes.		
7 IEEVSTAR and IEEVMNT1 check the commands and associated parameters for correct syntax. When either routine finds an error, it places the command name in an extended save area and invokes module IEE0503D to write a message using that name.		
IEEVSTAR and IEEVMNT1 continue error processing by freeing the SDT and issuing an XCTL macro instruction to the last routine of STC, IEEPRTN2. This module cleans up the data areas and ends the task that was begun for a START or MOUNT command.		

Diagram 74. Started Task Control Processing Part 5 of 10

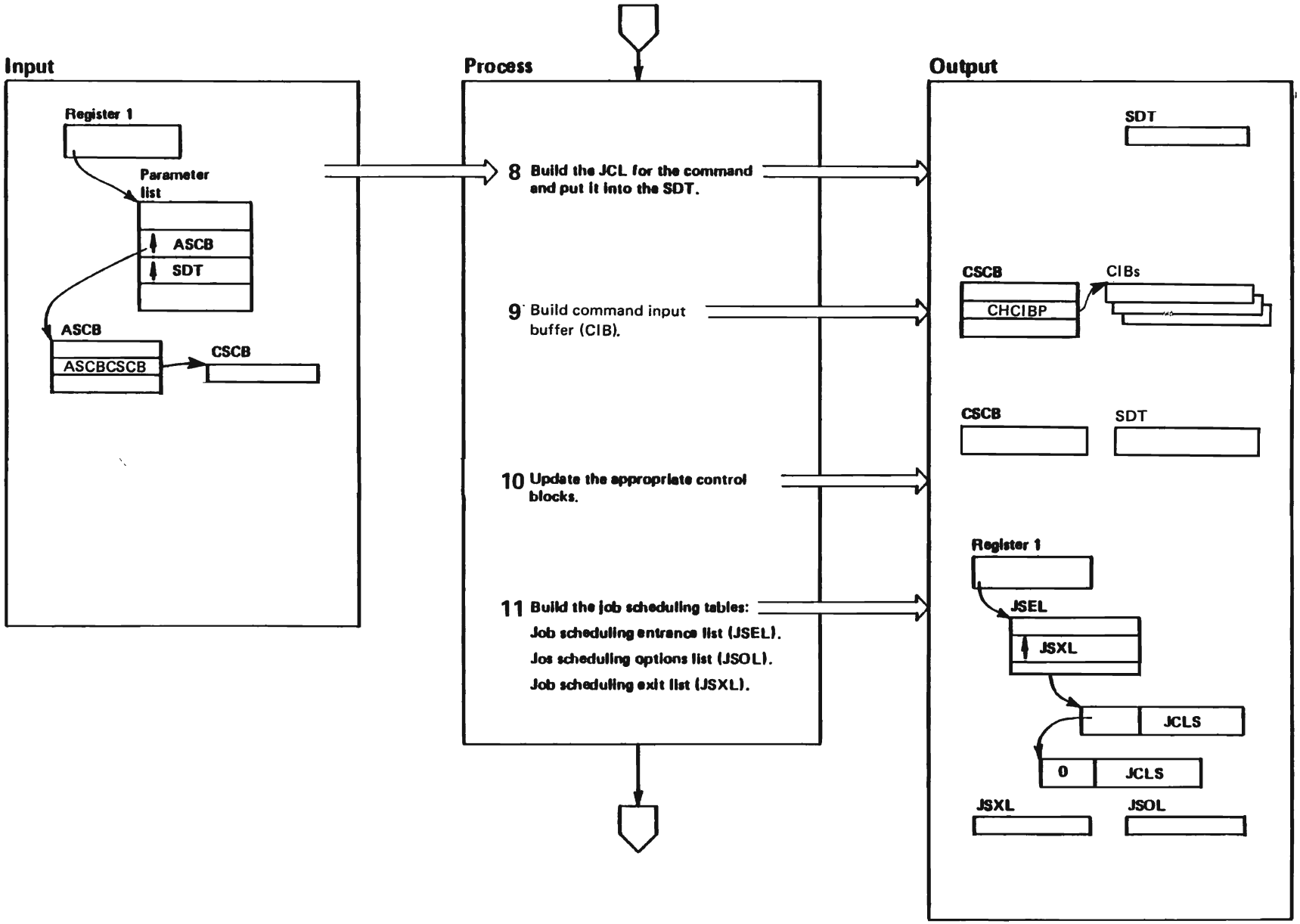


Diagram 74. Started Task Control Processing Part 6 of 10

Extended Description	Module	Label
8 Every time a MOUNT command is specified, JCL that identifies the device to be mounted is supplied by the user. When a START command is specified, the JCL must be created for it.		
IEEVSTAR uses the START command parameter to build internal JCL statements.	IEEVSTAR	
The procedure name that was specified with the START command becomes the JCL jobname. The name is placed in the CSCB and a pointer to it in the ASCB.		
When an ID was included on the START command, it becomes the stepname for an EXEC card. It too is saved in the CSCB. If no ID was specified, the stepname used is "STARTING".		
If a unit parameter and volume serial number were entered with START, they are used for a DD statement.		
As each JCL statement is generated, it is moved into the SDT.		
9 Both IEEVSTAR and IEEVMNT1 create command input buffer CIBs using storage from subpool 245. If the SUB operand is present on the START command, IEEVSTAR creates an STC communication CIB.	IEEVSTAR and IEEVMNT1 IEEVSTAR	
10 IEEVMNT1 stores JCL related information, which was provided with the MOUNT command, in the SDT.	IEEVMNT1	
IEEVMNT1 and IEEVSTAR both update the CSCB with information related to the command parameters. Then, before passing control to IEEVJCL, they build a parameter list for it.	IEEVMNT1 and IEEVSTAR	
After filling in the CSCB, if processing a system address space and the procedure name is "IEESYSAS", IEEVSTAR overlays the name of the procedure (IEESYSAS) with the name of the address space. The name of the address space is found in the step name.		
11 IEEVJCL is the JCL build routine. For each newly created JCL statement, IEEVJCL gets an 88-byte area of storage space called a JCLS. It moves each statement into a JCLS (job control language string) and chains each JCLS to another.	IEEVJCL	
After the JCLS chain is done, IEEVJCL issues a GETMAIN macro instruction for space for the JSXL. It places these pointers in the JSEL: • A pointer to the JSXL. • A pointer to the JCLS chain. • A pointer to the CSCB. • A pointer to the ASCB.		
IEEVJCL issues another GETMAIN macro instruction, this time for the JSOL, which is initialized with the command's jobname, EXEC name, and procedure name.		
IEEVJCL completes processing by freeing the SDT and invoking the job scheduling subroutine, IEESB605.		

Diagram 74. Started Task Control Processing Part 7 of 10

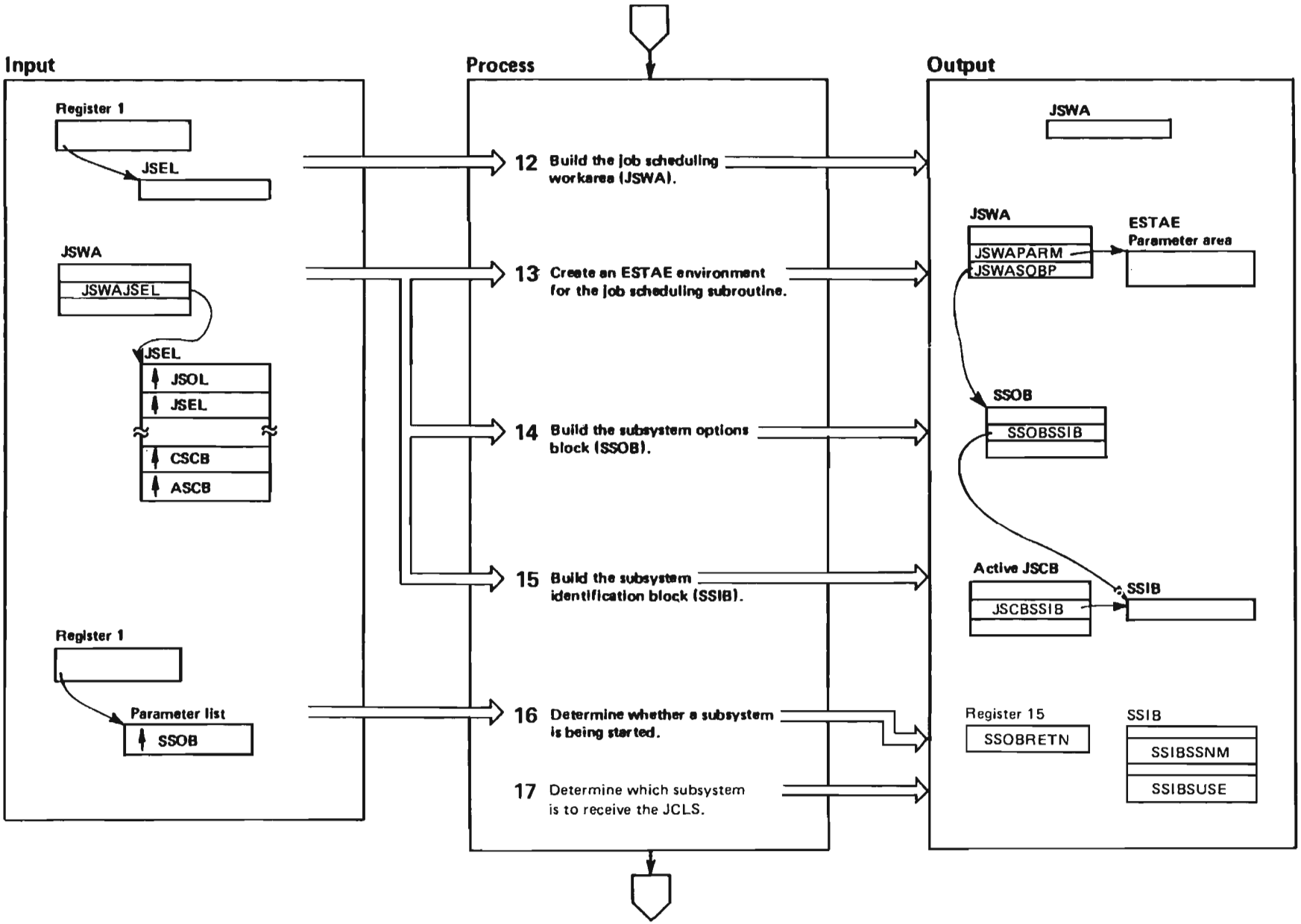


Diagram 74. Started Task Control Processing Part 8 of 10

Extended Description	Module	Label
12 IEESB605 creates the environment needed for initiator processing. It begins by obtaining storage space for its own work area, called the job scheduling work area (JSWA).	IEESB605	
13 IEESB605 creates its own ESTAE environment. It builds an ESTAE parameter list and issues the ESTAE macro instruction, then loads IEESB670, its own recovery exit routine. Then, if an error occurs in IEESB605, the RTM routines will receive control and, after preliminary processing, pass control on to IEESB670.	IEESB605	
If RTM provides an SDWA containing the necessary data, IEESB670 will schedule a retry of IEESB605 that terminates current STC processing by issuing a SETRP macro instruction.	IEESB670	
If no SDWA exists, IEESB670 will continue ABEND processing.		
In either case, the recovery routine will record the error in the SYS1.LOGREC data set and return to RTM.		
14 IEESB605 builds an SSOB to represent the command as a job. It places, in the JSWA, an SSOB pointer and an indicator that the SSOB exists.	IEESB605	
15 IEESB605 also builds an SSIB for the command and places pointers to it in the current JSCB and the SSOB.		
16 IEESB605 determines whether a subsystem is being started by issuing the IEFSSREQ macro instruction to invoke the master subsystem. When the master subsystem returns control, IEESB605 checks the return code in register 15.		
If STC is starting a subsystem that can provide its own sysout services, IEESB605 places a pointer to the JCLS chain in the SSIB (SSIBSUSE). This will allow the master subsystem access to the JCLS; otherwise STC treats the task as a non-subsystem.		
Note: In the event of a primary subsystem manual start at IPL, any START commands entered prior to starting the primary subsystem will result in the started task(s) waiting for the STC resource. After the primary subsystem is initialized and the STC resource becomes available, the primary subsystem starts and the waiting task(s) are executed.		
17 If the operator requests that STC use a particular subsystem to receive the JCLS, IEESB605 issues IEFSSREQ to validate the subsystem, IEESB605 checks the return code in register 15. If the subsystem is valid, IEESB605 places its name in the SSIB (SSIBSSNM). Otherwise, it issues message IEE825I, IEE826I, or IEE827I, depending on the error.		
If no subsystem was requested, IEESB605 places the name of the job entry subsystem in SSIBSSNM.		

Diagram 74. Started Task Control Processing Part 9 of 10

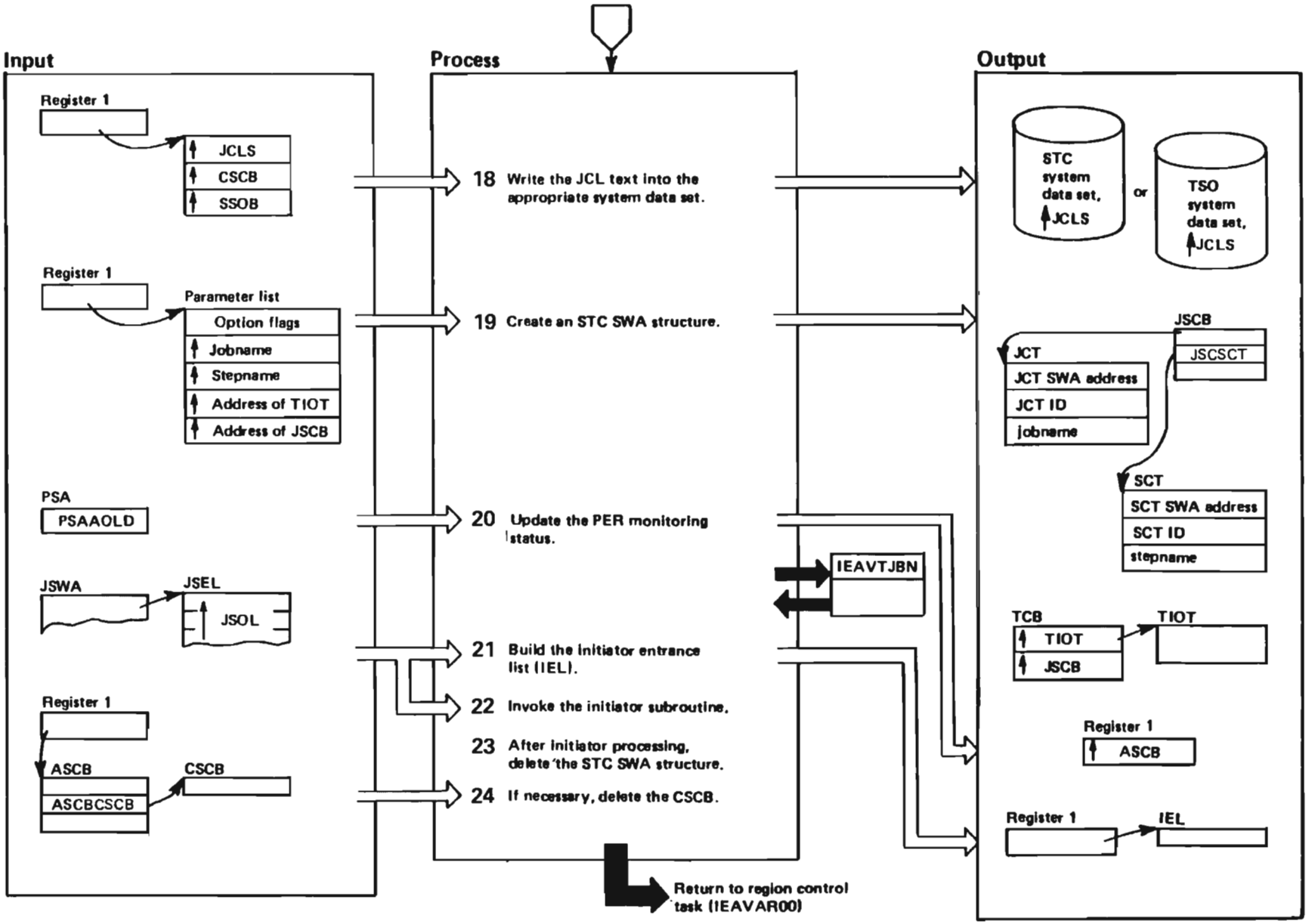


Diagram 74. Started Task Control Processing Part 10 of 10

Extended Description	Module	Label
18 When STC is not starting a subsystem, IEESB605 calls IEFJSWT, STC's internal JCL write routine. IEFJSWT first initializes a request parameter list (RPL) and an access control block (ACB), the control blocks associated with the system data set into which the JCLS will be written. IEFJSWT checks an indicator in the CSCB to determine which command is in progress. For a LOGON, the system data set used is TSOINRDR; for the other commands, it is STCINRDR. For every command, IEFJSWT opens the appropriate data set and writes each JCLS record into it. When the writing is done, IEFJSWT returns to IEESB605.	IEFJSWT	
19 IEESB605 clears all the existing JCLS pointers and then calls the STC SWA initialization routine, IEESB601. This initialization routine builds a skeletal SWA structure in preparation for initiator processing. The SWA structure includes these control blocks: • JSCB. • QMPA (queue manager parameter area). • JCT. • ACT • SCT. • ACT. • TIOT.	IEESB605 IEESB601	
20 IEESB605 branches to the PER monitor routine (IEAVTJBN) to determine if PER should update the monitoring status for the address space.	IEAVTJBN	
21 Once the SWA structure is complete, IEESB601 returns control to IEESB605, which initializes an initiator entrance list (IEL) with the following information: • Options from the JSOL. • A pointer to the JSEL. • A pointer to the initiator option list. • A pointer to the initiator exit list.	IEESB605	
22 IEESB605 clears pointers to the JSOL, which is no longer needed, and then issues a LINK macro instruction invoking an initiator subroutine, IEFSD060. From that routine, initiator processing proceeds normally until the command task has been executed and is in termination. At that point, IEESB605 again receives control. (During initiator processing of a MOUNT command, the initiator ATTACH routine, IEFSD263, attaches IEEVMNT2, the MOUNT command processor. IEEVMNT2 returns control to IEFSD263.)	IEFSD060	
23 IEESB605 performs STC clean-up functions by freeing the IEL, the JSWA, and the SSIB and SSOB. It invokes IEESB601 once again, this time to delete the SWA structure it previously created. When control returns to IEESB605, it issues an XCTL macro instruction to IEEPRTN2.	IEESB605	
24 The STC free region routine, IEEPRTN2, does not free storage space but simply checks for the existence of CSCB in the ASCB and frees it if it still exists. IEEPRTN2 returns to region control task.	IEEPRTN2	

Diagram 75. Master Scheduler Initialization (IEEVIPL, IEEMB860) Part 1 of 14

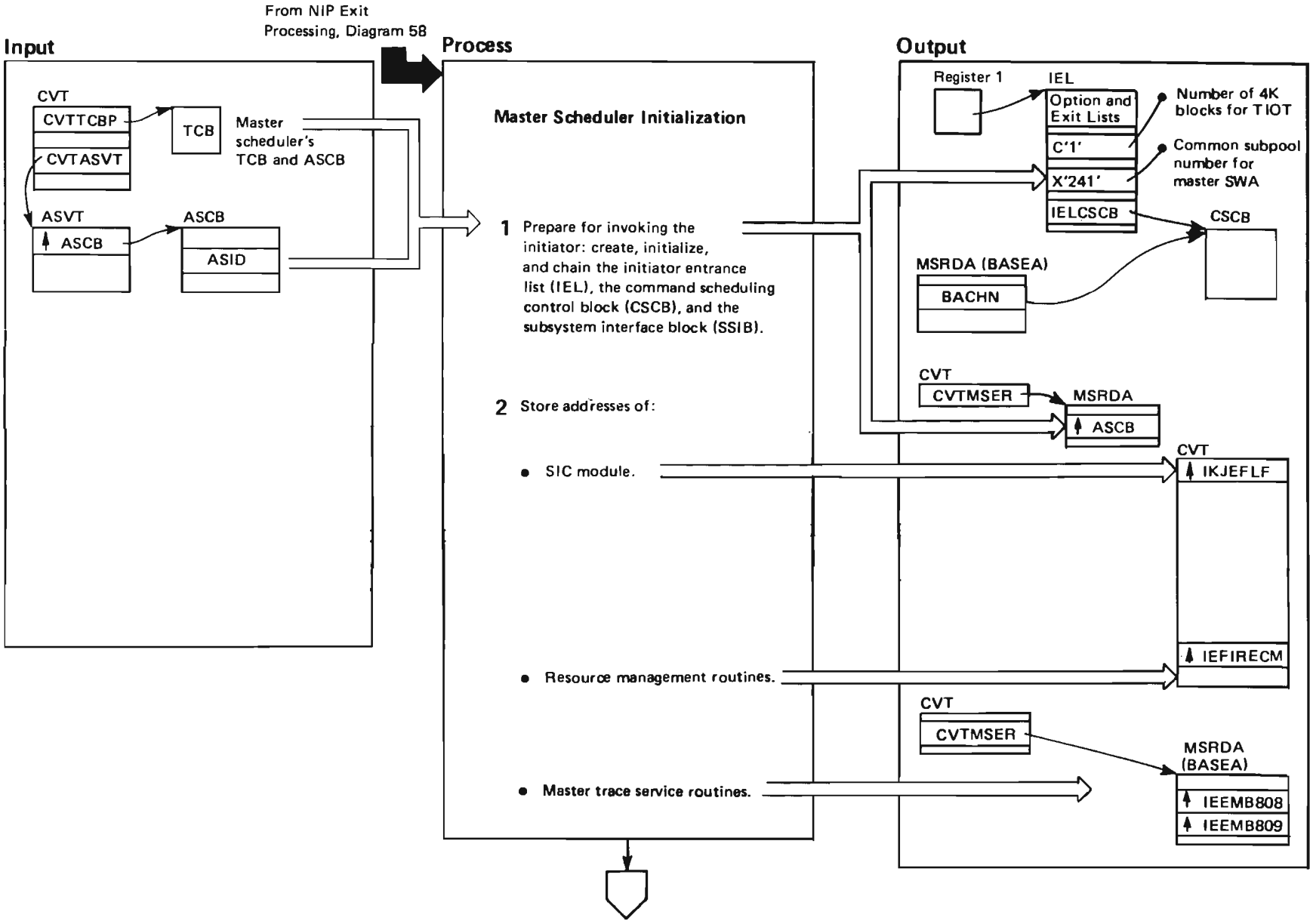


Diagram 75. Master Scheduler Initialization (IEEVIPL, IEEMB860) Part 2 of 14

Extended Description	Module	Label
Master scheduler initialization, invoked from the NIP module IEAVNIPX, is the final phase of system initialization. Following this phase of its initialization, the system is ready to receive work in the form of user jobs and commands from the console. Besides initializing parts of the system, master scheduler initialization attaches the permanent tasks that run in the master scheduler's region. It also invokes routines that write initialization records in an SMF data set and in SYS1.LOGREC.		
1 In this step, IEEVIPL creates the control blocks that the initiator, which will attach IEEMB860, uses. IEEVIPL places the master scheduler's CSCB at the beginning of the CSCB chain.	IEEVIPL	
2 IEEVIPL locates entry points for each of the following modules via the LOAD macro instruction. For more information on these modules, refer to the publication *System Logic Library*.	IEEVIPL	
• The system-initiated cancel (SIC) routine-schedules the orderly cancellation of a time-sharing user when requested to do so by the operator or when forced to do so by a disconnected telecommunications line.		
• The resource management routine — including initiator resource management.		
• The master trace service routines — include:		
- The IEEMB808 data entry routine whose address is stored in field BAMTRTN.		
- The IEEMB809 create/deactivate master trace table routine whose address is stored in BAMTCDR.		

Diagram 75. Master Scheduler Initialization (IEEVIPL, IEEMB860) Part 3 of 14

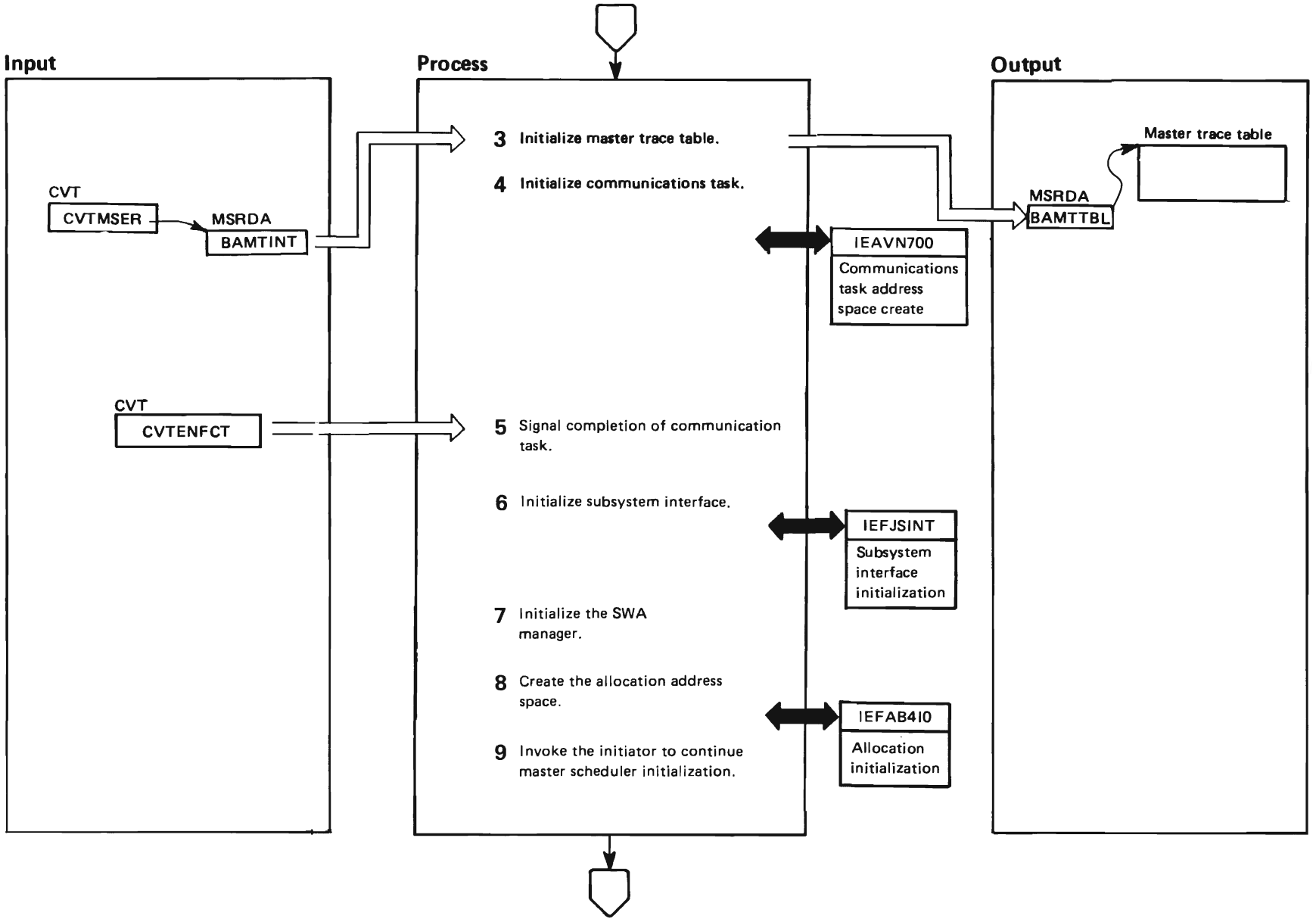


Diagram 75. Master Scheduler Initialization (IEEVIPL, IEEMB860) Part 4 of 14

Extended Description	Module	Label
3 Master scheduler initialization determines if a master trace table was requested during system initialization. If so, it obtains the required size from field BAMTINIT in the master scheduler resident data area (MSRDA). It issues the IEETRACE macro to initialize the master trace table. If the master trace table initialization fails, module IEEVIPL issues message IEE840I.		
4 Communications task initialization prepares the system operator consoles for sending and receiving data. This Initialization includes creating and initializing the communications task address space, initializing the task by setting up the communication task control blocks and environment, and initializing DIDOCS. For more detail, refer to Diagrams 64, 65, 66 and 67.	IEAVN700 IEAVVINT IEECVGCI	
5 IEEVIPL uses the event notification facility (ENF) routine, IEFENFXX, to signal that the initialization of the communications task is complete.		
6 Subsystem interface initialization creates the control blocks to be used by the subsystem interface to determine which subsystem routine is to receive control following a request. Initialization builds a subsystem CVTs (SSCVTs) for the subsystem specified on the SCHEDULR sysgen macro and in IEFJSSNT. Then initialization builds the subsystem vector table for the master subsystem. For more detail, refer to Diagrams 68 and 69.	IEFJSINT	
7 IEEVIPL calls IEFQBINT to initialize the SWA manager.	IEFQBINT	
8 IEEVIPL invokes the allocation initialization routine, IEAFAB4I0, to: • Store allocation entry points in the JESCT and CVT • Load the EDT and store its address in the JESCT (JESCDT) • Build the UCB pointer list (UPL) • Invoke the EDT verification routine (IEFEB400) • Create and initialize the allocation address space (ALLOCAS)	IEFAB4I0	
If IEFAB4I0 does not successfully complete the initialization of the allocation address space, the recovery routine, IEFAB4E6, indicates the failure in the JESCT (JESUASR=0) and issues message number IEF100I. IEFAB4I0 then returns to IEEVIPL with a return code of 4. IEEVIPL continues with the system initialization. If IEFAB4I0 loads an invalid EDT, IEFABI0 invokes IGFPTERM to issue message IEF927I or message IEF928I and places the system in a wait state with a code of X'400'.	IEFAB4E6	

Extended Description	Module	Label

9 The control blocks created in step 1 above (IEL, JSCB, CSCB, and SSIB) are used by the initiator, which, via the subsystem interface, requests job selection from the master subsystem. The master subsystem serves as a substitute for the job entry subsystem, which has not yet been initialized. Figure 4-8 portrays the initiation process for the master scheduler. The system initialization routine, IEFJJOBS obtains the master JCL suffix value from the scheduler NIP parameter list (IEEZB807) and appends it to MSTJCL to form a member name. The master subsystem obtains the JCL card images from the MSTJCLxx member of SYS1.LINKLIB. The MSTJCL00 member contains the following:
`//MSTJCL00 JOB MSGLEVEL=(0,0)`
`//  EXEC PGM=IEEMB860,DPRTY=(15,15)`
`//STCINRDR DD SYSOUT=(A,INTRDR)`
`//TSOINRDR DD SYSOUT=(A,INTRDR)`
`//IEFPDSI DD DSN=SYS1.PROCLIB,DISP=SHR`
`//IEFPARM DD DSN=SYS1.PARMLIB,DISP=SHR`
`//SYSUADS DD DSN=SYS1.UADS,DISP=SHR`
`//SYSLBC DD DSN=SYS1.BRODCAST,DISP=SHR`
`//START &SSNAME`
`/*`
The symbolic name, &SSNAME, is replaced by the name given to the job entry subsystem at system generation time. With MSTJCLxx as input, the converter and interpreter create first internal text then SWA control blocks. The EXEC statement of this JCL causes execution of IEEMB860, which continues the master scheduler initialization function. The DD statements permit allocation of TSO data sets and other system data sets required by those tasks that execute in the master scheduler region. The START command causes a CSCB to be created which, when processed by the master scheduler wait, will start the job entry subsystem (for example, JES2).
The function of the initiator is described in detail in the publication *System Logic Library*.
	IEFSD060	

Diagram 75. Master Scheduler Initialization (IEEVIPL, IEEMB860) Part 5 of 14

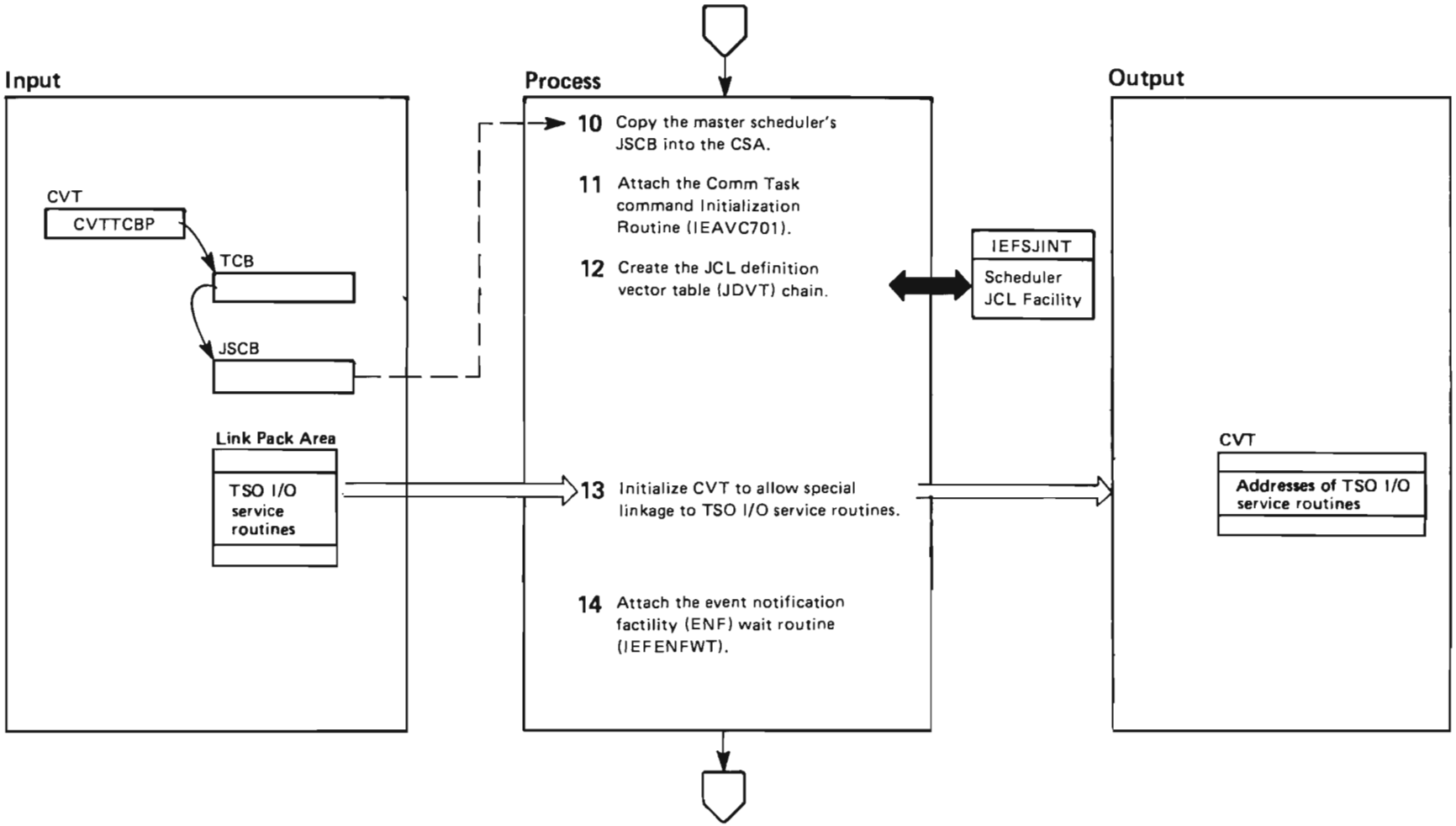


Diagram 75. Master Scheduler Initialization (IEEVIPL, IEEMB860) Part 6 of 14

Extended Description	Module	Label
10 The initiator fills in the JSCB, which is located in the master scheduler's local system queue area (LSQA). After being attached by the initiator, IEEMB860 moves this JSCB into the common service area (CSA). Normally, a task leaves its JSCB in the LSQA.	IEEMB860	
11 IEEMB860 attaches the Comm Task Command Initialization Routine (IEAVC701) and waits for it to complete.		
12 IEEMB860 calls the scheduler JCL facility routine, IEFSJINT, to create the default JCL definition vector table (JDVT). See *System Logic Library* for a description of IEFSJINT.		
13 The TSO module IKJEFXSR invokes the link-pack-area search routine (IEAVVMSR) to obtain the addresses of certain I/O service routines used by TSO command processors. Upon return from IEAVVMSR, register 0 points to an LPDE (link pack directory entry). This directory entry contains the entry point address of the routine it located. IEAVVMSR is invoked for each routine name that is coded in IKJEFXSR: IKJGETL – Get-line routine IKJPUTL – Put-line routine IKJPTGT – Put-get routine IKJSTCK – Stack routine IKJSCAN – Command scan routine IKJPARS – Parse routine IKJDAIR – Dynamic allocation interface routine IKJEHDEF – Default routine IKJEHCIR – Catalog information routine	IKJEFXSR	BUILD
As the routines are located, IKJEFXSR places their entry point addresses in the CVT and turns on the high-order bit in each CVT field containing an entry point address. To obtain the CVT field name corresponding to a routine, IKJEFXSR replaces the first three characters ('IKJ') of the routine name with the characters 'CVT'. If the search routine cannot find an I/O service routine in the link pack area, IKJEFXSR continues processing with the next routine name. It leaves the high-order bit off in the CVT field for the routine not found. When a TSO command processor issues the CALLTSSR macro instruction, it specifies the module name of the I/O service routine it is invoking. The macro instruction checks the high-order bit of the corresponding CVT field. If it is on, CALLTSSR branches to the address specified in the corresponding CVT field. If it is off, CALLTSSR issues a LINK macro instruction for the module name requested. If TSO/E Release 2 or later is installed, IKJEFXSR builds the TSO vector table (TSVT) and initializes it with the addresses of additional TSO service routines. The initialization process is the same as described for initializing the CVT fields.		
Recovery Processing If IKJEFXSR abnormally terminates, its ESTAE routine, after issuing a message, passes control back to master scheduler initialization (IEEVIPL). IEEVIPL continues with system initialization and does not terminate.	IKJEFXSR	XSRRETRY
14 IEEMB860 attaches the ENF wait routine (IEFENFWT), which completes ENF initialization. ENF can now process all requests for service. (See *System Logic Library,* for a description of IEFENFWT.)		

Diagram 75. Master Scheduler Initialization (IEEVIPL, IEEMB860) Part 7 of 14

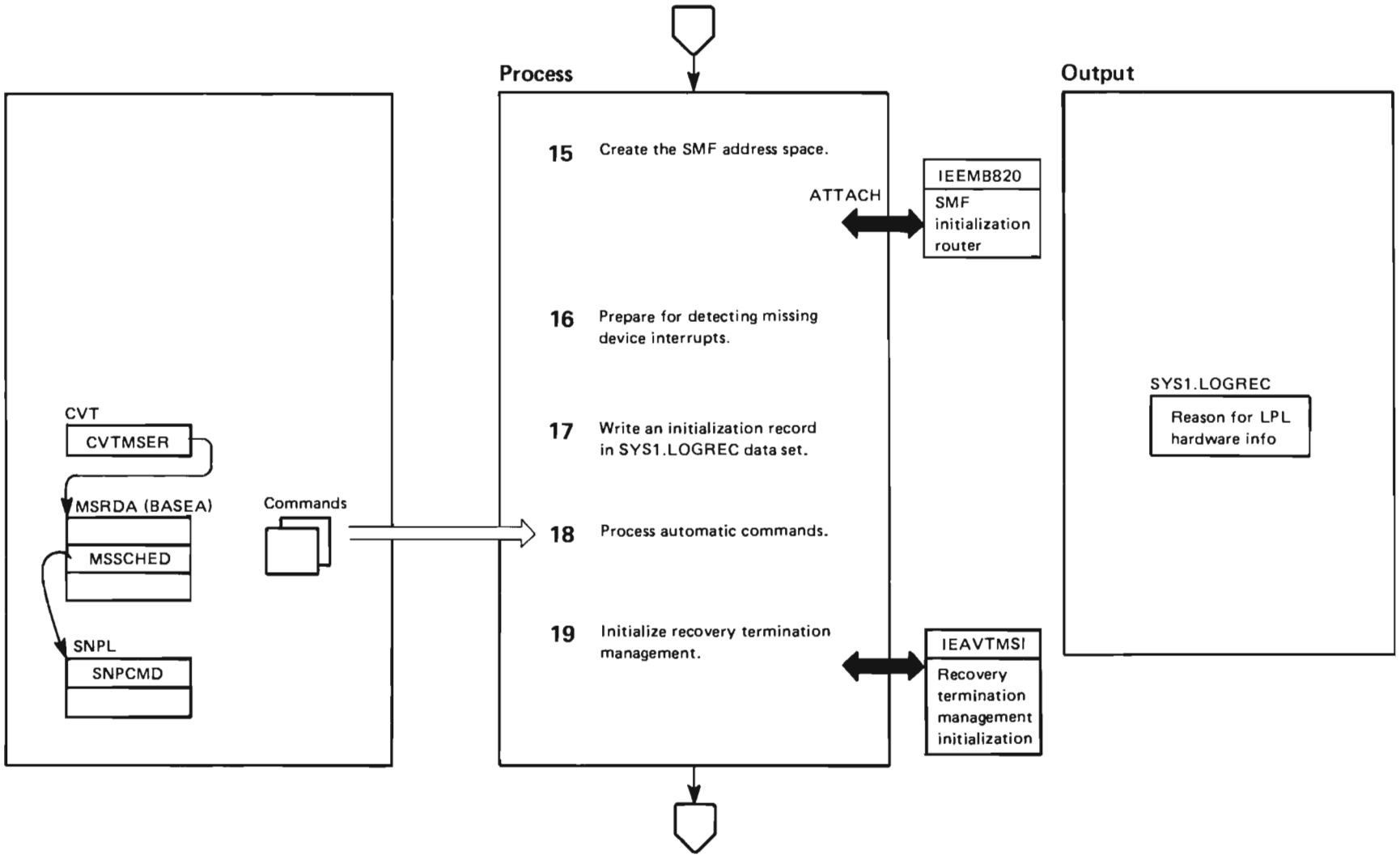


Diagram 75. Master Scheduler Initialization (IEEVIPL, IEEMB860) Part 8 of 14

Extended Description	Module	Label
15 IEEMB860 attaches the SMF create router (IEEMB820) to start processing to create the SMF address space via the system address space initialization.		
16 IEEMB860 attaches the missing interrupt handler (MIH), which initializes itself by determining the time intervals at which it checks for missing interruptions. A complete description of MIH logic is in *System Logic Library.*	IOSRMIHT	IGFINTVL
17 IEEMB860 attaches the IPL/outage recorder to write the initialization record to SYS1.LOGREC via SVC 76. This record contains the time and date of the IPL, the processor model and serial number, a code indicating the reason for the IPL, channel information, and the highest real storage address.	IOSROUTG	
18 The NIP module (IEAVNP13 has already read IEACMD00 and COMMNDxx members of SYS1.PARMLIB containing the automatic commands), copied the commands into storage, and placed a pointer to the commands in field SNPCMD of the scheduler NIP parameter list (IEEZB807). Master scheduler initialization internally issues each of these commands via SVC 34, and then frees the space taken up by the commands.	IEEMB860	
19 Recovery termination management (RTM) initialization attaches three permanent tasks: the first processes asynchronous recording requests, the second schedules SVC dumps for the master scheduler's address space, and the third processes requests for address space termination.	IEAVTMSI	

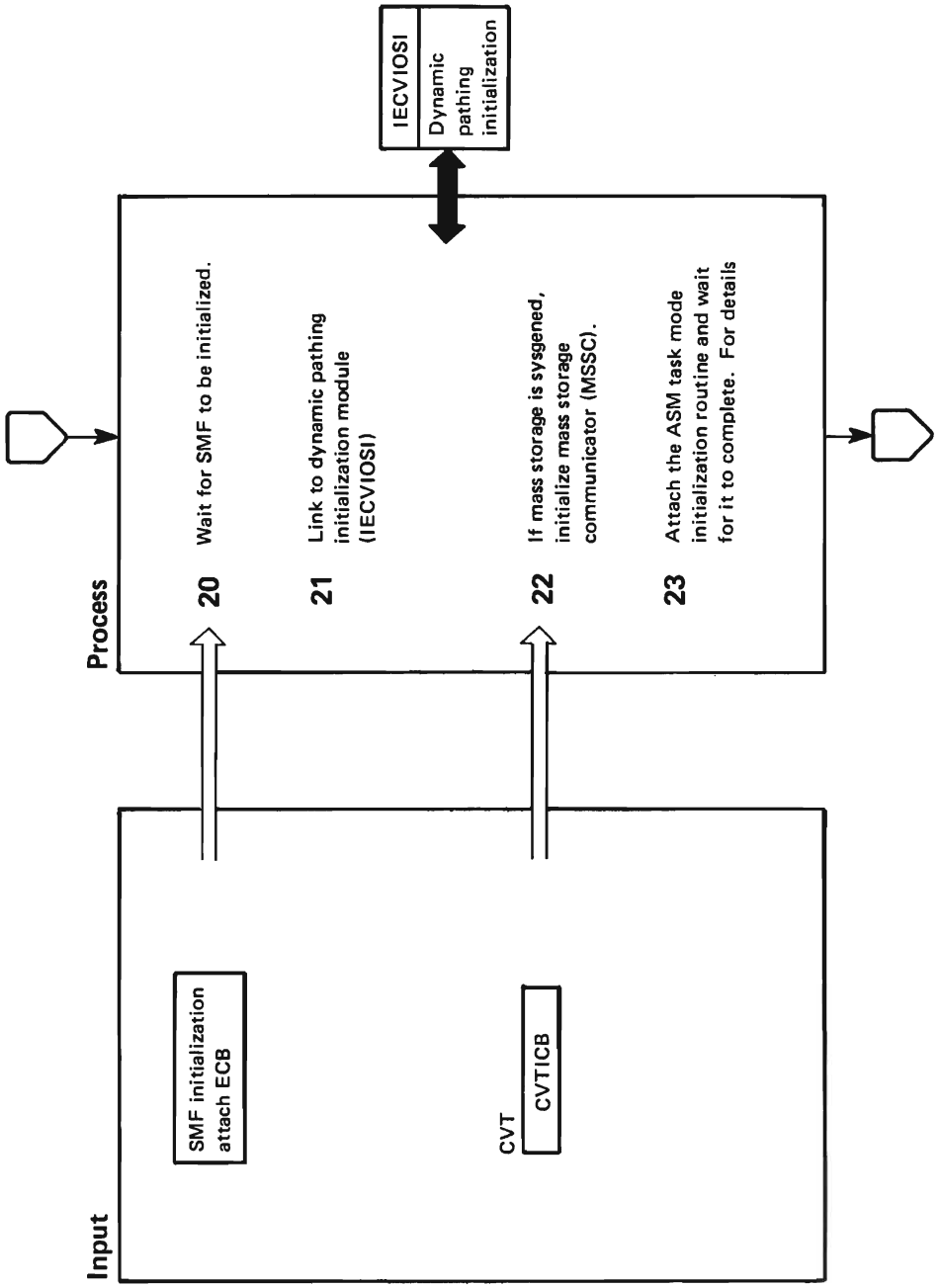


Diagram 75. Master Scheduler Initialization (IEEVIPL, IEEMB860) Part 9 of 14

Diagram 75. Master Scheduler Initialization (IEEVIPL, IEEMB860) Part 10 of 14

Extended Description	Module	Label
20 Because SMF initialization is an attached task, it runs in parallel with master scheduler initialization. Master scheduler initialization waits for completion of the SMF address space to be sure that SMF is ready to record before the system begins processing jobs.	IEEMB860	
21 IOSVIOSI calls IECVDPTH to initialize all online devices with the dynamic pathing feature. It also calls IOSVSMEI to establish IOSVSMGR as an event notification (ENF) listener for event code 4. (See *System Logic Library* for more details about dynamic pathing initialization and the IOS storage manager.)	IOSVIOSI	
22 If the mass storage system communicator (MSSC) has been included during system generation, and the MSSC RIM has processed successfully, control is passed via the LINK macro to the MSSC initialization module (ICBINIT). For more information, refer to *Mass Storage System Communicator Logic.*	IEEMB860	
23 Attach the ASM task mode initialization routine, ILRTMRLG and wait for it to complete. Module ILRTMRLG calls ILRTMI00 to complete ASM initialization.	ILRTMRLG ILRTMI00	

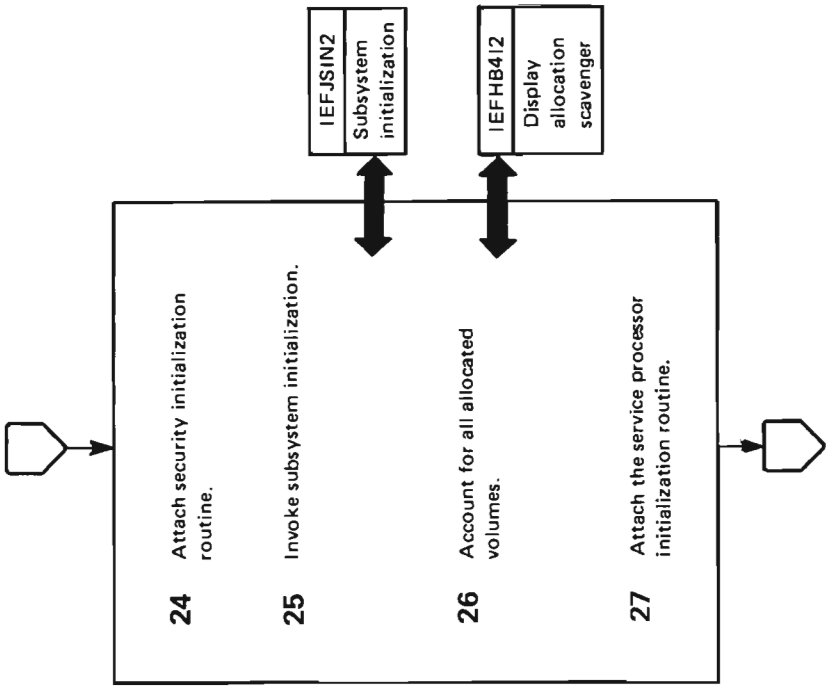


Diagram 75. Master Scheduler Initialization (IEEVIPL, IEEMB860) Part 11 of 14

Diagram 75. Master Scheduler Initialization (IEEVIPL, IEEMB860) Part 12 of 14

Extended Description	Module	Label
24 Attach the system security initialization routine (ICHSEC00). When it completes processing, detach it.	ICHSEC00	
25 Master scheduler initialization invokes IEFJSIN2 to initialize the subsystems.	IEFJSIN2	
26 IEEMB860 accounts for all allocated by volumes by invoking IEFHB4I2. IEFHB4I2 examines all UCBs and identifies whether or not the volume associated with the UCB is allocated. Up to this point in the master scheduler initialization process, volumes (such as the IPL volume) have been allocated for use by the various initialization routines without using the system allocation routines. IEFHB4I2 records the allocation status of each UCB in display allocation lookup tables (DALTs). When the system begins to use the system allocation routines, these DALTs serve as the starting status for current allocations. The system allocation routines then keep track of future allocation status.	IEFHB4I2	
27 IEEMB860 attaches IEAVSPDM to initialize the service processor.	IEAVSPDM	

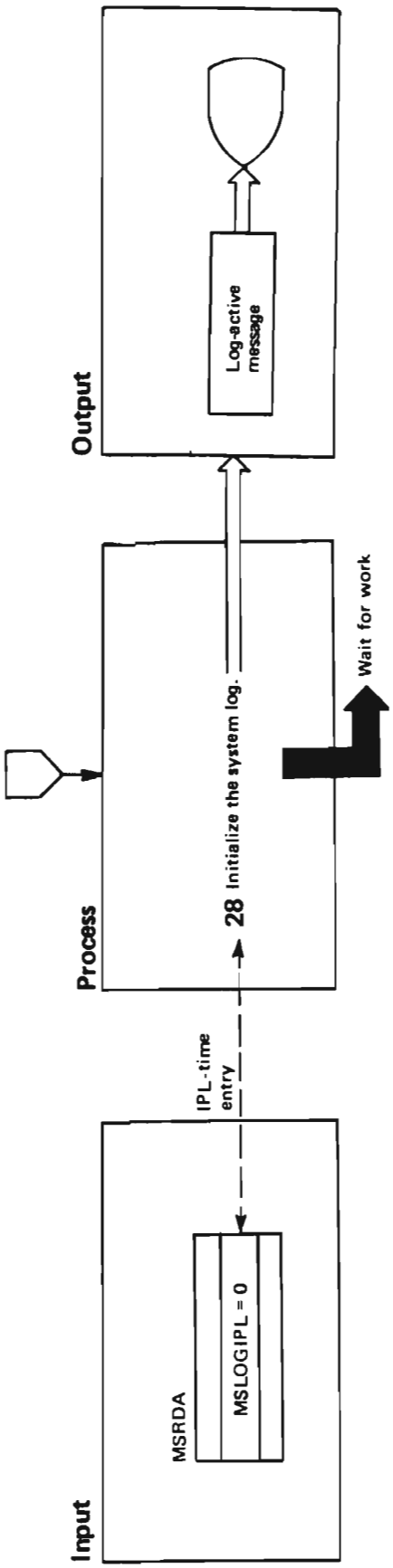


Diagram 75. Master Scheduler Initialization (IEEVIPL, IEEMB860) Part 13 of 14

Diagram 75. Master Scheduler Initialization (IEEVIPL, IEEMB860) Part 14 of 14

Extended Description	Module	Label
28 The final portion of master scheduler initialization runs under the control of the master scheduler wait module. First, it enqueues on serialization resources to prevent started task control (STC) and TSO LOGON from requesting job entry subsystem services before the subsystem is initialized: • STC serialization resource – major name SYSIEFSD, minor name STCQUE. • LOGON serialization resource – major name SYSIEFSD, minor name TSOQUE.	IEEVWAIT	
If step 13 of this diagram processed any START or MOUNT commands, started task control (STC), executing on behalf of these commands, suspends its processing until the job entry subsystem is initialized. STC accomplishes this by enqueuing on the STC resource (major name SYSIEFSD, minor name STCQUE).		
The wait module scans the CSCB chain for pending commands. One of the first commands to be processed is the one that starts the job entry subsystem. This START command is contained in the master scheduler's JCL, which was interpreted before the initiator attached IEEMB860. At that time, the CSCB for this command was put in the CSCB chain. A listing of the master scheduler's JCL is in the extended description for step 10. (The symbolic name, &SSNAME is replaced by the four-character name given to the job entry subsystem at system generation time.) Figure 4-9 portrays the initiation of the job entry subsystem.	IEEVWAIT	
When processing a START command CSCB for system address space, normal START processing is bypassed and IEEVWAIT posts the EAEASWT/CHASWT ECB to inform the system address space initialization wait/post routine, IEEMB883, that master scheduler initialization has completed.		
After the job entry subsystem is initialized, the wait module allows STC and TSO LOGON to process by dequeuing from the serialization resources.		
The wait module attaches the system log task after the job entry subsystem is initialized. The log task initializes itself by setting up the data areas necessary to write in the log data set. It verifies the log parameter values from SYS1.PARMLIB and stores them in the log control table. The log task notifies the system operator if log initialization fails and tells him whether or not the log is active. The log is initialized not only	IEEMB803	

Extended Description	Module	Label
at IPL time, but also in response to a WRITELOG START command. A detailed description appears under the topic "System Log Initialization" in the command processing section of *System Logic Library*. Once the log task is initialized, it notifies the communications task. If the hardcopy record is assigned to the system log data set, the communications task can now write the initialization hardcopy record in the system log data set.		
Recovery Processing		
IEEVIPL runs under an ESTAE environment which passes control to an exit routine if IEEVIPL fails or abnormally terminates. This exit routine performs the following functions depending on whether the communications task is initialized or not: • Issue an SVC dump, if necessary. • Communications task not initialized: put the system in a permanent wait state. – Wait state code X'008' indicates that a dump was taken. – Wait state code X'00D' indicates that a dump was not taken. • Communications task initialized: issue message IEE479W to inform the operator of the problem and of the dump status (dump taken or not taken) and put only the master scheduler task in a wait state.	IEEVIPL	IPSTAR
IEEMB860 also runs under an ESTAE environment. Its exit routine performs functions similar to those of IEEVIPL's exit routine: • Issue an SVC dump, if necessary. • Issue message IEE479W to inform the operator of the problem and of the dump status (dump taken or not taken). • Put the master scheduler task in a wait state.	IEEMB860	STAE00 STAE10
The ESTAE exit routine for IEEVWAIT is completely described in the diagram "Master Scheduler Wait Recovery and Retry" in the publication *System Logic Library*. IEEVWAIT's exit routine performs processing and, in certain cases, attempts a retry of master scheduler wait.	IEEVWAIT	STAE0000

Diagram 76. Communications Task Address Space Create (IEAVN700) Part 1 of 2

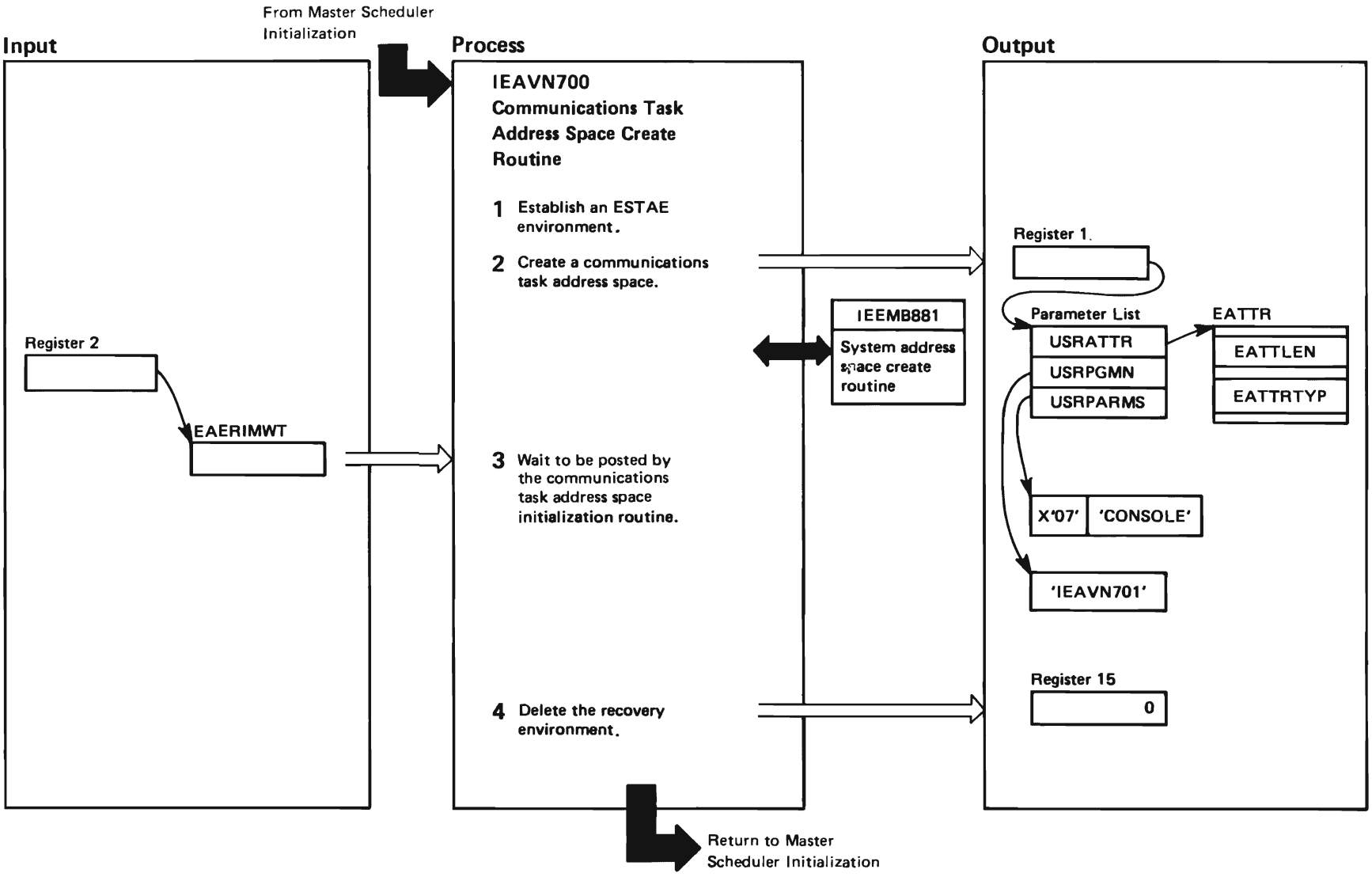


Diagram 76. Communications Task Address Space Create (IEAVN700) Part 2 of 2

Extended Description	Module	Label
This module invokes system address space initialization to create the communications task address space. The module waits for the communications task address space to be initialized before returning to its caller.		
1 IEAVN700 initializes a user parameter list and issues an ESTAE macro instruction to establish ESTAE700 as the ESTAE exit routine. If the ESTAE is not successful, it returns a non-zero return code in register 15, and control is passed to ESTAE700.	IEAVN700	SETESTAE
2 Set the attribute list, EATTR to indicate the type of address space to be created (via a link to IEEMB881). The attributes are: • Executable, high priority • Non-swappable • Has no special requirements for preferred storage • Cannot be canceled • Cannot be forced • Cannot be terminated • Wait state is loaded by RTM on DAT error In addition, set USRPGMN to point to the name of the initialization routine (IEAVN701) and set USRPARMS to point to the name of the address space (CONSOLE). If IEEMB881 returns a non-zero return code in register 15, control passes to the recovery routine, ESTAE700.	IEEMB881	

Extended Description	Module	Label
3 During its processing, IEEMB881 invokes the memory creation process and passes control to started task control processing (IEEPRWI2). Started task control processing in turn passes control to the communications task address space initialization routine (IEAVN701). IEAVN701 uses the EAERIMWT field to issue a cross memory post for IEAVN700.	IEEPRWI2 IEAVN701	
4 Cancel the ESTAE environment and return to the caller.		
Recovery Processing		
The recovery routine (ESTAE700): • Establishes addressability using the registers saved in the user parameter list • If a SDWA exists, records the appropriate diagnostic information in the variable recording area of the SDWA • Obtains the local lock, takes an SDUMP, and releases the local lock • Requests a retry	IEAVN700	ESTAE700
The retry routine calls IGFPTERM to: • Issue message IEA366W to an active console • Write a record in SYS1.LOGREC • Terminate the initialization process with a X'201' wait state	IEAVN700 IGFPTERM	RETRY700

Diagram 77. Communications Task Address Space Initialization (IEAVN701) Part 1 of 4

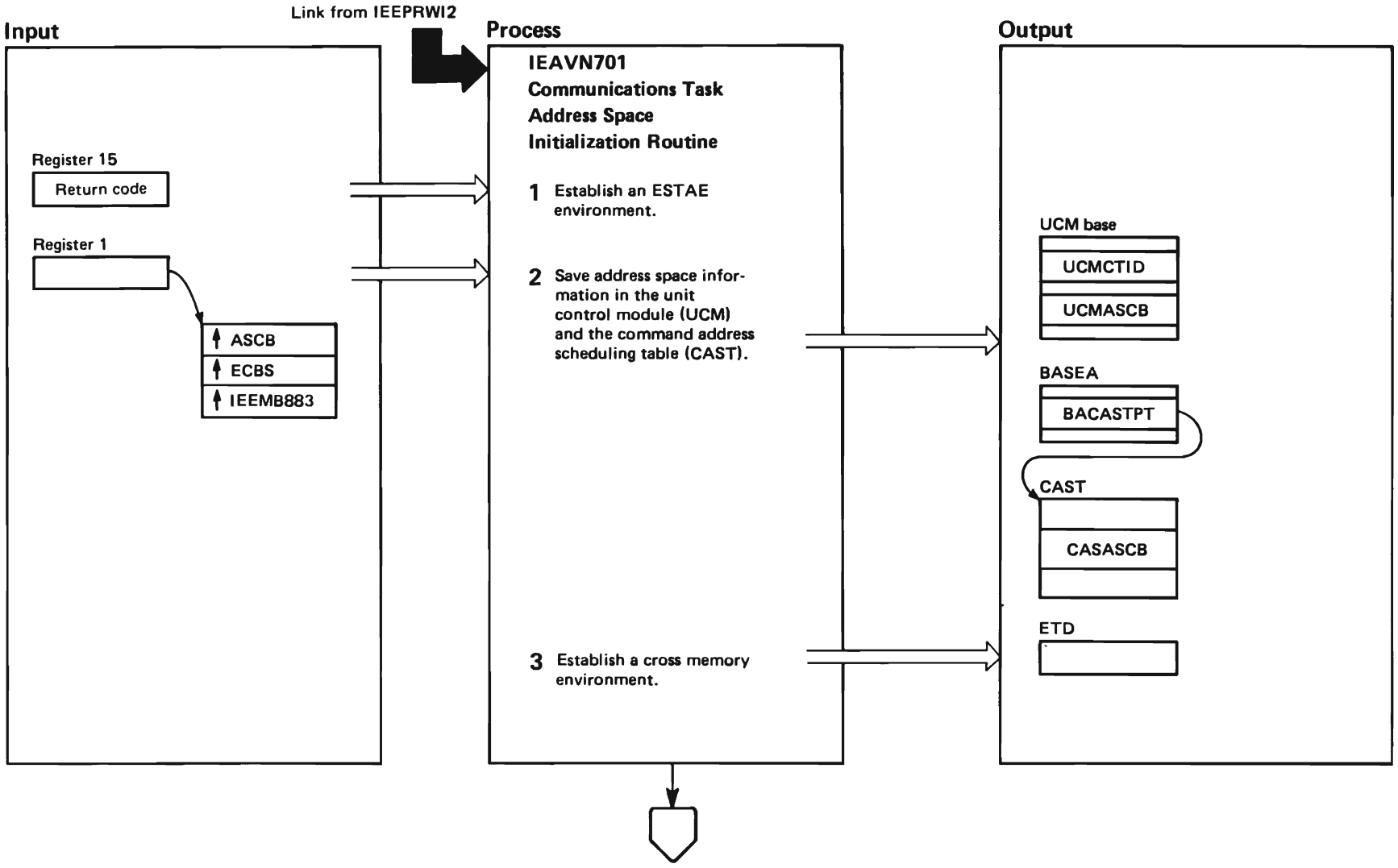


Diagram 77. Communications Task Address Space Initialization (IEAVN701) Part 2 of 4

Extended Description	Module	Label
This module initializes the communications task address space as follows: • Sets up a cross memory environment • Calls the communications task initialization routine (IEAVVINT) • Attaches the various tasks for communications task processing • Initializes the action message retention facility		
1 IEAVN701 initializes a user parameter list and issues an ESTAE macro instruction to establish ESTAE701 as the ESTAE exit routine. If the ESTAE is not successful, it returns a non-zero return code in register 15, and control is passed to ESTAE700.	IEAVN701	SETESTAE
2 IEAVN701 stores the ASCB address and the ASID in the respective unit control module (UCM) fields UCMASCB and UCMCTID. It also stores the ASCB address in the communications task entry of the command address scheduling table (CAST). If it cannot update the CAST, it sets an error condition and passes control to the recovery routine, ESTAE701.		TOP
3 IEAVN701 sets the program call authorization index to 1 and constructs and connects an entry table for the WQE service routine (IEAVH600).		SETXMEM

Diagram 77. Communications Task Address Space Initialization (IEAVN701) Part 3 of 4

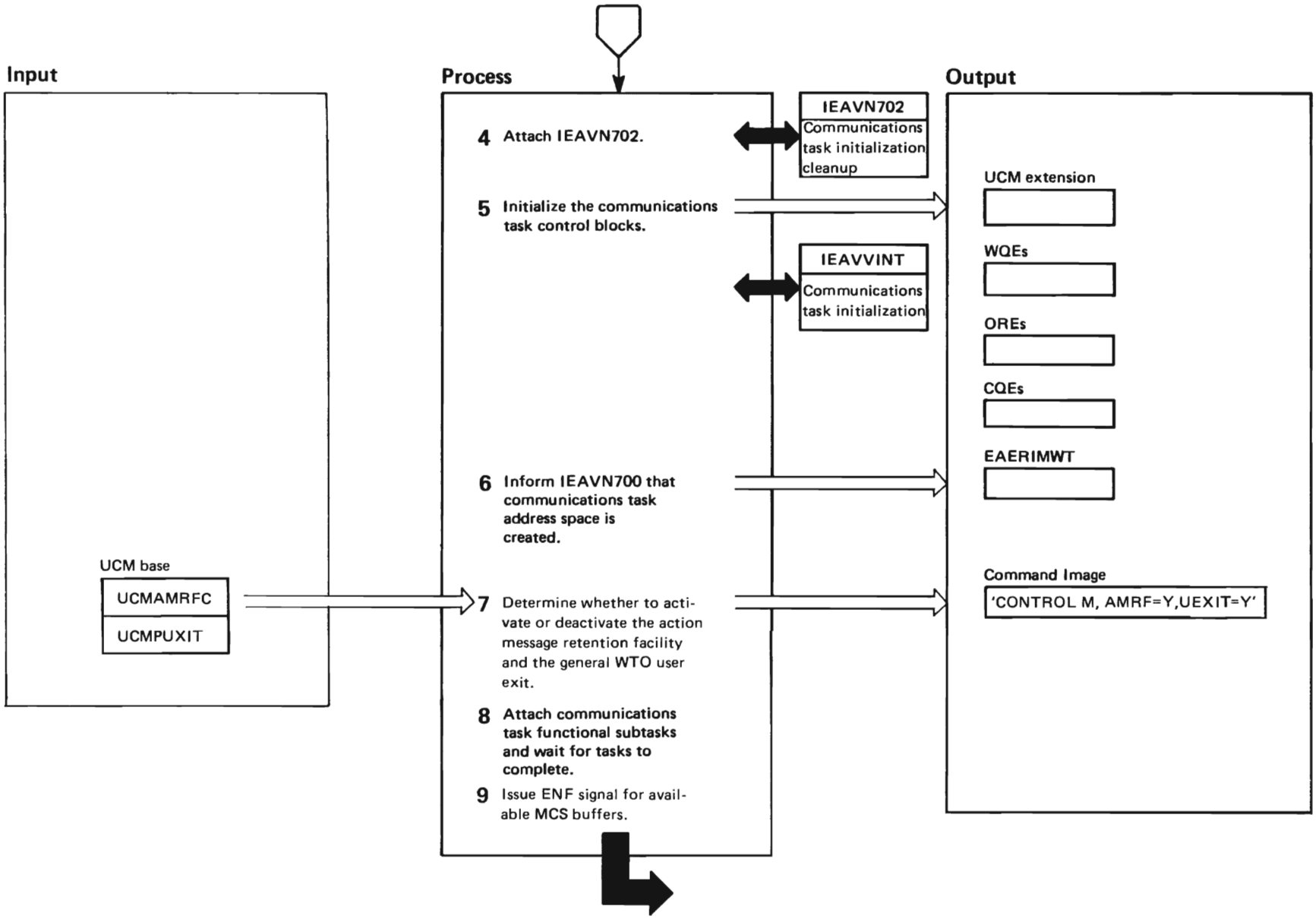


Diagram 77. Communications Task Address Space Initialization (IEAVN701) Part 4 of 4

Extended Description	Module	Label
4 IEAVN701 attaches the communications task address space initialization cleanup routine (IEAVN702) passing it the address of the system address space initialization WAIT/POST routine (IEEMB883). IEAVN702 invokes IEEMB883 and waits for master scheduler initialization to complete.	IEAVN702	
5 IEAVN701 links to the communications task initialization routine (IEAVVINT). IEAVVINT initializes the communications task's control blocks, creates the allocation control blocks required to open all the consoles, and determines the entry points of the modules the communications task uses.	IEAVVINT	
6 Upon successful initialization of the address space, IEAVN701 issues a cross memory post to IEAVN700 by means of EAERIMWT. If the post is not successful, control passes to the recovery routine, ESTAE701.		
7 IEAVN701 issues an internal command to activate or deactivate the action message retention facility and to activate or deactivate the user exit (IEAVMXIT) based on UCMAMRFC and UCMPUXIT fields in the UCM base. The fields are set according to the AMRF and UEXIT options in the CONSOLxx SYS1.PARMLIB member.	EAVN701	AMRFINIT
8 IEAVN701 attaches IEAVMQWR, the communications task wait and router routine, and IEEVWAIT, the command scheduler wait routine. IEAVN701 waits for each task to complete. If a task fails, IEAVN701 detaches the failing TCB and also clears the corresponding ECB. If IEAVMQWR fails, IEAVN701 re-attaches it. If IEEVWAIT fails, and an external restart for IEEVWAIT is not in process, IEAVN701 re-attaches IEEVWAIT. If an external restart is in progress, IEAVN701 does not re-attach IEEVWAIT; instead it issues message IEE365E.		ISSWAIT VMQWREND VWAITEND
9 If the IEAVMQUR Communication Task subtask has ended, IEAVN701 check bit UCMSDS1D in the UCM prefix. If UCMSCS1D is zero, meaning that MCS buffers are available, IEAVN701 issues an asynchronous ENF signal with an event code of ENFPC0113 and a qualifier code of '00000002'X.		

Extended Description	Module	Label
Recovery Processing		
The recovery routine (ESTAE701): • If a SDWA exists, records the appropriate diagnostic information in the variable recording area of the SDWA • Obtains the local lock, takes an SDUMP, and releases the local lock • Requests a retry		ESTAE701
The retry routine (RETRY701) calls IEEVLDWT. IEEVLDWT issues message IEA367A if the master console on its alternate is an active display console, or loads a restartable wait state code of X'202.'	IAAVN701 IEEVLDWT	RETRY701
The retry routine then waits on an ECB that is never posted and thus allows the system to continue processing even though the communications task is not functioning.		

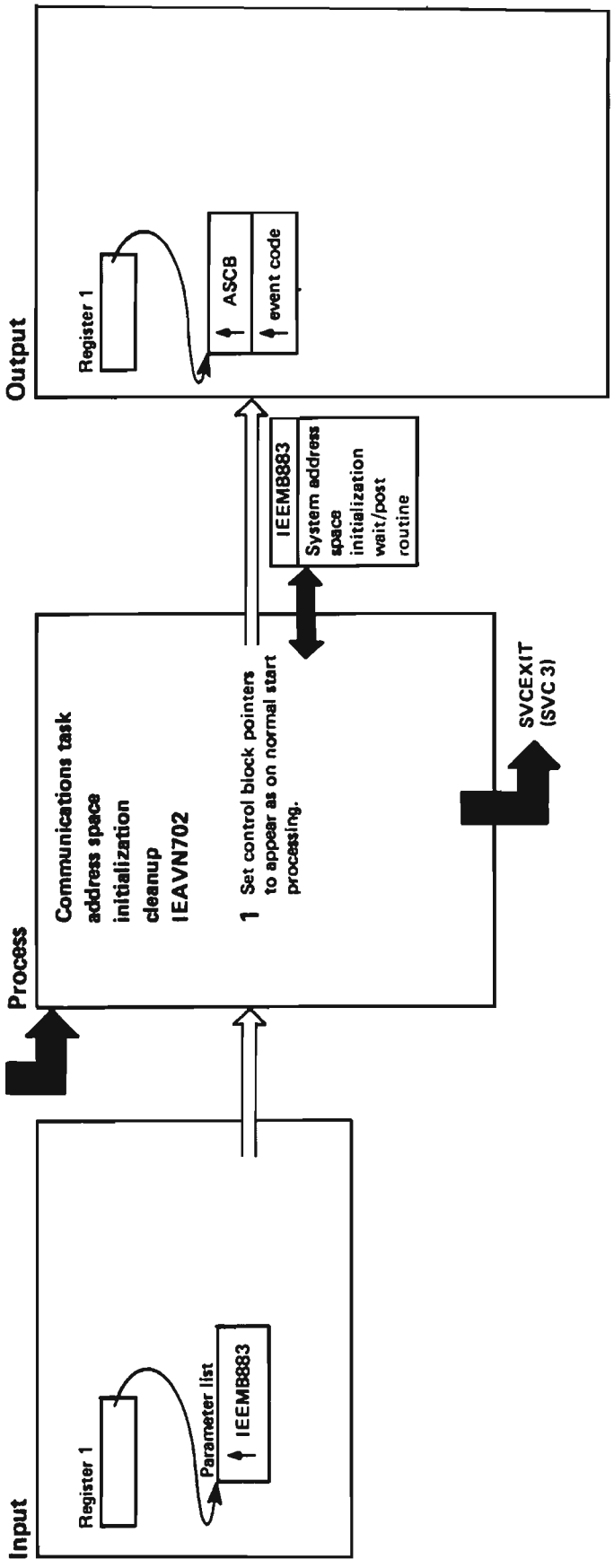


Diagram 78. Communications Task Address Space Initialization Cleanup (IEAVN702) Part 1 of 2

Diagram 78. Communications Task Address Space Initialization Cleanup (IEAVN702) Part 2 of 2

Extended Description	Module	Label
The communications task initialization cleanup routine (IEAVN702) is called by IEAVN701.		
1 IEAVN702 creates a two-word parameter list, pointed to by register 1, that it passes to the system address space initialization WAIT/POST routine (IEEMB883). The first word contains the address of the communications task address space ASCB; the second word contains the address of the master scheduler event code.	IEAVN702	
IEAVN701 calls IEEMB883 to wait for master scheduler initialization to complete. On return from IEEMB883, control returns to the dispatcher, by means of SVC EXIT (SVC 3).	IEEMB883	

Diagram 79. Communications Task Initialization (IEAVVINT) Part 1 of 21

```
IEAVVINT - MODULE DESCRIPTION


DESCRIPTIVE NAME: COMMUNICATIONS TASK INITIALIZATION ROUTINE

FUNCTION:
              IEAVVINT BUILDS THE VARIOUS ALLOCATION CONTROL BLOCKS
   NEEDED BY OPEN (DSAB, QDB, AND TIOT), INITIALIZES THE VARIOUS
   ADDRESS FIELDS IN THE SYSTEM (CVT, JSCB, SCVT, AND
   TCB), NOTIFIES COMTASK IF SYSLOG IS NOT THE HARDCOPY DEVICE,
   SETS UP THE INITIAL CELL POOLS FOR THE WQE'S, ORE'S, AND
   CQE'S. ACQUIRES AND INITIALIZES THE UCM FIXED AND PAGEABLE
   EXTENSION BASES AND THE FIXED AND PAGEABLE UCME
   EXTENSIONS. IT BUILDS A SUBSYSTEM INTERFACE BLOCK(SSIB)
   FOR COMTASK, GETMAINS AND INITIALIZES A WORK AREA FOR
   COMTASK'S USAGE OF MASTER TRACE AND GETMAINS A WORK AREA
   FOR IEAVH600. IT ALSO INITIALIZES THE CONSOLE ENTRY OF
   THE ATTENTION TABLE TO CONTAIN THE COMM TASK ADDRESS
   SPACE ASID,
   AND PASSES CONTROL TO THE DIDOCS (DISPLAY CONSOLE SUPPORT)
   INITIALIZATION ROUTINE.

ENTRY POINT: IEAVVINT

   PURPOSE:
              ENTERED WITH PROTECT KEY ZERO AND SUPERVISOR STATE
      DURING COMM TASK INITIALIZATION TIME TO INITIALIZE
      THE COMMUNICATION TASK'S CONTROL BLOCKS.

   LINKAGE:
              LINKED TO--   LINK  EP=IEAVVINT
      ONLY CALLER IS COMMUNICATION TASK ADDRESS SPACE
      INITIALIZATION, IEAVN701.

   CALLERS: None

   INPUT:
             AS FOLLOWS:
      CVTCUCB = ADDRESS OF UCM
      REG 13 = ADDRESS OF A REGISTER SAVE AREA
      REG 14 = RETURN ADDRESS
      REG 15 = ENTRY POINT ADDRESS

   OUTPUT: UPDATED COMM TASK CONTROL BLOCKS.

   EXIT NORMAL: VINTEXIT

      OUTPUT:
                  THE FOLLOWING CONTROL BLOCKS HAVE FIELDS
                  WHICH ARE INITIALIZED BY THIS MODULE:

                   DSAB                  SCVT
                   JSCB                  TCB
                                         TIOT
                   QDB                   SSIB
                                         UCM
                                         GENX
         NOTE:  THE DSAB'S, TIOT, AND QDB
                ARE CREATED BY THIS MODULE

EXTERNAL REFERENCES:

   ROUTINES: IEECVGCI, TO INITIALIZE FOR DIDOCS.
```


Diagram 79. Communications Task Initialization (IEAVVINT) Part 2 of 21

CONTROL BLOCKS:
AS FOLLOWS:

NAME	MAPPING MACRO	REFERENCE
AMRQ	IHACTM	R
ASVT	IHAASVT	R
CQE	IHACTM	R
CVT	IHACVT	R
DSAB	IHADSAB	C
HCL	IHAHCLOG	R
IOCOM	IECDCIOM	R
JSCB	IEZJSCB	W
ORE	IHAORE	R
PSA	IHAPSA	R
QDB	IHAQDB	C
SCVT	IHASCVT	RW
SSIB	IEFJSSIB	C
TCB	IKJTCB	RW
TIOT	IEFTIOT1	C
UCB	IEFUCBOB	R
UCM	IEECUCM	RW
WQE	IHAWQE	R

KEY = R-READ, C-CREATE, W-WRITE

TABLES:
SEE DESCRIPTION ABOVE.
IEZMTPRM, IEETRACE PARMLIST MAPPING MACRO
ATTENTION TABLE, ENTRIES MAPPED BY IECDATB

SERIALIZATION:
THE LOCAL LOCK IS OBTAINED AND HELD ACROSS THE BRANCH ENTRY TO GETMAIN.

Diagram 79. Communications Task Initialization (IEAVVINT) Part 3 of 21

IEAVVINT - MODULE OPERATION

THIS MODULE BEGINS BY BUILDING THE CONTROL BLOCK

CHAIN OF DSAB'S AND TIOT DD ENTRIES THAT WILL LATER BE USED
WHEN THE CONSOLE DEVICES ARE
OPENED. THE STORAGE FOR THE DSAB'S, QDB, AND TIOT ARE GOTTEN
FROM SP 230. A BRANCH ENTRY TO GETMAIN IS USED TO OBTAIN
THE STORAGE IN THE SCHEDULER'S STORAGE PROTECT KEY (LOCAL
LOCK IS HELD ACROSS THIS GETMAIN) SO THE BLOCKS WILL HAVE THE
SAME KEY AS THEY WOULD BE IF BUILT BY THE SCHEDULER.
THE AMOUNT OF STORAGE TO BE GOTTEN IS CALCULATED FROM THE
NUMBER OF CONSOLES IN THE SYSTEM.
WHEN THE COMPLETE CHAIN IS BUILT,
THE ADDRESS OF THE QDB IS PLACED IN THE JFCB
AND THE ADDRESS OF THE TIOT IN THIS
TASK'S TCB.

AN SSIB IS BUILT FROM STORAGE OBTAINED FROM SP 241, AND
THE BLOCK IS QUEUED OFF THE JSCB AND THE UCM.

THE MODULE NEXT USES THE LOAD MACRO TO RESOLVE ADDRESSES OF
ROUTINES IN LPA AND LINKLIB. ENTRY POINT ADDRESSES ARE
PLACED IN THE SCVT AND UCM. AFTER THE ENTRY POINTS
ARE RESOLVED, DELETE'S ARE ISSUED FOR THE LPA ROUTINES.

NEXT, THE CELL POOLS FOR WQE'S, ORE'S, AND CQE'S ARE
BUILT. THE WQE AND CQE STORAGE SIZE IS DETERMINED BY THE
NUMBER OF SYSGENED CONSOLES. (THE MINIMUM-MAXIMUM NUMBER
OF WQES IS 20-2000.(20 IS THE WTOBUF MINIMUM LIMIT.)
THE MINIMUM-MAXIMUM NUMBER OF CQES IS 10-295).
THE STORAGE FOR THE ORE'S IS OBTAINED FROM ABOVE 16 MB.

THE ECB LIST THAT IEAVMQWR
WAITS ON IN A NO-CONSOLES CONDITION IS INITIALIZED.
THE COMM TASK ENF LISTEN EXIT ROUTINE(IEAVG600) IS
LOADED AND THE ADDRESS STORED IN THE FIXED EXTENSION.

CALCULATE 60 AND 80 PERCENT VALUES OF THE ORE LIMIT
SPECIFIED AT PARMLIB MEMBER CONSOLXX.
STORE THESE NUMBERS IN UCMF60OR AND UCMF80OR.
CALCULATE 60, 80, AND 95 PERCENT VALUES OF THE WQE
LIMIT SPECIFIED IN PARMLIB MEMBER CONSOLXX.
STORE THESE NUMBERS IN UCMF60WQ, UCMF80WQ, AND UCMF95WQ.
THESE FIELDS ARE IN THE UCM FIXED EXTENSION BASE.

CALCULATE 75, 80 AND 95 VALUES OF THE NUMBER OF
MESSAGES WHICH CAN BE RETAINED BY THE ACTION MESSAGE
RETENTION FACILITY AND STORE THEM IN THE UCM FIXED
EXTENSION BASE.

ISSUE GETMAIN TO OBTAIN A WORK AREA FROM SUBPOOL 229,
TO BE USED BY IEAVH600. THE ADDRESS IS SAVED IN
UCMWQADA.

ISSUE GETMAIN TO OBTAIN A WORK AREA FROM SUBPOOL 229,
TO BE USED BY IEAVM601. THE ADDRESS IS SAVED IN
UCMFHCDA.

INITIALIZE THE CONSOLE ENTRY OF THE ATTENTION TABLE
TO CONTAIN THE COMM TASK ADDRESS SPACE (CTAS) ASID.

IF SYSLOG IS NOT THE HARDCOPY DEVICE, FIELD UCMNPECB IN
UCM IS POSTED.

NEXT TEST FOR WHETHER DISPLAY CONSOLES ARE
PRESENT, AND, IF SO LINK TO THE DIDOCS INITIALIZATION
ROUTINE.

Diagram 79. Communications Task Initialization (IEAVVINT) Part 4 of 21

IEAVVINT - DIAGNOSTIC AIDS

ENTRY POINT NAME: IEAVVINT

MESSAGES: None

ABEND CODES: None

WAIT STATE CODES: None

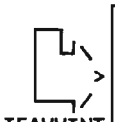
RETURN CODES: None

REGISTER CONTENTS ON ENTRY: Irrelevant

REGISTER CONTENTS ON EXIT: Irrelevant

Diagram 79. Communications Task Initialization (IEAVVINT) Part 5 of 21

IEAVVINT - COMMUNICATIONS TASK INITIALIZATION ROUTINE STEP 01



IEAVVINT BUILDS THE VARIOUS ALLOCATION CONTROL BLOCKS NEEDED BY OPEN (DSAB, QDB, AND TIOT), INITIALIZES THE VARIOUS ADDRESS FIELDS IN THE SYSTEM (CVT, JSCB, SCVT, AND TCB), NOTIFIES COMTASK IF SYSLOG IS NOT THE HARDCOPY DEVICE, SETS UP THE INITIAL CELL POOLS FOR THE WQE'S, ORE'S, AND CQE'S. ACQUIRES AND INITIALIZES THE UCM FIXED AND PAGEABLE EXTENSION BASES AND THE FIXED AND PAGEABLE UCME EXTENSIONS. IT BUILDS A SUBSYSTEM INTERFACE BLOCK(SSIB) FOR COMTASK, GETMAINS AND INITIALIZES A WORK AREA FOR COMTASK'S USAGE OF MASTER TRACE AND GETMAINS A WORK AREA FOR IEAVH600. IT ALSO INITIALIZES THE CONSOLE ENTRY OF THE ATTENTION TABLE TO CONTAIN THE COMM TASK ADDRESS SPACE ASID, AND PASSES CONTROL TO THE DIDOCS (DISPLAY CONSOLE SUPPORT) INITIALIZATION ROUTINE.

01 UCMEXTBD (A) APPROX. 079070 WQE THRESHOLD A (LOAD AND DELETE AREA) A (UCMEXTBD) D (PATCH) C (GETSPACE - SAVE REGS WHEN OBTAINING LOCKS) C (GETSPACE - SAVE REGS WHEN OBTAINING LOCKS) A (LOAD AND DELETE IEAVM700) C (GETSPACE - SAVE REGS WHERE LOCKS ARE HELD) D (IEAVVINT) INIT OF UCMCTID AND UCMASCB DELETED A (WTOSTOR) OBTAIN STORAGE FOR IEAVH600'S WORK AREA A BLDCPOOL FOR CQES FROM COMM TASK PRIVATE C BLDCPOOL FOR WQES TO GET STORAGE FROM COMM TASK PRIVATE A (LOADMODS) DELETE IEAVMFRR AND IEAVMEST AFTER LOAD C (UCMEXTBD) DON'T SET ACR ECB ADDRESS A (END) INITIALIZE COMM TASK ASID IN I/O ATTENTION TABLE D (UCMEXTBD) BUILD UCME PAGEABLE EXTENSION C (DECLARES) ADJUST AMRMINRM FOR NEW AMRQ SIZE C (BLDSSIB) CHANGE COMM TASK LIFE OF JOB SSIB FROM SUBPOOL 255 TO 241 A (UCMEXTBD) INITIALIZATION OF MESSAGE LEVEL A (LOADEXIT) LOAD THE GENERAL WTO USER EXIT (IEAVMX1\) A (GETWORK) ENLARGE THE IEETRACE WORK AREA A (LOADMODS) LOAD, SAVE THE ADDRESS AND DELETE IEECVXIT D (LOADMODS) DELETE LOAD/DELETE FOR IEAVMWQR AND IEECMENQ D (LOADMODS) DELETE THE DELETE FOR IEAVSWCH D (CONSOLCB) DELETE THE DELETE FOR DEVICE SERVICE PROCESSORS D (LOADEXIT) REMOVE LOAD OF IEAVMXIT C (UCMEXTBD) LOAD AND DELETE COMM TASK ENF LISTEN EXIT (IEAVG600). CALCULATE 95
OF THE IPL SPECIFIED WQE LIMIT. C (CELLPOOL) ISSUE GETMAIN TO OBTAIN ORE CELLPOOL C (UCMEXTBD) REMOVE GETMAIN FOR UCM FIXED EXTENSION BASE, UCME FIXED EXTENSION, UCM FIXED EXTENSION SAVE AREA, UCM PAGEABLE EXTENSION BASE AND UCME PAGEABLE EXTENSION. C (UCMEXTBD) CALCULATE 60
AND 80
OF THE SPECIFIED ORE LIMIT (THE VALUE IS SPECIFIED IN CONSOLXX) D (MAINLINE,DISPCONS) REMOVAL OF DCMLIB A (HCFSTOR) OBTAIN STORAGE FOR THE HARDCOPY DATA AREA D (GETWORK) MOVED TO IEAVM601

Diagram 79. Communications Task Initialization (IEAVVINT) Part 6 of 21

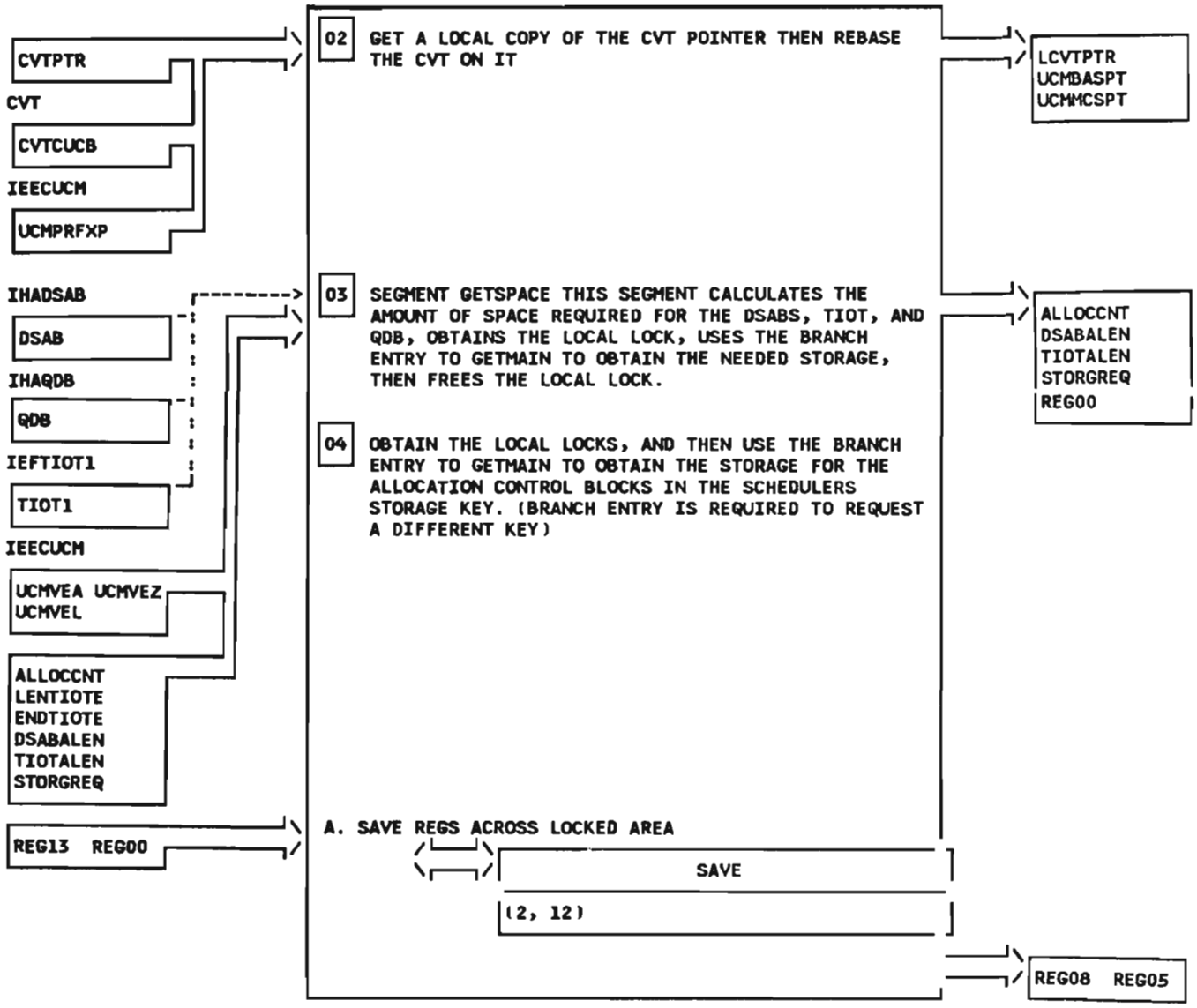


Diagram 79. Communications Task Initialization (IEAVVINT) Part 7 of 21

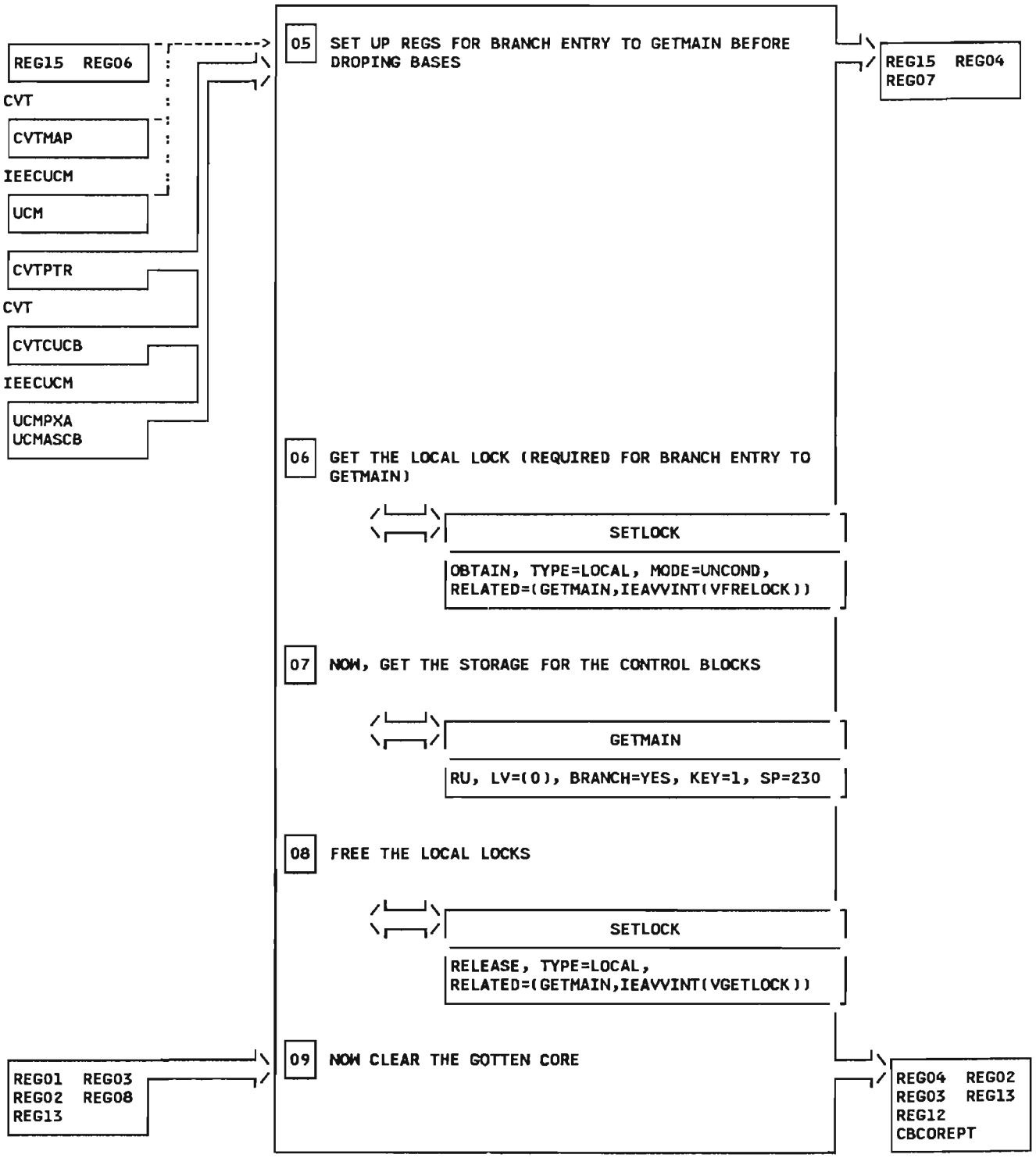


Diagram 79. Communications Task Initialization (IEAVVINT) Part 8 of 21

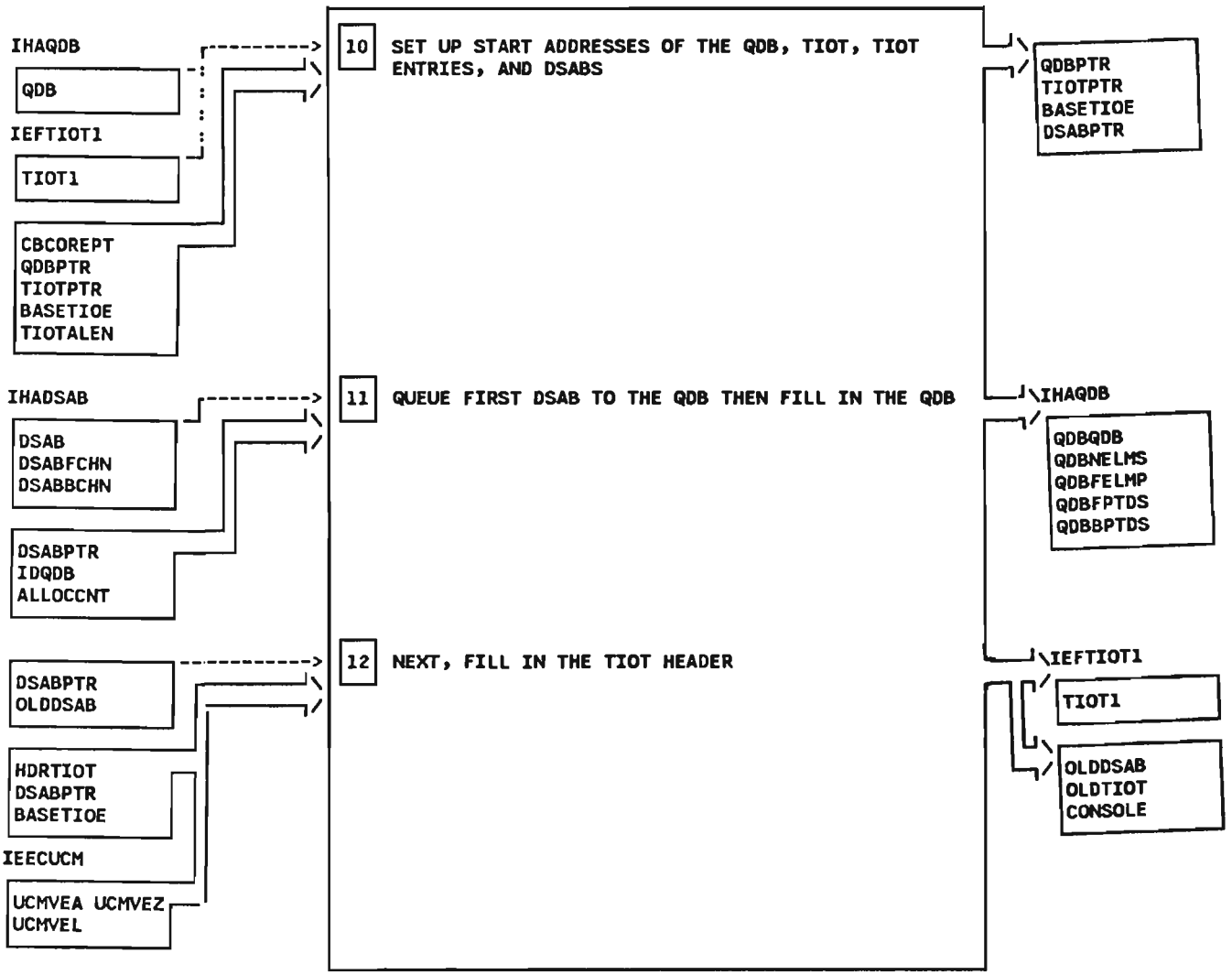


Diagram 79. Communications Task Initialization (IEAVVINT) Part 9 of 21

IEAVVINT - COMMUNICATIONS TASK INITIALIZATION ROUTINE

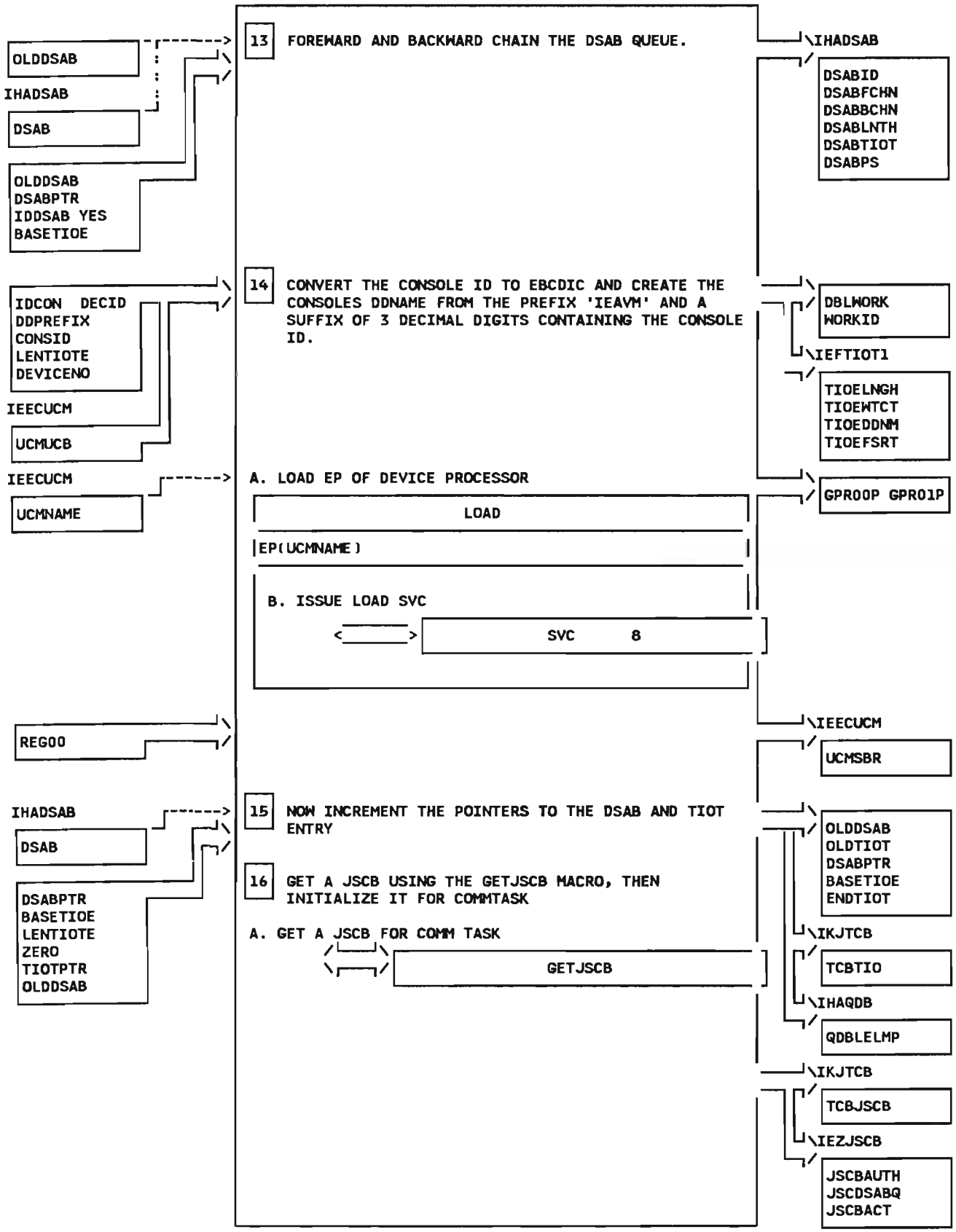


Diagram 79. Communications Task Initialization (IEAVVINT) Part 10 of 21

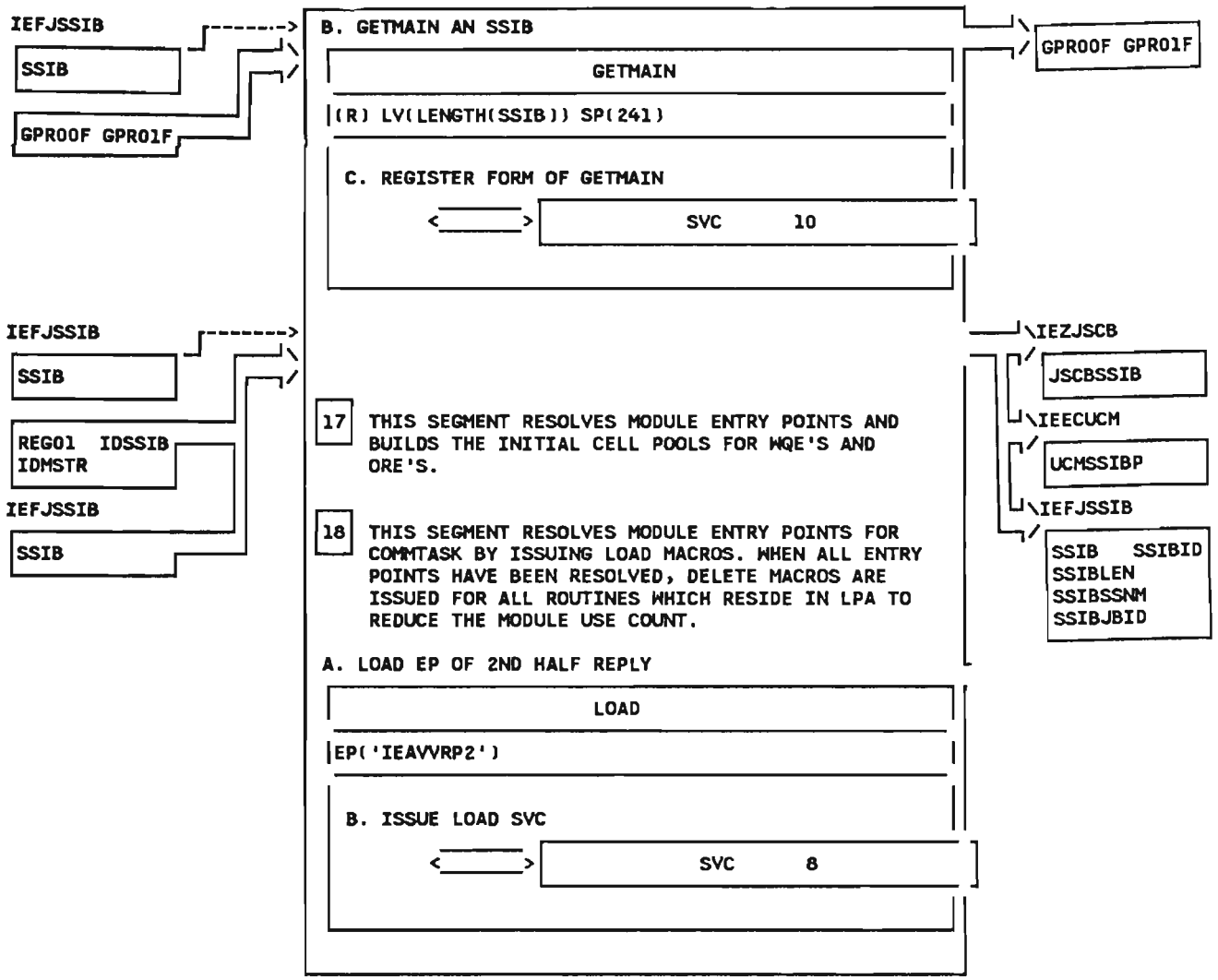


Diagram 79. Communications Task Initialization (IEAVVINT) Part 11 of 21

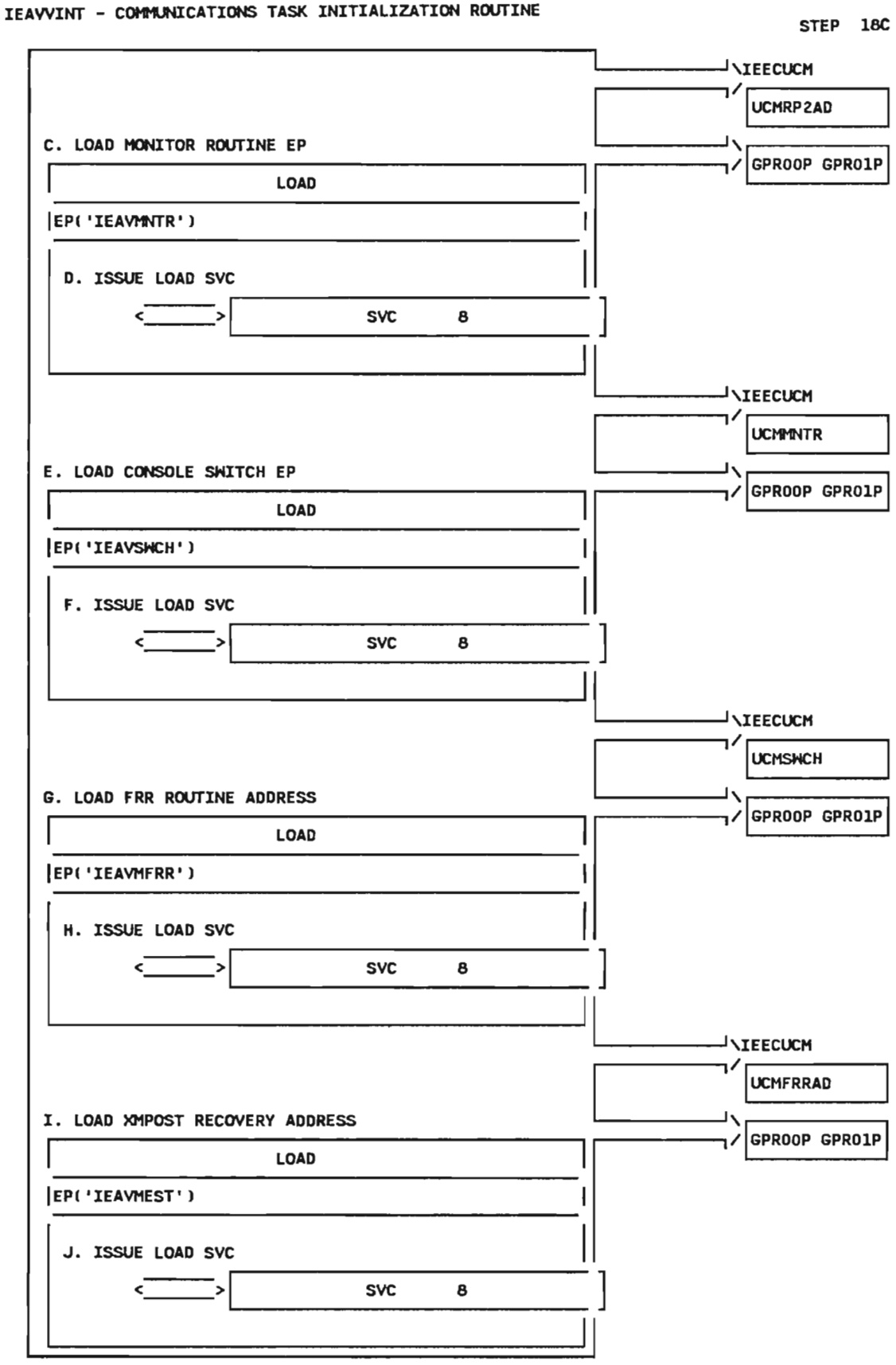


Diagram 79. Communications Task Initialization (IEAVVINT) Part 12 of 21

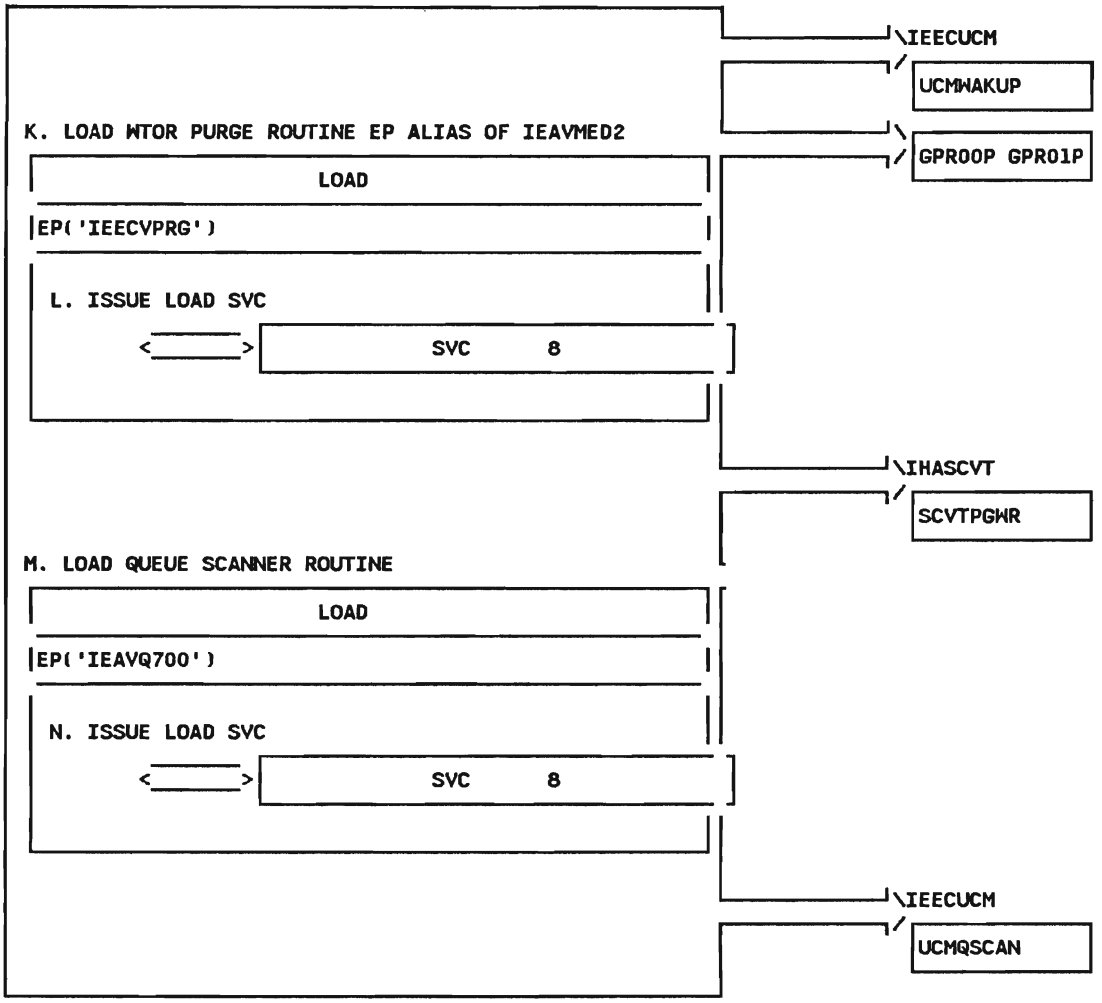


Diagram 79. Communications Task Initialization (IEAVVINT) Part 13 of 21

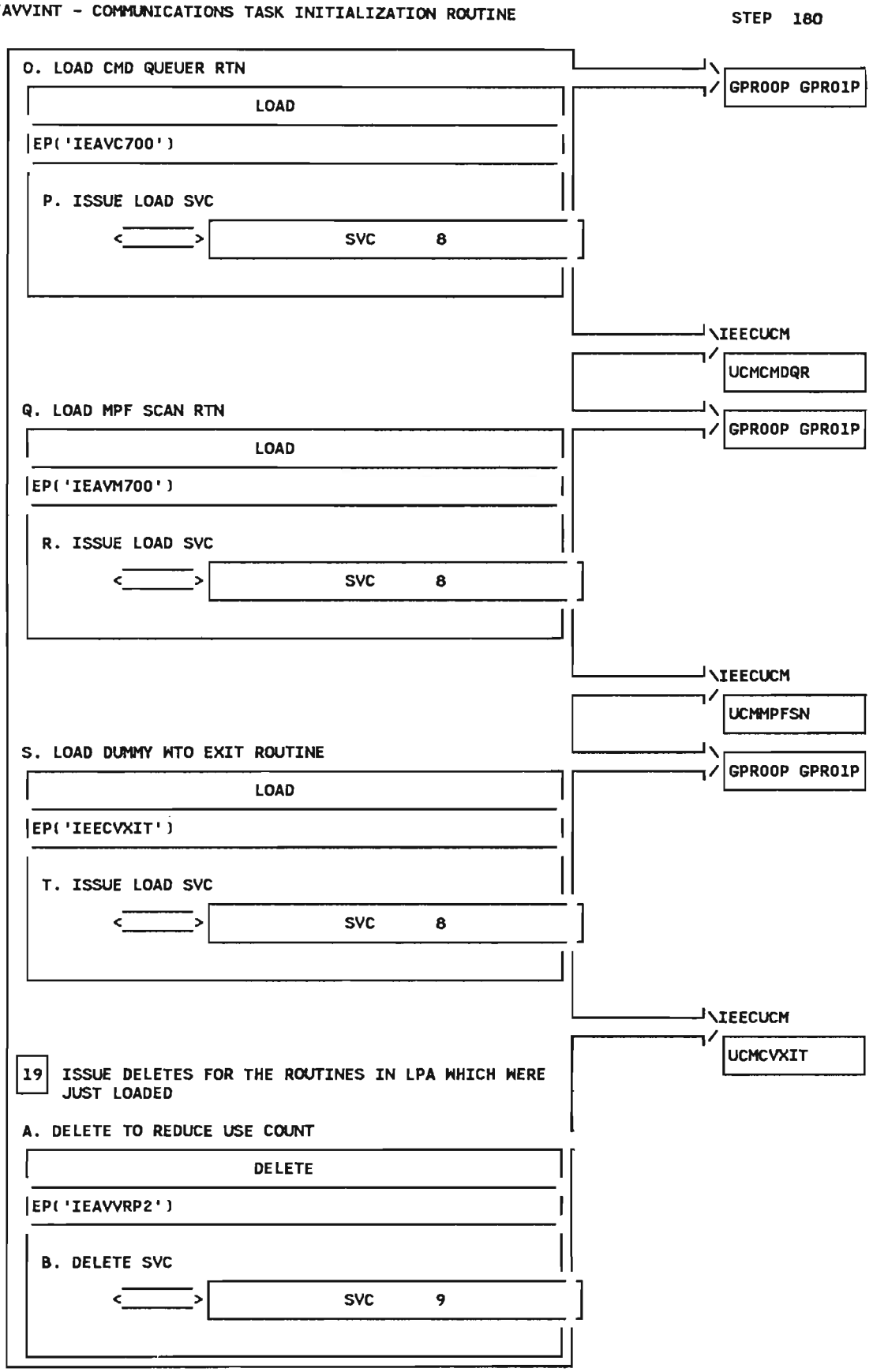


Diagram 79. Communications Task Initialization (IEAVVINT) Part 14 of 21

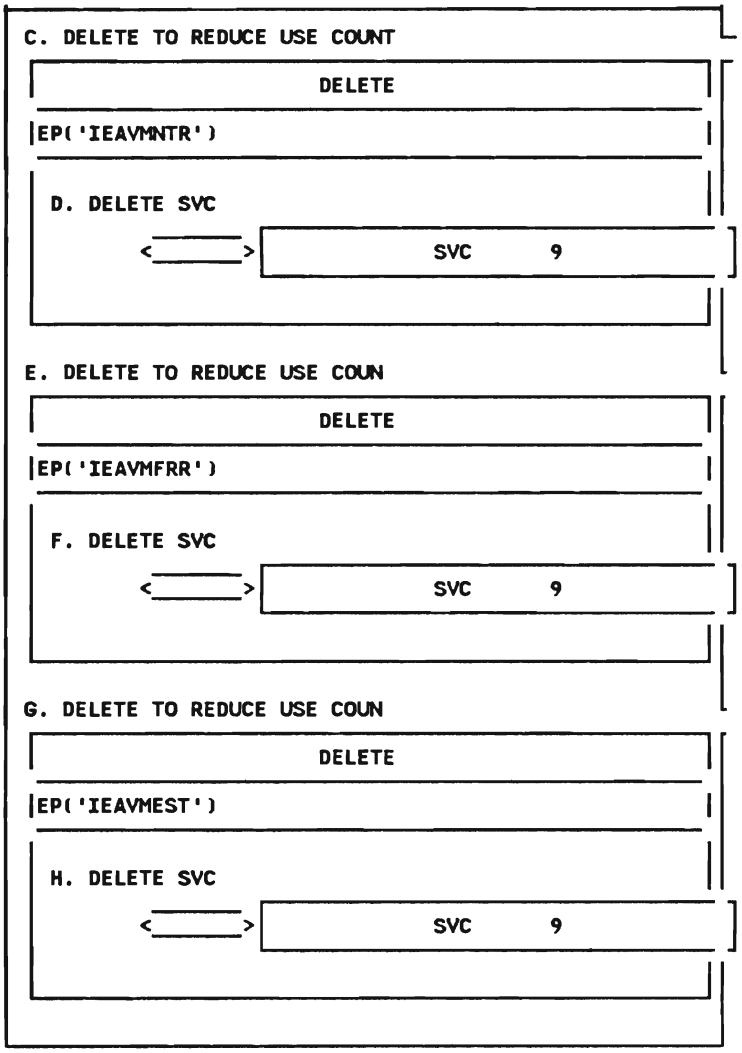


Diagram 79. Communications Task Initialization (IEAVVINT) Part 15 of 21

IEAVVINT - COMMUNICATIONS TASK INITIALIZATION ROUTINE STEP 19I

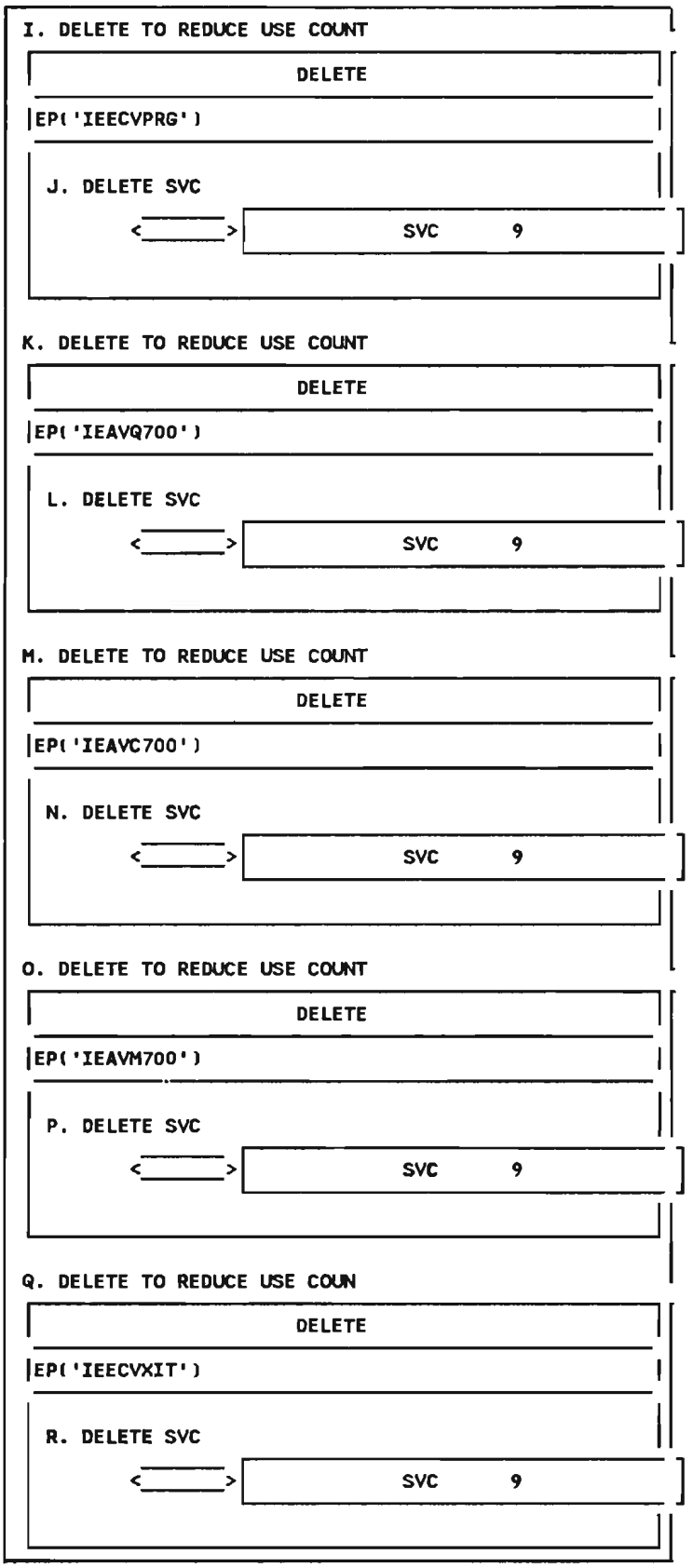


Diagram 79. Communications Task Initialization (IEAVVINT) Part 16 of 21

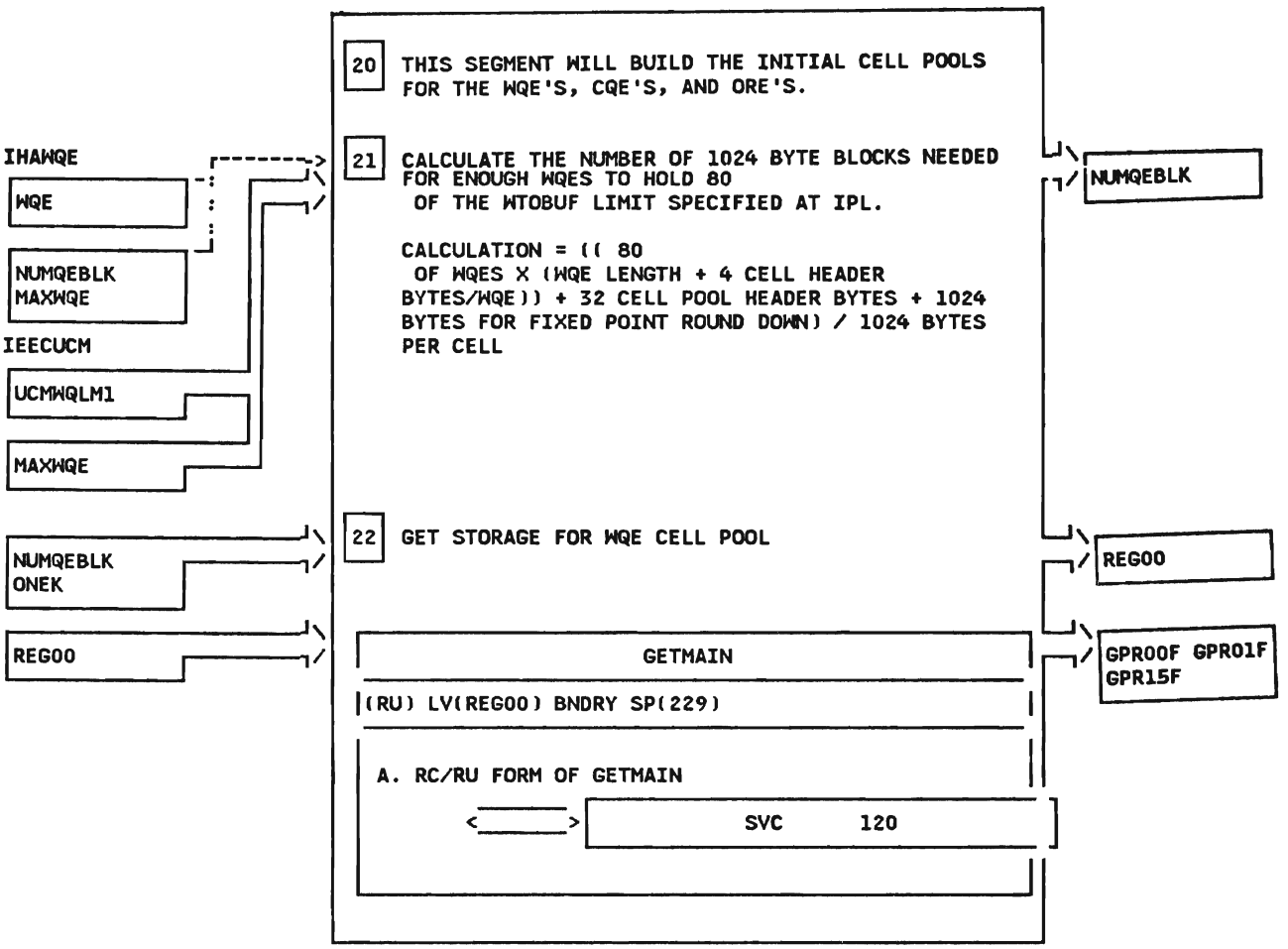


Diagram 79. Communications Task Initialization (IEAVVINT) Part 17 of 21

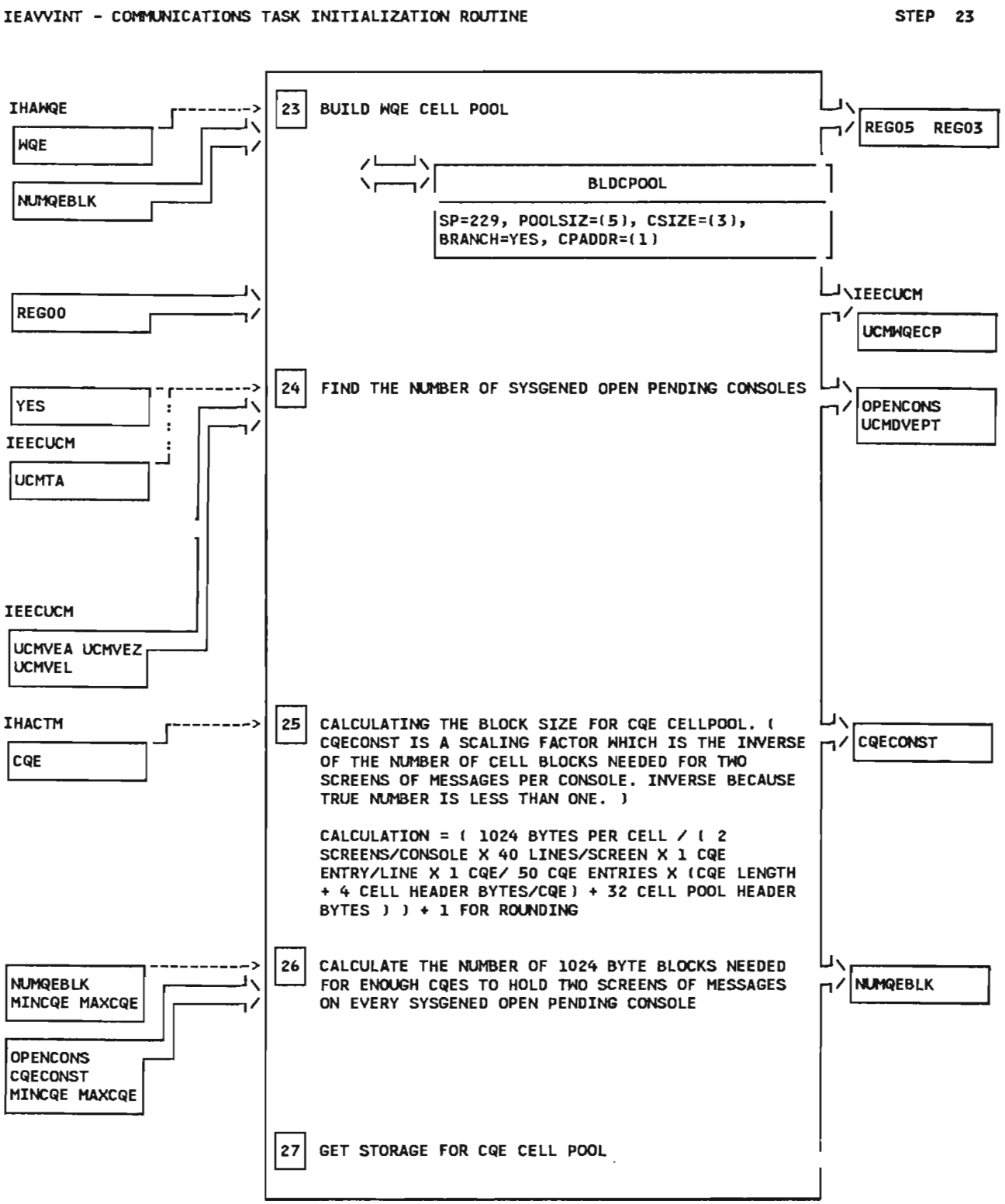


Diagram 79. Communications Task Initialization (IEAVVINT) Part 18 of 21

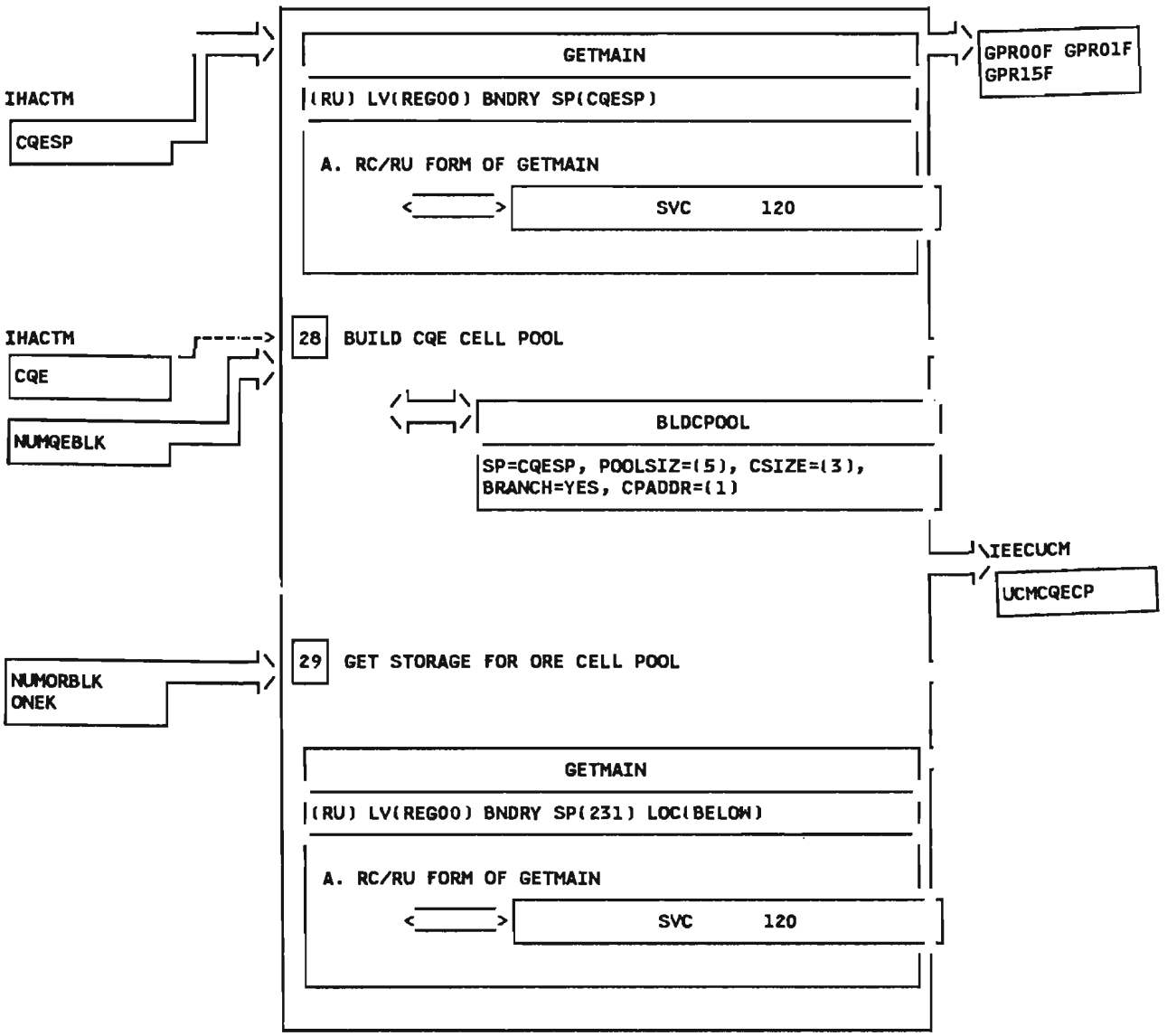


Diagram 79. Communications Task Initialization (IEAVVINT) Part 19 of 21

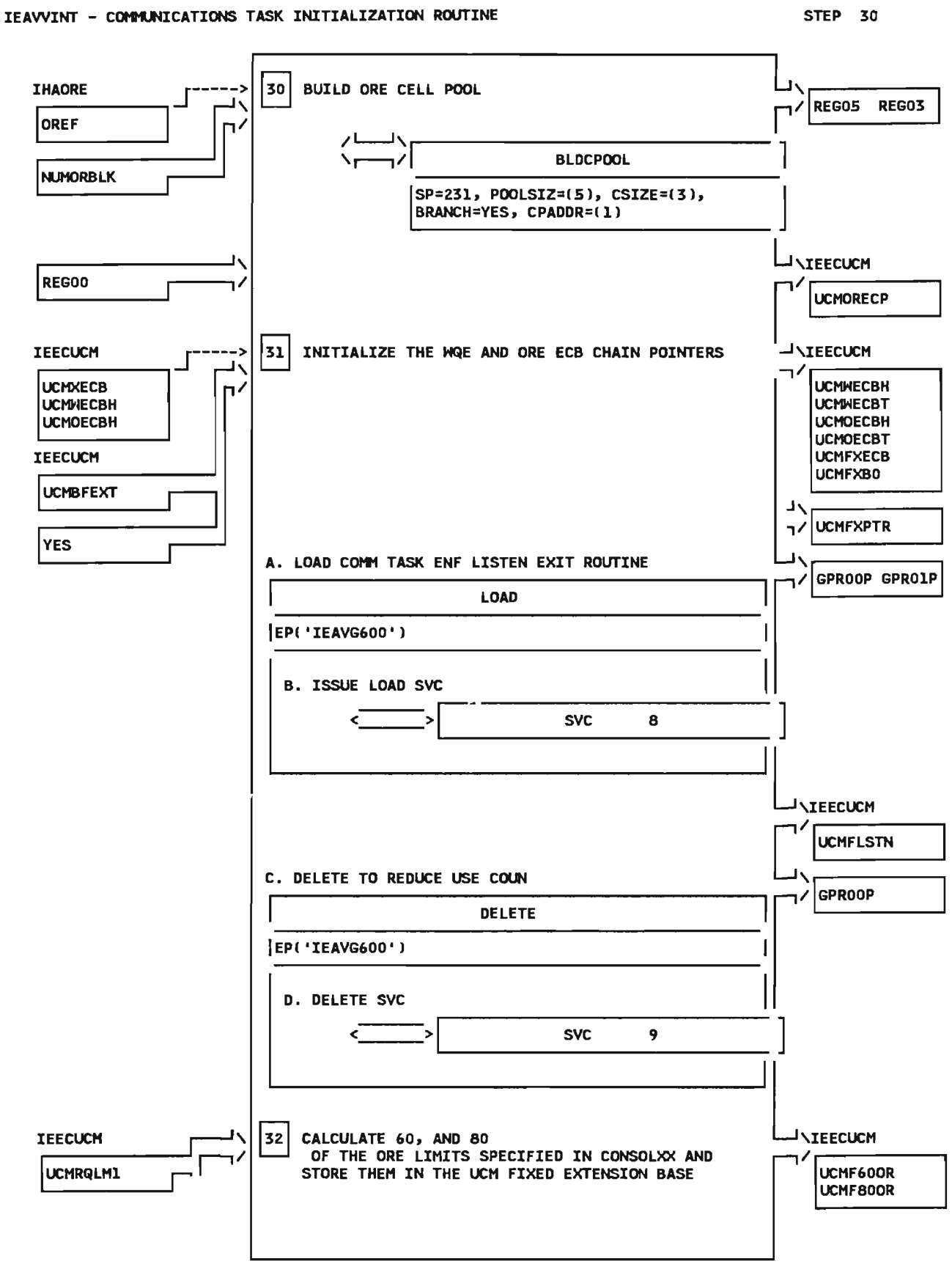


Diagram 79. Communications Task Initialization (IEAVVINT) Part 20 of 21

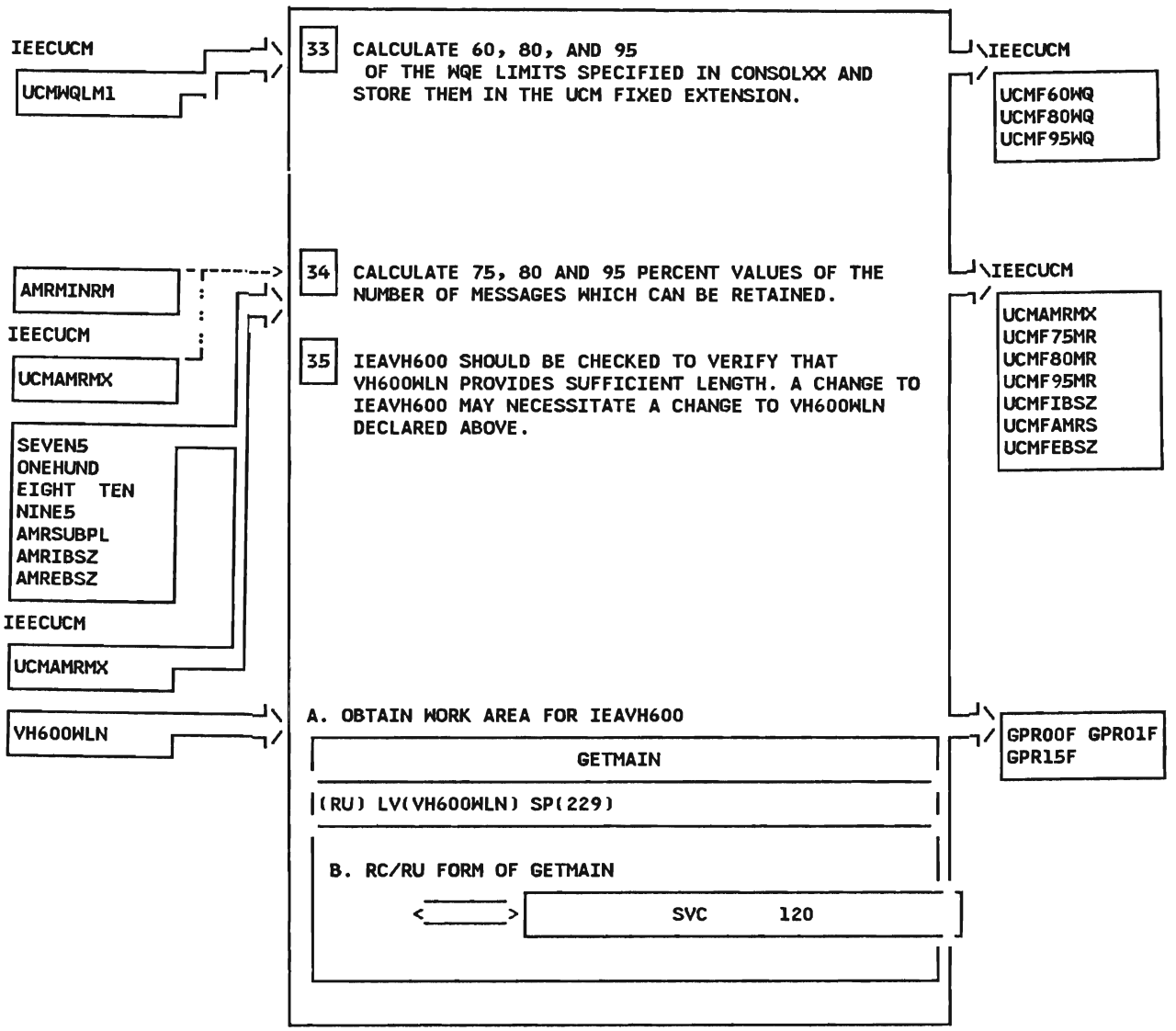


Diagram 79. Communications Task Initialization (IEAVVINT) Part 21 of 21

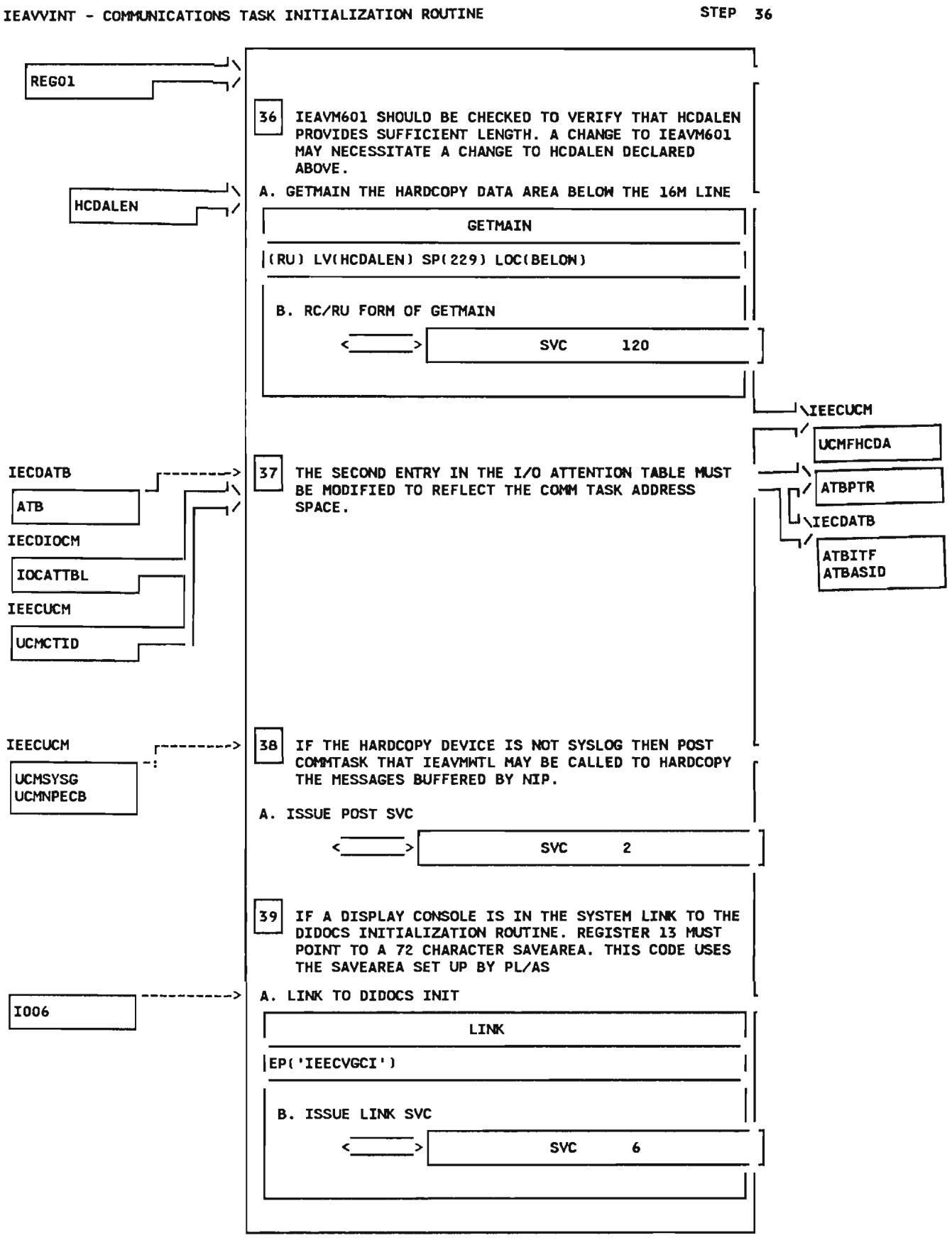


Diagram 80. Subsystem Interface Initialization (IEFJSINT) Part 1 of 6

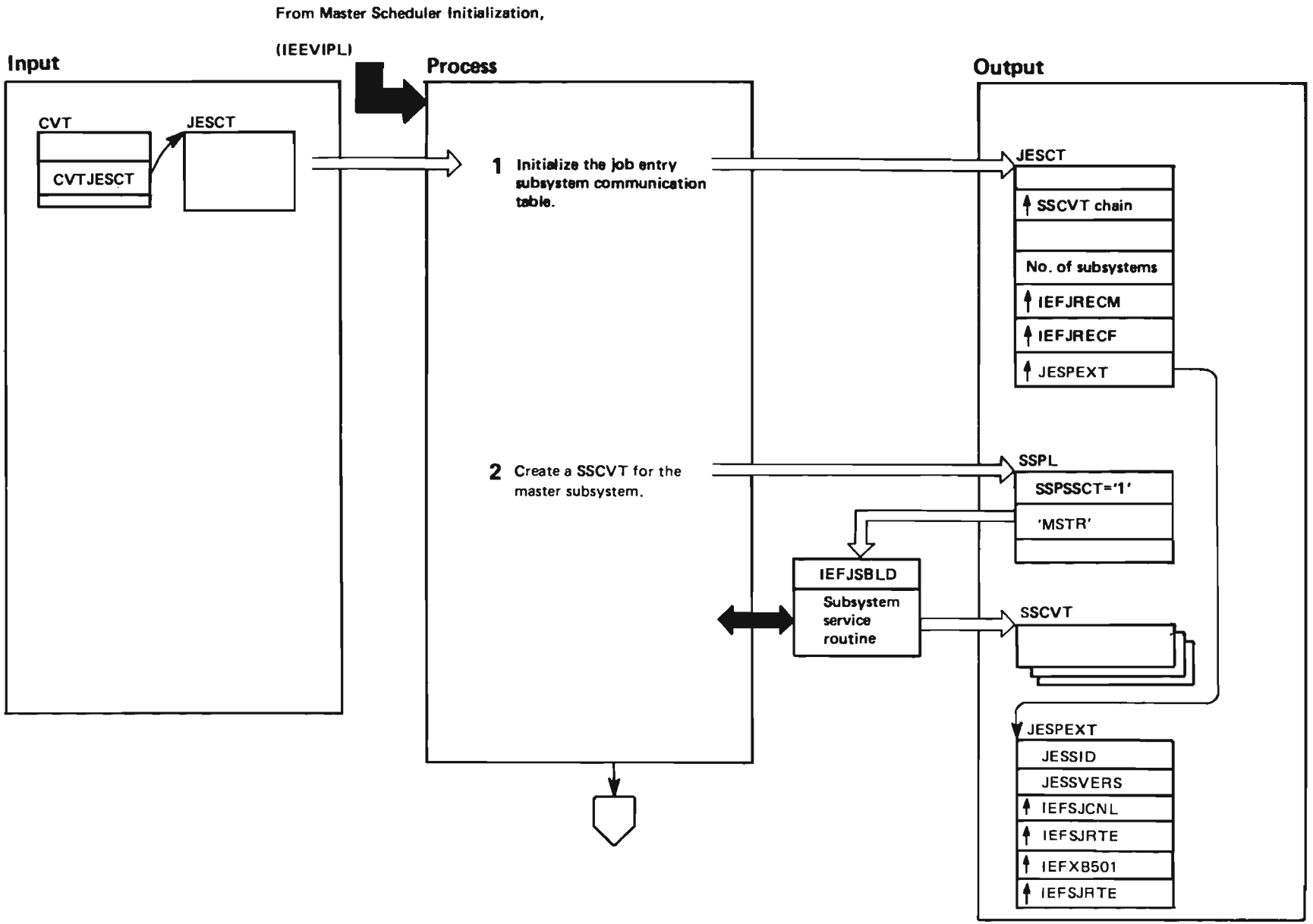


Diagram 80. Subsystem Interface Initialization (IEFJSINT) Part 2 of 6

Extended Description	Module	Label
Subsystem interface initialization (IEFJSINT) is called from master scheduler initialization (IEEVIPL) to create the control blocks that allow system routines to request services from the master subsystem. The subsystem interface uses these control blocks to identify the subsystem that will service a subsystem request. IEFJSINT builds an SSCVT for the master subsystem. Additional subsystems can be specified in the IEFSSNxx members of SYS1.PARMLIB; IEFJSIN2 (Diagram 75) creates the SSCVTs and initializes these additonal subsystems.		
1 IEFJSINT initializes fields in the job entry subsystem communication table (JESCT). In the JESCT, IEFJSINT initializes the total number of subsystems to zero, stores the pointer to the SSCVT chain in the JESCT field after the chain is built in step 2. IEFJSINT obtains storage for a pageable extension of JESCT (JESPEXT). Once storage has been obtained for the JESPEXT, IEFJSINT does the following: • Initializes the control block identifier (JESSID) of the JESPEXT. • Initializes the version number (JESSVERS) for the JESPEXT. • Loads the scheduler JCL facility control routine, IEFSJCNL, and saves its address in the JESPEXT. • Loads the scheduler JCL facility router routine, IEFSJRTE, and saves its address in the JESPEXT. • Loads the initiator message module, IEFIB651, and saves its address in the JESPEXT. IEFJSINT then does the following: • Loads the subsystem interface resource manager routine, IEFJRECM, and saves its address in the JESCT. • Loads the end-of-task entry point, IEFJRECF, of the subsystem interface resource manager routine and saves its address in the JESCT.	IEFJSINT	

Extended Description	Module	Label
2 For the master subsystem (MSTR), initialize the subsystem service routine parameter list (IEFZB601) to indicate that a SSCVT is to be built. Then call the subsystem service routine (IEFJSBLD). See Diagram 69.	IEFJSINT	

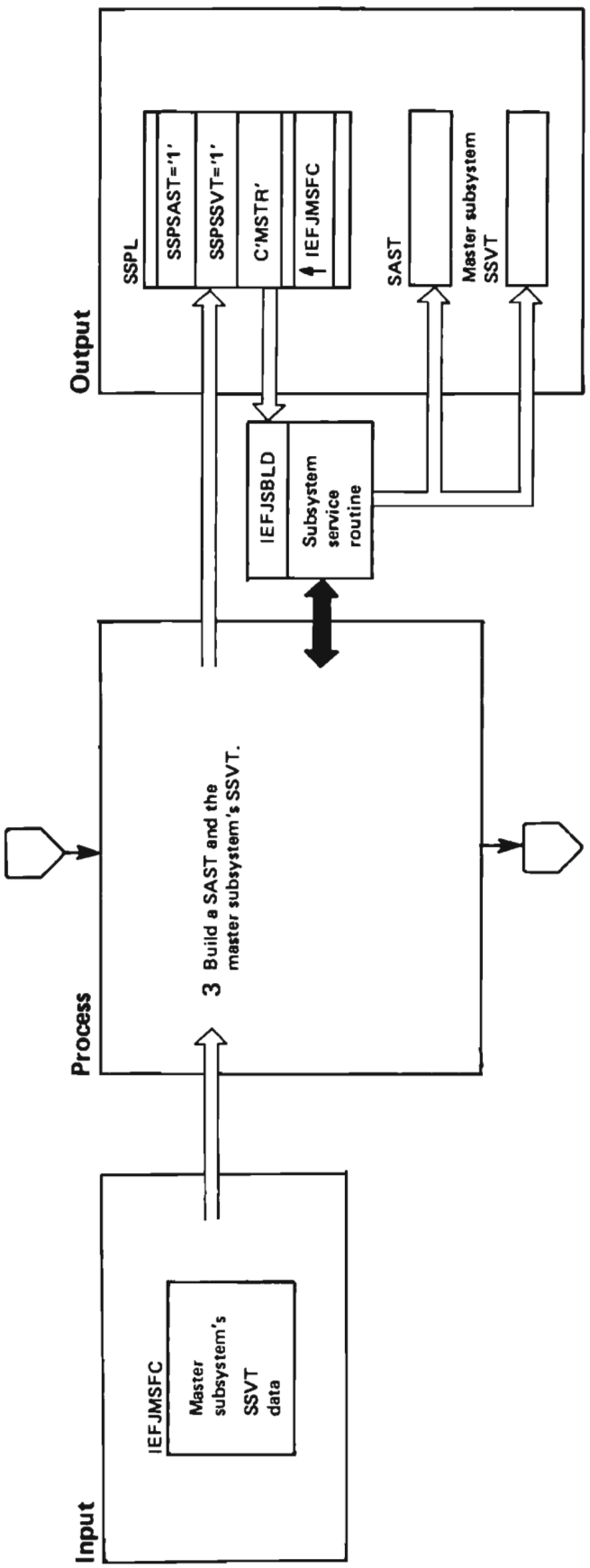


Diagram 80. Subsystem Interface Initialization (IEFJSINT) Part 3 of 6

Diagram 80. Subsystem Interface Initialization (IEFJSINT) Part 4 of 6

Extended Description	Module	Label
3 Set up the parameter list (SSPSAST='1') to build a subsystem allocation sequence table (SAST), (SSPSBCT='1') to build the subsystem communication vector table (SSCVT), and the master subsystem's vector table (SSVT) (SSPSSVT='1'). Ideniify the subsystem as the master subsystem (SSPNME=C 'MSTR'). The parameter list includes the address of a table (IEFJMSFC) containing the data needed to build the master subsystem's SSVT'. Invoke IEFJSBLD.	IEFJSINT	

Diagram 80. Subsystem Interface Initialization (IEFJSINT) Part 5 of 6

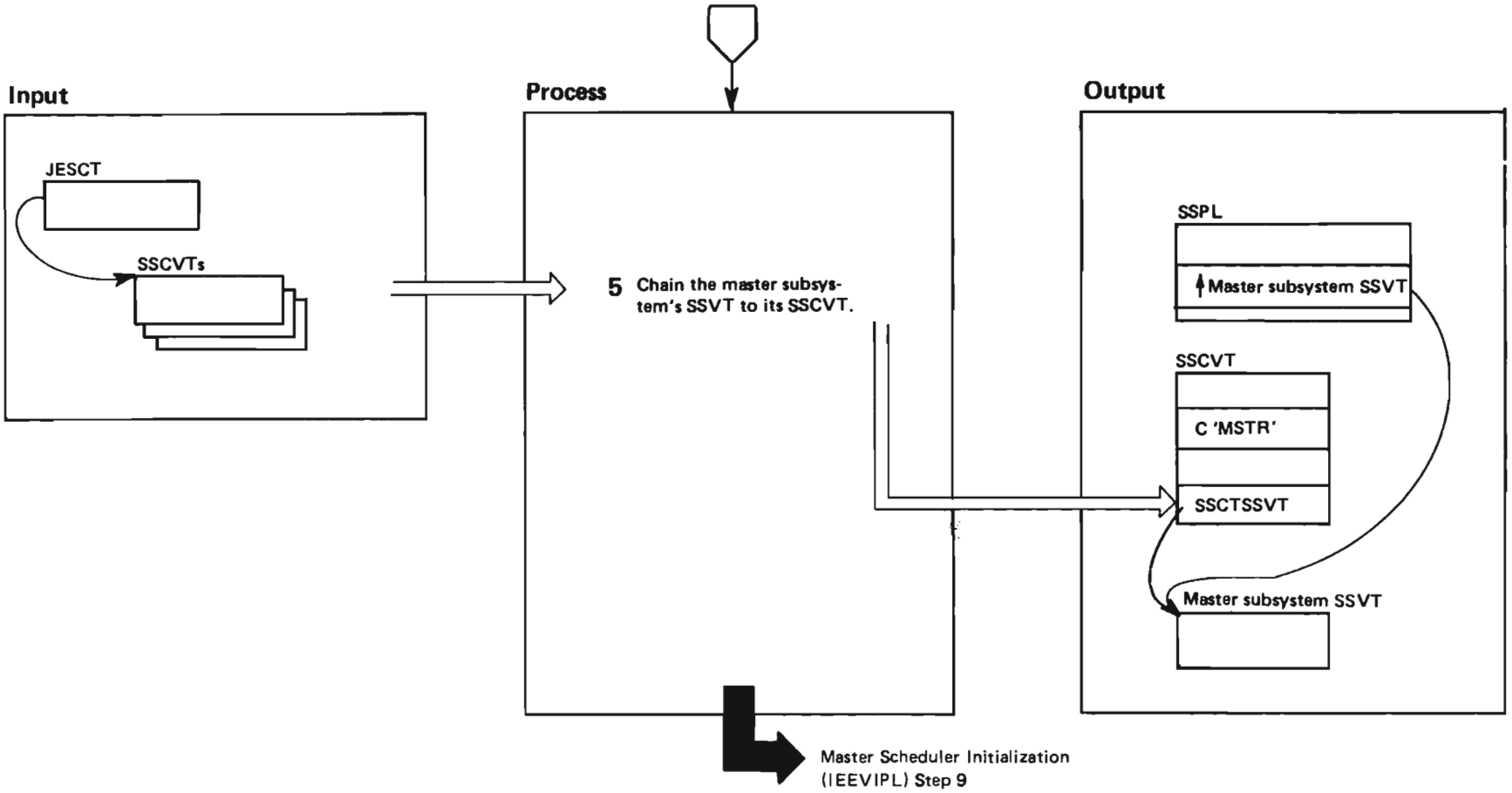


Diagram 80. Subsystem Interface Initialization (IEFJSINT) Part 6 of 6

Extended Description	Module	Label
5 Locate the master subsystem's SSCVT and store the SSVT address, returned in the parameter list, in the SSVT (field name SSCTSSVT).	IEFJSINT	
Return control to the caller.		

Recovery Processing:

None.

Diagram 81. Subsystem Service Routine (IEFJSBLD) Part 1 of 14

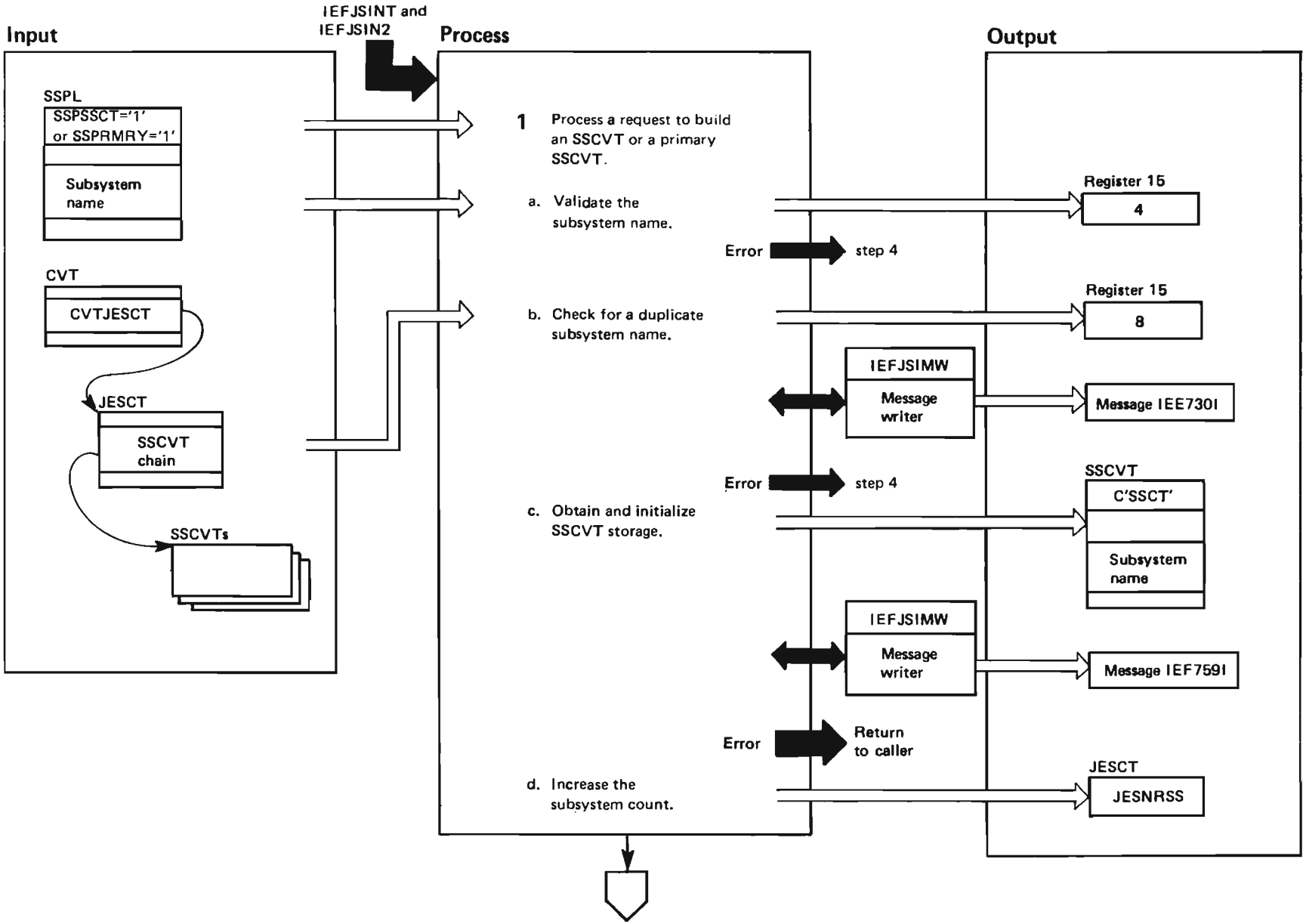


Diagram 81. Subsystem Service Routine (IEFJSBLD) Part 2 of 14

Extended Description	Module	Label
IEFJSBLD performs one or more of the following services: • Builds a subsystem communication vector table (SSCVT). • Builds a primary subsystem communication vector table (SSCVT). • Builds a subsystem vector table (SSVT). • Links to a subsystem initiallization routine. • Builds a hash table (SHAS) of SSCVT addresses. • Builds a subsystem allocation sequence table (SAST). • Disables selected SSVT function codes. • Enables selected SSVT function codes. IEFJSBLD is protected by its own ESTAE environment. The input to IEFJSBLD is the subsystem service routine parameter list (SSPL) that contains bits that indicate which function is being requested and what data is needed to process the request.		
1 If this is a SSCVT build request (the SSPSSCT bit is on), or a primary SSCVT build request (the SSPRMRY bit is on) then continue with this step. Otherwise, go to step 2.	IEFJSBLD	
a. Validate the subsystem name. The subsystem name cannot begin with a blank character. If the name is invalid, then set a return code of 4 and go to step 4; otherwise continue with step 1.		
b. If the SSCVT chain pointer in the JESCT is 0, then this is the first request to build an SSCVT, and no duplicate subsystem name exists. If this is not the first SSVT build request (the SSCVT chain pointer is not 0), then scan the SSCVT chain, searching for a subsystem name that matches the input subsystem name. If a duplicate name is found, invoke the message writer, IEFJSIMW, to issue message IEE730I, set a return code of 8, and proceed to step 4.	IEFJSBLD IEFJSIMW	SCANSSCT
If a duplicate subsystem name is not found, continue with step 4.		
c. Issue a GETMAIN macro instruction to obtain storage from subpool 241 (pageable CSA) for the SSCVT. If the GETMAIN fails, invoke the message writer, IEFJSIMW, to issue message IEF7591, set a return code of 24, and return to the caller.	IEFJSBLD IEFJSIMW	SSCTBLD
d. If the GETMAIN is successful, initialize the SSCVT with the control block identifier C'SSCT' and the subsystem name and increase the count of the number of subsystems (the JESNRSS field in the JESCT) by one.	IEFJSBLD	SSCTBLD

Diagram 81. Subsystem Service Routine (IEFJSBLD) Part 3 of 14

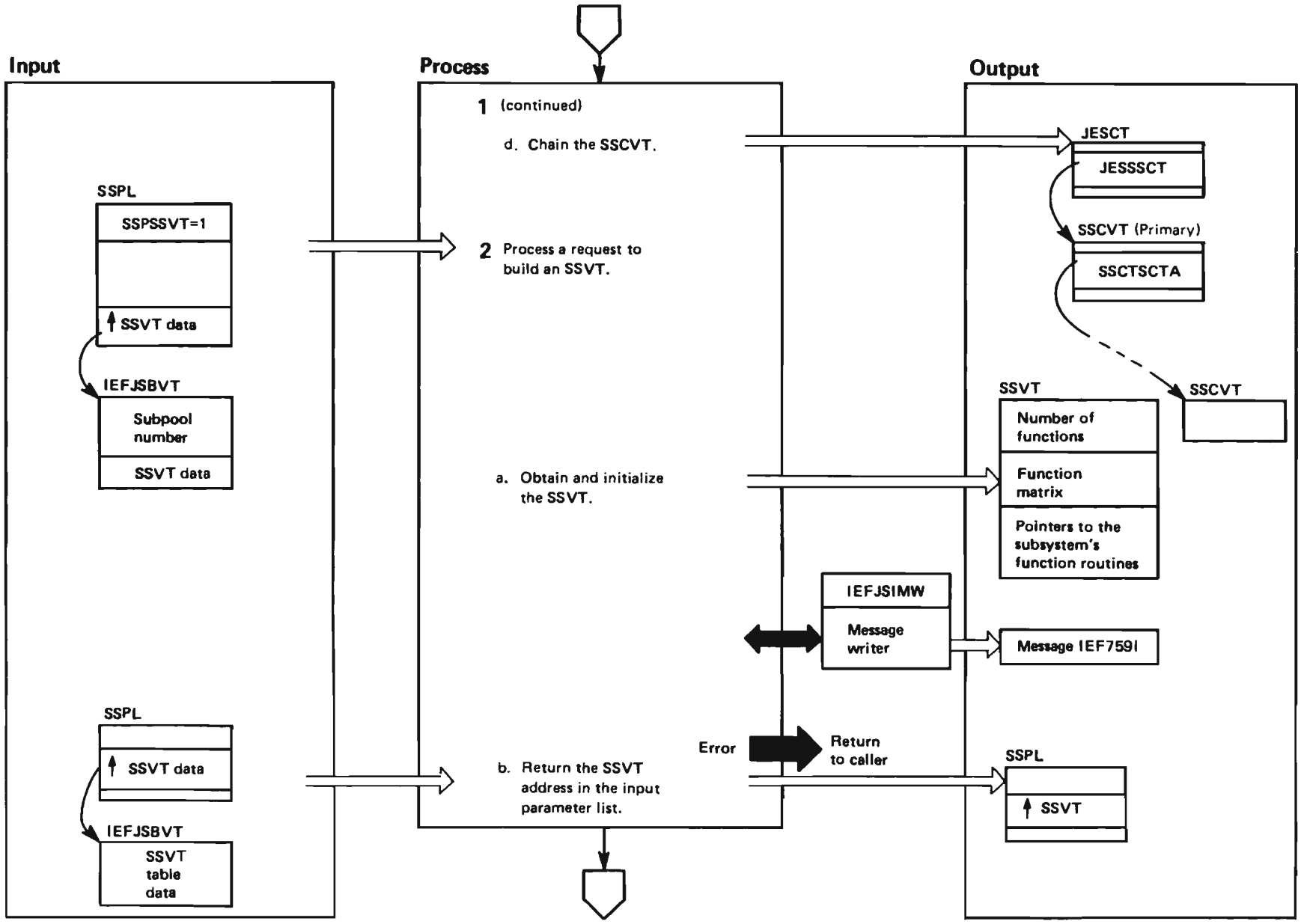


Diagram 81. Subsystem Service Routine (IEFJSBLD) Part 4 of 14

Extended Description	Module	Label
1 (continued)		
d. Chain the new SSCVT. If the request is for a primary subsystem or the first request, chain the SSCVT to the JESCT. Otherwise, chain the SSCVT to the previous SSCVT.	IEFJSBLD	SSCTBLD
2 If the SSPSSVT bit is on, then this is a request to build an SSVT. Continue with steps 2b-2c. If this is not a request to build a SSVT, go to step 3a.		
a. Issue the GETMAIN macro instruction to obtain storage for the SSVT from the subpool specified in the input SSVT data, mapped by IEFJSBVT. The storage size=SSVT fixed area (260 bytes) + (4 * the number of functions supported by the subsystem). If the GETMAIN fails, invoke the message writer, IEFJSIMW to issue message IEF759I, set a return code of 24, and return to the caller. If the GETMAIN is successful, clear the SSVT storage and build the SSVT. The subsystem interface uses the SSVT to route requests to subsystems.	IEFJSIMW	
Store the maximum number of functions supported into the SSVT.	IEFJSBLD	SSVTBLD
For each subsystem function routine specified in the input SSVT data, check the SSPGLOAD bit in the subsystem parameter list to determine if the caller requested that a LOAD macro instruction, global option be used. If so, issue the LOAD macro instruction with this option. Otherwise, issue the LOAD macro instruction without the global option, followed by a DELETE macro instruction. Save the address in the SSVT and initialize its corresponding function code.		
A system routine requests a subsystem service by passing a function code to the subsystem intarface. This code is a number between 1 and 256 that enables the subsystem interface routine to determine which subsystem routine is to get control. The code refers to a single byte in the SSVT's function matrix. A code of 1 refers to the first byte in the matrix, a code of 2 refers to the second byte, and so on. The matrix byte contains a value that is an index into the list of entry point addresses for the subsystem's function routines. A value of 1 refers to the first address, a value of 2 refers to the second address, and so on. A value of 0 indicates that the function is not offered by this subsystem.		
b. Store the SSVT address in the input parameter list in field SSPSVTAD.		

Diagram 81. Subsystem Service Routine (IEFJSBLD) Part 5 of 14

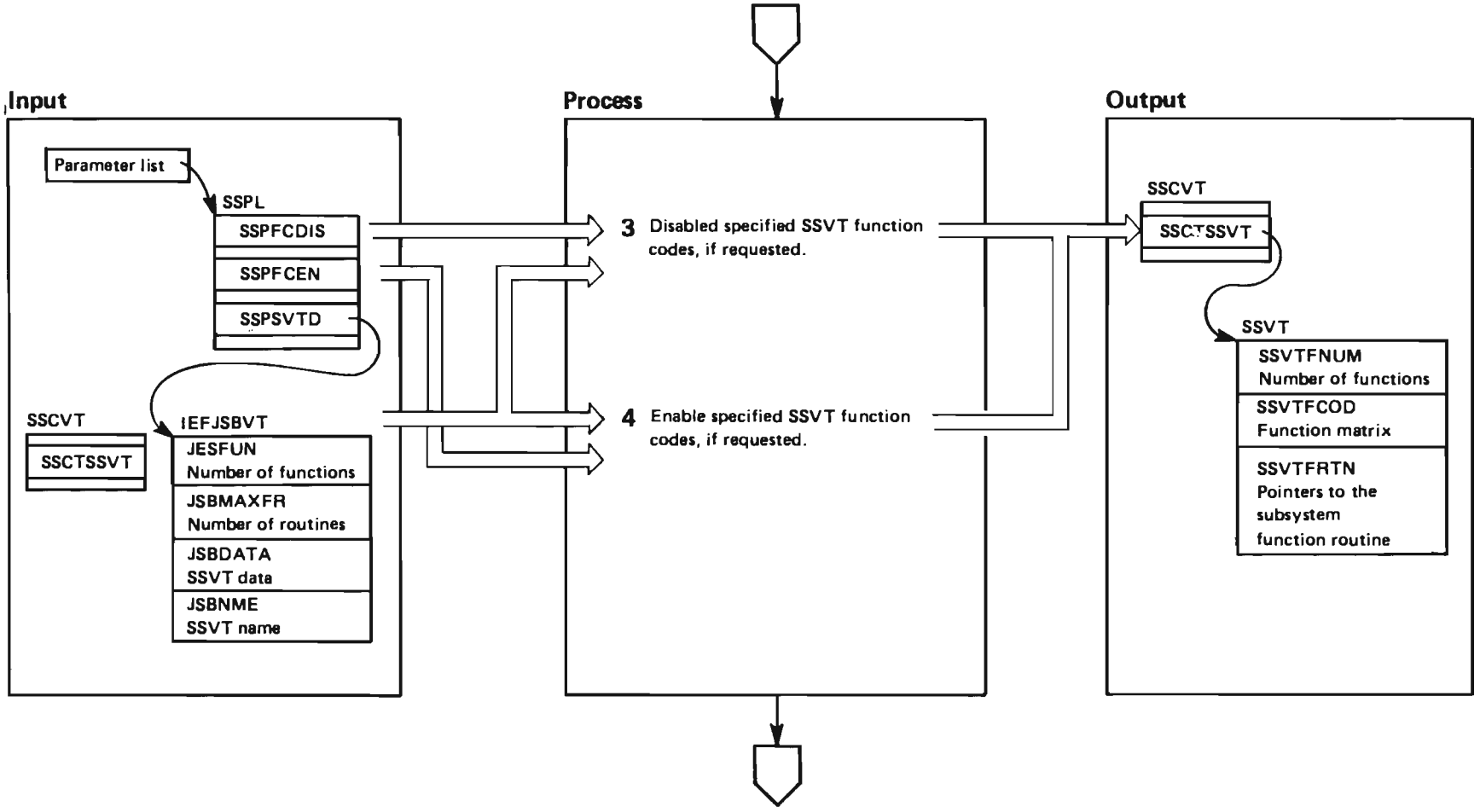


Diagram 81. Subsystem Service Routine (IEFJSBLD) Part 6 of 14

Extended Description	Module	Label
3 Check the bit, SSPFCDIS, to determine if this is a request to disable selected SSVT function codes. If the bit is off, continue with step 4. Otherwise, check the SSVT pointer field (SSCTSSVT) in the subsystem communications vector table (SSCVT), mapped by IEFJSCVT, to determine that there is an SSVT for the specified subsystem. **If there is an SSVT for the specified subsystem, zero the entry in the function code array for each SSVT function code and continue processing with step 5.** **If there is no SSVT for the specified subsystem, set a return code of 36 and return to the caller.**	IEFJSBLD	SSVTEDIS
4 Check the bit, SSPFCEN, to determine if this is a request to enable selected SSVT function codes. If the bit is off, continue with step 5. Otherwise, check the SSVT pointer field to determine that there is an SSVT for the specified subsystem. If there is an SSVT, for each subsystem function routine specified in the input SSVT data (JSBDATA), check the SSPGLOAD bit in the subsystem parameter list to determine if the caller requested that a LOAD macro instruction, global option, be issued. If so, issue the LOAD with this option. Otherwise, issue the LOAD without the global option, followed by a DELETE macro instruction. Compare the address of the subsystem function routine with each address in the SSVT. If there is a match, initialize the SSVT function code entry with the index to the function routine address. If the address is not in the SSVT, put the address in the next available entry in the SSVT function routine pointer array (SSVTFRTN) and store the index into the corresponding SSVT function code entry. If there is no match, set a return code of 44 to indicate that there is insufficient space in the SSVT for additional subsystem function routine addresses and return to the caller.		SSVTEN

Diagram 81. Subsystem Service Routine (IEFJSBLD) Part 7 of 14

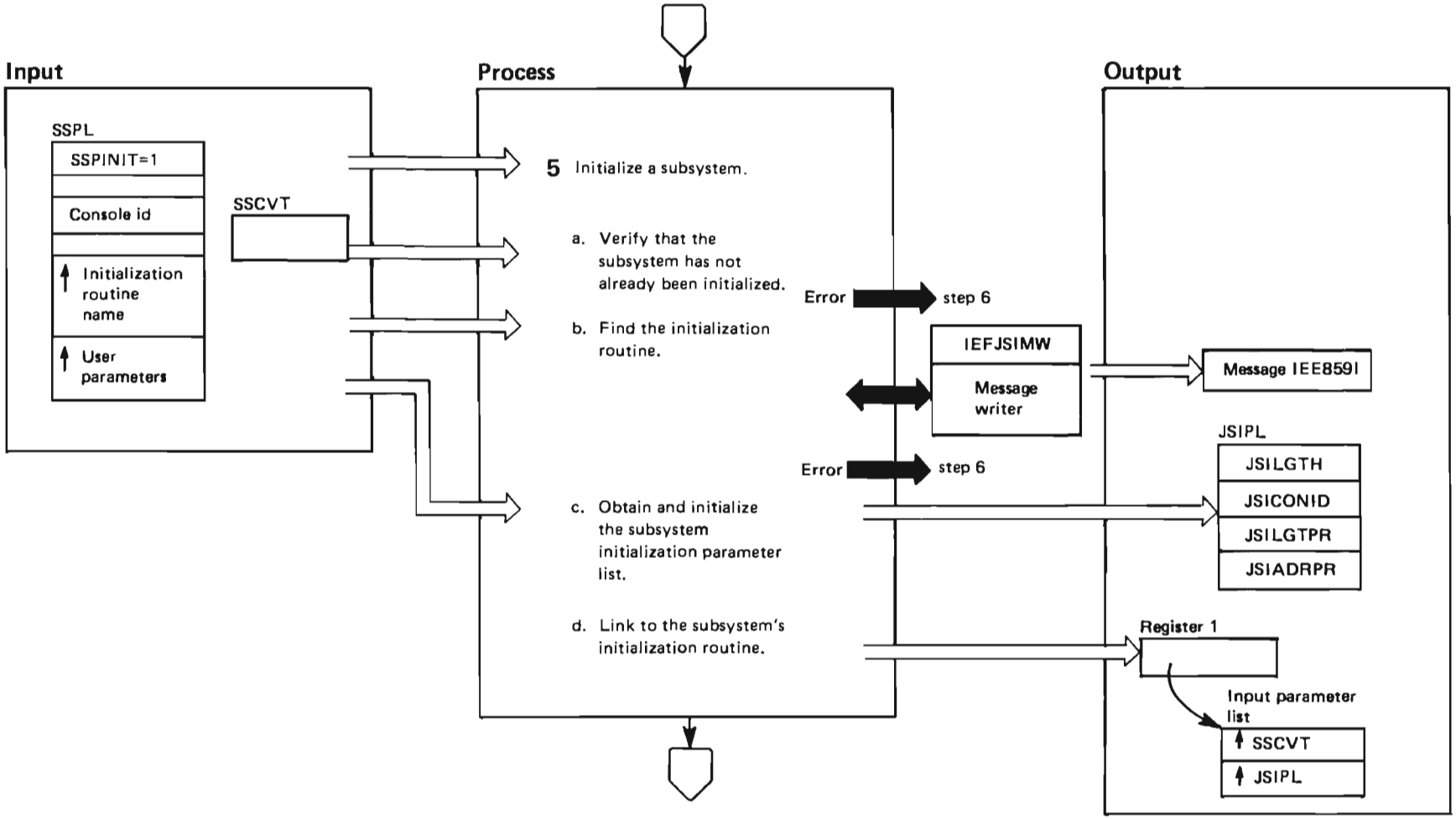


Diagram 81. Subsystem Service Routine (IEFJSBLD) Part 8 of 14

Extended Description	Module	Label
5 If this is a subsystem initialization request (bit SSPINIT of the input parameter list is set on), link to the subsystem initialization routine and continue with this step. Otherwise, go to step 6.	IEFJSBLD	LINKINIT

a. Determine if the subsystem has already been initialized. Scan the SSCVT chain to locate the subsystem's SSCVT. If either (1) the SSCTUPSS bit is on or (2) the pointer to the SSVT (SSCTSSVT) is not zero or (3) the SSCTSUSE field is not zero, the subsystem has already been initialized. In this case, set a return code of 12 and go to step 6.

b. If the subsystem has not been initialized, issue a BLDL macro instruction to determine if the initialization routine for this subsystem exists in SYS1.LINKLIB. If the initialization routine is not found, invoke the message writer, IEFJSIMW to issue message IEE859I, set a return code of 16, and go to step 6.

c. If the subsystem initialization routine is found, set up the subsystem initialization parameter list (IEFJSIPL) as follows:

- JSLGTH – length of the parameter list.
- JSICOND – console id to which the subsystem initialization routine should issue messages.
- JSILGTPR – length of user parameters.
- JSIADRPR – address of user parameters.

d. Link to the subsystem initialization routine, passing the SSCVT and the JSIPL.

If an ABEND occurs in the subsystem initialization routine, no dump is taken. IEFJSBLD issues message IEF759I and returns to its caller with a return code of 20.

Diagram 81. Subsystem Service Routine (IEFJSBLD) Part 9 of 14

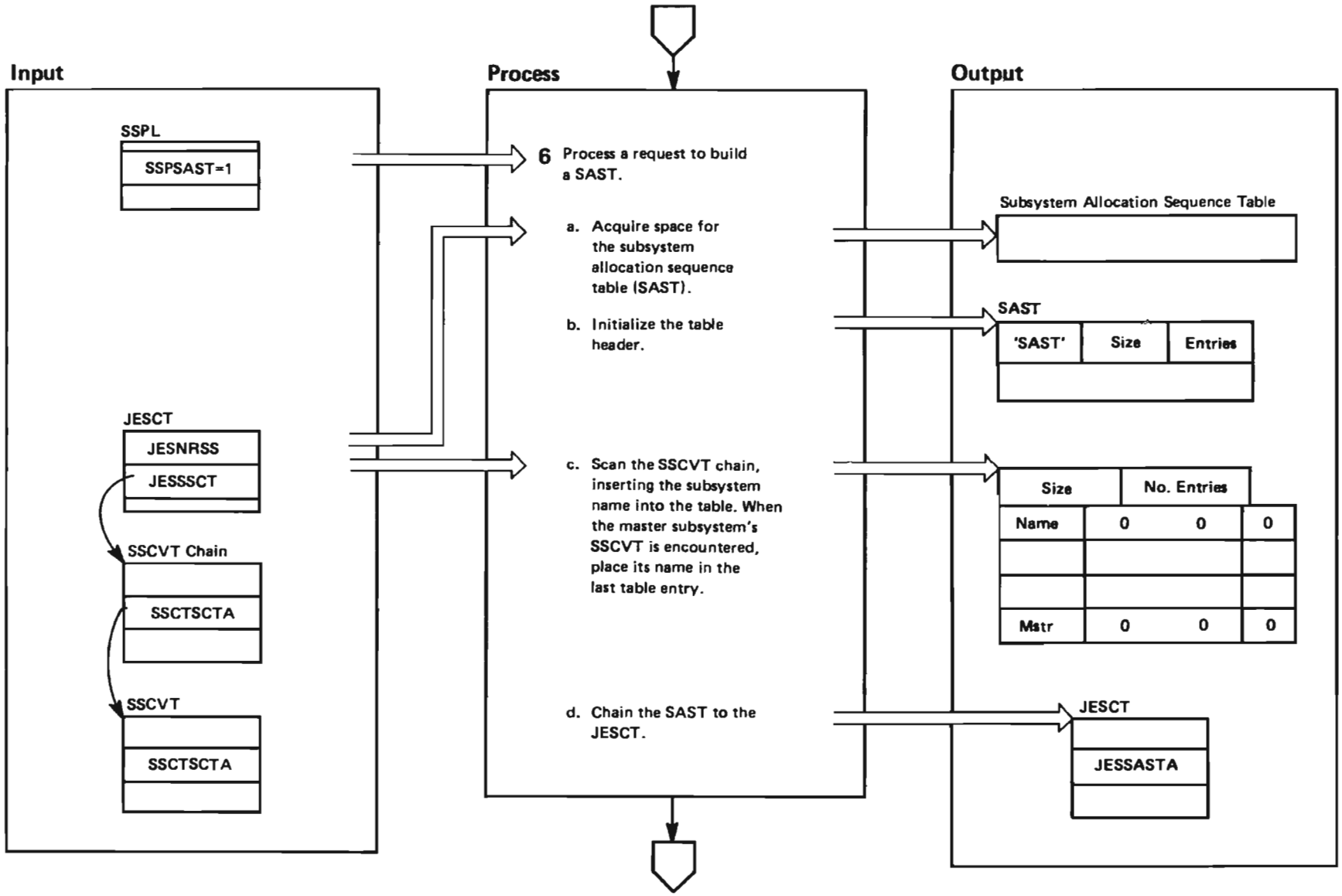


Diagram 81. Subsystem Service Routine (IEFJSBLD) Part 10 of 14

Extended Description	Module	Label
6 If this is a request to build a SAST (the SSPSAST bit is on), then continue with this step. If this is not a request to build a SAST (SSPAST bit is off), then go to step 7.	IEFJSBLD	SASTBLD
a. Obtain space for the SAST from subpool 241 (pageable) CSA). The size of the SAST is equal to 12 bytes times the number of subsystems plus 8 bytes for the header. The number of subsystems is in JESNRSS.		
b. Finish initializing the SAST header with the control block id, the number of entries in the table, and the size of the table.		
c. Scan the SSCVT chain to determine the subsystem names that have been defined to the system during master scheduler initialization. As a name is found, place it in the table. Place the name of the master subsystem in the last table entry.		
d. Chain the SAST to the JESCT (field JESSASTA). If a SAST address already exists in field JESSASTA, then chain the new SAST and free the old SAST.		

Diagram 81. Subsystem Service Routine (IEFJSBLD) Part 11 of 14

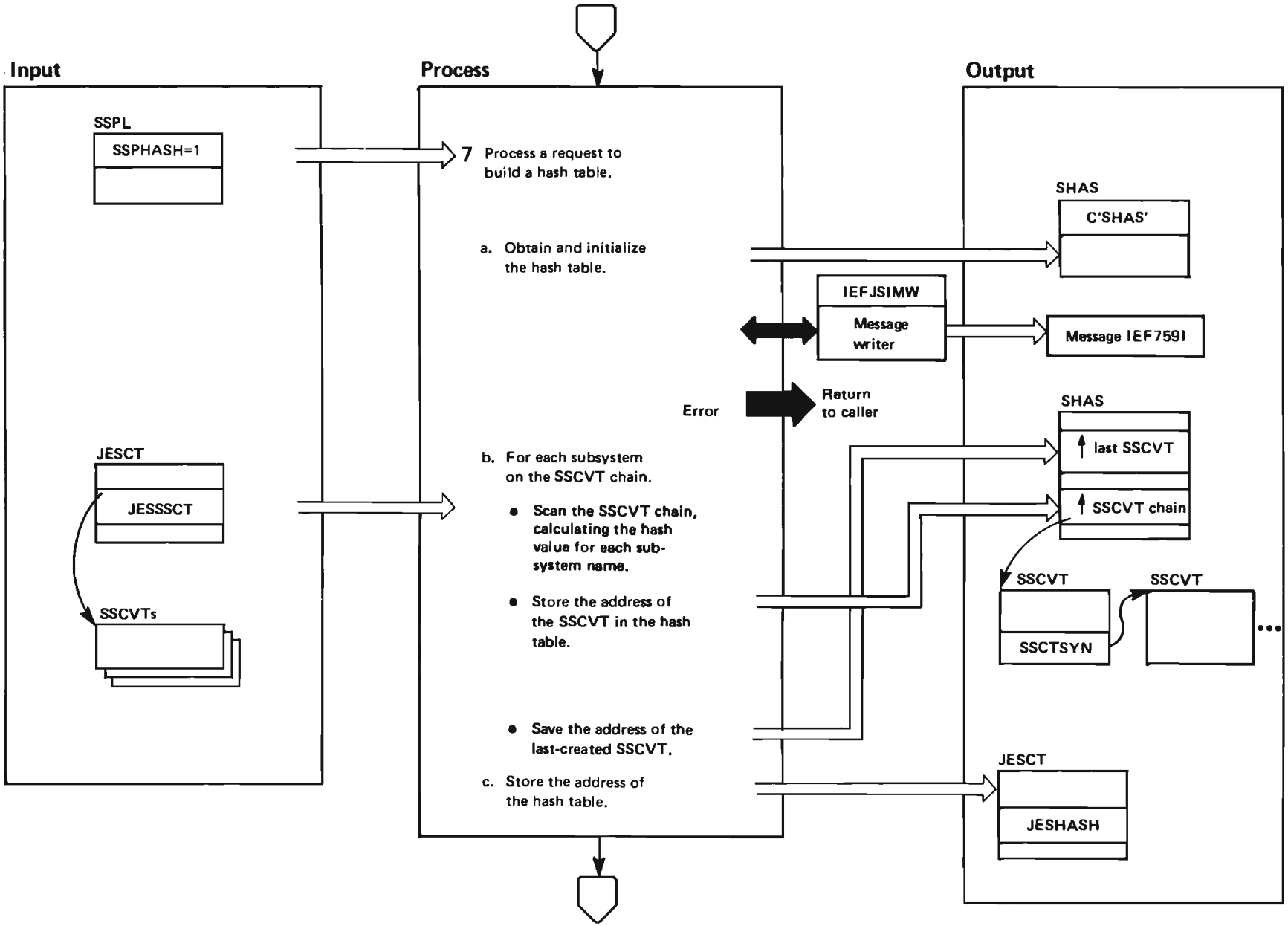


Diagram 81. Subsystem Service Routine (IEFJSBLD) Part 12 of 14

Extended Description	Module	Label
7 If a hash table is to be built (bit SSPHASH is on in the input parameter list), then continue with this step; otherwise, go to step 6. The hash table contains SSCVT addresses. The subsystem interface uses the hash table to locate an appropriate SSCVT directly rather than scanning the SSCVT chain to locate it.	IEFJSBLD	HASHBLD
a. Issue the GETMAIN macro instruction to obtain storage from subpool 241. The size of the storage area requested equals 12 bytes of fixed area plus (4 * number of entries in the hash table). Zero the acquired storage. If the GETMAIN fails, invoke the message writer, IEFJSIMW to issue message IEF759I, set a return code of 24, and return to the caller.		
b. Scan the SSCVT chain, calculating the hash value for each subsystem. The hash value is used as an index into the hash table. The hashing algorithm is:		
HASH = (The subsystem name divided by the number of slots in the table. The remainder + 1 is used as the hash value.)		
If a hash table entry is zero, then this subsystem is not a synonym; store the SSCVT address in the entry. If a hash table entry is not zero, then this subsystem is a synonym. Scan to the end of the synonym chain (indicated by a SSCVT synonym pointer [SSCTSYN] of zero). Store the SSCVT address for the newly-found synonym in the last SSCVT synonym pointer field.		
While scanning the SSCVT, the SSCVT address is kept in the hash table. Thus, when scanning is complete, the hash table will contain the address of the last SSCVT created during master scheduler initialization.		
c. Store the address of the hash table in the JESCT (field JESHASH).		

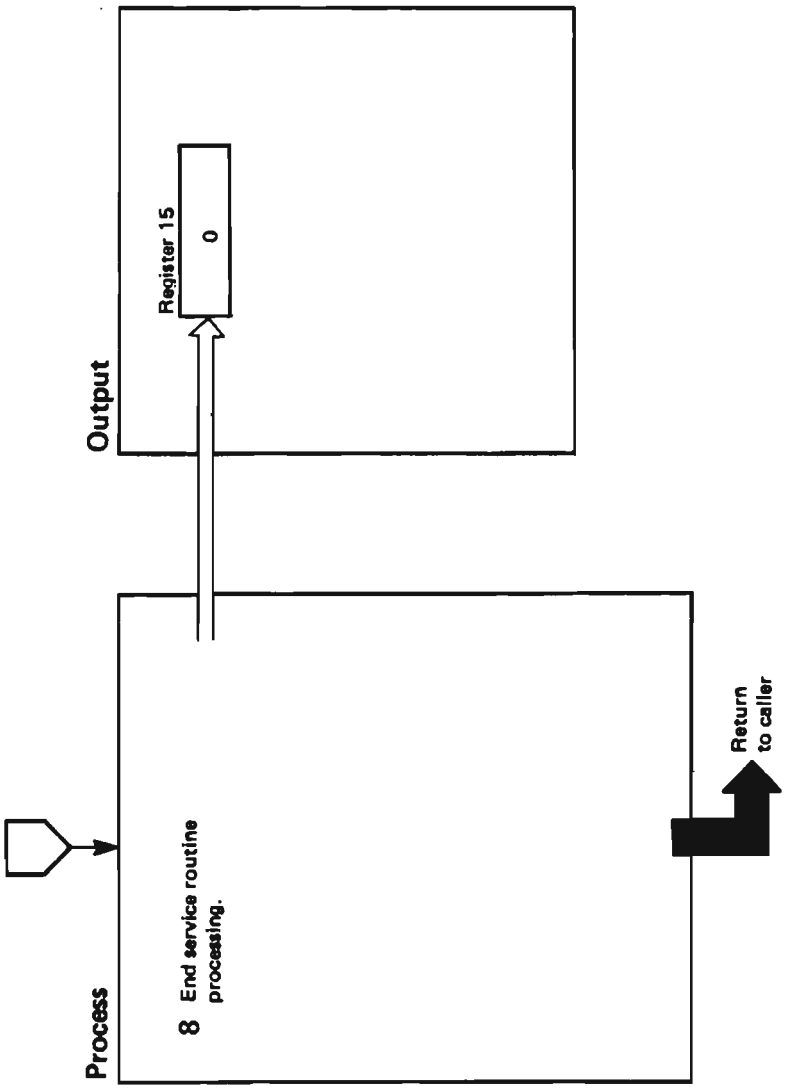


Diagram 81. Subsystem Service Routine (IEFJSBLD) Part 13 of 14

Diagram 81. Subsystem Service Routine (IEFJSBLD) Part 14 of 14

Extended Description	Module	Label
8 Set a successful return code of zero and return to the caller.	IEFJSBLD	
Recovery Processing		
An ESTAE protects all of IEFJSBLD processing from abends. If an abend occurs during IEFJSBLD processing, several actions are possible.	IEFJSBLD	ESTAEBLD
• Percolate the error if the GETMAIN ABENDs on the first attempt to build a SAST or if, in building the SSVT for the master subsystem, the LOAD for a subsystem function routine abends. For all other situations attempt a retry. • Take an SDUMP unless the error is a machine check or this ESTAE was percolated to or the subsystem initialization routine was in control when the error occurred. • Record diagnostic information in the variable recording area of the SDWA.		
The retry of IEFJSBLD consists of clean-up processing which depends on when the error occurred. The retry frees any storage that was obtained and issues the DELETE macro instruction for any outstanding LOADs.		
If a LOAD abended, return the name of the function routine that was being loaded.		

Diagram 82. Allocation Initialization (IEFAB410) Part 1 of 4

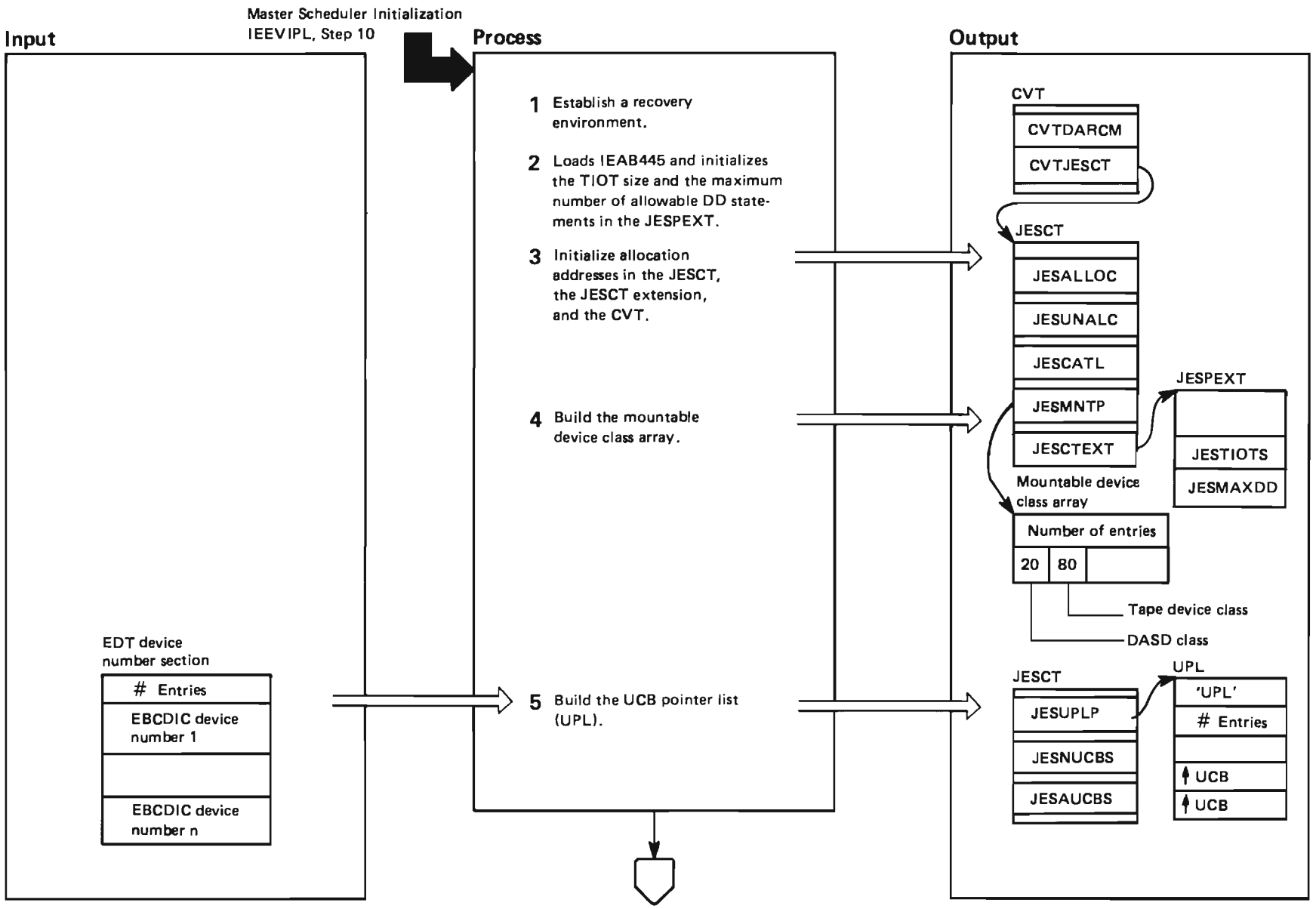


Diagram 82. Allocation Initialization (IEFAB410) Part 2 of 4

Extended Description	Module	Label
The allocation initialization routine initializes allocation control blocks, builds the UCB pointer list (UPL), and builds the mountable device class array. It passes to IEEMB881 the address of the initialization routine, IEFHB411, that will initialize the allocation address space (ALLOCAS). IEFAB410 attaches the EDT verification routine, IEFEB400, to verify that the description found in the EDT matches the UCBs in this system.		
1 IEFAB410 sets up an ESTAE environment for the non-address space initialization (Steps 1-5 in this diagram). The recovery routine is IEFAB4ED, which will route control to IEFAB4EH. If the ESTAE is unsuccessful, IEFAB410 issues an ABEND with a system completion code of X'05C' and a reason code of 5 in register 0.	IEFAB410	
2 IEFAB410 initializes the JESTIOTS and JESMAXDD fields in the JESPEXT to contain the TIOT size and the maximum number of DD statemtns that are allowable. The TIOT size is obtained from CSECT IEFAB445 and the maximum number of DD statements is a calculation based on the TIOT size.		
3 IEFAB410 establishes the allocation entry points by storing the entry point address obtained via a LOAD macro instruction into the JESCT (fields JESALLOC, JESUNALC, and JESCATL), the JESCT extension (fields JESDB401, JESXVNSL, JESGB4DC, JESGB4UV, JESAB445, JESGB400, JESGDTOK), and the CVT (field CVTDARCM).		INITCBS
4 IEFAB410 issues a GETMAIN macro instruction to obtain storage from subpool 241, key 1, for the mountable device class array. The array has two entries, one for DASDs and one for tape devices. IEFAB410 places the address of the array in JESMNTP.		
5 IEFAB410 issues a GETMAIN macro instruction to obtain storage for the UPL from the subpool 241, scheduler key, with option LOC=ANY. IEFAB410 initializes the UPL header, scans the device number entries in the EDT, and uses the IOSLOOK service to locate the UCB address that corresponds to the device number. If a device number in the EDT has a corresponding UCB address, IEFAB410 initializes the UPL with the address in the UCB. If a device number in the EDT does not have a corresponding UCB address, it sets the UPL field to zero. IEFAB410 stores the address of the UPL in JESUPL, the number of UCBs in the system in JESNUCBS, the total number of tape and DASD UCBs in JESAUCBS, and sets a flag (JESALRDY) to indicate that allocation is available.		BUILDUPL

Diagram 82. Allocation Initialization (IEFAB4I0) Part 3 of 4

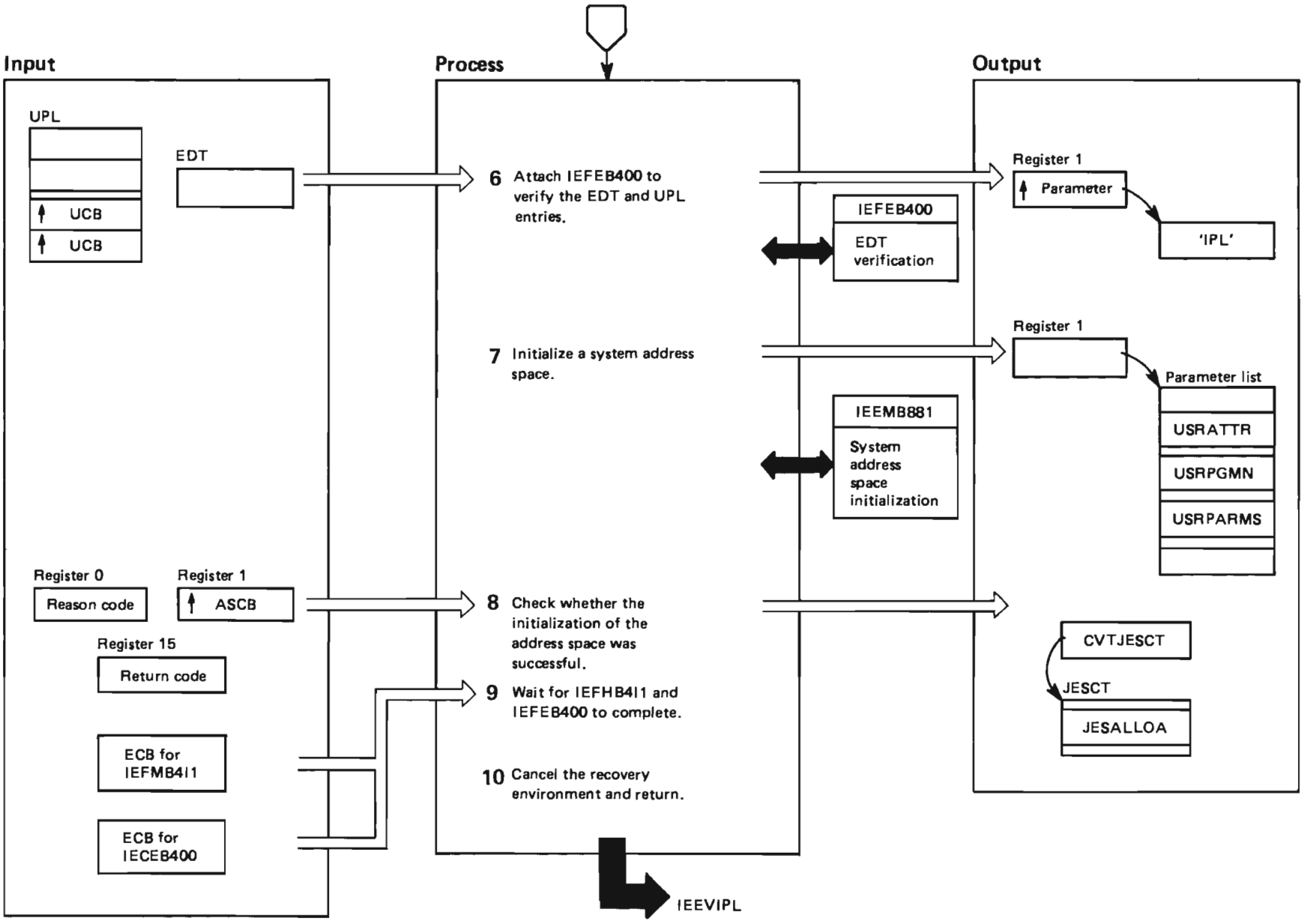


Diagram 82. Allocation Initialization (IEFAB410) Part 4 of 4

Extended Description	Module	Label
6 IEFAB410 attaches IEFEB400 (specifying the ESTAE routine, IEFAB4ED, which will route control to IEFAB4EH). IEFEB400 verifies that entries in the EDT match the UCBs in the system. It issues messages IEF924I and IEF925I for any discrepancies. If the ATTACH macro fails, IEFAB410 issues an ABEND with a system completion code of X'05C' and a reason code of X'101'.	IEFEB400 IEFAB410	
7 IEFAB410 sets up a second ESTAE environment to cover the allocation address space initialization processing. The recovery routine is IEFAB4E6. If the ESTAE is unsuccessful, IEFAB410 issues an ABEND with a system completion code of X'05C' and a reason code of 5.		SETSASI
IEFAB410 sets up the parameter list for IEEMB881 and calls IEEMB881 to create and initialize the allocation address space. The parameter list contains three fields: • USRATTR contains the length of the attribute (in double words) and describes the attributes of the allocation address space as follows: – The address space is for data use only (that is, it cannot be started from the operator's console), and is to remain in limited function start mode. – The address space is to be non-swappable. – The address space is not to be terminated (that is, RTM is to ignore any request to terminate it). • USRPGMN contains the address of IEFAB411, the routine that will initialize the allocation address space. • USRPARMS contains X'0007', the length of the name of the address space, and 'ALLOCAS', the name of the allocation address space.	IEEMB881	

Extended Description	Module	Label
8 IEFAB410 checks the return code from IEEMB881 in register 15. If the return code is not zero, then the initialization failed, and IEFAB410 issues an ABEND with a system completion code of X'05C' and a reason code of 5 in register 0. If it is zero, then the initialization was successful, and register 1 contains the address of the ASCB for the new address space. IEFAB410 extracts the ASID from this ASCB, for the new address space. IEFAB410 extract the ASID from this ASCB, places it in the JESALLOA field of the JESCT, and cancels the ESTAE, which covered the allocation address space initialization. This action re-establishes the recovery routine, IEFAB4ED.	IEFAB410	
9 IEFAB410 waits for IEFHB411 and IEFEB400 to complete. If IEFEB400 found any zero addresses in the UPL (indicating that a UCB was defined in the EDT but does not exist in the system), IEFAB410 calls IGFPTERM to issue message IEF928I and puts the system in a X'0200' wait state with a reason code of X'0002'.		
10 IEFAB410 cancels the recovery environment and returns to IEEVIPL.		

Recovery Processing

IEFAB4ED handles abnormal terminations of allocation failures not related to the allocation address space (ALLOCAS). IEFAB4ED places diagnostic data in the SDWA and produces a dump of the failing routine's address space. Areas dumped are the LPA, all the PSA, SQA, SWA, LSQA, and trace tables. Also, IEFAB4ED writes an error record SYS1.LOGREC. It then routes control to specialized recovery routines for the allocation subcomponent in control at the time of the failure. In the case of IEFAB410, the recovery routine is IEFAB4EH.

IEFAB4E6 handles abnormal terminations of tasks that create, initialize, or update the allocation address space (ALLOCAS). IEFAB4E6 places diagnostic data in the SDWA and produces an SDUMP of both ALLOCAS and the failing routine's address space. Areas dumped are the LPA, LSQA, all of the PSA, SQA, trace tables, and private regions. Also, IEFAB4E6 writes an error record to SYS1.LOGREC and issues message IEF1001.

Diagram 83. Allocation Address Space Initialization (IEFHB4I1) Part 1 of 2

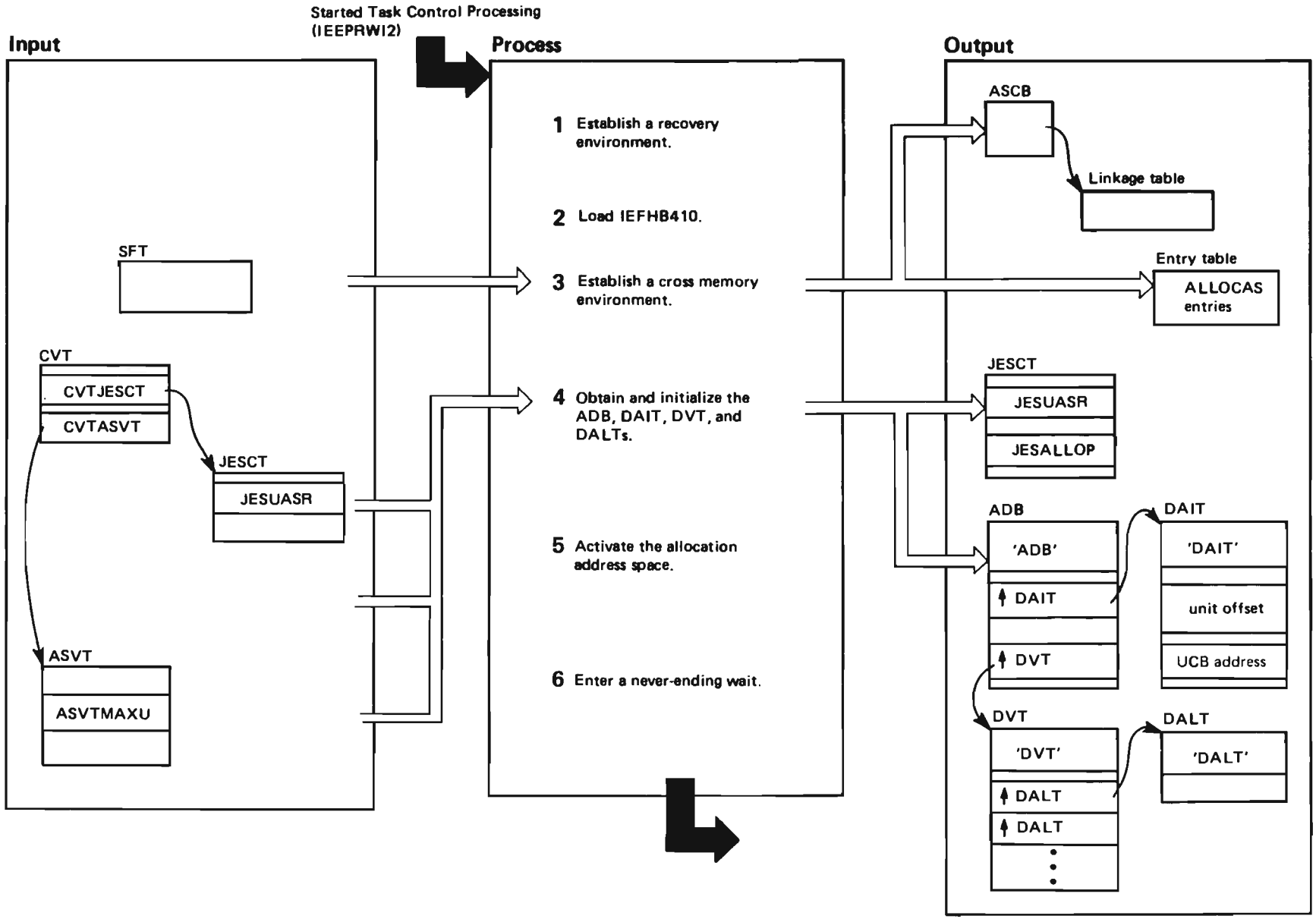


Diagram 83. Allocation Address Space Initialization (IEFHB4I1) Part 2 of 2

Extended Description	Module	Label
Allocation address space initialization obtains and initializes all the necessary control blocks in the allocation address space (ALLOCAS). This initialization includes obtaining storage in CSA for the program call display allocation lookup table (PCDALT) parameter list, loading the DALT manager (IEFHB4I0), establishing a cross memory environment, and entering a never-ending wait, leaving ALLOCAS in limited function start mode.		
1 Set up an ESTAE recovery environment. The ESTAE recovery routine is IEFAB4E6.	IEFHB4I1	
2 Load the display allocation lookup table (DALT) manager (IEFHB4I0) from SYS1.LINKLIB into ALLOCAS.	IEFHB4I1	
3 Set up a cross memory environment. Set the authorization index to one, using the AXSET macro instruction. ALLOCAS can then perform program transfers to any address space. A linkage index has already been reserved in the system function table (SFT) with the option to allow a global connection among all current and subsequent address spaces. Using the ETCRE macro instruction, create an entry table to define the entry points into ALLOCAS. Connect this entry table to the linkage table using the ETCON macro instruction with the linkage index from the SFT. Other address spaces can now perform program calls to ALLOCAS.	IEFHB4I1	XMEMPROC
4 Calculate the length of the allocation descriptor block (ADB), display allocation index table (DAIT), display allocation vector table (DVT), and all display allocation lookup tables (DALTs). Issue a GETMAIN macro instruction to obtain a block of key 0 storage from subpool 229 equal to the calculated length. Set the JESALLOP field of the JESCT equal to the address of the area obtained. The first section of this area becomes the ADB; store the ADB acronym in it; the next section becomes the DAIT; store the DAIT acronym, the unique index into the DAIT, and the UCB address in it; the third section becomes the DVT; store the DVT acronym in it, and store the DVT address and DAIT address in the ADB. The fourth section is split up into the required number of DALTs. Store the DALT acronym in each DALT and store its address in the DVT.	IEFHB4I1	GETBLOCK
5 Activate the allocation address space by setting the JESUASR bit on. Then post the EAERIMWT ECB to allow IEFAB4I0 to finish its processing.		
6 Enter a never-ending wait; that is, ALLOCAS remain in limited function start mode. If this wait is ever posted, then an SDUMP is in process for the allocation address space (ALLOCAS). WAIT for the SDUMP ECB, JESDUECB, to *oe* posted, then return to IEEPRWI2 with a return code of 4 indicating that an error has occurred.		
Recovery Processing		
See Diagram 70 for information on recovery processing.		

Diagram 84. System Management Facilities Initialization Router (IEEMB820) Part 1 of 2

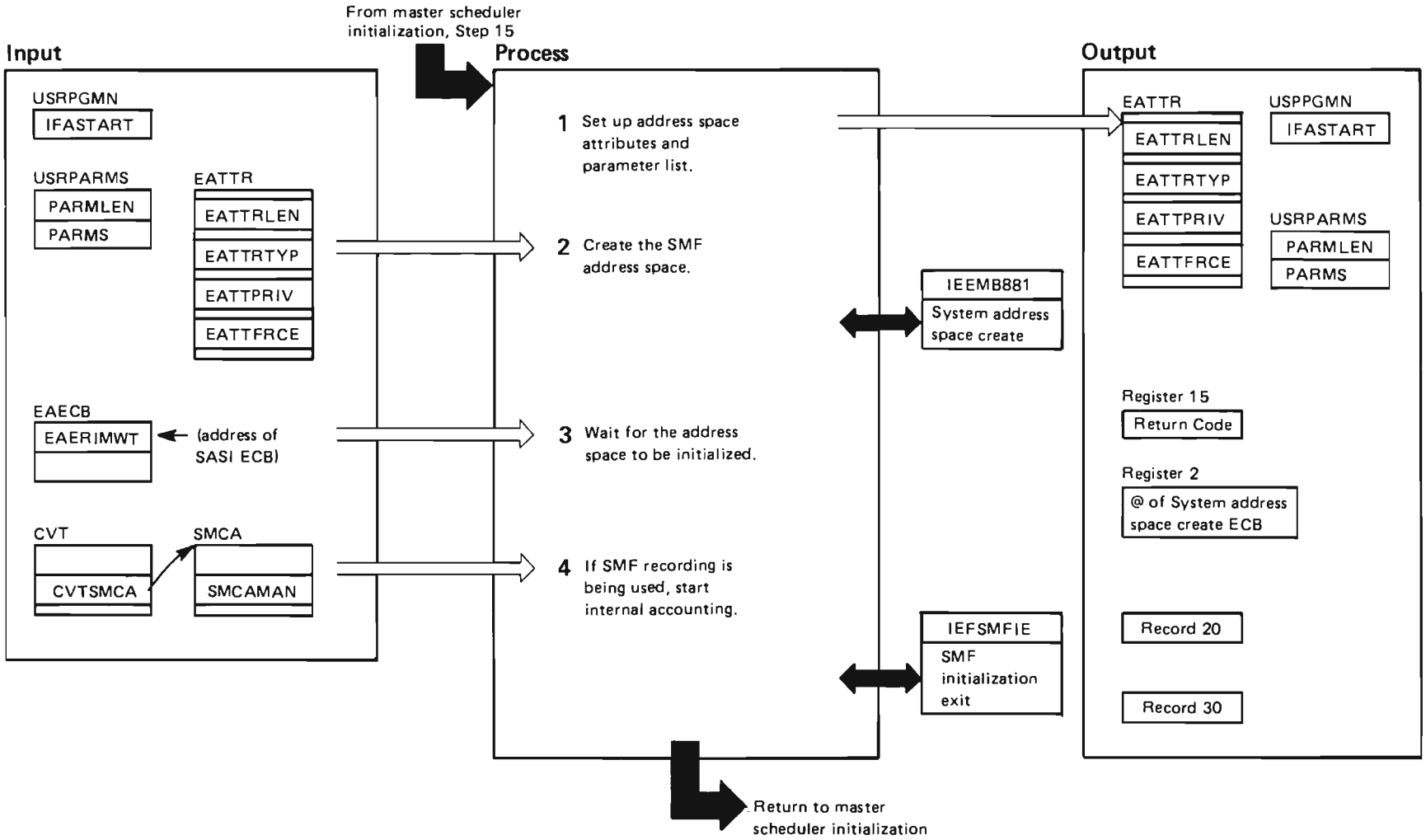


Diagram 84. System Management Facilities Initialization Router (IEEMB820) Part 2 of 2

Extended Description	Module	Label
This module starts the SMF address initialization process during Master Scheduler initialization. Once the address space is initialized, IEEMB820 links to IEFSMFIE to start interval accounting for the master address space.	IEEMB820	
1 IEEMB820 sets up the SMF address space attributes by filling in the system address space list (EATTR). SMF is a high-priority address space (EATTRTYP) that cannot be swapped out. The SMF address space can be forced (EATTFRCE=1). IEEMB820 also sets up SMF address space parameters. Specifying IFASTART as the SMF address space initialization routine.		
2 IEEMB820 links to the system address space create routine to set up the SMF address space. IEEMB881 sets a return code in register 15 and places the address of the address space initialization ECB in register 2. If the address space create fails, IEEMB820 issues message IEE480I to notify the operator.	IEEMB820	
3 IEEMB820 waits on the ECB returned by IEEMB881. IFASTART posts this ECB when initialization is complete so that IEEMB820 can continue.	IEEMB820	
4 If SMF recording is being used, IEEMB820 starts interval accounting for the master scheduler address space by linking to the SMF initialization exit routine.	IEFSMFIE	

SMF Keyword	Meaning and Use	Possible Values	Default	Corresponding Field in SMCA
ACTIVE	Specifies whether or not SMF recording is to be active.	–	ACTIVE	SMCAMAN=1, SMCAUSER=1
NOACTIVE		–		SMCAMAN=0, SMCAUSER=0
DSNAME (data set)	Specifies a list of up to 36 VSAM data sets that are to be used for SMF recording.	SYS1.MANn where n=A-Z, 0-9	DSNAME (SYS1.MANX, SYS1.MANY)	SMCAFRDS, SMCALRDS, SMCASVCR, SMCASRBR
JWT (hhmm)	Specifies the maximum amount of time that a job is allowed to wait continuously.	0001-2400	JWT(0010) [10 minutes]	SMCASJWT='hhmm', SMCAJWT=hhmm
LISTDSN	Specifies whether the system is to generate SMF data set status messages to the operator at IPL and SET SMF time.	–	LISTDSN	SMCALDSN=1
NOLISTDSN		–		SMCALDSN=0
MAXDORM (mmss)	Specifies the amount of time that data is permitted to remain in an SMF buffer before it is written to a recording data set.	0001-5959	MAXDORM(3000) [30 minutes]	SMCAMXDM=1, SMCASMDM='mmss'
NOMAXDORM				SMCAMXDM=0, SMCASMDM='0000' SMCAMAXT=0 SMCAMXT=address of timer element
PROMPT (option)	Specifies whether the selected SMF parameters are to be displayed on the system console at IPL time.	ALL	PROMPT(ALL)	SMCAIPLR=1, SMCALIST=1
		IPLR		SMCAIPLR=1, SMCALIST=0
		LIST		SMCAIPLR=0, SMCALIST=1
NOPROMPT				SMCAIPLR=0, SMCALIST=0
REC (option)	Specifies whether information for record type 17 (scratch data set status) is to be collected for temporary data sets.	ALL	REC(PERM)	SMCATDS=1
		PERM		SMCATDS=0
SID (system) (model, serial #)	Specifies the system identifier to be used in all SMF records.	The system and the model on which SMF is active and the processor serial #.	The processor model number found in the PCCA control block.	SMCASID=system
STATUS (hhmmss)	Specifies the amount of time between creations of record type 23 (SMF statistics).	000001-240000	STATUS (010000) [1 hour]	SMCASTUS=1, SMCASSTS='hhmmss' SMCASTTT=address of timer element
NOSTATUS				SMCASTUS=0, SMCASSTS='000000' SMCASTTT=0
SYS(options)	Specifies the global recording options for the entire system.	See Figure 5-6 for a list of options.	See each option	SMCASYSP=address of SYS SST entry
SUBPARM (name (parameters))	Specifies the information to be passed to a specific subsystem	Subsystem name followed by any parameters	None	SMCASUBP points to a chain of control blocks. Each control block identifies one subsystem name and its parameters
SUBSYS (name, options)	Specifies what data is to be collected and recorded for a specific subsystem.	Subsystem name followed by any of the options listed in Figure 5-6	See each option	SMCASSTD points to the block of SSTs.

Figure 5-5 The Relation of System Management Facilities (SMF) Keyword Parameters[1] to Fields in SMCA

[1]For more information of SMFPRMxx parameters, see *System Management Facilities (SMF)*

SMF Keyword	Meaning and Use	Possible Values	Default	Corresponding Field in SST
TYPE (aa,bb . . .)	TYPE specifies the record types to be collected by SMF. NOTYPE specifies that all records except those specified are to be written by SMF.	0-255	TYPE(0:255)	SSTRCDON – 256 bits string with bits turned on for only those record types requested.
NOTYPE (aa,bb . . .)		0-255		SSTRCDON – 256 bit string with all bits turned on except for those record types requested.
DETAIL	Specifies the level of data collection for type 32 TSO records.	–		SSTDETAL=1
NODETAIL		–	NODETAIL	SSTDETAL=0
INTERVAL (hhmmss)	Allows the user to checkpoint accounting data during the course of a long running job or TSO session by specifying the amount of real time between each checkpoint.	000001-240000	NOINTERVAL	SSTINTVL=interval in 64-bit form
NOINTERVAL				SSTINTVL=0
EXITS(exit name, exit name . . .)	Specifies whether SMF exits are to be invoked.	–	EXITS	SSTEXTAB=address of EXITTAB (Table contains up to 15 exits. EXITINAC (i)=1 if exit (i) is inactive.)
NOEXITS		–		SSTEXTAB=address of EXITTAB (Any exits specified previously will have EXITINAC (i)=1.)

Figure 5-6 The Relation of System Management Facilities (SMF) Keyword Parameters to Fields in the Subsystem Selectivity Table (SST)[1]

[1]An SST entry is created for SYS and for each SUBSYS specified during IPL and subsequent SET commands. Any SUBSYS parameters not specified default to the corresponding SYS parameter currently in effect.

Diagram 85. Recovery Termination Management Initialization (IEAVTMSI) Part 1 of 4

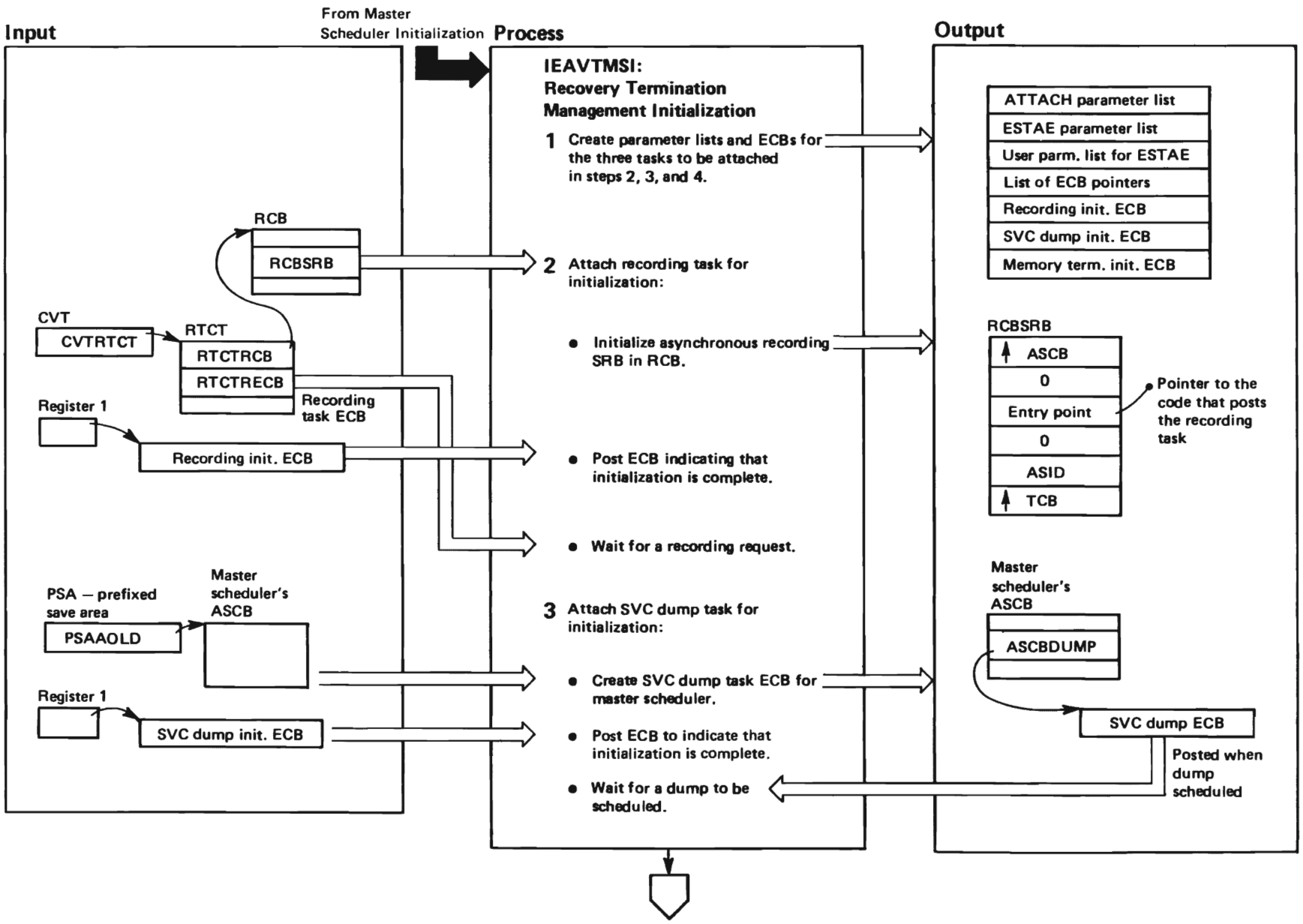


Diagram 85. Recovery Termination Management Initialization (IEAVTMSI) Part 2 of 4

Extended Description	Module	Label
Recovery termination management initialization is invoked from master scheduler region initialization (IEEMB860) to establish three permanent tasks. These tasks handle requests for writing records to SYS1.LOGREC or to the operator, requests for scheduled SVC dumps of the master scheduler's address space, and requests for address space termination. Each task performs its own initialization, notifies IEAVTMSI when initialization is complete, and then waits for a request.		
1 When each task is attached, it must have the address of an initialization ECB (event control block). IEAVTMSI places the address of an initialization ECB in register one for each of the three tasks. (These ECBs are also used as end-of-task ECBs should one or more of the attached tasks abnormally terminate before its initialization is complete.) Each task posts its ECB when it completes its initialization. When all three tasks are attached and initialized, they are ready to process and will wait for a request to do so.	IEAVTMSI	
2 The asynchronous recording task performs initialization as follows: • Initializes the SRB in the RBCB (an SRB created by IEAVNPA6 via the GETMAIN macro instruction). • Obtains storage (GETMAIN) in subpool 252 for a work area containing recovery tracking information. • Issues an ESTAE macro instruction to set up the recovery environment for the recording task. When there is a request to write records to SYS1.LOGREC or to the operator, the SRB just created notifies the recording task to process the request. The recording request routine, located in the nucleus, schedules this recording task SRB (RBCBEUB) which posts the recording task ECB. The recording task then issues an SVC 76 to write to SYS1.LOGREC or issues a WTO to write to the operator. System routines running in supervisor state with a protection key of zero can use the recording task for writing to the operator.	IEAVTRET	
3 SVC dump task initialization creates an ECB that is posted when a system routine requests a scheduled dump of the master scheduler's address space. An SVC dump task is normally attached and initialized by the region control task (RCT) for each address space. However, the SVC dump task for the master scheduler's address space is attached by IEAVTMSI. For a general discussion of this task, refer to the topic "SVC Dump Task" in *System Logic Library.*	IEAVTSDT	

Diagram 85. Recovery Termination Management Initialization (IEAVTMSI) Part 3 of 4

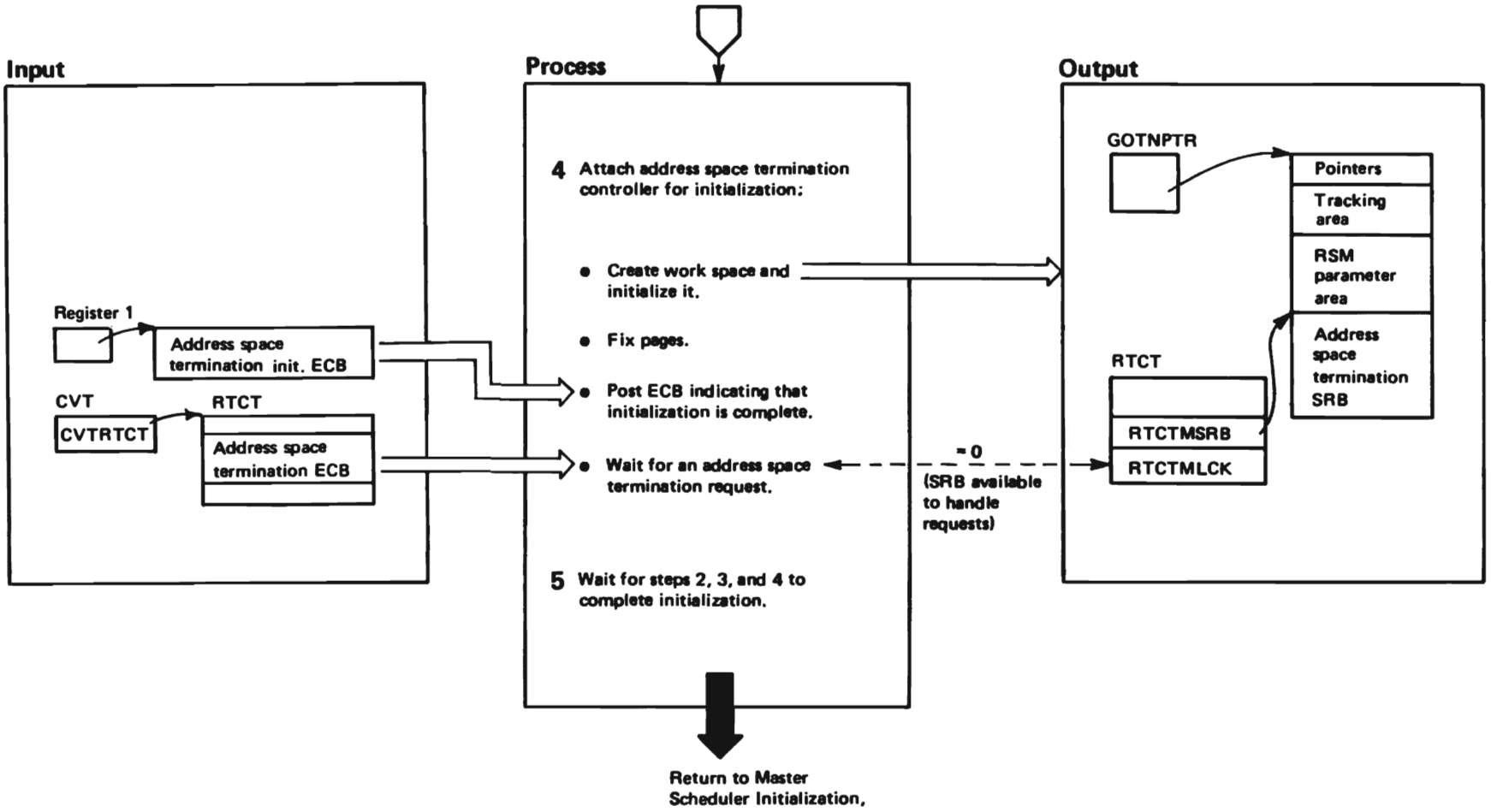


Diagram 85. Recovery Termination Management Initialization (IEAVTMSI) Part 4 of 4

Extended Description	Module	Label
4 Address space termination task initialization creates a workspace containing recovery tracking information, an RSM parameter list, and an SRB. IEAVTMTC also fixes the pages of IEAVTMTC (including ESTAE exit code) and issues the ESTAE macro instruction to set up a recovery environment. Finally, IEAVTMTC allows the termination SRB to be scheduled by setting a flag (RTCTMLCR=1). A request for address space termination causes the address space termination SRB to be scheduled. This termination SRB then notifies the address space termination task to process the request. The termination task, when posted for work, processes address space termination requests. It uses the recovery tracking information to retrace its steps if a program check occurs while it is terminating an eddress space. The termination task passes the RSM parameter list to a real storage management routine. This routine frees all pages and frames allocated to the terminating address space.	IEAVTMTC	
5 IEAVTMSI waits for the three tasks attached in steps 2, 3, and 4 to post the ECBs passed to them through register one. It then frees the space obtained in step 1. The three initialization ECBs remain to serve as end-of-task ECBs. Control is then returned to master scheduler region initialization (IEEMB860). **Error Processing** IEAVTMSI runs in an ESTAE environment. If the ESTAE exit is taken, it will pass a return code of 4 to master scheduler region initialization. If IEAVTMSI is posted by any of the three tasks with a completion code other than zero, it will pass a return code of 4 to region initialization. A return code of 4 causes region initialization to terminate the initialization. Each of the three attached tasks sets up its own ESTAE environment during its initialization phase. This environment allows the task to attempt recovery if it fails while processing a request. If any of these three tasks fails before its initialization phase is complete, the ESTAE routine that receives control will cause the abnormal termination to continue, thus causing the end-of-task ECBs to be posted.	IEAVTMSI	

Diagram 86. Auxiliary Storage Mangement Initialization (ILRTMI00) Part 1 of 4

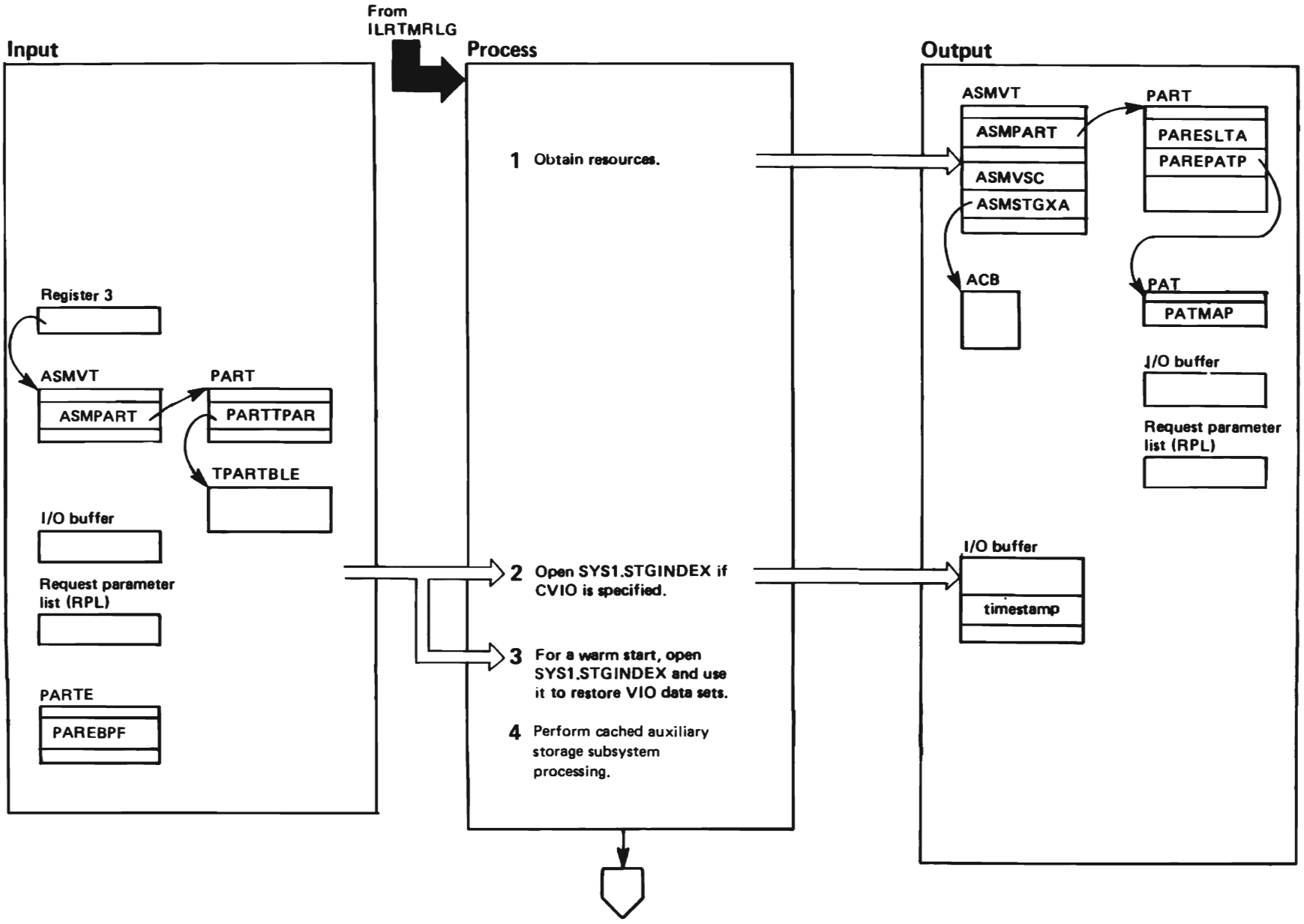


Diagram 86. Auxiliary Storage Mangement Initialization (ILRTMI00) Part 2 of 4

Extended Description	Module	Label
ILRTMI00 completes the initiallization process for ASM. It dynamically allocates, opens, and uses SYS1.STGINDEX; it builds and optionally displays the data set name list for all page and swap data sets; it backs the master scheduler address space with one megabyte of storage. ILRTMRLG, which is attached by IEEMB860, invokes ILRTMI00.	ILRTMI00	
1 ILRTMI00 issues the GETMAIN macro instruction to obtain storage from subpool 241 for an I/O buffer and an RPL (request parameter list) build area. ILRTMI00 issues another GETMAIN macro instruction to obtain storage for an ACB from subpool 245. ILRTMI00 dynamically allocates the SYS1.STGINDEX data set.	ILRTMI00	INITTMI
2 If the TPARTBLE warm-start flag is off (TPARWARM='0'), ILRTMI00 performs CVIO processing. a. ILRTMI00 issues a GENCB macro instruction to build the ACB for SYS1.STGINDEX. b. ILRTMI00 issues an OPEN macro instruction with the reuse option to reclaim all of SYS1.STGINDEX. c. ILRTMI00 copies the timestamp (TPARTIME) from the TPARTBLE to the I/O buffer. ILRTMI00 issues a GENCB macro instruction to build the RPL and then writes record zero of SYS1.STGINDEX. d. ILRTMI00 issues a CLOSE macro instruction for SYS1.STGINDEX; then reopen SYS1.STGINDEX for updating.	ILRTMI00	CVIOSTRT
3 If the TPARTBLE warm start flag is on (TPARWARM='1'), ILRTMI00 performs warm start processing. a. ILRTMI00 issues a GENCB macro instruction to build the ACB for SYS1.STGINDEX. b. ILRTMI00 opens and updates SYS1.STGINDEX. ILRTMI00 issues a GENCB macro instruction to build the RPL. ILRTMI00 issues a VERIFY macro instruction to update the VSAM catalog.	ILRTMI00	WARMSTRT
3 *(continued)* c. ILRTMI00 reads in the records from SYS1.STGINDEX. These records identify slots used for VIO data sets from the previous IPL. For each VIO data set, ILRTMI00 calculates the actual slots used by the VIO data set (to update ASMVSC). While reading these records, ILRTMI00 marks the slot in the appropriate local page data set's PAT (bit map of slots) as unavailable and updates the available slot count for this local page data set (PARESLTA). ILRTMI00 checks for VIO slots on unavailable local paging data sets. If any VIO slots were on local paging data sets and the data sets are not available for this IPL, then the warm start fails and a quick start is forced.		
4 ILRTMI00 scans all the PARTEs. If any data sets are on a cached auxiliary storage subsystem, ILRTMI00 does the following: • Page fixes ILRTMI00 to obtain the ASMLG lock. • Obtains the ASMGL lock to serialize the manipulation of the PART and PARTEs. • For a PARTE associated with a PLPA, common, or duplex data set, rechains the paging-mode IORB as the current IORB chain. • For a PARTE associated with a page data set, sets the PARESPP field to cause paging-mode to be used for I/O to the paging subsystem. • Releases the ASMGL lock. • Page frees ILRTMI00. • Turns off the warm start invalid bit (TPARTRAP) in the TPARTBLE. • Writes out the TPARTBLE.		

Diagram 86. Auxiliary Storage Management Initialization (ILRTMI00) Part 3 of 4

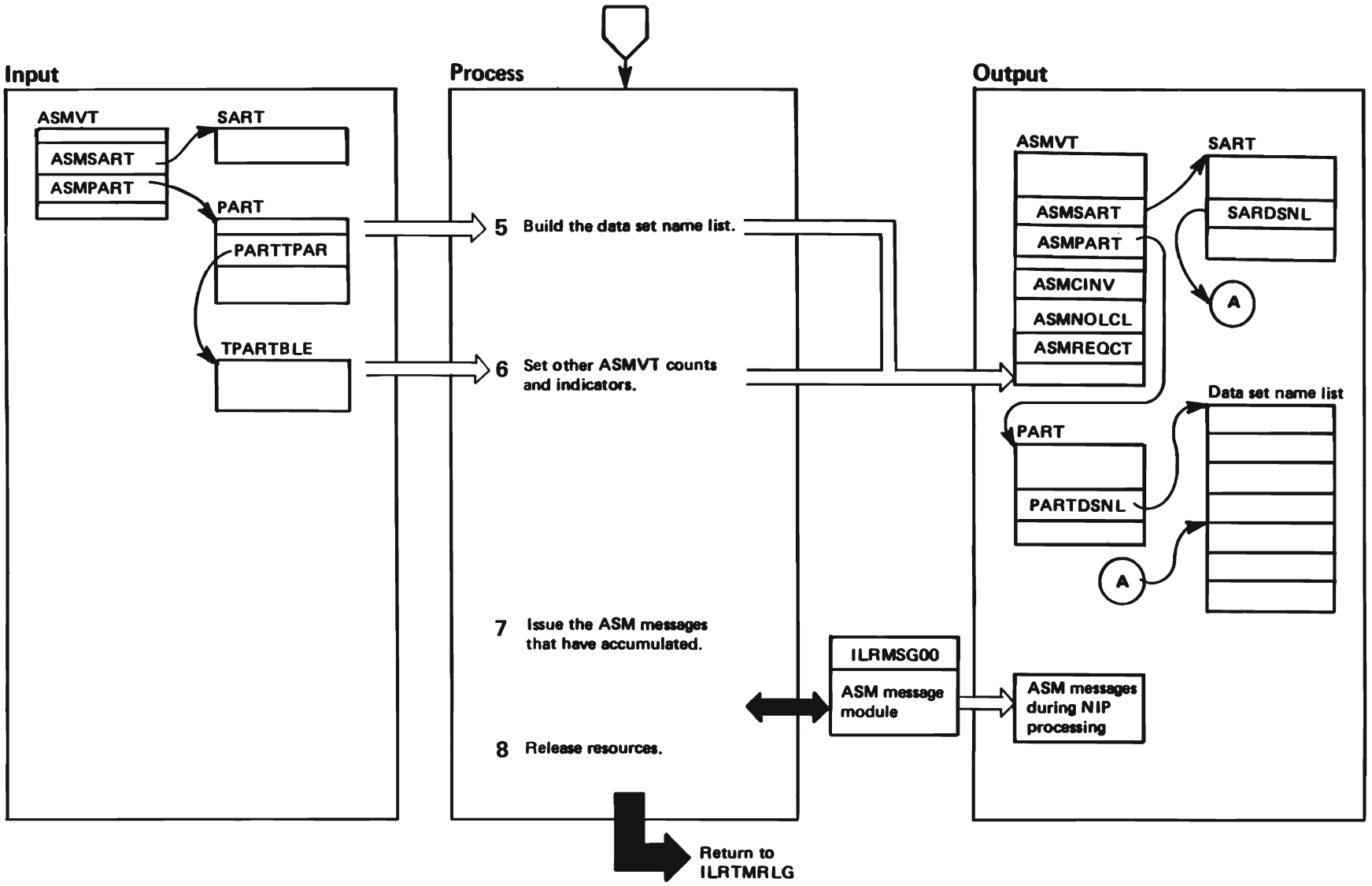


Diagram 86. Auxiliary Storage Mangement Initialization (ILRTMI00) Part 4 of 4

Extended Description	Module	Label
5 ILRTMI00 adds the total number of possible page and swap data sets and issues a GETMAIN macro instruction for subpool 241 storage for the data set name list. ILRTMI00 copies the local page data set names from the TPARTBLE to the data set name list. ILRTMI00 puts the address of the list in the PART. Copy the swap data set names from the TPARTBLE to the last section of the data set name list. ILRTMI00 also puts the address of this portion of the data set name list in the SART.	ILRTMI00	BUILDSNL
6 ILRTMI00 sets the ASMREQCT field to two, allowing the group operator starter routine of ASM to make only two simultaneous requests to VSAM. ILRTMI00 also sets the ASMCINV field equal to the number of control intervals in SYS1.STGINDEX (that is, the maximum number of records allowed). Now that VIO slots are currently marked unavailable (step 3). ILRTMI00 sets the ASMNOLCL bit off; this action allows slots to be allocated to local page data sets.	ILRTMI00	
7 ILRTMI00 invokes the ILRMSGSP entry point of ILRMSG00 to write out any ASM messages queued during NIP processing.	ILRMSG00	ILRMSGSP
8 ILRTMI00 issues a FREEMAIN macro instruction to free the I/O buffer RPL build area, the TPARTBLE, and the page storage table. ILRTMI00 then returns to the caller.	ILRTMI00	

Recovery Processing:

All recovery for ILRTMI00 is handled by module ILRTMI01. ILRTMI01 is an ESTAE established by ILRTMRLG to protect itself and those modules it calls, one of which is ILRTMI00. ILRTMI01 retries ILRTMI00 at one of three retry points.

ILRCRTM2 – used if activating warm start processing fails.
ILRCRTM3 – used if starting CVIO processing fails.
ILRCRTM4 – used if building data set name lists fails.

Diagram 87. Subsystem Initialization (IIEFJSIN2) Part 1 of 8

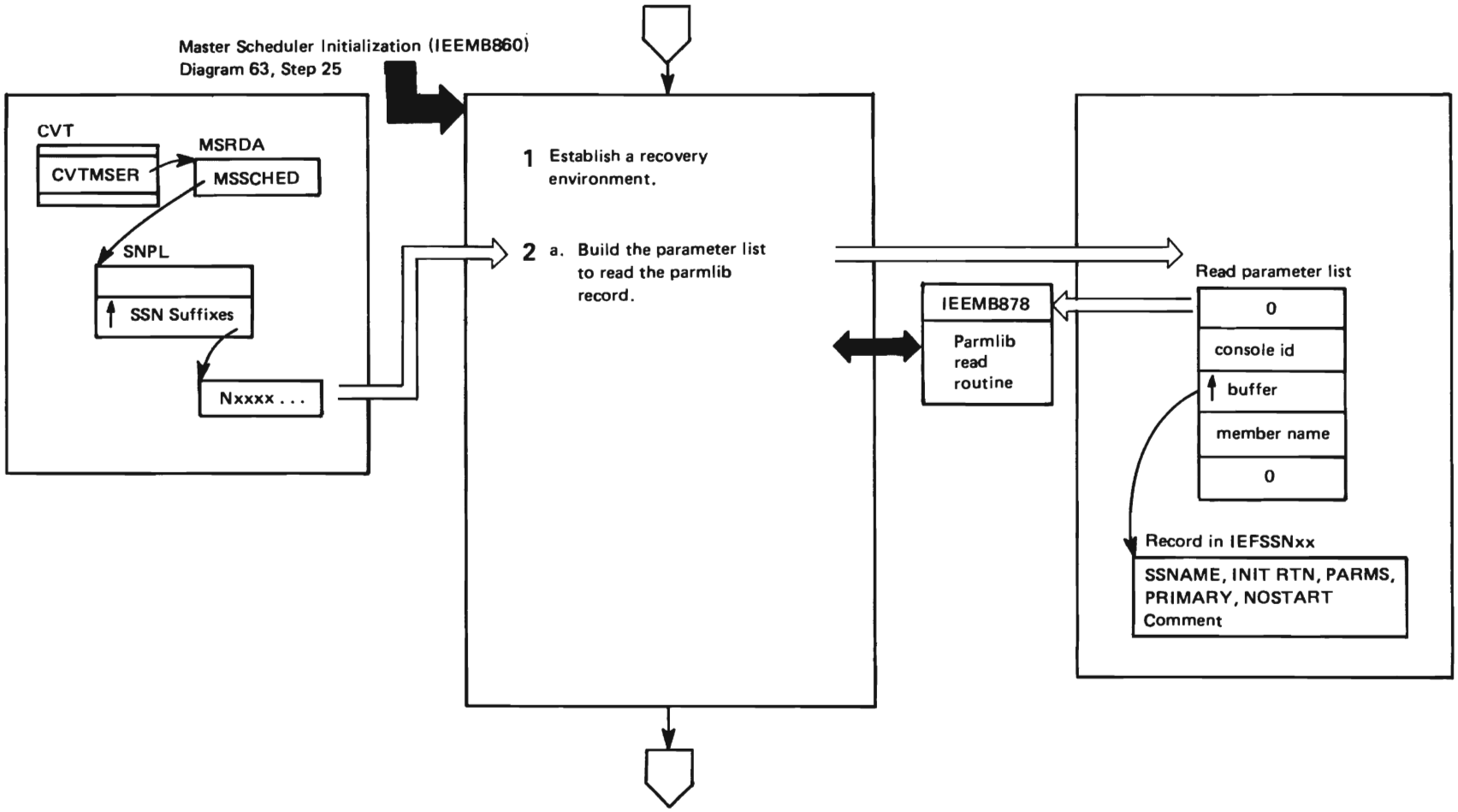


Diagram 87. Subsystem Initialization (IIEFJSIN2) Part 2 of 8

Extended Description	Module	Label
IEFJSIN2 initializes the subsystems identified in the IEFSSNxx members of SYS1.PARMLIB. If IEFSSNxx members are missing, IEFJSIN2 issues a message explaining this condition. IEFSSNxx contains two keywords: PRIMARY and NOSTART. PRIMARY allows the installation to specify a primary subsystem name and NOSTART indicates that the primary subsystem should not be automatically started. To specify NOSTART, you must specify PRIMARY. Also IEFJSIN2 uses the subsystem service routine (IEFJSBLD) to rebuild the subsystem allocation sequence table (SAST) and to build the hash table (SHAS) of SSCVT addresses.		
1 Establish the ESTAE environment.		
2 The master scheduler resident data area (MSRDA) points to the scheduler NIP parameter list (SNPL). If the SNPL indicates that there are any IEFSSNxx parmlib members to process (SNPSSN≠0), then IEFJSIN2 performs steps 2a-2e for each record in each IEFSSNxx parmlib member. After all members are processed, control passes to step 2f. The SSN parameters are in the format, Nxxxx . . . where N is the number of parmlib members and each xx is a two character suffix appended to IEFSSN that identifies a particular parmlib member. If there are no IEFSSNxx parmlib members to process (SNPSSN=0), go to step 7.	IEFJSIN2	
a. Call the parmlib read routine, IEEMB878, to read an 80-byte parmlib record. The input parameter list to IEEMB878 must contain a console id to which messages can be issued, the address of an 80-byte buffer, and the name of the parmlib member to be read.	IEEMB878	

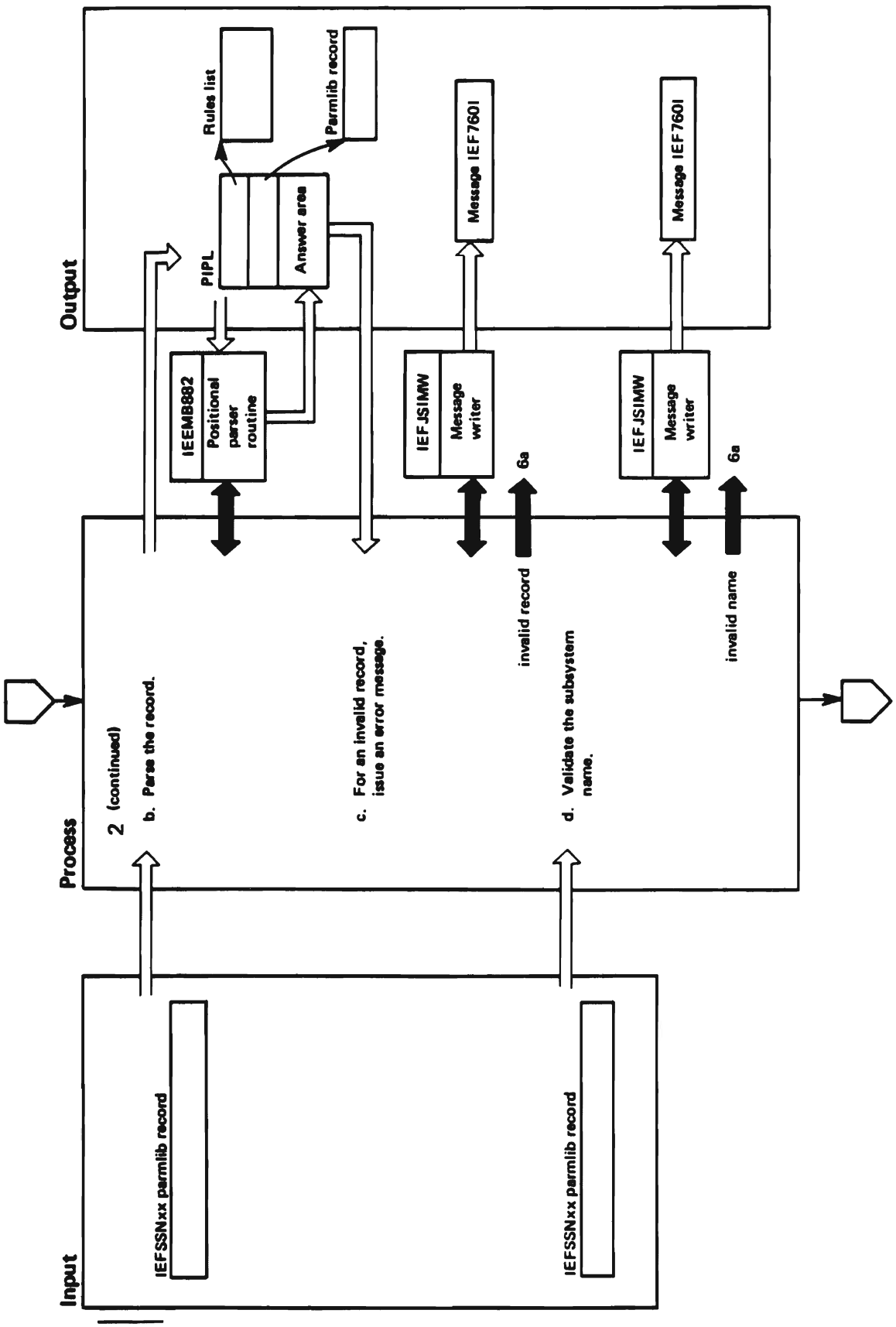


Diagram 87. Subsystem Initialization (IIEFJSIN2) Part 3 of 8

Diagram 87. Subsystem Initialization (IIEFJSIN2) Part 4 of 8

Extended Description	Module	Label
2 (continued)		
b. Store the address of the parmlib record and the rules list in the input parameter list to IEEMB882, the positional parser routine. Invoke IEEMB882 to check the syntax of the parmlib record. IEEMB882 verifies that the parameters on the record comply with the rules list for such items as the number of parameters on the record, the minimum and maximum length of each parameter, whether the parameter is optional or mandatory, and whether the parameter can be enclosed in apostrophes. (For a description of IEEMB882 see *MVS System Logic Library*.)	IEEMB882	PLIBPROC
c. If the return code from IEEMB882, located in the answer area of the positional parser parameter list (PIPL), indicates either that an invalid parameter exists or that there are too many parameters in the record, call the message writer, IEFJSIMW, passing the console id, the number of lines in the message, and a pointer to the message data for message IEF760I. The message indicates that the parmlib record is in error and includes the first 70 bytes of the record in error. After issuing the message continue processing at step 6a to process the next parmlib record. If the parmlib record is valid, proceed to step 6d.	IEFJSIN2 IEFJSIMW	PLIBPROC
d. Verify that the first character of the subsystem name is an alphabetic or national character and that the remaining characters are alphameric or national characters. If the subsystem name is not valid, invoke the message writer, IEFJSIMW, to issue message IEF760I. After issuing the message, continue processing at steps 6a to process the next parmlib record. If the subsystem name is valid, proceed to step 6e.	IEFJSIN2 IEFJSIMW	VALSSNME

Diagram 87. Subsystem Initialization (IIEFJSIN2) Part 5 of 8

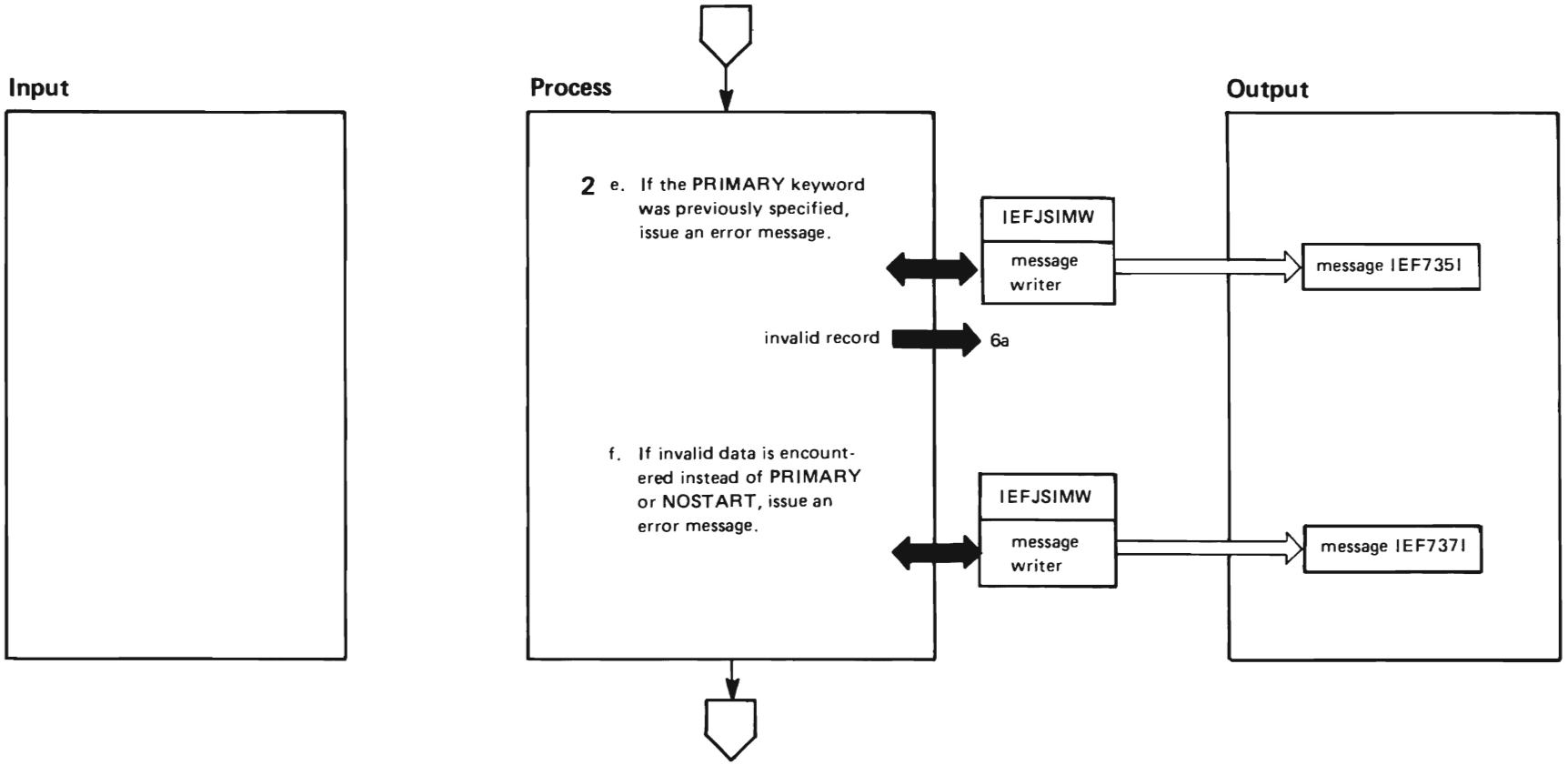


Diagram 87. Subsystem Initialization (IIEFJSIN2) Part 6 of 8

Extended Description	Module	Label
e. If the PRIMARY keyword is specified, determine if JESPJESN contains zeroes indicating it is not initialized. If it does, initialize JESPJESN with the primary subsystem name spcified in the IEFSSNxx member. If it does not contain zeroes, indicating it is initialized, invoke IEFJSIMW to issue message IEF735I. This message indicates that PRIMARY has been previously specified.		
f. If invalid date is encounted in place of the PRIMARY or or NOSTART keywords, invoke IEFJSIMW to issue message IEF737I. This message indicates that unrecognized data was encountered and that it is ignored.	IEFJSIMW	

Diagram 87. Subsystem Initialization (IIEFJSIN2) Part 7 of 8

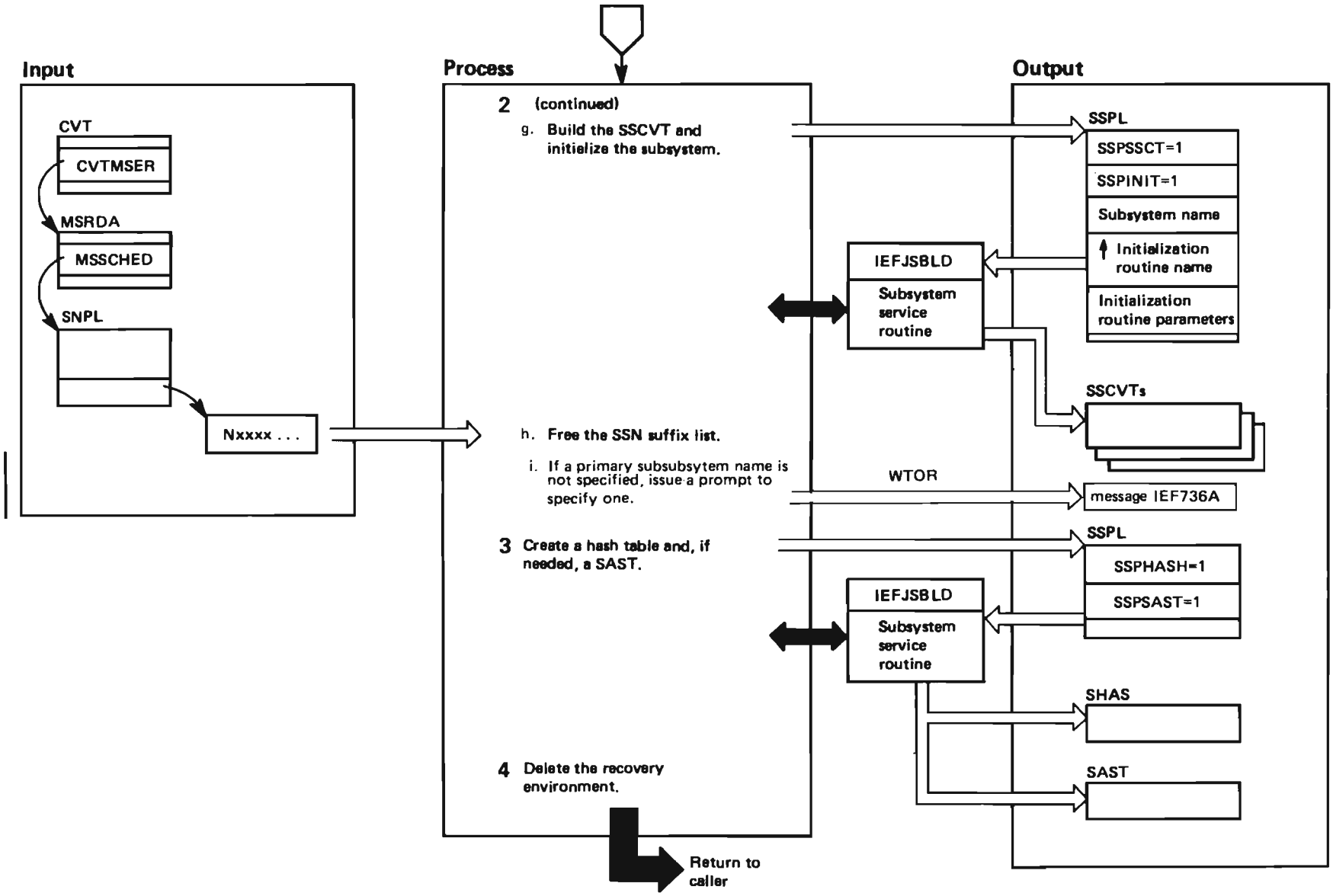


Diagram 87. Subsystem Initialization (IIEFJSIN2) Part 8 of 8

Extended Description	Module	Label
2 (continued)		
g. Initialize the parameter list to indicate that an SSCVT be built and that the subsystem initialization routine get control via a LINK macro instruction. Include the address of the initialization routine name and the address of initialization routine parameters that were obtained from the parmlib record. Then call IEFJSBLD. After IEFJSBLD returns, proceed back to 6a to process the next parmlib record in IEFSSNxx.	IEFJSIN2 IEFJSBLD	PLIBPROC
h. Issue a FREEMAIN macro instruction for the SSN suffix list.	IEFJSIN2	FREESSN
i. If a PRIMARY keyword is not specified, issue message IEF736A which prompts the system operator to name a primary subsystem.		
3 Initialize the parameter list to indicate that a subsystem hash table (SHAS) and, if additional subsystems were specified in parmlib, a subsystem allocation sequence table (SAST) needs to be built. The hash table contains SSCVT addresses; it is used by the subsystem interface to locate SSCVTs. (The SAST is used in the allocation of subsystem data sets.) Invoke IEFJSBLD to build the SAST.	IEFJSIN2 IEFJSBLD	BLDTABLS
4 Return to the caller of IEFJSIN2, master scheduler initialization (IEEMB860), after deleting the ESTAE environment.		

Recovery Processing

If RTM invokes the ESTAE recovery routine as a result of an error in IEFJSIN2, the ESTAE recovery routine:

- Re-establishes the code and data registers.
- Requests retry unless a previous retry attempt was made, the error is a percolated one, or the error is a machine check; in these cases, the ESTAE requests RTM to continue termination.
- Records appropriate diagnostic information in the variable recording area of the SDWA.
- Takes an SDUMP only if the error is not a machine check or percolated error.

The retry routine cleans up IEFJSIN2 processing and issues message IEF758I to inform the operator thet an ABEND occurred during subsystem initialization.

Diagram 88. Display Allocation Scavenge Routine (IEFHB412) Part 1 of 2

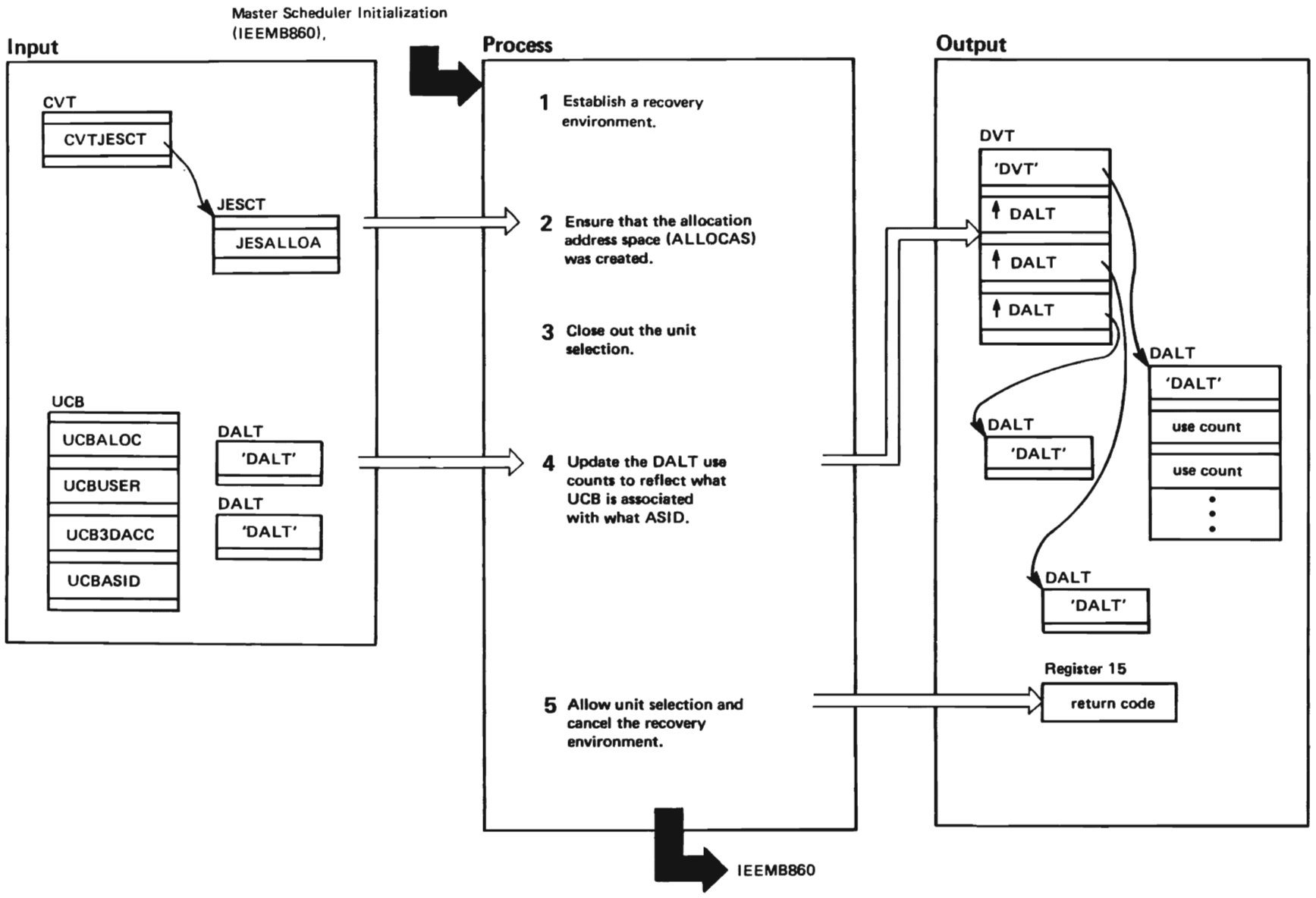


Diagram 88. Display Allocation Scavenge Routine (IEFHB412) Part 2 of 2

Extended Description	Module	Label
The display allocation scavenge routine (IEFHB412) scans all the UCBs in the system and associates the allocation use count in each UCB with a particular address space or a number of address spaces.		
1 Set up an ESTAE environment. IEFAB4E6 is the ESTAE.	IEFHB412	ESTAPROC
2 Check the JESALLOA field in the JESCT to determine if the allocation address space (ALLOCAS) was created. If it has not been created, then set a return code of 8 and return to the caller.	IEFHB412	
3 Otherwise, issue an ENQ macro instruction to request shared use of the SYSZTIOT resource and exclusive use of the Q4, DDRTPUR, and DDRDA resources.	IEFHB412	
4 Scan the UCBs using IOS scan service (IOSVSUCB). Assign all allocated non-direct access devices to the diaplay allocation lookup table (DALT) for the ASID found in the UCB field UCBASID. For all other allocated UCBs (that is, those in use), determine what use count is already stored in the DALTs. If the use count is zero, then store a one in the use count field of the DALT. Add the difference to the DALT for the pseudo ASID, ASID 0. (ASID 0 is the pseudo ASID for a DALT that contains system-related allocation counts.) Note that all references and updates to the DALTs are made by issuing a program call (PC) to IEFHB410, the display allocation tables manager.	IEFHB412	SCAVENGE
5 Release the resources acquired in step 3 by issuing a DEQ macro instruction. Cancel the ESTAE environment and return to the caller.	IEFHB412	

Recovery Processing

See Diagram 70 for information on recovery processing.

SECTION 6. MODULE DESCRIPTIONS

IPL RIMS

IEAVNIPL00 knows a RIM by the name 'IEAVIPLxx', where 'xx' is the entry in the suffix list. This name might be an entry point of a module that has a different name. The following table gives the names of the RIMs in order of invocation, the module name, and the corresponding component.

RIM Name	Component Name
IEAIPL10	IPL
IEAIPL20	IPL
IEAIPL30	IPL
IEAIPL40	IOS
IEAIPL41	IPL
IEAIPL05	IPL
IEAIPL02	IPL
IEAIPL04	IPL
IEAIPL06	VSM
IEAIPL07	Supervisor Control
IEAIPL03	IOS
IEAIPL99	IPL

NIP RIMS

IEAVNIPM knows a RIM by the name 'IEAVNPxx', where 'xx' is the entry in the suffix list. This name might be an entry point of a module that has a different name. The following table gives the names of the RIMs in order of invocation, the module name, and the corresponding component. If there is no entry under Module Name, the RIM Name and the module name are identical.

RIM Name	Module Name	Component Name
IEAVNPE6		Service Processor
IEAVNPA6		RTM
IEAVNP06	IGFRIM00	MCH
IEAVNP27		Reconfiguration
IEAVNPA2	IEAVNP02 (entry point IEAVNPA2)	IOS
IEAVNPC1		NIP
IEAVNPB2	IEAVNP02 (entry point IEAVNPB2)	IOS
IEAVNP11		VSAM
IEAVNP76		NIP
IEAVNP03		NIP
IEAVNPE8	IARMU (entry point IARMURIM)	RSM
IEAVNP23	ISGNCBIM	Global resource serialization
IEAVNP04	ILRASRIM	ASM
IEAVNPA8	IEAVNP08 (entry point IEAVNPA8)	VSM
IEAVNP14	ILRASRM1	ASM
IEAVNP25		Supervisor control
IEAVNP05		Contents supervisor
IEAVNPB8	IEAVNP08 (entry point IEAVNPB8)	VSM
IEAVNP47		ENF

RIM Name	Module Name	Component Name
IEAVNPD6		RTM
IEAVNP09		Supervisor control
IEAVNPD8	IARMN IARMS IARMT IARMF (entry point IARMNRIM)	RSM
IEAVNP10		SRM
IEAVNP1F		SRM
IEAVNPD1	IEAVTABI	ABDUMP
IEAVNPD2	IEAVTSDI (entry point IEAVTSDI)	SVCDUMP
IEAVNPX1		NIP
IEAVNPF5		PC/AUTH
IEAVNP51		System trace
IEAVNP33	ISGNTASC	Global resource serialization
IEAVNP57		Dumping services
IEAVNPA1		Communications task
IEAVNP16		Data management
IEAVNP13		Master scheduler
IEAVNP17		GTF
IEAVNP18		Master scheduler
IEAVNP19		MSSC
IEAVNP20		Content supervisor
IEAVNPF2		IOS
IEAVNP15	IEAVAP00	Allocation
IEAVNP1B		VSAM
IEAVNP00		Reconfiguration
IEAVNIPX		NIP

IARMF (entry point IARMFRIM) — RSM Extended Storage Control Block Routine

OPERATION: Builds and initializes extended storage control blocks and queues that RSM uses to control any extended storage that might be configured in the system. IARMFRIM is in load module IEAVNPD8.

ENTRY FROM: IEAVNPD8

EXIT: Returns to the caller.

IARMI (entry point IARMIIPL) — RSM Initial Program Loader

OPERATION: Initializes the page frame table (PFT) and the RSM address space block (RAB), and maps the existing segment table into ELSQA and the existing page tables into ESQA and ELSQA. IARMIIPL is in load module IEAIPL06.

ENTRY FROM: IEAIPL00

EXIT: Returns to the caller.

IARMN (entry point IARMNRIM) — RSM Initialization Routine

See the module description for IEAVNPD8.

IARMS (entry point IARMSRIM) — RSM System Parameter Routine

OPERATION: Completes processing of the REAL and VRREGN system parameters.

ENTRY FROM: IEAVNPD8

EXIT: Returns to the caller.

IARMT (entry point IARMTRIM) — RSM RSU Parameter Routine

OPERATION: Processes the RSU system parameter.

ENTRY FROM: IEAVNPD8

EXIT: Returns to the caller.

IARMU (entry point IARMURIM) — RSM REAL Parameter Routine

OPERATION: Sets up an initial V=R region for the REAL parameter.

ENTRY FROM: IEAVNIPM

EXIT: Returns to the caller.

IARVB (entry point IARVBCSG) — RSM Create Segment Support

OPERATION: Creates a segment of common storage by formatting a page table for that segment.

ENTRY FROM: IEAVNPA8, IEAVNPB8.

EXIT: Returns to the caller.

IARXT (entry point IARXTIGU) — RSM Contiguous Real Support

OPERATION: Backs the virtual range specified by VSM with contiguous preferred real storage.

ENTRY FROM: IGVNIPCR.

EXIT: Returns to the caller.

IEAIPL00 — Initial Program Loader

OPERATION: Clears real storage and registers, prepares an environment in which the IRIMs can execute, provides a set of common service routines for the IRIMs, and finds, passes control to, and deletes the IRIMs.

ENTRY FROM: The operator when he or she activates the load process.

EXIT: IEAIPL99.

ERROR EXIT TO: System wait state.

IEAIPL01 — IRIM List

OPERATION: List of IDs of IPL resource initialization modules to be loaded by IEAIPL00.

ENTRY FROM: Not applicable (non-executable module).

EXIT: Not applicable (non-executable module).

IEAIPL02 — Nucleus Load Routine

OPERATION: Loads the DAT-off nucleus and the DAT-on nucleus.

ENTRY FROM: IEAIPL00.

EXIT: Returns to the caller.

IEAIPL03 — Initializes UCB for System Resident Volume

OPERATION: Initializes the UCB and subchannel for the device from which the system was IPLed, builds the resident ERP table and sets the DDT pointers in the UCBs.

ENTRY FROM: IEAIPL00.

EXIT: Returns to the caller.

IEAIPL04 — Virtual Storage Management IRIM

OPERATION: Allocates storage for the page frame tables, the system queue area (SQA), the extended SQA (ESQA), and the extended local system queue area (ELSQA).

ENTRY FROM: IEAIPL00.

EXIT: Returns to the caller.

IEAIPL05 — Nucleus Map Creation

OPERATION: Creates a map of all modules and entry points in the DAT-on nucleus.

ENTRY FROM: IEAIPL00.

EXIT: Returns to the caller.

IEAIPL06 — Real Storage Management IRIM

OPERATION: Initializes the page frame table, the RSM internal table (RIT), the RSM address space block (RAB), and the RAB extension (RAX).

ENTRY FROM: IEAIPL00.

EXIT: Returns to the caller.

IEAIPL07 — SVC and ESR Table Amode Bit Update

OPERATION: Updates the AMODE indicators for nucleus resident SVC routines in the SVC and ESR tables.

ENTRY FROM: IEAIPL00.

EXIT: Returns to the caller.

IEAIPL10 — Service Processor Interface Routine

OPERATION: Invokes the SCP INFO functions of the service processor.

ENTRY FROM: IEAIPL00.

EXIT: Returns to the caller.

IEAIPL20 — IPL Storage Validation Routine

OPERATION: Validates the system storage and places all valid frames on the available frame queue.

ENTRY FROM: IEAIPL00.

EXIT: Returns to the caller.

IEAIPL30 — IPL IRIM that loads the IPL WTO Facility Module

OPERATION: Obtains storage for MQH (message queue header) and IEAIPL35, then loads IEAIPL35 into real storage.

ENTRY FROM: IEAIPL00.

EXIT: Returns to the caller.

IEAIPL35 — IPL WTO (Write to Operator) Facility Module

OPERATION: Places a message passed by the caller on the IPL/NIP message queue.

ENTRY FROM: IPL resource initialization modules.

EXIT: Returns to the caller.

IEAIPL40 — Device Support Module List

OPERATION: Builds the device support modules list.

ENTRY FROM: IEAIPL00

EXIT: Returns to the caller.

IEAIPL41 — DAT-ON Nucleus Load Table Builder

OPERATION: Builds a nucleus load list element for the DAT-on nucleus, prepares the nucleus load list for IEAIPL02 to load the DAT-on nucleus into virtual storage.

ENTRY FROM: IEAIPL00

EXIT: Returns to the caller.

IEAIPL99 — IPL Cleanup IRIM

OPERATION: Performs IPL cleanup, allocates and initializes the address increment map (AIM), sets the prefix to the address of SYSGEN PSA, and transfers control to IEAVNIP0.

ENTRY FROM: IEAIPL00.

EXIT: IEAVNIP0 via LPSW.

IEAVAP00 — Volume Attribute RIM

OPERATION: Sets status of permanently resident DASD volumes to public, sets status of devices without volumes to not-ready, and processes the volume attributes specified via the VAL system parameter.

ENTRY FROM: IEAVNIPM.

EXIT: Returns to the caller.

IEAVM200 — Generalized Message Service Module for SYS1.PARMLIB

Processing

OPERATION: Builds the requested message text for immediate issuance or for queueing.

ENTRY FROM: IEAVN600, IEAVN601, IEAVN602, IEAVN610, IEAVN611, IEAVN612, IEAVN613, IEEMB888, and IEEMB889.

EXIT: Returns to the caller.

IEAVNIPM — NIP Control and Service Routine

OPERATION: Initializes the NIP vector table (NVT); provides diagnostic support for software and hardware failures that occur during NIP before system recovery is available; finds, loads, and deletes NIP RIMs; finds and loads IEAVNIPX; contains the NIPOPIO, NIPPRMPT, NIPREAL, NIPSENSE, NIPSWAIT, and NIPUCBFN service routines; establishes initial operator communications and checks the operator's reply.

ENTRY FROM: IEAVNIP0.

EXIT: Returns to the caller.

IEAVNIPX — NIP Exit Processing

OPERATION: Deletes all NIP control blocks and all system traps.

ENTRY FROM: IEAVNIPM.

EXIT: LINK to IEEVIPL.

IEAVNIP0 — IPL/NIP Interface Routine

OPERATION: Performs post-IPL housekeeping, allocates and initializes global system control blocks, and sets up an environment in which the RIMs can run.

ENTRY FROM: IEAIPL99.

EXIT: IEAVNIPM.

IEAVNPA1 — Communications Task RIM

OPERATION: Verifies CON= system parameter and passes control to the front end processor for CONSOLxx member of SYS1.PARMLIB (IEAVN600).

ENTRY FROM: IEAVNIPM.

EXIT: Returns to the caller.

IEAVNPA2 — Non-Direct Access Devices Initialization

OPERATION: Initializes all subchannels and verifies paths to non-direct access devices.

ENTRY FROM: IEAVNIPM.

EXIT: Returns to the caller.

IEAVNPA5 — Contents Supervisor APF Table Build

OPERATION: Builds the APF table using user-defined and system-defined authorized libraries.

ENTRY FROM: IEAVNP03.

EXIT: Returns to the caller.

IEAVNPA6 — System Recovery Initialization

OPERATION: Initializes RTM for software recovery by

- Initializing the RTCT.
- Acquiring a quickcell pool for RTM use.
- Initializing the recording buffer for the RTM recording task.

ENTRY FROM: IEAVNIPM.

EXIT: Returns to the caller.

IEAVNPA8 — SQA Parameter Initialization

OPERATION: Processes the SQA parameter and builds the VSM control blocks and calls real storage management (RSM) to build page tables necessary to describe the new SQA and extended SQA storage.

ENTRY FROM: IEAVNIPM.

EXIT: Returns to the caller.

IEAVNPB2 — Direct Access Device Initialization

OPERATION: Verify paths to non-virtual direct access devices.

ENTRY FROM: IEAVNIPM.

EXIT: Returns to the caller.

IEAVNPB8 — CSA Parameter Initialization

OPERATION: Processes the CSA parameter, builds the VSM control blocks, and calls real storage management (RSM) to build the page tables necessary to describe the new CSA and extended CSA areas.

ENTRY FROM: IEAVNIPM.

EXIT: Returns to the caller.

IEAVNPC1 — NIP Console Initialization Routine

OPERATION: Find a valid console in the NIP Console Table (NCT).

ENTRY FROM: IEAVNIPM.

EXIT: Returns to the caller.

IEAVNPC2 — Input/Output Supervisor Initialization

OPERATION: Verifies paths to mass storage system (MSS) devices.

ENTRY FROM: IEAVNP19.

EXIT: Returns to the caller.

IEAVNPC5 — Contents Supervision Extended PLPA Creation

OPERATION: Creates the extended PLPA, EPLPA, and LPA directory. IEAVNPC5 is only called on cold start processing.

ENTRY FROM: IEAVNP05.

EXIT: Returns to the caller.

IEAVNPC8 — LPA and ELPA Page Table Initialization

OPERATION: Calls real storage management (RSM) to build the page tables for LPA and ELPA and marks the space that a GETMAIN macro instruction assigned.

ENTRY FROM: IEAVNP05, IEAVNPC5, and IEAVNP04/ILRASRIM.

EXIT: Returns to the caller.

IEAVNPD1 — ABDUMP Initialization Routine

OPERATION: Initializes the ABDUMP. See the module description for IEAVTABI.

ENTRY FROM: IEAVNIPM.

EXIT: Returns to the caller.

IEAVNPD2 — SVC Dump Initialization Routine

OPERATION: Initializes the SVC dump. See the module descriptions for IEAVTSDD and IEAVTSDI.

ENTRY FROM: IEAVNIPM.

EXIT: Returns to the caller.

IEAVNPD6 — Task Recovery Initialization

OPERATION: Completes the initialization for RTM by placing the branch entry address of SVC 60 in the CVT.

ENTRY FROM: IEAVNIPM.

EXIT: Returns to the caller.

IEAVNPD8 — REAL, VRREGN, and RSU Initialization

OPERATION: Initializes RSM control blocks and processes the REAL, VRREGN and RSU system parameters.

ENTRY FROM: IEAVNIPM.

EXIT: Returns to the caller.

IEAVNPE6 — Service Processor CALL SVC Initialization

OPERATION: Initializes the software control blocks required when using the service processor CALL SVC to interface with the hardware's service processor. The control blocks initialized are the service processor control block (MSFCB) and its buffer, the service processor attention block (MSFAB) and its buffer, the service processor communication block (MSFKB) and its buffer, and a service processor recovery buffer.

ENTRY FROM: IEAVNIPM.

EXIT: Returns to the caller.

IEAVNPE8 — REAL Parameter Initialization

OPERATION: Processes the REAL parameter, marks all the page frame table entries (PFTEs) for frames in the V=R region, and marks the address increment map entries (AIMEs).

ENTRY FROM: IEAVNIPM.

EXIT: Returns to the caller.

IEAVNPF2 — Input/Output Supervisor RIM

OPERATION: Processes the IECIOSxx member of SYS1.PARMLIB; IECIOSxx specifies time intervals to be used by the missing interrupt handler (MIH) when it scans UCBs for missing interrupt conditions. IEAVNPF2 also allows an installation to change the hot I/O threshold or default recovery actions.

ENTRY FROM: IEAVNIPM.

EXIT: Returns to the caller.

IEAVNPF5 — Program Call/Authorization RIM

OPERATION: Causes the PC/AUTH address space to be initialized.

ENTRY FROM: IEAVNIPM.

EXIT: Returns to the caller.

IEAVNPM2 — NIP WTOR and WTO Service Routines

OPERATION: Writes NIP messages to the master operator's console; for WTOR, reads the replies associated with the WTOR messages.

ENTRY FROM: IEAVNIPM and NIP RIMs.

EXIT: Returns to the caller.

IEAVNPM3 — NIPOPEN and NIPMOUNT Service Routines

OPERATION: Provides a general direct access and tape device mount facility and a general direct access and tape data set open facility.

ENTRY FROM: IEAVNIPM and NIP RIMs.

EXIT: Returns to the caller.

IEAVNPM4 — NIP SYS1.PARMLIB Member Read Routine

OPERATION: Locates and reads text records of SYS1.PARMLIB members.

ENTRY FROM: IEAVNIPM and NIP RIMs.

EXIT: Returns to the caller.

IEAVNPM5 — NIP Console Attention Routine

OPERATION: Posts the NIP Console Table ECB whenever the NIP console issues attention. Otherwise, the attention is ignored.

ENTRY FROM: IOS when IOS has received an attention.

EXIT: Returns to the caller.

IEAVNPST — SVC Table Initialization for Program Products

OPERATION: Determines if certain access methods or program products are installed and, if so, sets up the SVC table entries they require.

ENTRY FROM: IEAVNPS5

EXIT: Returns to the caller.

IEAVNPS5 — Supervisor Control SVC Table Initialization

OPERATION:

1. Initializes the system and ESR SVC tables, resolving the entry point addresses of LPA-resident SVC routines.
2. Completes the installation of LPA-resident user-SVC routines specified in SYS1.PARMLIB member(s) IEASVCxx.
3. Calls IEAVNPST to perform SVC table initialization for program products.
4. Sets any SVC table entries with unresolved entry point addresses to IGCERROR.

ENTRY FROM: IEAVNP05.

EXIT: Returns to the caller.

IEAVNP00 — Reconfiguration RIM

OPERATION: Invokes IEAVRTOD to initialize the time-of-day (TOD) clock on the IPL processor, writes the 3081 console restart text to the console screen of the IPL processor, and invokes IEEVCPR to bring the non-IPL processor online.

ENTRY FROM: IEAVNIPM.

EXIT: Returns to the caller.

IEAVNP03 — System Parameter Analysis and SYS1.LINKLIB Initialization

OPERATION

- Opens SYS1.PARMLIB.
- Analyzes and merges system parameters from the operator's console and IEASYSxx in SYS1.PARMLIB.
- Opens SYS1.LINKLIB and its concatenated data sets specified in member LNKLSTxx of SYS1.PARMLIB.
- Prompts the operator for new specifications of system parameters.

ENTRY FROM: IEAVNIPM.

EXIT: Returns to the caller.

IEAVNP04 — Auxiliary Storage Management RIM, Part 1

See the module description for ILRASRIM.

IEAVNP05 — Link Pack Area Initialization RIM

OPERATION: Performs resource initialization for the Contents Supervisor at IPL time. It builds the pageable link pack area for cold start processing, the fixed link pack area and the modified link pack area. IEAVNP05 initializes the SVC table for the entry point addresses of type 3 and 4 SVCs and the extended router (ESR). This module is also used, prior to the PLPA being initialized, in order to build the LPALST library concatenation.

ENTRY FROM: IEAVNIPM.

EXIT: Returns to the caller.

IEAVNP06 — Machine Check Handler RIM

OPERATION: Initializes control registers 14 and 15 and the machine check handler control blocks for the IPL processor.

ENTRY FROM: IEAVNIPM.

EXIT: Returns to the caller.

IEAVNP08 — Virtual Storage Management RIMs

See the module descriptions for:

IEAVNPA8 — SQA Parameter Process
IEAVNPB8 — CSA Parameter Process
IEAVNPC8 — LPA and ELPA Page Table Initialization

IEAVNP09 — Supervisor Control RIM

OPERATION: IEAVNP09 does the following:

- Initializes the address space vector table (ASVT) with the maximum number of address spaces (specified in the MAXUSER, RSVNONR, and RSVSTRT system parameters).
- Initializes an ASVT entry for each possible address space.
- Initializes each ASVT entry reserved for non-reusable address spaces.
- Initializes each ASVT entry reserved for START/SASI address spaces.
- Builds the three available entry queues.
- Initializes the address space first tables (AFTs) and the address space second tables (ASTs) for the maximum number of address spaces.

ENTRY FROM: IEAVNIPM.

EXIT: Returns to the caller.

IEAVNP1A — VSAM-Data-Set-Open Service Routine

OPERATION: Constructs VSAM control blocks chained off the ACB for VSAM data sets and opens VSAM data sets.

ENTRY FROM: IEAVNP11.

EXIT: Returns to the caller.

IEAVNP1B — Master Catalog Utility RIM

OPERATION: Closes the system catalog by freeing storage used by the catalog control blocks, except that used for AMCBS.

ENTRY FROM: IEAVNIPM, IEAVNP11, IEAVNP12.

EXIT: Returns to the caller.

IEAVNP1F — System Resources Management Channel Measurement Initialization

OPERATION: Builds and initializes the SRM measurement functions.

ENTRY FROM: IEAVNIPM.

EXIT: Returns to the caller.

IEAVNP10 — System Resources Manager RIM

OPERATION: Initializes the system resources manager (SRM) using the APG, OPT, ICS, and IPS system parameters.

ENTRY FROM: IEAVNIPM.

EXIT: Returns to the caller.

IEAVNP11 — Master Catalog Open RIM

OPERATION: Locates the system catalog, then uses IEAVNP1A to open the system catalog.

ENTRY FROM: IEAVNIPM.

EXIT: Returns to the caller.

IEAVNP12 — NIP LOCATE Service Routine

OPERATION: Locates data sets that are cataloged in the system catalog.

ENTRY FROM: IEAVNIPM, IEAVNP03, IEAVNP05, IEAVNP11, ILROPS00.

EXIT: Returns to the caller.

IEAVNP13 — Master Scheduler Initialization Interface RIM

OPERATION: Prepares the interface between NIP and master scheduler initialization by:

- Obtaining the SMF system parameter values and storing them.
- Obtaining the LOGCLS and LOGLMT system parameter values and storing them.
- Storing the suffixes of the SSN members of SYS1.PARMLIB.
- Reading the IEACMD00 and COMMNDxx members of SYS1.PARMLIB and storing the automatic commands.
- Obtaining the suffix value for the MSTJCLxx member of SYS1.LINKLIB used to start the master scheduler address space.

ENTRY FROM: IEAVNIPM.

EXIT: Returns to the caller.

IEAVNP14 — Auxiliary Storage Management RIM, Part 2

See the module description for ILRASRM1.

IEAVNP15 — Volume Attribute RIM

See the module description for IEAVAP00.

IEAVNP16 — Data Management RIM

OPERATION: Builds table of user-written EXCP appendages that may be used by unauthorized programs, builds CVAF table, processes the RER and RDE system parameters. and branches to event notification facility (ENF) to establish listeners for DASD volume unload events and SQA shortage events.

ENTRY FROM: IEAVNIPM.

EXIT: Returns to the caller.

IEAVNP17 — Generalized Trace Facility RIM

OPERATION: Initializes GTF's interface with monitor call by placing the addresses of the MC routing module, the MC routing module's functional recovery routine, the two SETEVENT routines, and the cross-address-space-read module into MCHEAD.

ENTRY FROM: IEAVNIPM.

EXIT: Returns to the caller.

IEAVNP18 — Scheduler Interface to the General Parmlib Scan Routine

OPERATION: Initializes a parameter list (IEEZB819) and invokes the General Parmlib Scan Routine, (IEEMB888), to process the SCHEDxx member of SYS1.PARMLIB.

ENTRY FROM: IEAVNIPM.

EXIT: Returns to the caller.

IEAVNP19 — Mass Storage System Communicator RIM

OPERATION: Initializes the software support for mass storage system communicator (MSSC) for mounting and demounting mass storage volumes on the 3850. (See Mass Storage System Communicator Logic: MSS Communicator (MSSC) for details.)

ENTRY FROM: IEAVNIPM.

EXIT: Returns to the caller.

IEAVNP20 — CLOCK Processing Resource Initialization Module

OPERATION: Processes the CLOCKxx member of SYS1.PARMLIB.

ENTRY FROM: IEAVNIPM

EXIT: Returns to caller.

IEAVNP23 — Global Resource Serialization Control Block Initialization

See the module description for ISGNCBIM.

IEAVNP25 — SVC Parmlib Processing IRIM

OPERATION: Processes the IEASVCxx member(s) of SYS1.PARMLIB. Nucleus-resident routines are installed immediately. Specifications for LPA-resident user-SVCs are saved in a table until IEAVNP05 builds the LPA.

ENTRY FROM: IEAVNIPM

EXIT: Returns to the caller.

IEAVNP27 — Reconfiguration Installed Resources RIM

OPERATION: Obtain and initialize the installation channel path table (ICHPT) and the configuration management table (CMT).

ENTRY FROM: IEAVNIPM.

EXIT: Returns to the caller.

IEAVNP33 — Global Resource Serialization Address Space Creation

See the module description for ISGNTASC.

IEAVNP47 — Event Notification Facility RIM

OPERATION: Initializes the control blocks required for using the event notification facility (ENF).

ENTRY FROM: IEAVNIPM.

EXIT: Returns to the caller.

IEAVNP51 — System Trace RIM

OPERATION: Calls module IEAVTSAS to create the system trace address space and initializes the trace function.

ENTRY FROM: IEAVNIPM.

EXIT: Returns to the caller.

ERROR ENTRY: None.

ERROR EXIT: IGFPTERM.

IEAVNP57 — Dumping Services Address Space RIM

OPERATION: Calls module IEAVTSAS to create the dumping services address space (DUMPSRV).

ENTRY FROM: IEAVNIPM.

EXIT: Returns to the caller.

IEAVNP76 — LOGREC Initialization RIM

OPERATION: Locates and opens a cataloged SYS1.LOGREC. If this fails, it attempts to open an uncataloged SYS1.LOGREC on SYSRES.

ENTRY FROM: IEAVNIPM.

EXIT: Returns to the caller.

IEAVN600 — CONSOLxx Member Front End Processor

OPERATION: Reads the records for the CONSOLxx member of SYS1.PARMLIB specified by the CON system parameter. Invokes IEEMB887, IEAVN610, and IEAVN613 to validate the CONSOLE, INIT, HARDCOPY and DEFAULT statements in the member to select the master console and to initialize the UCM control blocks.

ENTRY FROM: IEAVNPA1

EXIT.: Returns to the caller.

IEAVN601 — CONSOLxx Member Exit — Part 1

OPERATION: Sets up an internal work area to record the valid values for various keywords on the CONSOLE, HARDCOPY, and DEFAULT statements in a CONSOLxx member of SYS1.PARMLIB.

ENTRY FROM: IEEMB887 through the parse definitions in IEAVN603.

EXIT: Returns to the caller.

IEAVN602 — CONSOLxx Member Exit — Part 2

OPERATION: Sets up an internal work area to record the valid values for various keywords on the INIT, HARDCOPY, and DEFAULT statements in a CONSOLxx member of SYS1.PARMLIB.

ENTRY FROM: IEEMB887 through the parse definitions in IEAVN603.

EXIT: Returns to the caller.

IEAVN603 — CONSOLxx Member Parse Definitions

OPERATION: This module in non-executable and contains the parse rules for the CONSOLE, INIT, HARDCOPY and DEFAULT statements in a CONSOLxx member of SYS1.PARMLIB.

ENTRY FROM: N/A

EXIT: N/A

IEAVN604 — CONSOLxx Member Exits — Part 3

OPERATION: Sets up an internal work area to record the valid values for various keywords on the CONSOLE statements in a CONSOLxx member of SYS1.PARMLIB.

ENTRY FROM: IEEMB887 through the parse definitions in IEAVN603.

EXIT: Returns to the caller.

IEAVN610 — CONSOLxx Statement Validation Module

OPERATION: Processes the data in a CONSOLxx member of SYS1.PARMLIB by validating the device types of the MCS consoles, by performing semantic checking for the keywords on the INIT, HARDCOPY, and DEFAULT statements, and by providing default values, when required, for those keywords.

ENTRY FROM: IEAVN600, IEAVN601, IEAVN602, and IEAVN613,

EXIT: Returns to the caller.

IEAVN611 — CONSOLE Statement Semantic Module —Part 1

OPERATION: Validates and applies defaults, when necessary, for some of the keywords on the CONSOLE statements in a CONSOLxx member of SYS1.PARMLIB. IEAVN612 validates the remaining keywords.

ENTRY FROM: IEAVN610.

EXIT: Returns to the caller.

IEAVN612 — CONSOLE Statement Semantic Module —Part 2

OPERATION: Validates and applies defaults, when necessary, for some of the keywords on the CONSOLE statements in a CONSOLxx member of SYS1.PARMLIB. IEAVN611 validates the remaining keywords.

ENTRY FROM: IEAVN611.

EXIT: Returns to the caller.

IEAVN613 — CONSOLE Configuration Validation Routine

OPERATION: Validates the master console specification in a CONSOLxx member of SYS1.PARMLIB and selects one, if necessary. This module also validates the alternate console chain and calls IEAVN614 to initialize the UCM control block structure.

ENTRY FROM: IEAVN600.

EXIT: Returns to the caller.

IEAVN614 —

OPERATION: Initializes the Unit Control Module (UCM) base and its extensions. Also initializes a Unit Control Module Entry (UCME) and its extensions for each valid console definition in a CONSOLxx member of SYS1.PARMLIB.

ENTRY FROM: IEAVN613.

EXIT: Returns to the caller.

IEAVN700 — Communications Task Address Space Create Routine

OPERATION: Creates the console communications task address space by invoking system address space initialization (IEEMB881).

ENTRY FROM: IEEVIPL.

EXIT: Returns to the caller.

ERROR ENTRY: From RTM.

ERROR EXIT: To IGFPTERM.

IEAVN701 — Communications Task Address Space Initialization

OPERATION: Initializes the console communications task address space by:

- Setting up the cross memory environment
- Invoking communications task initialization (IEAVVINT)
- Activating the action message retention facility
- Attaching the various tasks for communications task processing

ENTRY FROM: IEEPRWI2.

EXIT: No normal exit. This is a permanent task.

ERROR ENTRY: From RTM.

ERROR EXIT: No normal exit. This is a permanent task.

IEAVN702 — Communications Task Initialization Cleanup

OPERATION: Handles the initialization process separate from IEAVN701.

ENTRY FROM: IEAVN701.

EXIT: Dispatcher.

IEAVN800 — Console Default Table

OPERATION: This non-executable module contains default SEG, RNUM, RTME, and AREA values for each device type supported as an MCS console.

ENTRY FROM: N/A

EXIT: N/A

IEAVRTOD — Time-of-Day Clock Initialization

OPERATION: Sets the TOD clock to the correct GMT.

ENTRY FROM: IEAVNP00.

EXIT: Returns to the caller.

IEAVTABI — Recovery Termination Management ABDUMP RIM

OPERATION: Initializes the RTM control table with installation-defined dump options from SYS1.PARMLIB members IEAABD00, IEADMP00, and IEADMR00 (for SYSABEND, SYSMDUMP, and SYSUDUMP dumps).

ENTRY FROM: IEAVNIPM.

EXIT: Returns to the caller.

IEAVTMSI — Recovery Termination Management Initialization

OPERATION: Establishes three permanent tasks for RTM: the asynchronous recording task, an SVC dump task, and the address space termination task.

ENTRY FROM: IEEMB860.

EXIT: Returns to the caller.

ENTRY ERROR: At entry point MSIESTAE from ABEND.

EXIT ERROR: Returns to IEEMB860.

IEAVTMTC — Address Space Termination Controller Task Initialization

OPERATION: Creates an SRB that is scheduled when a system routine requests address space termination.

ENTRY FROM: IEAVTMSI.

EXIT: No normal exit. This is a permanent task.

IEAVTRET — Recording Task Initialization

OPERATION: Creates an SRB that is scheduled when a system routine requests asynchronous recording.

ENTRY FROM: IEAVTMSI.

EXIT: No normal exit. This is a permanent task.

IEAVTSAI — Dumping Services Address Space Initialization

OPERATION: Initializes the dumping services address space (DUMPSRV) with a summary dump extended work area and the SDUMP virtual storage buffer for eventual use by SDUMP services.

ENTRY FROM: IEEPRWI2.

EXIT: IEEPRWI2 terminates the address space or continues to initialize the address space.

ERROR ENTRY: None.

ERROR EXIT: Returns to the caller with an error return code.

IEAVTSAS — DUMPSRV Address Space Creation

OPERATION: Invokes IEEMB881 to initiate the creation of the dumping services (DUMPSRV) address space.

ENTRY FROM: IEAVNP57, IEAVTSDR.

EXIT: Returns to the caller.

IEAVTSDD — SVC Dump Data Set Initialization

OPERATION: Initializes the dump data sets and tapes that are specified on the DUMP system parameter.

ENTRY FROM: IEAVTSDI.

EXIT: Returns to the caller.

IEAVTSDI — Recovery Termination Management SVC Dump RIM

OPERATION: Initializes system areas for SVC dump.

ENTRY FROM: IEAVNIPM.

EXIT: Returns to the caller.

IEAVTSDT — SVC Dump Task for Master Scheduler (Initialization Function)

OPERATION: Performs SVC dump processing in each address space for which there is a scheduled SVC dump request.

ENTRY FROM: IEAVTMSI.

EXIT: No normal exit. This is a permanent task.

IEAVVINT — Communications Task Initialization

OPERATION: Sets up for opening the console devices and performs initialization for the communications task. It invokes IEECVGCI to initialize display consoles. It obtains storage for COMM TASK console queueing service of the IEETRACE parameter list.

ENTRY FROM: IEAVN701.

EXIT: Returns to the caller.

IEAVXEPM — Program Call/Authorization Nucleus Entry Point Search Routine

OPERATION: Locates the entry points for various nucleus - resident routines that receive control through the program call instruction.

ENTRY FROM: IEAVXMAS and IEAVXECR.

EXIT: Returns to the caller.

IEAVXMAS — Program Call/Authorization Address Space Initialization

OPERATION: Initializes the PC/AUTH address space (when requested by the PC/AUTH RIM, IEAVNPF5) with the PC/AUTH control blocks and tables that are needed for cross memory operations.

ENTRY FROM: IEEPRWI2.

EXIT: None. Enters a never-ending wait state.

IEAVXSEM — Program Call/Authorization System Entry Table Descriptor Module

OPERATION: This module contains an entry table description list. The PC/AUTH address space initialization module, IEAVXMAS, uses this list to build the system entry table (SET).

ENTRY FROM: Not applicable (non-executable module).

EXIT: Not applicable (non-executable module).

IECVIOSI — IOS Initialization

OPERATION: Initializes all dynamic pathing devices and establishes the IOS storage manager as an ENF listener.

ENTRY FROM: IEEMB860.

EXIT: Returns to the caller.

IEECVGCI — DIDOCS Initialization

OPERATION: Determines which display consoles have output-only and/or color capability, and opens SYS1.DCMLIB.

ENTRY FROM: IEAVVINT.

EXIT: Returns to the caller.

IEECVUCM — Unit Control Module (UCM) Builder

OPERATION: This non-executable module contains the UCM extension, the MCS prefix, the UCM base, the UCM event indication list, and 99 individual device entries (UCMEs).

ENTRY FROM: N/A

EXIT: N/A

IEEMB803 — System Log Task Initialization

OPERATION: Sets up the data areas necessary to write in the system log data set. It then posts the communications task and waits for a request.

ENTRY FROM: IEEVWAIT at system initialization time.

EXIT: No normal exit. When the log is closed, IEEMB803 waits for it to be activated again.

IEEMB809 — Master Trace Table Initialization

OPERATION: Creates or deactivates a master trace table based on information passed in a parameter list via the IEETRACE macro.

ENTRY FROM: IEEVIPL at master scheduler initialization time or from IEECB806 via the TRACE command.

EXIT: Returns to the caller.

IEEMB820 — SMF Initialization Router

OPERATION: Starts the SMF address space initialization process during master scheduler initialization. IEEMB820 sets up the address space attributes and parameters specifying IFASTART as the SMF address space initialization module. It then links to IEEMB881 (system address space create routine). Once the address space is created, it links to IEFSMFIE to start interval recording for the master scheduler address space.

ENTRY FROM: IEEMB860

EXIT: Returns to the caller.

IEEMB860 — Master Scheduler Region Initialization

OPERATION: Attaches the SMF initialization router (IEEMB820) to start, the SMF address space. It attaches the missing interrupt handler (IOSRMIHT) and the IPL/outage recorder (IOSROUTG), invokes IEAVTMSI, and waits for SMF initialization and ASM initialization to finish. It links to the IOS module, IECVIOSI, to initialize dynamic pathing and establish the IOS storage manager as an ENF listener. It calls IEFJSIN2 to initialize subsystems. It attaches the system security initialization routine (ICHSEC00) and detaches it when complete. It links to IEFHB4I2, the allocation use count gatherer.

ENTRY FROM: IEFSD263.

EXIT: To IEEVWAIT via XCTL.

ERROR ENTRY: At entry point STA00 from failure detection or from ABEND.

ERROR EXIT: To a wait state.

IEEMB881 — System Address Space Create Routine

OPERATION: Creates a system address space by

- Creating and initializing a JSCB for the master scheduler's TCB.
- Obtaining and initializing a START command CSCB.
- Calling IEAVEMRQ to obtain an ASID/ASCB.
- Setting the address space attributes specified by the caller.
- Using IEEVEMCR to create an address space for the caller.

ENTRY FROM: IEAVNPF5, ISGNTASC (IEAVNP33), IEFAB4I0, IEAVNP57, and any other callers of the system address space create routine.

EXIT: Returns to the caller.

IEEMB883 — System Address Space Initialization WAIT/POST Routine

OPERATION: Waits on events supplied as input in the form of event codes. Only one event, event code 1, is defined. This event code causes IEEMB883 to wait for the completion of master scheduler initialization. IEEMB883 is notified of the completion of master scheduler initialization by IEEVWAIT.

ENTRY FROM: IEEPRWI2 and any routine initializing a system address space, such as IEAVXMAS, ISGNASIM, IEFHB4I1, and IEAVTSAI.

EXIT: Returns to the caller.

IEEMB888 — General Parmlib Scan Routine

OPERATION: Scans the specified SYS1.PARMLIB member, deletes comments and collects a record containing the operands for a particular statement type. It then passes control to the appropriate statement processor routine to handle the record.

ENTRY FROM: IEAVNP18, IEAVNP20, and IEAVNP25.

EXIT: Returns to the caller.

IEEPRWI2 — Started Task Control Processing

OPERATION: Gets a new region for started task control processing. If IEEPRWI2 is executing for a system address space, it links to the address space function initialization routine if one was specified and calls IEEMB883 to wait for the completion of master scheduler initialization. Then IEEPRWI2 determines whether a START, MOUNT, or LOGON command is being processed, and, based on that decision, invokes the appropriate command processor.

ENTRY FROM: Region control task via ATTACH.

EXIT: To the region control task.

IEEVCPR — CONFIG CPU Reconfiguration Processor

OPERATION: Brings the non-IPL processor online.

ENTRY FROM: IEAVNP00.

EXIT: Returns to the caller.

IEEVIPL — Master Scheduler Base Initialization

OPERATION: Initializes the master scheduler, invokes the subsystem interface initialization function (IEFJSINT), and other initialization routines, specifically IEAVN701, IEFSD060, IEFAB4I0, IKJEFXSR, and IEFQBINT.

ENTRY FROM: IEAVNIPX.

EXIT: IEFSD060.

ERROR ENTRY: At entry point IPSTAR from failure detection or from ABEND.

ERROR EXIT: To a wait state.

IEEVWAIT — Command Scheduler Wait

For The Master Address Space

OPERATION: Scans the CSCB chain and, for each pending CSCB that represents a command the executes in the master address space, attaches the corresponding command processor. Normally, the first CSCB processed represents the START command for the primary job entry subsystem. After the subsystem is started, IEEVWAIT terminates system trace (if specified) and attaches IEEMB803, the system log. When IEEVWAIT processes a START command CSCB, it checks to see if a system address space is being started. If so, it bypasses normal START processing and just posts the EAEASWT/CHASWT ECB.

ENTRY FROM: IEEMB860.

EXIT: No normal exit. This is a permanent task.

ERROR ENTRY: At STAE0000 from failure detection or from ABEND.

ERROR EXIT: To IEEVWAIT's main processing for restart or to a wait state.

For The Communications Task Address Space

OPERATION: Scans the CSCB chain and processes pending commands that execute in the communications task address space.

ENTRY FROM: IEAVN701.

EXIT: No normal exit. This is a permanent task.

ERROR ENTRY: At entry point STAE0000 from failure or from ABEND.

ERROR EXIT: To dispatcher.

IEFAB4E6 — Allocation ESTAE/FRR Recovery Routine

OPERATION: Handles abnormal terminations of tasks that create, initialize, or update the allocation address space (ALLOCAS).

ENTRY FROM: Recovery termination manager (RTM).

EXIT: Returns to the caller.

IEFAB4I0 — Allocation Initialization Routine

OPERATION: Invokes the system address space create routine, IEEMB881 to create the allocation address space (ALLOCAS). (IEFHB4I1 initializes the new address space.) It establishes the entry points of various allocation functions by storing their entry points into the JESCT, the JESCT extension, or the CVT. IEFAB4I0 builds the UCB pointer list (UPL). Stores the address of the eligible device list (EDT) into the JESCT and invokes the EDT verification routine IEFEB400.

ENTRY FROM: IEEVIPL

EXIT: Returns to the caller.

Called Routine: IEEMB881,IEFEB400, IGFPTERM

IEFEB400 — EDT Verification Routine

OPERATION: Verifies that the devices defined in the eligible device table (EDT) match the devices currently existing in the nucleus.

ENTRY FROM: IEFAB4I0.

EXIT: Returns to the caller.

IEFENFFX — Event Notification Request Router

OPERATION: Checks the validity of the input event parameter list of the caller and routes the request for processing.

ENTRY FROM: ISGNASIM, IEAVNP1F.

EXIT: Returns to the caller.

ERROR ENTRY: None.

ERROR EXIT: Returns to caller with an error return code.

IEFENFNM — Event Notification Mainline Processing

OPERATION: Handles all the processing of the event notification facility listen elements such as processing for signalling of events, processing for listening of events, processing for deleting listeners.

ENTRY FROM: IEFENFFX, IEFENFWT.

EXIT: Returns to the caller.

ERROR ENTRY: None.

ERROR EXIT: ABEND with a system completion code of X'055' and a reason code of either 36 or 40.

IEFENFWT — Event Notification Facility Wait Routine

OPERATION: Processes all pending requests in the event notification facility process table.

ENTRY FROM: Attached as a task by master scheduler initialization (IEEMB860).

EXIT: Remains as a never-ending task.

ERROR ENTRY: At entry point ESTAEXIT (as an ESTAE) to store data in the SDWA, record a SYS1.LOGREC entry if an SDWA was supplied, take a dump, and request retry.

ERROR EXIT: None.

IEFHB4I1 — Allocation Address Space Initialization

OPERATION: Obtains and initializes all the necessary control blocks in the allocation address space and sets up a cross memory environment.

ENTRY FROM: IEEPRWI2.

EXIT: None. This module enters a never-ending wait state.

ERROR ENTRY: None.

ERROR EXIT:

- Control goes to IEFAB4E6 (FRR/ESTAE) for abnormal termination of this task.
- Return to caller when the ADBUECB is posted indicating that ALLOCAS has failed and an SDUMP is in progress.

IEFHB4I2 — Display Allocation Scavenge Routine

OPERATION: Scans all the UCBs in the system and associates the allocation use count in each UCB with an address space or address spaces.

ENTRY FROM: IEEMB860.

EXIT: Returns to the caller.

ERROR ENTRY: None.

ERROR EXIT:

- Return to caller when unit allocation status recording is not active.
- To IEFAB4E6 (FRR/ESTAE) via RTM for abnormal termination of this task.

IEFHB410 — Display Allocation Tables Manager

OPERATION: Handles all updates to the display allocation tables in the allocation address space (ALLOCAS). IEFHB4I0 executes in cross memory mode. It is entered by means of a program call (PC) to perform one or more of the following functions.

- Updates the DALT use count field
- Returns the DALT use count for a unit
- Returns the DALT use count for one address space and one unit
- Clears the DALT for an address space

ENTRY FROM: IEFAB4A4, IEFAB4E5, IEFAB4E6, IEFAB434, IEFAB435, IEFHB4I2, IEFAB477, IEFDB440.

EXIT: Returns to the caller by means of a program transfer (PT).

ERROR ENTRY: None.

ERROR EXIT:

- Return to caller when unit allocation status recording is not active.
- To IEFAB4E6 (FRR/ESTAE) via RTM for abnormal termination of this task.

IEFJESDM — Job Entry Subsystem Control Table

OPERATION: Contains the csect for the IEFJESCT control block.

ENTRY FROM: Not applicable (non-executable module).

EXIT: Not applicable (non-executable module).

IEFJMSFC — Master Subsystem Vector Table Data

OPERATION: Contains the subpool number for the master subsystem's subsystem vector table (SSVT), the names of the master subsystem's function routines, and the function codes associated with each function routine.

ENTRY FROM: Not applicable (non-executable module).

EXIT: Not applicable (non-executable module).

IEFJSBLD — Subsystem Service Routine

OPERATION: Handles common services used in the creation of subsystems. The services include building SSCVTs, the master subsystem's SSVT, a hash table of SSCVT addresses (SHAS), and the SAST. IEFJSBLD also links to individual subsystem initialization routines.

ENTRY FROM: IEFJSINT, IEFJSIN2.

EXIT: Returns to the caller.

ERROR ENTRY: At entry point, ESTAEBLD, from ABEND.

ERROR EXIT: Percolate the error if the error occurred on the first SAST build or a LOAD macro failed; otherwise, return to the caller.

IEFJSIMM — Subsystem Initialization Messages

OPERATION: Contains the messages used during subsystem initialization.

ENTRY FROM: Not applicable (non-executable module).

EXIT: Not applicable (non-executable module).

IEFJSIMW — Subsystem Initialization Message Writer

OPERATION: Issues messages contained in IEFJSIMM for IEECB805, IEFJSBLD, and IEFJSIN2.

ENTRY FROM: IEECB805, IEFJSBLD, and IEFJSIN2.

EXIT: Returns to the caller.

IEFJSINT — Subsystem Interface Initialization

OPERATION: Initializes the JESCT and calls the subsystem service routine (IEFJSBLD) to build an SSCVT for the master subsystem.

ENTRY FROM: IEEVIPL.

EXIT: Returns to the caller.

IEFJSIN2 — Subsystem Initialization

OPERATION: For each record in the specified IEFSSNxx members, IEFJSIN2 calls:

1. The parmlib read routine (IEEMB878) to read the record
2. The positional parser routine (IEEMB882) to parse the record
3. IEFJSBLD to build a SSCVT and link to the subsystem's initialization routine (if one is specified).

IEFJSIN2 also requests that IEFJSBLD rebuild the SAST if additional subsystems were defined in SYS1.PARMLIB and build the hash table (SHAS).

ENTRY FROM: IEEMB860.

EXIT: Returns to the caller.

ERROR ENTRY: At entry point ESTASIN2 from ABEND.

ERROR EXIT: Percolate the error if a retry was previously attempted; otherwise, return to IEEMB860.

IEFPPT — Program Properties Table (PPT) Statement Processor

OPERATION: Processes the PPT statement in SYS1.PARMLIB member, SCHEDxx, and builds a Program Properties Table (PPT) using the default PPT and the PPT statements.

ENTRY FROM: IEEMB888

EXIT: Returns to caller.

IEFRCSTP — Restart Codes Statement Processor

OPERATION: Processes the RESTART/NORESTART statements in the SYS1.PARMLIB member, SCHEDxx.

ENTRY FROM: IEEMB888

EXIT: Returns to caller.

IEFSJINT — Scheduler JCL Facility JDVT Initialization

OPERATION: Creates the default JCL definition vector table (JDVT).

ENTRY FROM: IEEMB860 (via the scheduler JCL facility control routine (IEFSJCNL)

EXIT: IEEMB860.

ERROR ENTRY: At entry point INENTRY from ABEND.

ERROR EXIT: Return to IEEMB860.

IEFQBINT — SWA Manager Initialization Routine

OPERATION: Initializes fields in the JESCT with the addresses of the SWA manager, Journal Write and SWA manager diagnostic information routines. It also initializes the JESCT with the address of the SWA manager storage table (QMST).

ENTRY FROM: IEEVIPL

EXIT: Returns to IEEVIPL via a branch.

IEFQB550 — SWA Manager Move Mode Interface Routine

OPERATION: Intercepts branches to the SWA Manager Move Mode Routine and routes the requests to IEFQB551 in the EPLPA.

ENTRY FROM: Initiator/terminator, scheduler restart, convertor/interpreter, JES3.

EXIT: Returns to IEFQB551 via BSM.

IFASMF — SMF Control Task

OPERATION: Completes the initialization of the SMF address space (data set initialization). IFASMF controls the execution of the SMF commands and IEEMB829 events.

ENTRY FROM: Attached by initiator to complete initialization. Posted by command processors or utility modules to process IEEMB829 events.

EXIT: Returns to the caller.

IFASTART — SMF Address Space Initialization Routine

OPERATION: IFASTART initializes the SMF control blocks (SMCA, ACT and SLCA). It then routes control to the proper routine to handle SMF parameter processing and the writing of the SMF initialization records that describe the current IPL.

ENTRY FROM: IEEPRW12

EXIT: Returns to the caller.

IGFPBUCR — Processor MCH Initialization

OPERATION: Allocates and initializes the processor work area (PWA) and the LOGREC buffer (LRB) for each online processor.

ENTRY FROM: IGFPINIT

EXIT: Returns to the caller.

IGVNIPCR — VSM NIP Contiguous Virtual Storage Routine

OPERATION: Obtains CSA virtual storage and calls real storage management (RSM) to obtain contiguous real frames for SRM channel measurement.

ENTRY FROM: IEAVNP1F.

EXIT: Returns to the caller.

IKJEFXSR — Initialization for Linkage to TSO I/O Service Routines

OPERATION: Searches the link pack area for certain TSO I/O service routines and puts their addresses in the CVT. (The names of the service routines are coded into this module.) The service routines can now be invoked by TSO command processors via the CALLTSSR macro instruction. If TSO/E Release 2 or later is installed, IKJEFXSR builds the TSO vector table (TSVT). If TSO/E Release 3 or later is installed, IKJEFXSR sets the upper limit of broadcast records allowed for the TSO SEND command processor.

ENTRY FROM: IEEVIPL.

EXIT: Returns to the caller.

ILRASRIM — Auxiliary Storage Management RIM

OPERATION: Initializes ASM for cold, warm, and quick starts by initializing ASM control blocks. Opens PLPA, common, and duplex (if requested) page data sets. Restores PLPA and EPLPA slot information of a previous IPL for quick and warm starts.

ENTRY FROM: IEAVNIPM.

EXIT: Returns to the caller.

ILRASRM1 — Auxiliary Storage Management RIM — Part 2

OPERATION: Completes the initialization of ASM control blocks. Opens local page data sets and swap data sets.

ENTRY FROM: IEAVNIPM.

EXIT: Returns to the caller.

ILRASRM2 — Auxiliary Storage Management Initialization Service Routines

OPERATION: Contains service routines to:

- Parse the page, swap, and NONVIO data set name strings.
- Call ILROPS00 to open page data sets and swap data sets.
- Complete the TPARTBLE and PAT information for page data sets.
- Write the TPARTBLE to the PLPA page data set.

ENTRY FROM: ILRASRIM, ILRASRM1.

EXIT: Returns to the caller.

ILRIMMSG — Auxiliary Storage Management RIM Message Module

OPERATION: Contains and writes all ASM initialization messages to the operator.

ENTRY FROM: ILRASRIM, ILRQSRIT.

EXIT: Returns to the caller.

ILRMSG00 — Auxiliary Storage Management Message Module

OPERATION: Writes messages to the operator concerning the status of PLPA, common, duplex, local, and swap data sets. Terminates the system when auxiliary storage management is unable to continue processing for one of the following reasons.

- The PLPA or common data set has become unusable and there is no duplex data set available.
- The duplex data set has become unusable and the PLPA/common data set is unavailable.
- Both the duplex data set and the PLPA/common data set are full.
- The last local page data set has become unusable or full.

ENTRY FROM: ILRTMI00.

EXIT: Returns to the caller.

ERROR ENTRY: None.

ERROR EXIT: Put the system into either a X'02E', X'03C', or X'03E' wait state.

ILROPS00 — Auxiliary Storage Management Open Processing

OPERATION: Opens page and swap data sets; creates and initializes I/O control blocks; calls ILRPREAD to initialize and obtain addressability to cached auxiliary storage subsystems.

ENTRY FROM: ILRASRM2, ILRPGEXP.

EXIT: Returns to the caller.

Note: ILRASRIM, ILRASRM1, and ILRPGEXP load ILROPS00, use it, and then delete it. ILROPS00 resides in SYS1.LINKLIB.

ILRPREAD — Auxiliary Storage Management Special Read/Write Routine

OPERATION: Reads from, and writes to, page data sets; initializes and obtains addressability to cached auxiliary storage subsystems; selects slots for XQSRDs during cold starts (quick start record processing).

ENTRY FROM: ILRASRIM, ILRASRM1, ILRASRM2, ILROPS00, ILRQSRIT, ILRTMI00.

EXIT: Returns to the caller.

Note: Loaded by callers from SYS1.LINKLIB.

ILRQSRIT — Auxiliary Storage Management Quick Start Record Initialization

OPERATION: For a cold start only, fills in the quick start record (QSRCD), the extended quick start record (EQSRD) and quick start record extensions (XQSRDs) writes the TPARTBLE, QSRCD, EQSRD and XQSRDs to the PLPA data set.

ENTRY FROM: IEAVNP05.

EXIT: Returns to the caller.

ILRTMI00 — Auxiliary Storage Management Task Mode Initialization

OPERATION: Completes the initialization of ASM in the system by processing SYS1.STGINDEX. Specifically, restores virtual I/O (VIO) data set page usage information from the previous IPL, builds a list of page and swap data set names, and enqueues on SYS1.STGINDEX.

ENTRY FROM: ILRTMRLG.

EXIT: Returns to the caller.

ERROR ENTRY: From ILRTMI01 ESTAE routine at one of three possible entry points:

- ILRCRTM2 - when warm start processing fails.
- ILRCRTM3 - when CVIO processing fails.
- ILRCRTM4 - when data set name list processing fails.

ERROR EXIT: Returns to the caller.

ILRTMI01 — ESTAE Routine for ILRTMRLG and ILRTMI00

OPERATION: Provides recovery for ILRTMRLG and its calls to ILRTMI00 so that ILRTMI00 finishes its initialization and ILRTMRLG doesn't terminate.

ENTRY FROM: RTM.

EXIT: Returns to the caller or retries into ILRTMRLG or ILRTMI00.

ILRTMRLG — Auxiliary Storage Management Release Logical Group Module

OPERATION: Loads and deletes ILRTMI00, puts the addresses of group operator modules into the ASMVT control block, and waits to do "release logical group" requests coming to ASM for saved logical groups.

ENTRY FROM: IEEMB860.

EXIT: Remains for entire IPL.

IOSRMIHT — Missing Interrupt Handler Initialization

OPERATION: Initializes missing interrupt handler (MIH).

ENTRY FROM: IEEMB860.

EXIT: Returns to the caller.

IOSVNPTH — IOS NIP Path Verification Routine

OPERATION: Validates paths for a device.

ENTRY FROM: IEAVNPA2, IEAVNPB2, IEAVNPC2, and IEAVNIPM.

EXIT: Returns to the caller.

IRARMANL — SRM Parameter Syntax Analyzer RIM

OPERATION: Checks the syntax of parameters in the IEAIPSxx, IEAICSxx, and IEAOPTxx members of SYS1.PARMLIB.

ENTRY FROM: IRARMIPS, IRARMICS, IRARMOPT, and IRARMANL (recursive).

EXIT: Returns to the caller.

IRARMCPU — SRM Processor Adjustment Table

OPERATION: Used by IEAVNP10 to adjust processor model dependent fields.

ENTRY FROM: Not applicable (nor-executable module).

EXIT: Not applicable (non-executable module).

IRARMICS — SRM Installation Control Specification Processor RIM

OPERATION: Scans the installation control specification list in the IEAICSxx member of SYS1.PARMLIB and builds a control block for installation control specification information.

ENTRY FROM: IEAVNP10.

EXIT: Returns to the caller.

IRARMIPM — IPS, Installation Control Specification, and OPT Error Message Module

OPERATION: Contains the text of messages denoting syntax errors detected by IRARMIPS, IRARMICS, and IRARMOPT.

ENTRY FROM: Not applicable (non-executable module).

EXIT: Not applicable (non-executable module).

IRARMIPS — SRM IPS Processor RIM

OPERATION: Scans the installation performance specification (IPS) list in the IEAIPSxx member of SYS1.PARMLIB and builds control blocks for IPS information.

ENTRY FROM: IEAVNP10.

EXIT: Returns to the caller.

IRARMOPT — SRM OPT Processor RIM

OPERATION: Scans the OPT list in the IEAOPTxx member of SYS1.PARMLIB and builds the parameter list containing the SRM tuning parameters.

ENTRY FROM: IEAVNP10.

EXIT: Returns to the caller.

ISGBCI — Global Resource Serialization Ring Processing Command Interface

OPERATION: Extracts the status of the global resource serialization systems and the CTC connections between systems, communicates data and commands between systems, and changes the configuration of the main ring.

ENTRY FROM: ISGBTC, ISGCMDR, ISGCOMRG, ISGCPRG, ISGCRST, ISGNGRSP, ISGCOSC.

EXIT: Returns to the caller.

ERROR ENTRY: None.

ERROR EXIT: ABEND with completion code of X'09A' and reason code that identifies the specific nature of the error.

ISGBTC — Global Resource Serialization CTC Ring Processing Task - Mode Controller

OPERATION: Performs task-mode services for global resource serialization ring processing such as initialization, responding to unusual events and exceptions, initiating ring processing operations, and cleaning up ring resources.

ENTRY FROM: ISGNASIM via ATTACH.

EXIT: None. Posts ISGNASIM, then becomes a never-ending task that waits for unusual events or exceptions to occur.

ISGCMDR — Global Resource Serialization Command Router

OPERATION: Processes the following global resource serialization requests by attaching the appropriate request processor.

- Restart request
- Quiesce request
- Purge request
- Display request
- Message request

ENTRY FROM: ISGNASIM via ATTACH.

EXIT: Enters a wait state, waiting to process command requests.

ERROR ENTRY

- ISGCDRRV - error recovery routine to recover from errors in ISGCMDR.
- ISGCRR01 - error retry entry point in ISGCMDR.
- ISGCRR02 - error retry entry point in ISGCTXR1.
- ISGCTXR1 - part of ISGCMDR to detach the command processor task and release unneeded storage.

ERROR EXIT: ABEND with completion code of X'09A' and reason code that indicates that the command processor resources to be cleaned up were not on the command clean-up queue.

ISGCQMRG — Global Resource Serialization Queue Merge

OPERATION: Ensures that the global resource queue for a system joining a global resource serialization complex is identical to those of the other active systems already in the complex.

ENTRY FROM: ISGNGRSP, ISGCRST.

EXIT: Returns to the caller.

ERROR ENTRY: ISGCQMRR is the entry point in ISGCQMRG that cleans up resources following an error.

ERROR EXIT: ABEND with completion code of X'09A' and reason code that indicates the specific nature of the error.

ISGCRCV — Global Resource Serialization Command Recovery Module

OPERATION: Provides error recovery and minimal cleanup for those modules of global resource serialization that establish it (that is, ISGCRCV) as their ESTAE.

ENTRY FROM: RTM.

EXIT: Returns to RTM.

ERROR ENTRY: ISGCRCVY is the entry point in ISGCRCV that recovers from errors that occur in ISGCRCV.

ERROR EXIT: None.

ISGCRST — Global Resource Serialization Restart Request Processor

OPERATION: Controls the processing necessary to bring a new system into the global resource serialization complex and ensures that the new system is compatible with the existing complex.

ENTRY FROM: ISGCMDR as an attached task.

EXIT: Returns to the dispatcher at the normal end of task.

ERROR ENTRY: ISGCRS02 is an entry point in ISGCRST that issues message ISG015I and performs back-out processing.

ISGGQSRV — Global Queue Services

OPERATION: This module is called in three situations:

- When global QCBs need to be processed by ISGCQMRG, this module sets the queue merge bit in each QCB to one.
- When a global QCB has been processed by ISGCQMRG, this module sets the queue merge bit to zero.
- When global QCBs are to be dequeued by ISGCQMRG, this module builds the DEQ QWBs.

ENTRY FROM: ISGCQMRG.

EXIT: Returns to the caller.

ERROR ENTRY: At entry points ISGGRECB and ISGGRTRY by RTM.

ERROR EXIT: None.

ISGGRNLV — Global Resource Serialization RNL Verification Routine

OPERATION: Checks the validity of the SYSTEM inclusion resource name list (RNL), the SYSTEMS exclusion RNL, and the reserve conversion RNL (loaded from SYS1.LINKLIB) to ensure that no errors occur during mainline processing of these lists. This routine is invoked once per IPL, if the installation opts to have RNLs loaded from SYS1.LINKLIB.

ENTRY FROM: ISGNRNLP.

EXIT: Returns to the caller.

ISGGRP00 — Global Resource Serialization Resource Processor

OPERATION: Processes global resource requests as defined by the queue work blocks (QWBs) that exist on the process queue. The global resource requests are:

- ENQ/DEQ/RESERVE SVC request
- TCB/DEQ purge request
- ASID DEQ purge request
- SYSID DEQ purge request
- Synchronization request

ENTRY FROM: ISGNASIM via ATTACH.

EXIT: Returns to the dispatcher at the normal end of task.

ERROR ENTRY: At entry point, GRPESTAE, as an ESTAE routine to ensure that ISGGRP00 is not terminated before the ISGGRP00 wait.

ERROR EXIT: ABEND with completion code of X'09A' and reason code that identifies the specific nature of the error.

ISGGSRVI — Global Resource Serialization Mainline Service Routine Initialization

OPERATION: Initializes the entry points for all the global resource serialization service routines used to provide global resource serialization services to the system.

ENTRY FROM: ISGNASIM.

EXIT: Returns to the caller.

ISGJPARM — Global Resource Serialization CTC Configuration Parmlib Processor

OPERATION: Parses the GRSCNF= system parameter as well as the entire GRSCNFxx parmlib member.

ENTRY FROM: ISGNCBIM.

EXIT: Returns to the caller.

ISGMSG00 — Global Resource Serialization Message Processor

OPERATION: Provides the global resource serialization services with the ability to communicate with the operator using both informational (WTO/MLWTO) and reply (WTOR) messages.

ENTRY FROM: ISGCMDR, ISGNASIM, ISGNGRSP, ISGCMDI, ISGCRST, ISGCQSC, ISGCPRG, ISGBTC, ISGGFRR0.

EXIT: Returns to the caller.

ERROR ENTRY: None.

ERROR EXIT: ABEND with completion code of X'09A' and a reason code that identifies the specific nature of the error.

ISGNASIM — Global Resource Serialization Address Space Initialization

OPERATION: Initializes the internal control blocks in the global resource serialization address space so that global resource serialization services can be available to the system.

ENTRY FROM: IEEPRWI2.

EXIT: Enters a never-ending wait.

ERROR ENTRY: ISGNASRV is the entry point in ISGNASIM that performs primary recovery of errors occurring during ISGNASIM processing. ISGNASIM's ESTAE, ISGCRCV, passes control to this entry point.

ERROR EXIT: ABEND with a completion code of X'09A' and reason code identifying the specific nature of the error.

ISGNCBIM — Global Resource Serialization Control Block Initialization

OPERATION: Obtains storage for and initializes global resource serialization control blocks before the creation of the global resource serialization address space and processes the GRS, GRSCNF and SYSNAME system parameters.

ENTRY FROM: IEAVNIPM.

EXIT: Returns to the caller.

ISGNGRSP — Global Resource Serialization Option Processor

OPERATION: Processes the GRS=JOIN/START system parameter and validates whether the specified GRS=option is consistent with the existing global resource serialization complex.

ENTRY FROM: ISGNASIM via ATTACH.

EXIT: Returns to the caller.

ERROR ENTRY: At entry point, ISGNERRX, to investigate the nature of the error and retry or terminate processing.

ERROR EXIT: ABEND with completion code of X'09A' and reason code that identifies the specific nature of the error.

ISGNPARS — Global Resource Serialization Parse Setup Routine

OPERATION: Initializes the parameter list for the generalized parser (IEEMB887) and then invokes IEEMB887 to process the specified SYS1.PARMLIB member.

ENTRY FROM: ISGJPARM, ISGNRNLP.

EXIT: Returns to the caller.

ISGNPGIM — POST Global Resource Serialization Initialization

OPERATION: Notifies global resource serialization address space initialization, ISGNASIM, that timer and console services are available.

ENTRY FROM: IEFENFNM (event notification facility).

EXIT: Returns to the caller.

ERROR ENTRY: At entry point, ISGNPRET, to abnormally terminate ISGNASIM.

ISGNRNLP — Global Resource Serialization Resource Name List Parmlib Processor

OPERATION: Validates the GRSRNL= system parameter, parses the GRSRNL member(s) of SYS1.PARMLIB, and builds the RNLs in SQA.

ENTRY FROM: ISGNCBIM.

EXIT: Returns to the caller.

ISGNTASC — Global Resource Serialization Address Space Creation

OPERATION: Starts the creation of the global resource serialization address space by invoking system address space initialization, IEEMB881.

ENTRY FROM: IEAVNIPM.

EXIT: Returns to the caller.

ERROR ENTRY: None.

ERROR EXIT: ABEND with completion code of X'09A' and reason code that identifies the specific nature of the error.

ISGNWMSI — Global Resource Serialization Wait for Master Scheduler Initialization

OPERATION: Waits for master scheduler initialization to complete on behalf of global resource serialization address space initialization, ISGNASIM.

ENTRY FROM: Dispatcher as a task attached by ISGNASIM.

EXIT: To the dispatcher at normal end of task.

ERROR ENTRY: None.

ERROR EXIT: ABEND with completion code of X'09A' and reason code that identifies the specific nature of the error.

ISGQSCAN — Global Resource Serialization Queue Scanning Services

OPERATION: Searches the global resource internal queues for information requested by the caller on the GQSCAN macro instruction. Returns the requested information in an area specified by the caller.

ENTRY FROM: ISGCQMRG.

EXIT: Returns to the caller.

ERROR ENTRY: None.

ERROR EXIT: ABEND with a completion code of X'09A' and reason code that identifies the specific nature of the error.

ISGSALC — Global Resource Serialization Storage Manager

OPERATION: Allocates storage cells for global resource serialization services to use.

ENTRY FROM: ISGNASIM.

EXIT: Returns to the caller.

ERROR ENTRY: None.

ERROR EXIT: ABEND with completion code of X'09A' and reason code that identifies the specific nature of the error.

APPENDIX A. IPL MACROS

The following macro instructions and mapping macros are available to the RIMs for IPL control block and parameter list mapping and for linkage to the NIP service routines. (For a description of the NIP service routines, refer to Appendix B: NIP Service Routines. For a description of IPL control blocks and parameter lists, refer to MVS/XA Debugging Handbook.

IPLWTO

The IPLWTO macro instruction generates linkage to the IPL WTO service routine IEAIPL35.

IHAIVT

The IHAIVT macro instruction generates a mapping of the IPL vector table (IVT).

APPENDIX B. NIP MACROS

The following macro instructions and mapping macros are available to the RIMs for NIP control block and parameter list mapping and for linkage to the NIP service routines. (For a description of the NIP service routines, refer to Appendix B: NIP Service Routines. For a description of NIP control blocks and parameter lists, refer to Section 4: Data Areas and to the publication Data Areas.)

IEAPMNIP

The IEAPMNIP macro instruction generates linkage to the following NIP service routines: NIPMOUNT, NIPOPEN, NIPPRMPT, NIPSENSE, NIPSWAIT, NIPTIME, NIPUCBFN, NIPWTO, NIPWTOR, and NIPWTOR2. For the following service routines, IEAPMNIP generates parameter lists before establishing linkage: NIPMOUNT, NIPOPEN, and NIPWTOR. For the NIPWTO service routine, IEAPMNIP creates the message header before establishing linkage.

IEAPPNIP

The IEAPPNIP macro instruction generates a mapping of the NIP parameter area (NIPPAREA) and the NIP parameter address table (PARMTAB).

IEAPSPIO

The IEAPSPIO macro instruction serves as an interface to the routine that finds and reads text records from the SYS1.PARMLIB data set.

IHANVT

The IHANVT macro instruction generates a mapping of the NIP vector table (NVT). IHANVT also generates the system wait state codes NIP uses.

ILREQSRD

The ILREQSRD mapping macro generates a mapping of the 32-byte extended quick start record (EQSRD) header and the 4-byte slot map entries contained in the EQSRD.

ILRQSRCD

The ILRQSRCD mapping macro generates a mapping of the 32-byte quick start record (QSRCD) header and the 4-byte slot map entries contained in the QSRCD.

ILRXQSRD

The ILRXQSRD mapping macro generates a mapping of the 32-byte quick start record extension (XQSRD) header and the 8-byte slot map entries contained in the XQSRD.

APPENDIX C. IPL SERVICE ROUTINES

IPL service routines provide IPL modules with functions that are frequently required, or they simulate functions that are not yet available in the system.

IPLWTO

OPERATION: Writes messages to the message queue.

INPUT: Register 1 points to the parameter list.

CALLING MACRO

```
?IPLWTO MSG(message) TYPE(WTO).
?IPLWTO MSG(message) TYPE(WTL).
```

CONTAINED IN: IEAIPL35.

APPENDIX D. NIP SERVICE ROUTINES

NIP service routines provide NIP modules with functions that are frequently required, or they simulate functions that are not yet available in the system. All NIP service routines are contained in the IEAVNIPM load module.

NIPMOUNT

OPERATION: Ensures that required direct access or tape volumes are online and available for further NIP processing.

INPUT: NIPMOUNT parameter list.

CALLING MACRO: IEAPMNIP TYPE=MOUNT, PARAM=label of NIPMOUNT parameter list.

CONTAINED IN: IEAVNPM3.

NIPOPEN

OPERATION: Performs the OPEN function of creating and initializing a DEB to describe a specified data set.

INPUT: NIPOPEN parameter list.

CALLING MACRO: IEAPMNIP TYPE=OPEN, PARAM=label of NIPOPEN parameter list.

CONTAINED IN: IEAVNPM3.

NIPOPIO

OPERATION: Issues messages queued prior to master console initialization, then issues SPECIFY SYSTEM PARAMETERS message. (RIM IEAVNP01 is the only caller of NIPOPIO.)

INPUT: Register 2 contains the address of the NVT; register 3 contains the address of the CVT.

CALLING MACRO: IEAPMNIP TYPE=OPIO.

CONTAINED IN: IEAVNIPM.

NIPPRMPT

OPERATION: Passes control to IEAVNP03, which prompts the operator for required parameters that have not been specified or that have been invalidly specified.

INPUT: Label of 8-character field containing the name of the parameter for which prompting is to take place.

CALLING MACRO: IEAPMNIP TYPE=PRMPT, PARAM=label of parameter field.

CONTAINED IN: IEAVNIPM.

NIPSENSE

OPERATION: Interprets sense data after an I/O error occurs and writes a diagnostic message to the master operator console.

INPUT: Register 1 points to the IOB.

CALLING MACRO: IEAPMNIP TYPE=SENSE, PARAM=register pointing to IOB.

CONTAINED IN: IEAVNIPM.

NIPSWAIT

OPERATION: Places system in disabled wait state, abnormally terminating system initialization.

INPUT: Wait state code and optional diagnostic information in field NPMPSW2 of the PSW (NPMWTPSW) passed to NIPSWAIT.

CALLING MACRO: IEAPMNIP TYPE=SWAIT.

CONTAINED IN: IEAVNIPM.

NIPTIME

OPERATION: Provides the time of day in decimal for time-stamping messages, or provides a fullword binary value that reflects the number of hundredths of seconds elapsed since IEAVNIPM was first entered for the current system initialization process.

INPUT: Register 1 contains either a zero for decimal time requests or a four for binary time requests.

CALLING MACRO: IEAPMNIP TYPE=TIME, PARAM=DEC or PARAM=BIN

CONTAINED IN: IEAVNIPM.

NIPUCBFN

OPERATION: Locates a UCB using the corresponding EBCDIC unit name or hexadecimal device address (CUA).

INPUT: Register 1 contains either the EBCDIC unit name or the hexadecimal unit address.

CALLING MACRO: IEAPMNIP TYPE=UCBFN, PARAM=register containing EBCDIC or hexadecimal unit.

CONTAINED IN: IEAVNIPM.

NIPWTO

OPERATION: Writes messages to the master operator console.

INPUT: Register 1 contains a pointer to the NIPWTO message header.

CALLING MACRO: IEAPMNIP TYPE=WTO, PARAM=message header label.

CONTAINED IN: IEAVNPM2.

NIPWTOR

OPERATION: Writes messages to the master operator console and reads the associated replies.

INPUT: Register 1 contains a pointer to the NIPWTOR parameter list.

CALLING MACRO: IEAPMNIP TYPE=WTOR, PARAM=parameter list label.

CONTAINED IN: IEAVNPM2.

NIPWTOR2

OPERATION: (1) Waits for the completion of an operator's reply associated with an earlier NIPWTOR message, (2) frees an SQA reply buffer, or (3) frees the master-console-image buffer.

INPUT: Register 1 contains (1) the address of the associated NIPWTOR parameter list, (2) the address in two's complement form of the SQA reply buffer, or (3) the value zero.

CALLING MACRO: (1) IEAPMNIP TYPE=WTOR2, PARAM=parameter register; (2) IEAPMNIP TYPE=WTOR2, BUFREL=YES, PARAM=parameter register; (3) IEAPMNIP TYPE=WTOR2, BUFREL=YES, PARAM=0.

CONTAINED IN: IEAVNPM2.

APPENDIX E. ABBREVIATIONS

ABDUMP (snap dump)
ABEND (abnormal end)
abnormal end (ABEND)
ACB (access method control block)
access method control block (ACB)
access method work area (AMWA)
ACE (ASM control element)
ADB (allocation descriptor block)
address increment map (AIM)
address increment map entry (AIME)
address queue anchor table (AQAT)
address queue anchor table index (AQATINDX)
address space control block (ASCB)
address space extension block (ASXB)
address space identifier (ASID)
address space table entry (ASTE)
address space vector table (ASVT)
ADRTABLE (contains the addresses at which the csect of the nucleus is loaded)
AFC (available frame count)
AFQ (available frame queue)
AIM (address increment map)
AIME (address increment map entry)
allocation descriptor block (ADB)
AMDSB (VSAM access method data statistics block)
AMWA (access method work area)
APF (authorized program facility)
APG (automatic priority group)
AQAT (address queue anchor table)
AQATINDX (address queue anchor table index)
ASCB (address space control block)
ASID (address space identifier)
ASM (auxiliary storage management)
ASM control element (ACE)
ASM page allocation table (PAT)
ASMVT (auxiliary storage management vector table)
ASTE (address space table entry)
ASVT (address space vector table)
ASXB (address space extension block)
authorization index allocation table (AXAT)
authorized program facility (APF)
automatic priority group (APG)
auxiliary storage management (ASM)
auxiliary storage management vector table (ASMVT)
available frame count (AFC)
available frame queue (AFQ)
AXAT (authorization index allocation table)

BASEA (master scheduler resident data area)
BUFC (buffer control block)
buffer control block (BUFC)
build directory list (BLDL)

CACE (cache array element)
CACHE (cache array table)
CCT (CPU control table)
CCW (channel command word)
CDE (contents directory entry)
cell pool anchor block (CPAB)
central processing unit (CPU)
CESD (composite external symbol table)
channel command word (CCW)
channel measurement block (CMB)
channel path measurement table (CPMT)
channel report word (CRW)
CMB (channel measurement block)
command recovery work area (CRWA)
command request block (CRB)

command scheduling control block (CSCB)
common service area (CSA)
common system data area (CSD)
common VTOC access facility (CVAF)
communications vector table (CVT)
composite external symbol table (CESD)
contents directory entry (CDE)
control section (CSECT)
CPAB (cell pool anchor block)
CPMT (channel path measurement table)
CPU (central processing unit)
CPU control table (CCT)
CRB (command request block)
cross memory directory (XMD)
CRW (channel report word)
CRWA (command recovery work area)
CSA (common service area)
CSCB (command scheduling control block)
CSD (common system data area)
CSECT (control section)
CVAF (common VTOC access facility)
CVT (communications vector table)

DAE (dump analysis and elimination)
DAIT (display allocation index table)
DALT (display allocation lookup table)
DASD (direct access storage device)
DAT (dynamic address translation)
data control block (DCB)
data definition (DD)
data definition name (DD name)
data event control block (DECB)
data extent block (DEB)
data set association block (DSAB)
data set control block (DSCB)
data set work area (DSTBL)
DBVT (device block vector table)
DCB (data control block)
DCM (display control module)
DCQ (device class queue)
DCTI (device connect time interval)
DD (data definition)
DD name (data definition name)
DEB (data extent block)
DECB (data event control block)
descriptor queue element (DQE)
device block vector table (DBVT)
device class queue (DCQ)
device connect time interval (DCTI)
device measurement block (DMB)
device-independent display operator console support (DIDOCS)
device table (DEVTAB)
DEVTAB (device table)
DFE (double free element)
DIDOCS (device-independent display operator console support)
direct access storage device (DASD)
display allocation index table (DAIT)
display allocation lookup table (DALT)
display allocation vector table (DVT)
display control module (DCM)
DFL (DAE default option)
DMB (device measurement block)
DMDT (domain descriptor table)
domain descriptor table (DMDT)
double free element (DFE)
DQE (descriptor queue element)
DSAB (data set association block)
DSCA (DAE communication area)
DSCB (data set control block)
DSPD (DAE pre-dump/post dump parameter list)
DSTBL (data set work area)
DVCT (device characteristic table)

DVT (display allocation vector table)
dynamic address translation (DAT)

EBCDIC (extended binary coded decimal interchange code)
EC (extended control)
ECB (event control block)
ECSA (extended common service area)
EDT (eligible device table)
eligible device table (EDT)
ELPA (extended link pack area)
ELSQA (extended local system queue area)
EMLPA (extended modified link pack area)
ENF (event notification facility)
entry point (EP)
entry table descriptor (ETD)
entry table information block (ETIB)
EPFP (extended precision floating point)
EPLPA (extended pageable link pack area)
EQSRD (extended quick start record)
ERP (error recovery procedure)
ERPIB (error recovery procedure information block)
error recovery procedure (ERP)
error recovery procedure information block (ERPIB)
ESD (external symbol dictionary)
ESDID (external symbol dictionary ID)
ESQA (extended system queue area)
ESR (extended SVC router)
ESW (extended status word)
ETD (entry table descriptor)
ETIB (entry table information block)
event control block (ECB)
event notification facility (ENF)
extended binary coded decimal interchange code (EBCDIC)
extended common service area (ECSA)
extended control (EC)
extended link pack area (ELPA)
extended local system queue area (ELSQA)
extended modified link pack area (EMLPA)
extended pageable link pack area (EPLPA)
extended precision floating point (EPFP)
extended quick start record (EQSRD)
extended status word (ESW)
extended SVC router (ESR)
extended system queue area (ESQA)
external page table (XPT)
external page table entry (XPTE)
external symbol dictionary (ESD)
external symbol dictionary ID (ESDID)

FBLDL (fixed build directory list)
FBQE (free block queue element)
first level interrupt handler (FLIH)
fixed build directory list (FBLDL)
fixed link pack area (FLPA)
FLIH (first level interrupt handler)
FLPA (fixed link pack area)
FQE (free queue element)
free block queue element (FBQE)
free queue element (FQE)
FRR (functional recovery routine)
functional recovery routine (FRR)

GCC (global resource serialization CTC-driver control card table)
GDA (global data area)
Generalized Trace Facility (GTF)
GGSA (global group summary area)
global data area (GDA)
global group summary area (GGSA)
global queue hash table (GQHT)
global queue work area (GQWA)
global resource pool table (GRPT)
global resource serialization vector table (GVT)
global resource serialization vector table extension (GVTX)
global system duplex area (GSDA)

global work/save area vector table (WSAG)
GMT (Greenwich Mean Time)
GQHT (global queue hash table)
GQWA (global queue work area)
Greenwich Mean Time (GMT)
GRPT (global resource pool table)
GSDA (global system duplex area)
GTF (Generalized Trace Facility)
GVT (global resource serialization vector table)
GVTX (global resource serialization vector table extension)

hardware save area (HSA)
HOTIO (hot input/output)
HSA (hardware save area)

ICHPT (installed channel path table)
ICT (input/output control table)
ID (identifier)
identifier (ID)
IEL (initiator entrance list)
initial program loader (IPL)
initiator entrance list (IEL)
input/output (I/O)
input/output block (IOB)
input/output control table (ICT)
input/output management block (IOMB)
input/output supervisor (IOS)
input/output supervisor communication extension table (IOCX)
installation performance specifications (IPS)
installed channel path table (ICHPT)
interruption queue element (IQE)
interruption response block (IRB)
I/O (input/output)
IOB (input/output block)
IOCX (input/output supervisor communication extension table)
IOMB (input/output management block)
IOS (input/output supervisor)
IOSB (input/output supervisor block)
IPL (initial program loader)
IPL resource initialization module (IRIM)
IPL vector table (IVT)
IPS (installation performance specifications)
IQE (interruption queue element)
IRB (interruption response block)
IRIM (IPL resource initialization module)
IVT (IPL vector table)

JCL (job control language)
JES (job entry subsystem)
JESCT (job entry subsystem control table)
job control language (JCL)
job entry subsystem (JES)
job entry subsystem control table (JESCT)
job step control block (JSCB)
JSCB (job step control block)
JSSNT (subsystems name table)

K (1024 bytes)

LCCA (logical configuration communications area)
LDA (local data area)
LGN (logical group number)
LGRAS (logical group of unusable slots)
LGSA (local group summary area)
LGVT (logical group vector table)
link pack area (LPA)
link pack directory entry (LPDE)
linkage index allocation table (LXAT)
LLE (load list element)
load list element (LLE)
local data area (LDA)
local group summary area (LGSA)

local queue hash table (LQHT)
local queue work area (LQWA)
local resource pool table (LRPT)
local system queue area (LSQA)
logical configuration communications area (LCCA)
logical group number (LGN)
logical group vector table (LGVT)
logical path control block (LPB)
logical slot identifier (LSID)
LOGREC record block (LRB)
LPA (link pack area)
LPB (logical path control block)
LPB table (LPBT)
LPBT (LPB table)
LPDE (link pack directory entry)
LQHT (local queue hash table)
LQWA (local queue work area)
LRB (LOGREC record block)
LRPT (local resource pool table)
LSID (logical slot identifier)
LSQA (local system queue area)
LXAT (linkage index allocation table)

machine check handler (MCH)
main storage control table (MCT)
mass storage system (MSS)
mass storage system communicator (MSSC)
master scheduler resident data area (BASEA, MSRDA)
Mb (megabyte)
MC (monitor call)
MCH (machine check handler)
MCHEAD (monitor call tables head)
MCRWSA (monitor call routing work/save area)
MCS (multiple console support)
MCT (main storage control table)
megabyte (Mb)
message request block (MRB)
MIH (missing interruption handler)
missing interruption handler (MIH)
MLPA (modified link pack area)
modified link pack area (MLPA)
monitor call (MC)
monitor call routing work/save area (MCRWSA)
monitor call tables head (MCHEAD)
monitoring and service support facility (MSSF)
MP (multiprocessing)
MPL (multiprogramming level)
MQE (message queue element)
MQH (message queue header)
MRB (message request block)
MSRDA (master scheduler resident data area)
MSS (mass storage system)
MSSC (mass storage system communicator)
MSSF (monitoring and service support facility)
multiple console support (MCS)
multiprocessing (MP)
multiprogramming level (MPL)

NCT (NIP console table)
NIP (nucleus initialization program)
NIP parameter address table (PARMTAB)
NIP parameter area (NIPPAREA)
NIP vector table (NVT)
NIPPAREA (NIP parameter area)
NLL nucleus load list
NLLE nucleus load list element
nucleus initialization program (NIP)
nucleus load list (NLL)
nucleus load list element (NLLE)
NVT (NIP vector table)

operation request block (ORB)
operator reply element (ORE)

ORB (operation request block)
ORE (operator reply element)

page control block (PCB)
page frame table (PFT)
page frame table entry (PFTE)
page table (PGT)
page table entry (PGTE)
pageable display control module (TDCM)
pageable link pack area (PLPA)
paging activity reference table (PART)
paging channel command work area (PCCW)
PARMTAB (NIP parameter address table)
PART (paging activity reference table)
partitioned data set (PDS)
PAT (ASM page allocation table)
PC (program call)
PCB (page control block)
PCCA (physical communications configuration area)
PCCA vector table (PCCAVT)
PCCAVT (PCCA vector table)
PCCW (paging channel command work area)
PCI (program-controlled interrupt)
PDS (partitioned data set)
PEXB (pool extent block)
PFK (program function key)
PFT (page frame table)
PFTE (page frame table entry)
PGT (page table)
PGTE (page table entry)
physical communications configuration area (PCCA)
PLPA (pageable link pack area)
pool extent block (PEXB)
PRB (program request block)
prefixed SAVE area (PSA)
processor work area (PWA)
program call (PC)
program-controlled interrupt (PCI)
program function key (PFK)
program request block (PRB)
program status word (PSW)
PSA (prefixed SAVE area)
PSW (program status word)
PVT (RSM paging vector table)
PWA (processor work area)

QCB (queue control block)
QDB (queue descriptor block)
QEL (queue element)
QSRCD (quick start record)
queue control block (QCB)
queue descriptor block (QDB)
queue element (QEL)
queue extension block (QXB)
queue work block (QWB)
quick start record (QSRCD)
QWB (queue work block)
QXB (queue extension block)

RAB (RSM address space control block)
RB (request block)
RBA (relative byte address)
RBN (real block number)
RCB (recording control block)
RCBSRB (recording task SRB)
RCT (region control task, routing control table)
RD (region descriptor)
RDCM (resident display control module)
real block number (RBN)
real storage management (RSM)
reconfigurable storage unit (RSU)
recording control block (RCB)
recording task ECB (RTCTECB)
recording task SRB (RCBSRB)

recovery management support (RMS)
recovery management support secondary communications vector table (RVT)
recovery termination management (RTM)
recovery termination management control table (RTCT)
region control task (RCT)
region descriptor (RD)
relative byte address (RBA)
relocation dictionary (RLD)
request block (RB)
resident display control module (RDCM)
resource information block (RIB)
resource information block extension (RIBE)
resource initialization module (RIM)
resource name list (RNL)
resource queue area (RQA)
RIB (resource information block)
RIBE (resource information block extension)
RIM (resource initialization module)
RIT (RSM internal table)
RLCT (system resources manager logical channel table)
RLD (relocation dictionary)
RMCT (system resources manager control table)
RMEP (system resources manager algorithm entry point block)
RMPT (system resources manager parameter table)
RMS (recovery management support)
RNL (resource name list)
RQA (resource queue area)
RSM (real storage management)
RSM address space control block (RAB)
RSM external page table (XPT)
RSM external page table entry (XPTE)
RSM internal table (RIT)
RSM paging vector table (PVT)
RSU (reconfigurable storage unit)
RTCT (recovery termination control table, recovery termination management control table)
RTCT extension for SVC dump (RTSD)
RTCTECB (recording task ECB)
RTM (recovery termination management)
RTSD (RTCT extension for SVC dump)
RVT (recovery management support secondary communications vector table)

SAHT (SYSID/ASID hash table)
SALLOC (storage allocation)
SART (swap activity reference table)
SAST (subsystem allocation sequence table)
SAT (swap allocation table)
SCCW (swap channel command work area)
SCD (status collection data area)
scheduler NIP parameter list (SNPL)
scheduler work area (SWA)
SCHIB (subchannel information block)
SCSW (subchannel status word)
SCVT (secondary communications vector table)
SDDSQ (SDUMP dump data set queue)
SDT (start descriptor table)
SDUMP dump data set queue (SDDSQ)
SDWA (system diagnostic work area)
second level interrupt handler (SLIH)
second level table (SLT)
secondary communications vector table (SCVT)
segment table (SGT)
segment table entry (SGTE)
service request block (SRB)
SET (system entry table)
set task asynchronous exit (STAE)
SFQ (SQA frame queue)
SFT (system function table)
SGT (segment table)
SGTE (segment table entry)
SHAS (subsystem hash table)

SIC (system-initiated cancel)
SIRB (system interrupt request block)
size queue anchor table (SQAT)
SLH (subchannel logout handler)
SLIH (second level interrupt handler)
SLR (subchannel logout record)
SLT (second level table)
SMCA (system management control area)
SMEW (summary dump extended work area)
SMF (system management facilities)
SMPL (storage manager parameter list)
snap dump (ABDUMP)
SNPL (scheduler NIP parameter list)
SPE (system parameter element)
SPQE (subpool queue element)
SQA (system queue area)
SQA frame queue (SFQ)
SQAT (size queue anchor table)
SRB (service request block)
SRM (system resources manager)
SSCVT (subsystem communications vector table)
SSIB (subsystem identification block)
SSPL (subsystem service routine parameter list)
SSVT (subsystem vector table)
STAE (set task asynchronous exit)
start descriptor table (SDT)
started task control internal reader (STCINRDR)
STCINRDR (started task control internal reader)
storage manager parameter list (SMPL)
subchannel information block (SCHIB)
subchannel logout handler (SLH)
subchannel logout record (SLR)
subchannel status word (SCSW)
subpool queue element (SPQE)
subsystem allocation sequence table (SAST)
subsystem communications vector table (SSCVT)
subsystem hash table (SHAS)
subsystem identification block (SSIB)
subsystem service routine parameter list (SSPL)
subsystem vector table (SSVT)
subsystems name table (JSSNT)
summary dump extended work area (SMEW)
supervisor call (SVC)
supervisor request block (SVRB)
supervisor vector table (SVT)
SVC (supervisor call)
SVT (supervisor vector table)
SVRB (supervisor request block)
SWA (scheduler work area)
swap activity reference table (SART)
swap allocation table (SAT)
swap channel command work area (SCCW)
SYSGEN (system generation)
SYSID/ASID hash table (SAHT)
SYSRES (system resident device)
system diagnostic work area (SDWA)
system entry table (SET)
system function table (SFT)
system generation (SYSGEN)
system-initiated cancel (SIC)
system management control area (SMCA)
system management facilities (SMF)
system parameter element (SPE)
system queue area (SQA)
system resident device (SYSRES)
system resources manager (SRM)
system resources manager algorithm entry point block (RMEP)
system resources manager control table (RMCT)
system resources manager logical channel table (RLCT)
system resources manager user control block (UOCB)
system resources manager user extension block (UOXB)
system trace buffer (TBUF)
system trace buffer vector table (TBVT)

task control block (TCB)
task input-output table (TIOT)
TBUF (system trace buffer)
TBVT (system trace buffer vector table)
TCB (task control block)
TCWA (TOD clock work area)
TDCM (pageable display control module)
temporary page activity reference table (TPARTBLE)
time-of-day (TOD)
time sharing option (TSO)
time sharing option internal reader (TSOINRDR)
timer queue element (TQE)
timer work area (TPC)
TIOT (task input-output table)
TLB (translation lookaside buffer)
TOD (time-of-day)
TOD clock work area (TCWA)
TPARTBLE (temporary page activity reference table)
TPC (timer work area)
TQE (timer queue element)
translation lookaside buffer (TLB)
TSO (time sharing option)
TSOINRDR (time sharing option internal reader)

UADS (user attribute data set)
UCB (unit control block)
UCB pointer list (UPL)
UCM (unit control module)
UCME (unit control module entry)
unit control block (UCB)
unit control module (UCM)
unit control module entry (UCME)
UOCB (system resources manager user control block)
UOXB (system resources manager user extension block)
UPL (UCB pointer list)
user attribute data set (UADS)

V=R (virtual equals real)
VBP (virtual block processor)
VIO (virtual input/output)
virtual block processor (VBP)
virtual equals real (V=R)
virtual input/output (VIO)
virtual storage access method (VSAM)
virtual storage management (VSM)
volume table of contents (VTOC)
VSAM (virtual storage access method)
VSAM access method data statistics block (AMDSB)
VSM work area (VSWK)
VSWK (VSM work area)
VTOC (volume table of contents)

WMST (workload manager specification table)
workload manager specification table (WMST)
WQE (write queue element)
write queue element (WQE)
write-to-operator (WTO)
write-to-operator with reply (WTOR)
WSAG (global work/save vector table)
WTO (write-to-operator)
WTOR (write-to-operator with reply)

XMD (cross memory directory)
XPT (RSM external page table)
XPTE (RSM external page table entry)

INDEX

A

D

E

F

G

H

I

J

K

L

M

N

Q

R

S

T

U

V

W

MVS/Extended Architecture System Initialization Logic

LY28-1200-4

S370-36

Printed in U.S.A.

MVS/Extended Architecture
System Initialization
Logic

READER'S
COMMENT
FORM

LY28-1200-4

This manual is part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Note: ***Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.***

Possible topics for comment are:

Clarity Accuracy Completeness Organization Coding Retrieval Legibility

If you wish a reply, give your name, company, mailing address, and date:

_____

_____

_____

_____

What is your occupation? _____

How do you use this publication? _____

Number of latest Newsletter associated with this publication: _____

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 or you may mail directly to the address in the Edition Notice on the back of the title page.)

MVS/Extended Architecture System Initialization Logic

LY28-1200-4 S370-36

Reader's Comment Form

Cut or Fold Along Line

Fold and tape Please Do Not Staple Fold and tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines Corporation
Department D58, Building 921-2
PO Box 390
Poughkeepsie, New York 12602

Fold and tape Please Do Not Staple Fold and tape

Printed in U.S.A.

IBM®