SY27-7243-1

Systems

OS/VS2 IPL and NIP Logic

VS2 Release 1.6

IBM

Second Edition (September, 1974)

This edition is a minor revision of SY27-7243-0 incorporating TNL GN27-1406. These editions are still current.

This publication applies to release 1.6 of OS/VS2 and to all subsequent releases unless otherwise indicated in new editions or technical newsletters. Changes are continually made to the specifications in this book; before using this book, consult the most recent edition of the IBM System/360 and System/370 Bibliography, GA22-6822, and the current SRL Newsletter for editions that are applicable and current.

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

A form for reader's comments appears at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, Programming Publications, Department 636, Neighborhood Road, Kingston, New York 12401. All comments become the property of IBM.

© Copyright International Business Machines Corporation 1972

This publication describes the logic of IPL (initial program loader) and NIP (nucleus initialization program) for OS/VS2. The information is intended for use by personnel responsible for program maintenance. IPL loads the nucleus designated by the user and prepares the system for initialization. NIP initializes the nucleus and the remainder of real storage in preparation for system execution.

ORGANIZATION OF THIS BOOK

This publication contains six sections.

"Section 1. Introduction" provides an overview of the functions performed by IPL/NIP and contains background information on linkage between modules and subroutines. The section also describes real storage areas, data sets used by IPL/NIP, formatting page data sets, and the quickstart/coldstart process.

"Section 2. Method of Operation" describes in more detail the functions performed by the IPL/NIP modules. Method-of-operation diagrams show the input, processing, and output of modules and subroutines.

"Section 3. Program Organization" describes the structure of the IPL/NIP modules. It also contains flowcharts of complex areas of code.

"Section 4. Directory" contains a directory of entry points, showing module names, function, and diagram number.

"Section 5. Data Areas" contains descriptions of control blocks and data areas used by IPL/NIP.

"Section 6. Diagnostic Aids" contains messages and wait state codes issued by the IPL/NIP modules. It also contains a register usage table and a module data field cross-reference table.

PREREQUISITE READING

Readers of this publication should be familiar with the following OS/VS publications:

<u>OS/VS2 Supervisor Logic</u>, SY27-7244.
<u>OS/VS Supervisor Services and Macro Instructions</u>, GC27-6979.

Supplementary Reading

Information in the following publications may be helpful in understanding IPL and NIP:

<u>VS2 System Messages</u>, GC28-1002.
<u>OS/VS System Codes</u>, GC28-1003.
<u>VS2 Operator's Reference</u>, GC38-0210.
<u>VS2 System Data Areas</u>, SY28-0606.

CONTENTS

IPL (initial program loader) and NIP (nucleus initialization program) initialize the VS2 system by loading the supervisor and preparing the system for execution. Because the functions of IPL and NIP are related, and because NIP execution immediately follows IPL execution, IPL and NIP are treated in this publication as one component.

Diagrams 0.1 and 0.2 show the main functions performed by IPL and NIP.

THE INITIAL PROGRAM LOADER

IPL is brought into real storage when the operator presses the LOAD button. (See "Section 2: Method of Operation" for a description of how IEAIPL00 is loaded.)

IPL loads the resident nucleus that was created during system generation (either a nucleus designated by the user or one assigned by the system). IPL then limits real storage to the size specified by the user (if defaulted, it is to the actual size of real storage), and clears real storage up to the effective size to 0's. IEAIPL00 initializes system tables and general registers for use during NIP execution.

After IPL has been executed, control is passed to the first NIP module, IEAVNIP0, which was loaded as part of the resident nucleus.

THE NUCLEUS INITIALIZATION PROGRAM

NIP initializes real storage, provides access to the various system data sets, and provides access to virtual storage by initializing the appropriate paging tables. Because NIP performs its functions sequentially, and because each function is performed primarily by one module (often with assistance from a service module), NIP execution is described on a module basis.

The modules IEAVNIPM and IEAVNP02 each performs special sequential processing, but those modules also contain subroutines that perform common functions for one or more other NIP modules. IEAVNPA4 and IEAVNPA5 function solely as service routines.

The NIP modules perform the following functions:

- Define the initial part of the NVT (NIP vector table), define the SQA (system queue area), define the nucleus buffer, define NIP's dynamic area, provide access to the SYS1.NUCLEUS data set, and place the CPU in EC (extended control) mode with translation enabled (IEAVNIP0).
- Control the sequence of execution for NIP modules and handle requests for execution of nonresident SVC routines (IEAVNIPM).
- Initialize system consoles (IEAVNP01).
- Initialize the system device configuration, the SVC library (SYS1.SVCLIB), and the system log data set (SYS1.LOGREC) (IEAVNP02).
- Process parameters specified by the system programmer and the operator (IEAVNP03).
- Initialize paging control blocks and data areas for the paging supervisor (IEAVNP04 and IEAVNPA4).
- Define link pack area modules for quickstart or coldstart processing (IEAVNP05 and IEAVNPA5).
- Initialize the reliability and serviceability (RAS) features (IEAVNP06).
- Define the hardcopy console, pageable DCMs (display control modules), and the paging algorithm limit values (IEAVNP07).
- Define the nonpageable dynamic area, quickcell tables, available storage, the master scheduler's region, and the dispatcher's APG (authorized priority group) table and time-slice queues (IEAVNIPX).

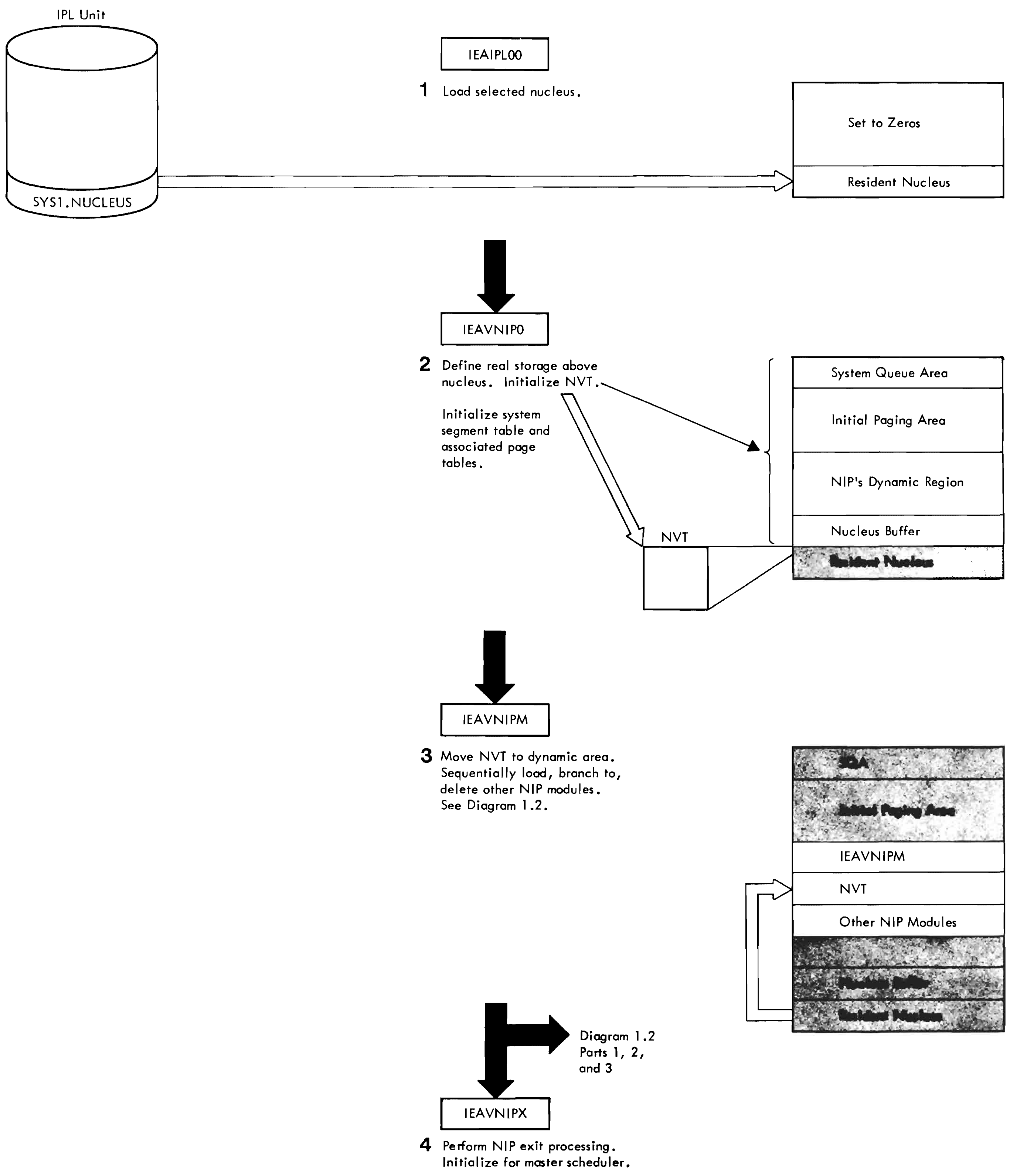

IPL Unit
SYS1.NUCLEUS
IEAIPL00
1 Load selected nucleus.
Set to Zeros
Resident Nucleus
IEAVNIP0
2 Define real storage above nucleus. Initialize NVT.
Initialize system segment table and associated page tables.
System Queue Area
Initial Paging Area
NIP's Dynamic Region
Nucleus Buffer
NVT
Resident Nucleus
IEAVNIPM
3 Move NVT to dynamic area. Sequentially load, branch to, delete other NIP modules. See Diagram 1.2.
SQA
Initial Paging Area
IEAVNIPM
NVT
Other NIP Modules
Nucleus Buffer
Resident Nucleus
Diagram 1.2 Parts 1, 2, and 3
IEAVNIPX
4 Perform NIP exit processing. Initialize for master scheduler.

Diagram 0.1: IPL/NIP processing

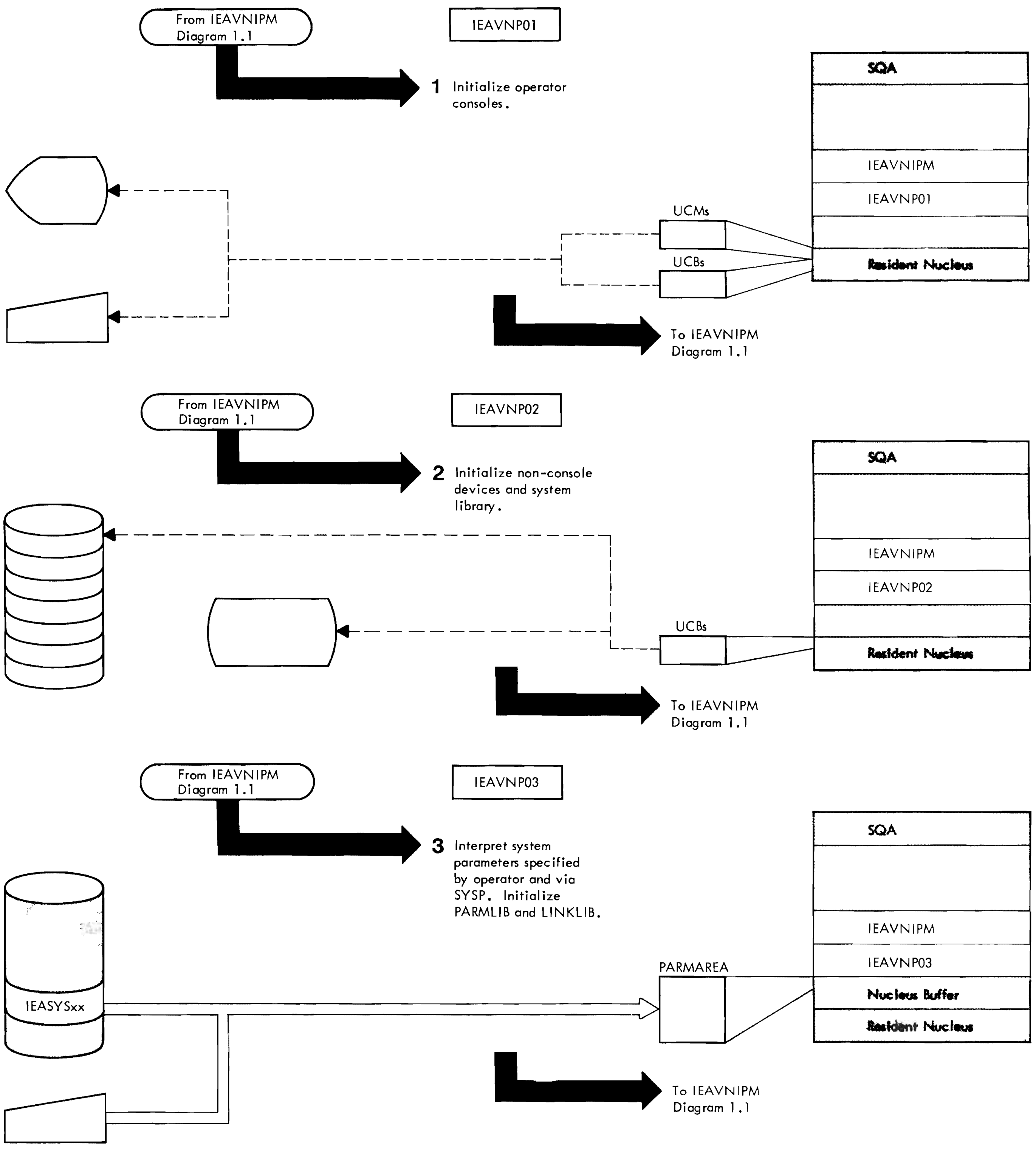
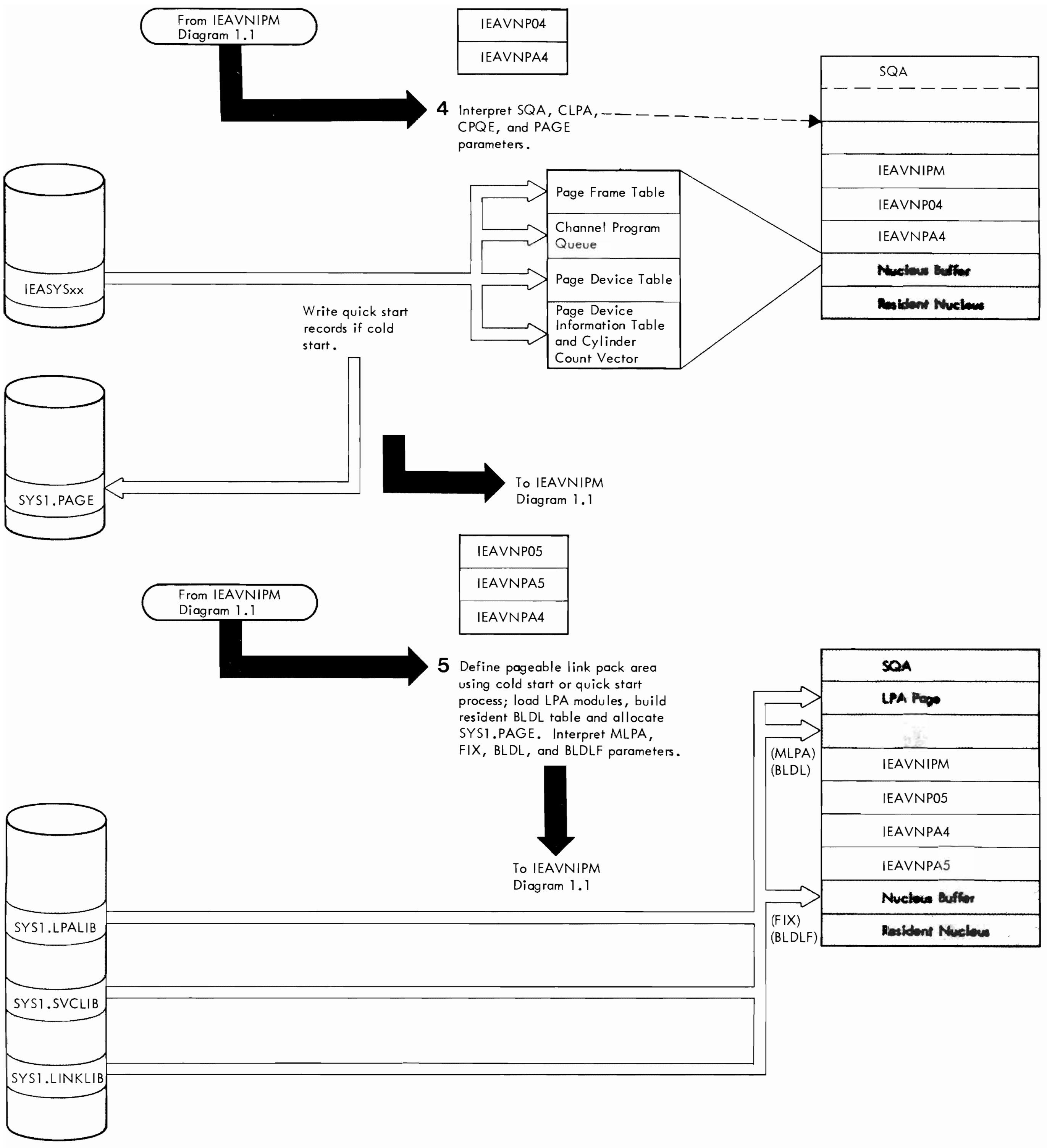

From IEAVNIPM Diagram 1.1
IEAVNP01
1 Initialize operator consoles.
SQA
IEAVNIPM
IEAVNP01
UCMs
UCBs
Resident Nucleus
To IEAVNIPM Diagram 1.1
From IEAVNIPM Diagram 1.1
IEAVNP02
2 Initialize non-console devices and system library.
SQA
IEAVNIPM
IEAVNP02
UCBs
Resident Nucleus
To IEAVNIPM Diagram 1.1
From IEAVNIPM Diagram 1.1
IEAVNP03
3 Interpret system parameters specified by operator and via SYSP. Initialize PARMLIB and LINKLIB.
SQA
IEAVNIPM
IEAVNP03
PARMAREA
Nucleus Buffer
Resident Nucleus
IEASYSxx
To IEAVNIPM Diagram 1.1

Diagram 0.2: NIP processing under control of IEAVNIPM

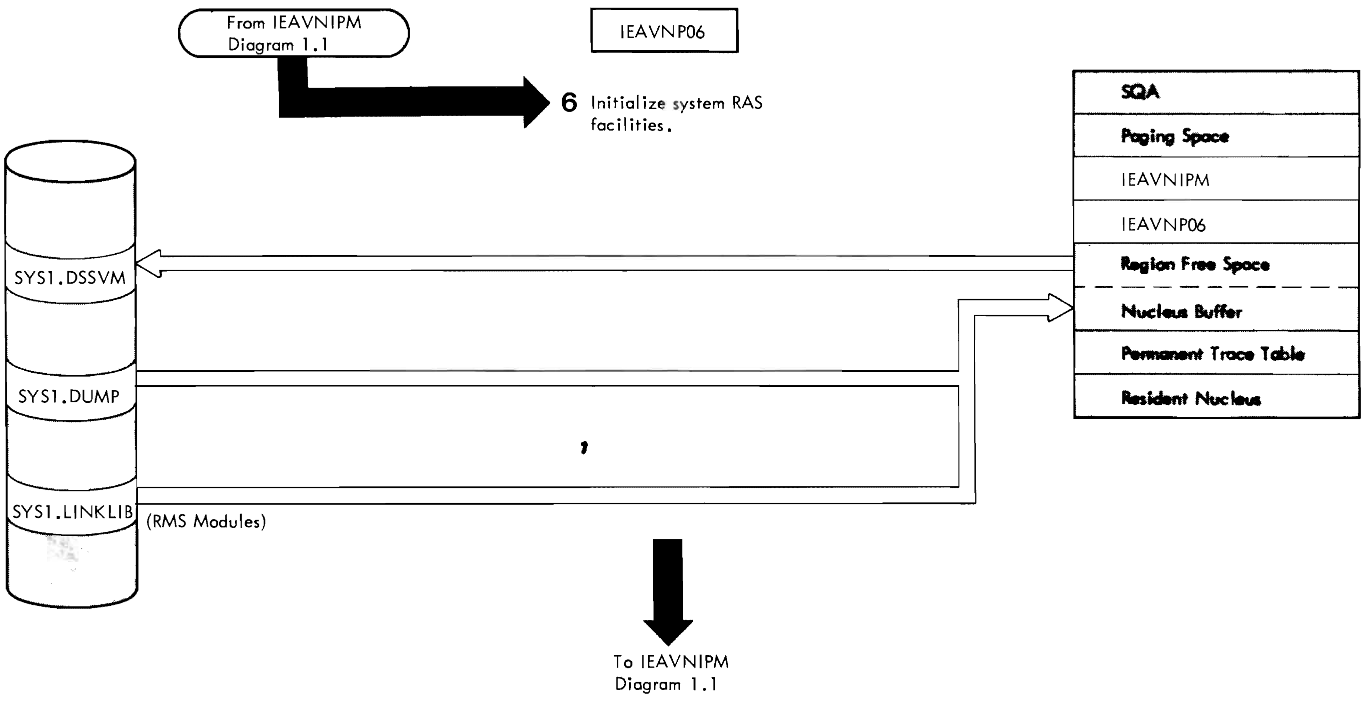


4 Interpret SQA, CLPA, CPQE, and PAGE parameters.

Write quick start records if cold start.

5 Define pageable link pack area using cold start or quick start process; load LPA modules, build resident BLDL table and allocate SYS1.PAGE. Interpret MLPA, FIX, BLDL, and BLDLF parameters.

Diagram 0-2: NIP processing under control of IEAVNIPM (Part 2 of 3)



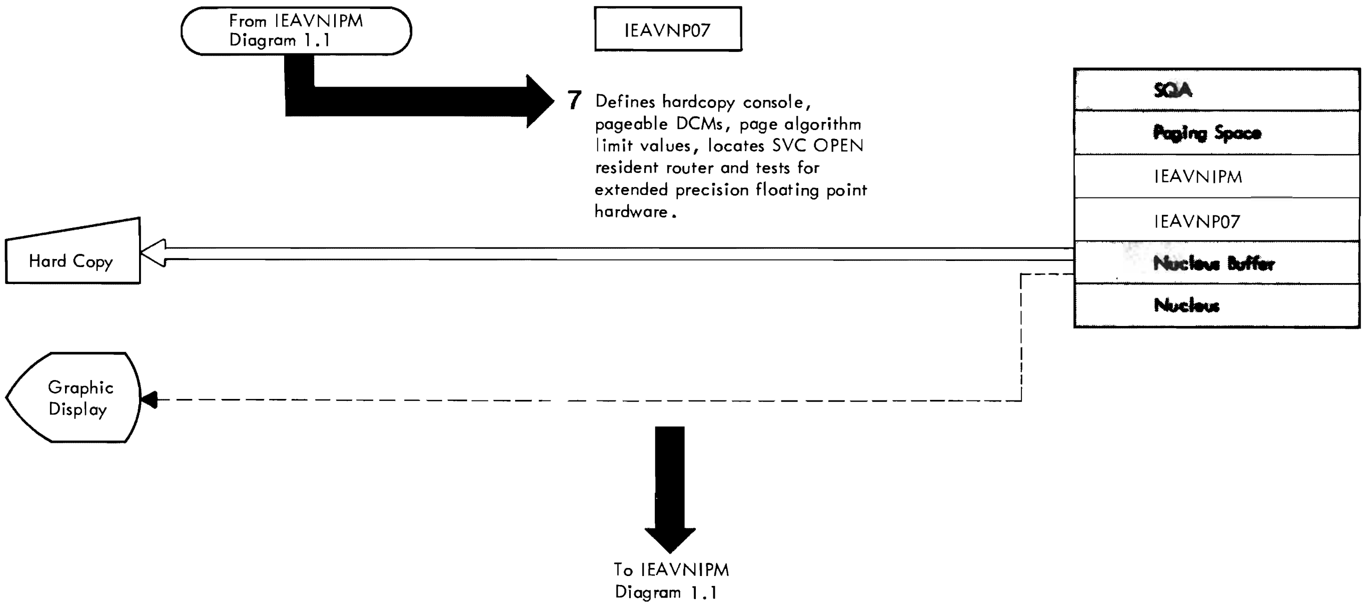

From IEAVNIPM Diagram 1.1
IEAVNP07
7 Defines hardcopy console, pageable DCMs, page algorithm limit values, locates SVC OPEN resident router and tests for extended precision floating point hardware.
SQA
Paging Space
IEAVNIPM
IEAVNP07
Nucleus Buffer
Nucleus
Hard Copy
Graphic Display
To IEAVNIPM Diagram 1.1

Diagram 0-2: NIP processing under control of IEAVNIPM (Part 3 of 3)

LINKAGE TO SERVICE SUBROUTINES

In addition to mainline processing, the NIP modules IEAVNIPM, IEAVNP02, and IEAVNPA4 contain subroutines that perform common functions for other NIP modules. The IEAVNIPM and IEAVNP02 subroutines are executed in response to the NIP macro instruction IEAPMNIP. When an IEAVNIPM subroutine is requested, the macro expands to contain the address of the required subroutine and a BAL instruction to that address. When the IEAPMNIP macro instruction specifies a subroutine within IEAVNP02, the macro expansion contains a BAL to the NIPLOAD subroutine within IEAVNIPM to load IEAVNP02, a BAL to the IEAVNP02 subroutine entry point, and an SVC 9 (DELETE) instruction.

IEAVNPA4 is initially brought into storage when the IEAPMNIP macro instruction is issued by IEAVNP04 (the expansion contains the BAL to NIPLOAD to load IEAVNPA4 and a BAL to the entry point). Subsequent entries from IEAVNP04, IEAVNP05, and IEAVNPA5 are by a BALR instruction which contains the entry address found in the NIP vector table. The particular subroutine required is indicated by bit settings in the parameter list passed in register 1. The last linkage from IEAVNP05 to an IEAVNPA4 subroutine is also by the IEAPMNIP macro. This expansion contains an SVC 9 (DELETE) instruction to logically remove IEAVNPA4 from storage.

PAGE DATA SET FORMATTING AND THE QUICKSTART/COLDSTART PROCESS

One of the required NIP functions is to initialize the page data sets. Each paging device specified during system generation or by the operator using the PAGE parameter is considered a page data set and is referred to as such in the following discussion.

For each unformatted page data set, only the "home address" and record 0 contents of any track are known, and the remainder of the track must be formatted into records of specified size and characteristics. NIP formats each page data set by executing a channel program which writes the count, key, and data for the specified number of records. The channel program varies depending on the type of device containing the data set being formatted and the PAGE parameter specifications.

For each data set that is formatted, NIP maintains a record, called a quickstart record, or NIPQSR1. Each NIPQSR1 record contains the number of tracks formatted, the number of pages in the data set that are available (a page is unavailable if the track is defective or if it will be allocated to NIPQSR1), and a "bit map" that indicates which pages are available. Each NIPQSR1 record is written to record 1 of the first track of the page data set.

The NIPQSR1 records serve an important function when the operator must restart the system by performing another IPL. During NIP processing for the restart, NIP checks the NIPQSR1 record for each page data set to determine whether the data set must be reformatted. NIP performs the I/O operations required to reformat a page data set only when:

- The NIPQSR1 record cannot be read (either the data set was not formatted or the NIPQSR1 record is defective).

- As part of the system restart, the operator has entered new specifications for the data set, and the data set is too small to meet the requirements.

- The operator has entered new specifications for the data set, and a factor other than size requires the data set to be reformatted.

Because the pages for the LPA (link pack area) remain static during execution, two additional quickstart records are created to provide additional information about the LPA page data set. The additional records are NIPQSR2 and NIPQSR3.

NIPQSR2 contains the virtual address of the LPA directory, the highest virtual address assigned to the LPA, and a bit map of available pages in the data set. NIPQSR3 contains a slot/group map of the sequential LPA address space. There may be one to three NIPQSR3 records for the LPA page data set.

<u>Quickstart and Coldstart Process</u>: During the first IPL for a system, NIP reads the LPA modules from the SYS1.LPALIB data set and writes (loads) them into the LPA page data set. At the same time, NIP builds the NIPQSR2 and NIPQSR3 records and writes them to the data set.

Normally, the LPA modules are not reloaded during a subsequent IPL. The only time they are reloaded is when:

- The NIPQSR1 record for the LPA page data set cannot be read, forcing NIP to reformat the data set and reload the modules because it has no assurance that the data set has previously been formatted.

- The operator specifies the CLPA parameter, indicating that the contents of the LPA are to be modified during the restart.

- The PAGE parameter specifies that another data set is to contain the LPA.
- The SQA system parameter is so specified that SQA infringes on the LPA virtual address space, and the operator accepts this condition.

The initialization process in which the LPA modules must be loaded into the data set is called the coldstart process.

The process of reinitializing the system without reloading the LPA modules (that is, by using the LPA page data set created during a previous initialization) is called the quickstart process. This name refers to the time saved by not having to reload the LPA modules.

DATA SETS USED BY IPL/NIP

Because all data used by IPL/NIP originates in or will reside in system data sets, it is important to be familiar with the following list of these data sets and brief descriptions of their contents.

- SYS1.NUCLEUS -- Contains the resident nucleus and all NIP modules.
- SYS1.LOGREC -- Used for recording I/O and hardware errors.
- SYS1.LINKLIB -- Contains user and system programs.
- SYS1.SVCLIB -- Contains the type 3 and 4 SVC routines used during NIP processing and RMS routines.
- SYS1.LPALIB -- Contains all of the modules for the link pack area.
- SYS1.PAGE -- Contains the auxiliary space for system paging.
- SYS1.DUMP -- Used for dumping the contents of storage.
- SYS1.PARMLIB -- Contains the system parameters referred to with the SYSP parameter from the operator console and the LINKLIB concatenation member, LNKLST00, and the MLPA, FIX, and BLDL module name lists.
- SYS1.DSSVM -- Contains the dynamic support system virtual storage.

REAL STORAGE AREAS DURING NIP EXECUTION

During NIP execution, real storage is logically divided into several areas. Because these areas are continually referred to in the method-of-operation diagrams and the text in Section 2, brief descriptions of these areas and their contents are provided. The following are descriptions of the real storage areas in ascending order (from the lowest address to the highest):

- Nucleus: This is the lowest area of real storage and contains the modules, system control blocks, and system data areas that remain in real storage during execution. Because they remain in real storage, no paging is required. The nucleus is created during system generation, and some of its control blocks are initialized during IPL/NIP execution.
- Nucleus Buffer: This area occupies the space immediately above the nucleus. It contains additional control blocks and data areas created by NIP. The nucleus buffer becomes a logical and physical extension of the nucleus. Two fields in the NIP vector table (NVTNUCND and NVTBUFND) point to the next area in the nucleus buffer into which additional data can be placed, and the upper limit of the nucleus buffer, respectively.
- NIP's Dynamic Area: This area is immediately above the nucleus buffer and contains the NIP modules that are required for execution of NIP. The modules are loaded into the highest available area within the dynamic area.
- Initial Paging Area: This area is immediately above NIP's dynamic area and is used for paging functions during NIP execution.
- System Queue Area: This area is in the highest real storage space. It contains the control blocks and queue elements required for virtual storage supervision and the segment and page tables required for paging supervision.

DATA AREAS AND CONTROL BLOCKS USED BY NIP

The following data areas and control blocks are used only during NIP execution. They are either overlaid during system execution (for example, the NVT and the PARMAREA) or are not used (for example, the quickstart records).

NIP Vector Table (NVT)

The NVT contains pointers and flags that are used only by NIP. Its relationship to NIP execution is similar to that of the CVT to system execution. The NVT base is created by IEAVNIP0 and is moved into and further initialized by IEAVNIPM. It is used by all NIP modules.

Quickstart Records

There are three types of quickstart records. The first type, NIPQSR1, is created and written to a page data set when the data set is formatted. The other two types of records, NIPQSR2 and NIPQSR3, are created and written only for page data sets containing link pack area modules. The contents and use of the quickstart records are more fully described under "Page Data Set Initialiization and the Quickstart/ Coldstart Process" in this section.

PARMAREA

The PARMAREA is created by IEAVNP03 and initialized with the system parameters specified by the system programmer or by the operator. It also contains a table of addresses (PARMTAB) that point to the system parameters and a data area required for reading the system parameters from the SYS1.PARMLIB.

This section uses text and diagrams to describe the functions performed by IPL/ NIP. The method-of-operation diagrams are the primary vehicle for describing the IPL/ NIP functions. Notice that the diagrams emphasize the sequence of functions rather than the detailed step-by-step logic found in flowcharts. For information on how to use the diagrams, please read the next sub-section, "Using Method-of-Operation Diagrams."

The text in this section supplements the diagrams. The text describes briefly the functions performed by each module and the major subroutines within each module. Detailed descriptions of complex processing are given in "Section 3: Program Organization."

The initial program loader is the fourth record on cylinder 0, track 0 of the system residence volume. The nucleus initialization program modules are members of the SYS1.NUCLEUS data set. IPL loads the resident nucleus and the first NIP module, IEAVNIP0, which was previously link-edited with the resident nucleus to form the IEANUC0x member of the SYS1.NUCLEUS data set. IEAVNIP0 in turn causes the loading of the NIP control module, IEAVNIPM, which sequentially loads and deletes IEAVNP01, IEAVNP02, IEAVNP03, IEAVNP04, IEAVNP05, IEAVNP06, and IEAVNP07. A subroutine within IEAVNIPM also causes the loading and deletion of IEAVNP02 when subroutines within it are required by other NIP modules.

Other special loading and deletion operations are:

- IEAVNPA4 is loaded by IEAVNIPM for IEAVNP04 and deleted by IEAVNP05.
- IEAVNPA5 is loaded by IEAVNIPM for IEAVNP05 and deleted by IEAVNP05.
- IEAVNIPX is loaded by IEAVNIPM and is self deleted.

The IPL and NIP modules are described in this section in the order in which they are normally executed.

USING METHOD-OF-OPERATION DIAGRAMS

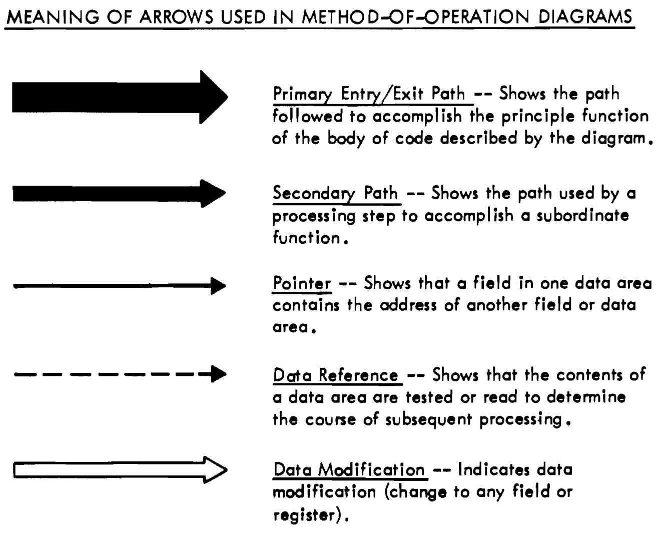
Method-of-operation diagrams are provided for all IPL/NIP modules and most of the subroutines within these modules. These diagrams describe the functions performed by the modules and show the input and output associated with the functions.

Where necessary, the processing steps in a method-of-operation diagram are further explained by "Notes on Processing." Each note amplifies a functional statement in the diagram and gives the label in the code that performs the function. Each note is related to the appropriate processing step by the function number (the number adjacent to the step in the diagram). Where no note is given, it is presumed that the statement in the diagram and the comments in the code provide an adequate explanation of the function. The reference labels included in the "Notes on Processing" provide a means of finding the point in the code at which each function is performed.

To help the reader understand and use these diagrams, this section contains a sample diagram, an explanation of diagramming conventions (Figure 1), and text that explains the sample diagram.

The sample diagram describes the functions performed by module IEAVNzzz. The first five characters in all NIP module names are "IEAVN". The characters "zzz" represent the unique suffix in each module name.

The wide, shaded arrow indicates entry to a module, exits from a module, or major paths of execution within or between modules. Entry to the module in the sample diagram is from IEAVNxxx, Step y.



MEANING OF ARROWS USED IN METHOD-OF-OPERATION DIAGRAMS

Primary Entry/Exit Path -- Shows the path followed to accomplish the principle function of the body of code described by the diagram.

Secondary Path -- Shows the path used by a processing step to accomplish a subordinate function.

Pointer -- Shows that a field in one data area contains the address of another field or data area.

Data Reference -- Shows that the contents of a data area are tested or read to determine the course of subsequent processing.

Data Modification -- Indicates data modification (change to any field or register).

Figure 1. Key to symbols in method-of-operation diagrams

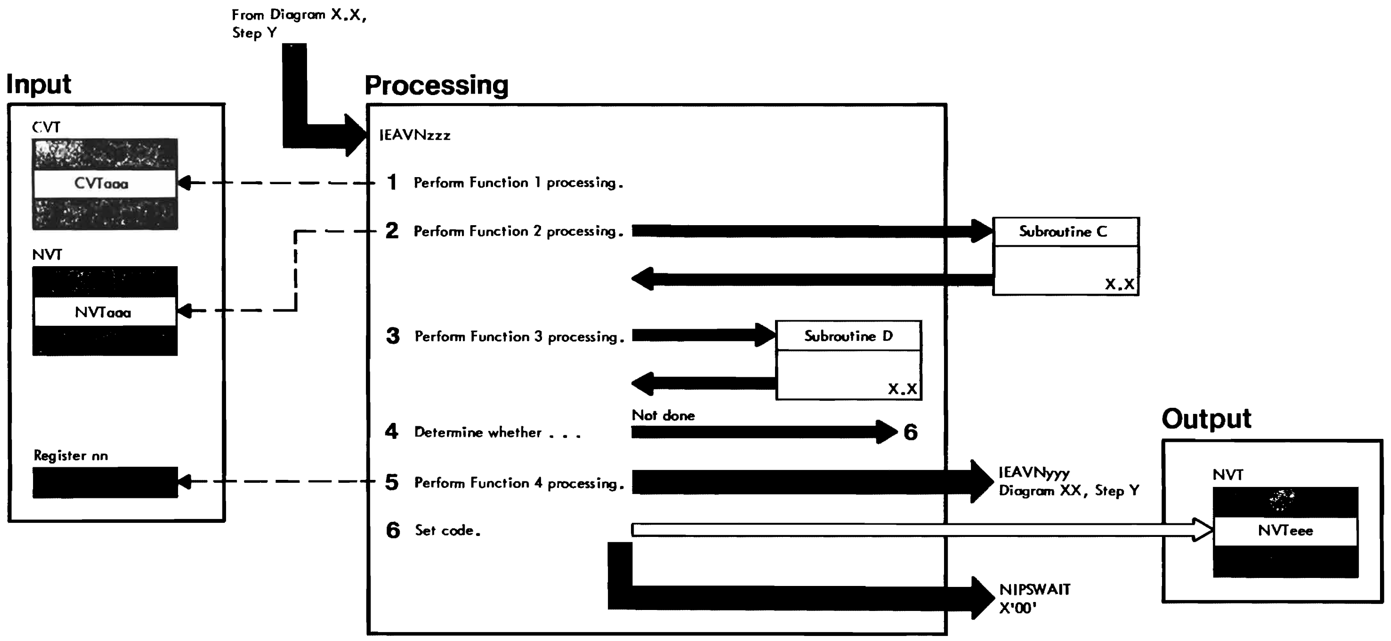

From Diagram X.X, Step Y
Input
CVT
CVTaaa
NVT
NVTaaa
Register nn
Processing
IEAVNzzz
1 Perform Function 1 processing.
2 Perform Function 2 processing.
Subroutine C
X.X
3 Perform Function 3 processing.
Subroutine D
X.X
4 Determine whether . . .
Not done
6
5 Perform Function 4 processing.
IEAVNyyy
Diagram XX, Step Y
6 Set code.
NIPSWAIT
X'00'
Output
NVT
NVTeee

Diagram X.X IEAVNzzz

Notes on Processing	Label
1. This note would explain in more detail the processing for function 1.	IEAVNzzz
2. Control is passed to a subroutine that is not part of this module.	Label2
3. Control is passed to a subroutine that is part of this module.	Label3
4. This note would amplify the decision made.	Label4
5. Exit is to IEAVNyyy.	
6. The system is put in a wait state with the code X'00'.	Label5

Processing Steps 1 and 2 refer to fields CVTaaaaa and NVTbbbbb. Data areas used across the system are described fully in <u>VS2 Data Areas</u>. Data areas pertinent only to NIR are described in Section 5 of this manual. Referring to a field usually involves loading the contents of the field into a register for testing or addressing. A reference to a field does not include modification of the field.

To perform functions 2 and 3, the module in the sample diagram requires the services of other routines. Module C is another module (it may be a NIP module) used to perform function 2. Linkage to another NIP module is achieved by issuing an internal NIP macro instruction, the expansion of which causes (1) the NIPLOAD subroutine in IEAVNIPM to be entered to load the module, (2) a BAL instruction to give control to the module, (3) the required input parameters to be expanded inline, and (4) a subsequent SVC instruction to delete the module. Function 3 requires the processing of Subroutine D which is within this module. Subroutines described in this manner are entered by a BAL instruction and normally return control to the next sequential instruction. Where modules and subroutines are described by method-of-operation diagrams in this manual, the number of the diagram is given in the lower right-hand corner.

In cases where secondary or supportive processing is required, the narrower shaded arrows indicate the flow of processing.

The fifth processing step uses the contents of register nn. The contents of other registers are not shown as input because they are not used by this module.

The normal exit from this module is to module IEAVNqqq. Where appropriate, the number of the step in the diagram to which exit is made is shown.

The error processing that occurs in entering a disabled wait state includes storing a wait state code and branching to the NIPSWAIT subroutine in IEAVNIPM, which places the system in the disabled wait state.

THE IPL RECORDS AND IEAIPL00 (DIAGRAM 1.0)

The initial program loader is the program that initializes real storage and loads the specified control program nucleus. The nucleus is assembled during system generation and resides on the SYS1. NUCLEUS data set. The nucleus contains system control blocks, data areas, resident control program modules, and the first module of the nucleus initialization program that will be executed (IEAVNIP0).

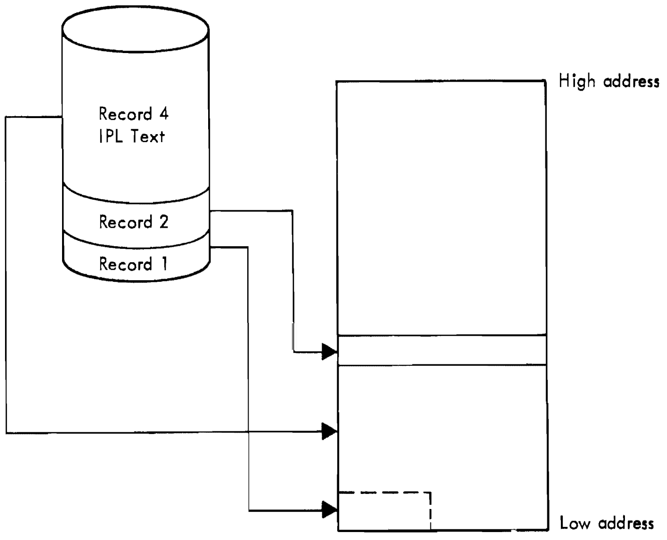


Figure 2. Location of IPL records in real storage

IEAIPL00 gains control when the operator presses the LOAD button. The operator prepares for loading IEAIPL00 by mounting the system residence volume on a direct access device and setting the load unit address switches to the unit address of that device. Pressing the LOAD button causes a system reset, turns on the LOAD light, turns off the MANUAL light, and initiates a read operation from the selected direct access device.

When the read operation is started (see Figure 2), the selected input device starts transferring data. The first IPL record is read into real storage locations 0 through 23. The doubleword read into location 8 (through 15) is a CCW (channel command word) that causes the loading of the second record into real storage at an address higher than the size of the third IPL record, so that the second record will not be overlaid. The Transfer-in-Channel command at location 16 (in the first record) specifies the address of the second record. The second record is a chain of CCWs that cause the IPL control section (third record) to be read into real storage beginning at location 0.

When the channel end is received (indicating that the last CCW in the second record has been executed), the unit address of the device is stored in bits 21-31 of the first word in storage. Bits 16-20 are set to 0, and bits 0-15 remain unchanged. The doubleword in storage location 0 is loaded as a new PSW, and normal operation proceeds. The LOAD light then turns off. IEAIPL00 does these things:

- Clears general and floating-point registers.

- Clears real storage to the highest addressable location (the highest addressable location may be specified by the operator as a location lower than the highest actual address in real storage).
- Sets all storage keys to 0.
- Relocates unexecuted coding and data areas above the real storage area to be occupied by the resident nucleus and loads the nucleus specified by the operator (IEANUC0x) or the system-assigned nucleus IEANUC01 (see Figure 3).
- Transfers control to the first NIP module, IEAVNIP0, which is loaded as part of the resident nucleus.

Processing Expected Program Checks

At certain points during execution, IEAIPL00 expects program checks to occur. These are caused when IEAIPL00 clears real storage and sets the storage keys for 2K multiples of storage. IEAIPL00 uses loop processing to perform these functions. When the instructions in the loop address a location greater than the size of real storage, a program check occurs. Anticipating this action, IEAIPL00 sets the program-check new PSW to the address of the next sequential instruction beyond the particular loop processing. When this processing is complete, the program-check new PSW is set to the address of the IEAIPL00 subroutine that processes unexpected program checks. Unexpected program checks result in a system wait state with a code of X'19' being placed in the address portion of the current PSW.

Operator Interface During IPL

IEAIPL00 provides two interfaces that allow the operator to limit the size of real storage (specify that the full size of real storage is not to be effective), and to specify a nucleus other than the system-assigned nucleus, IEANUC01. To use these interfaces, the operator sets an address stop at location X'80' before pressing the LOAD button. When this address stop is reached (by the CPU processing the third IPL record), the operator selects an alternate nucleus by storing in location X'08' the EBCDIC character to be appended to IEANUC0. This suffix is used by

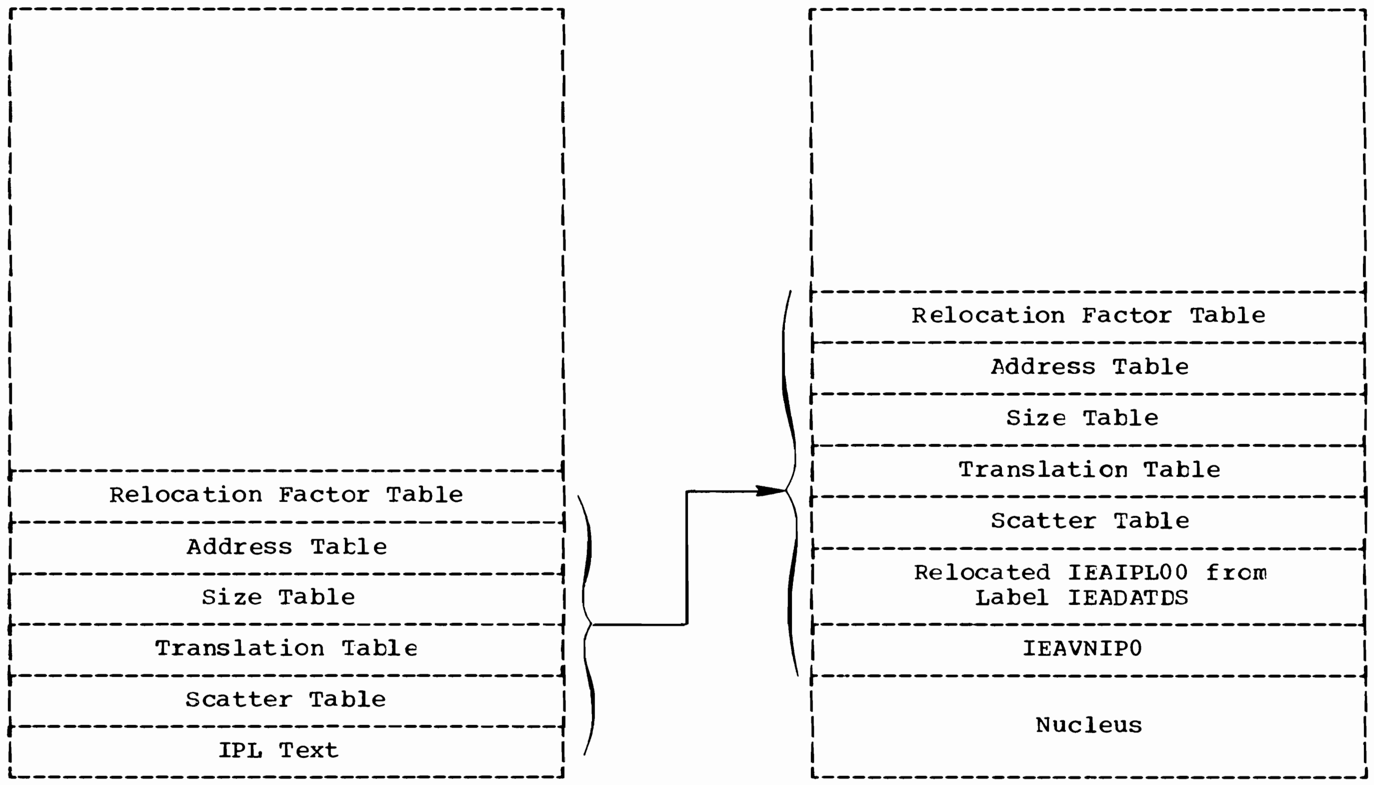


Figure 3. Contents of real storage before and after IPL relocation

IEAIPL00 to determine which nucleus is to beloaded.

The real storage size is limited by storing one of the following hexadecimal values in location X'09':

Value	Effective Size
X'A7'	192K
X'A8'	384K
X'C6'	64K
X'C7'	128K
X'C8'	256K
X'C9'	512K
X'D0'	768K
X'D1'	1024K

If the operator limits real storage, IEAIPL00 clears real storage only up to the operator-selected limit. However, storage keys for all 2K segments of real storage are set, regardless of the artificial size limit.

THE NUCLEUS INITIALIZATION PROGRAM

NIP receives control from the Initial Program Loader to initialize system tables and control blocks, to make data sets and devices available for system use, and to bring into virtual storage those modules specified to be resident. The modules that compose NIP have the following residency characteristics:

IEAVNIPO	Loaded by IEAIPL00 as part of the nucleus; overlaid by subsequent NIP execution.
IEAVNIPM	Loaded into the highest portion of NIP's dynamic area by IEAVNIPO and deleted by IEAVNIPX; remains in real storage during execution of all NIP modules except IEAVNIPO.
IEAVNP01	Loaded and deleted by IEAVNIPM; remains in NIP's dynamic area only during its execution.
IEAVNP02	Loaded and deleted by IEAVNIPM for use by NIP modules; remains in real storage only during its execution.
IEAVNP03	Loaded and deleted by IEAVNIPM; remains in NIP's dynamic area only during its execution.
IEAVNP04	Loaded and deleted by IEAVNIPM; remains in NIP's dynamic area only during its execution.
IEAVNPA4	Loaded by IEAVNIPM for IEAVNP04 and deleted by IEAVNP05; remains in NIP's dynamic area during execution of IEAVNP04 and IEAVNP05.
IEAVNP05	Loaded and deleted by IEAVNIPM; remains in NIP's dynamic area only during its execution.
IEAVNPA5	Loaded by IEAVNIPM for IEAVNP05 and deleted by IEAVNP05; remains in NIP's dynamic area during execution of IEAVNP05; is executed under a TCB created as a result of an ATTACH macro instruction issued by IEAVNP05.
IEAVNP06	Loaded and deleted by IEAVNIPM; remains in NIP's dynamic area only during its execution.
IEAVNP07	Loaded and deleted by IEAVNIPM; remains in NIP's dynamic area only during its execution.
IEAVNIPX	Loaded by IEAVNIPM; remains in NIP's dynamic area only during its execution. Passes control to the master scheduler's Initialization routine (IEEVIPL).

IEAVNIPO (DIAGRAM 2.0)

The IEAVNIPO routine in NIP is link-edited with the nucleus modules to form the nucleus member IEANUC0x (IEANUC01 if unspecified) of the SYS1.NUCLEUS data set. IEAVNIPO is entered only from the initial program loader (IEAIPL00) and performs the initialization needed to load the first NIP module into NIP's dynamic area. Execution subsequent to IEAVNIPO causes the nucleus area occupied by IEAVNIPO to be overlaid by system data, and the area becomes a logical and physical extension of the nucleus.

IEAVNIPO does the following things:

- Initializes the NVT (NIP vector table) and the CVT (communications vector table).
- Defines the real storage areas above the nucleus (see Figures 4 and 5).
- Defines the initial copy of the PVT (page vector table) and the PFT (page frame table).
- Defines the initial copy of the system trace table.

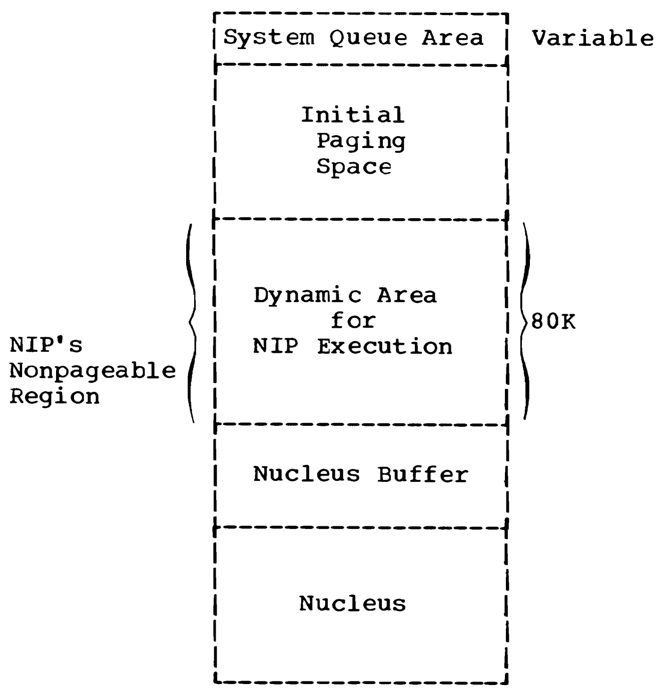


Figure 4. Definition of real storage

- Initializes the segment table and page tables (see Figure 6).
- Sets the initial values in the System/370 clocks.
- Places the CPU in extended control (EC) mode with DAT enabled.
- Defines a temporary nucleus buffer.
- Defines NIP's nonpageable region.
- Reserves initial system paging space.
- Initializes SYS1.NUCLEUS control blocks.
- Passes control to IEAVNIPM (see Figure 7).

Paging Initialization

IEAVNIP0 creates the system segment table and the page tables in the high-address portion of the system queue area of real storage. The tables are used by the dynamic address translation feature to convert virtual addresses to real addresses. To better understand NIP's execution, it is useful to summarize the function of these tables in a virtual environment.

Virtual addressing allows programs to refer to addresses that are greater than the size of real storage up to a maximum address of 16m-1. Virtual storage is divided into 64K segments, which are in turn divided into 4K blocks. Each 64K segment is described by a 16-entry page table, each entry describing a 4K block. Each 64K segment is also associated with one of the 256 entries in the system segment table which is pointed to by the CVTSEGD field. The addressing scheme (see Figure 8) multiplies the first eight bits of a 24-bit virtual address by 4 and adds the result to the address of the system segment table to arrive at the segment table entry for that virtual address. The next four bits in the virtual address are multiplied by 2, and the result is added to the 3-byte value in the segment table entry to determine the page table entry for the virtual address. The page table entry contains the first 12 bits of the real storage address of the 4K block that contains the data referred to by the virtual address. The last 12 bits of the virtual address are used as the displacement into the 4K block to arrive at the real storage address of the data.

IEAVNIP0 creates only the page tables that describe virtual addresses corresponding to real storage addresses plus a single SQA segment. The appropriate (segment table entries) are initialized for the page tables that are created. Unused segment table entries are set to indicate that no corresponding page tables have been created for them.

Because the virtual addresses of the nucleus are real addresses, the page tables that describe the virtual area corresponding to the real storage area never change and are marked as valid. The area above NIP's region and below the SQA is reserved for paging. The page tables for this area are marked as invalid, because the real storage area associated with these page tables contains no valid data, and any reference to this area must be invalid. The page table created for the virtual storage area corresponding to the real storage SQA is also indicated as invalid, because the virtual area logically associated with the SQA is in the high-address portion of virtual storage. Note that external page tables are not built, because neither the nucleus nor the SQA has a copy in virtual storage. External page tables normally follow page tables and describe page residency characteristics and location on paging devices.

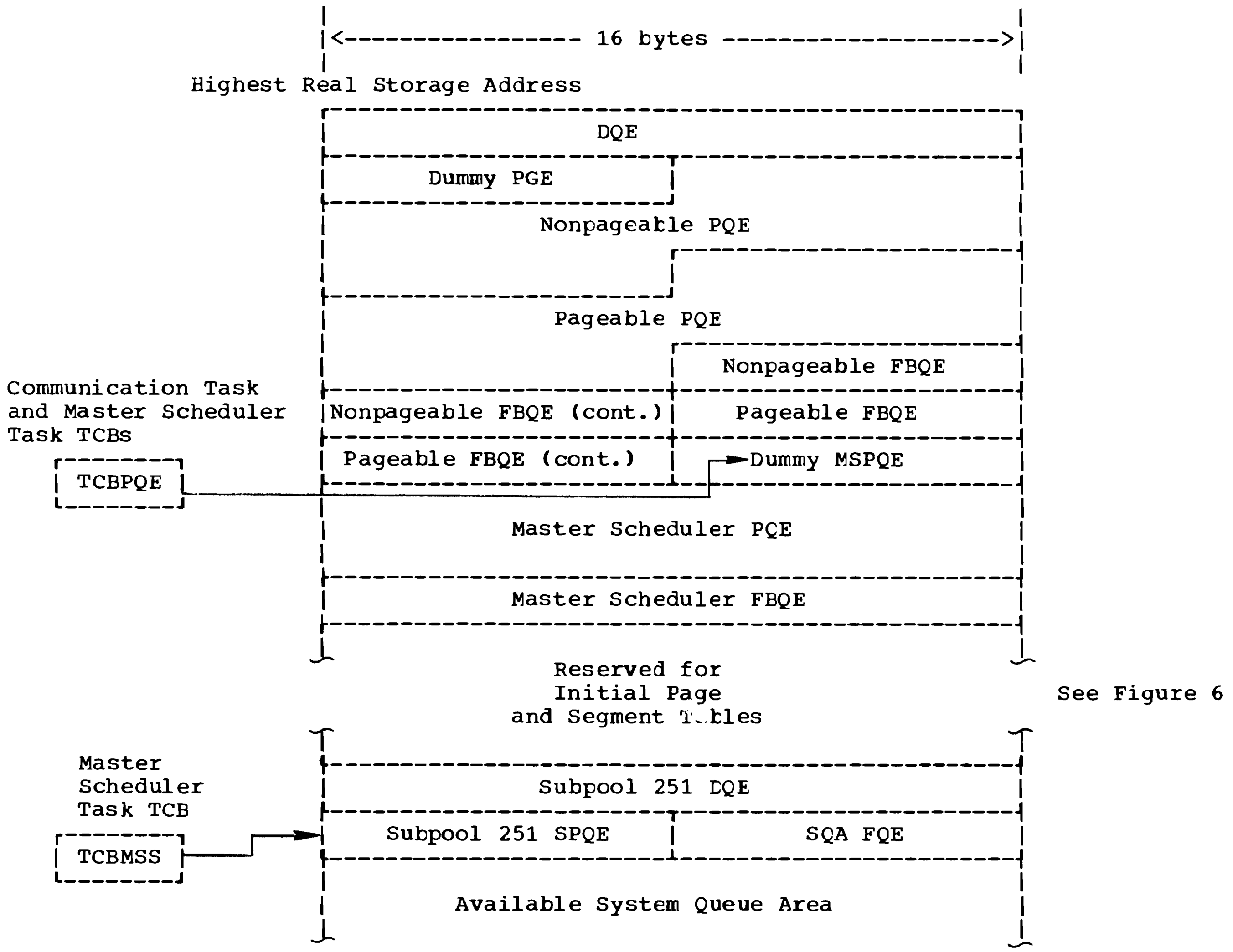


Figure 5. Organization of the initial system queue area

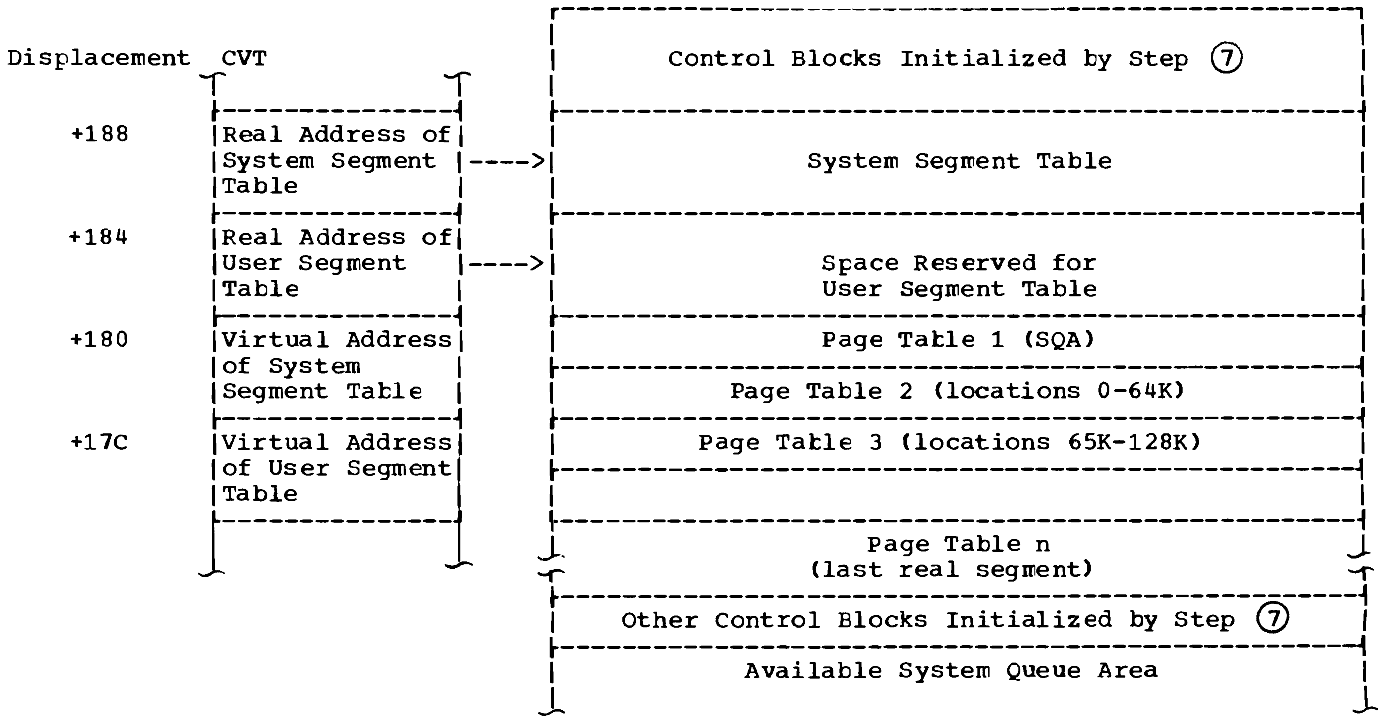
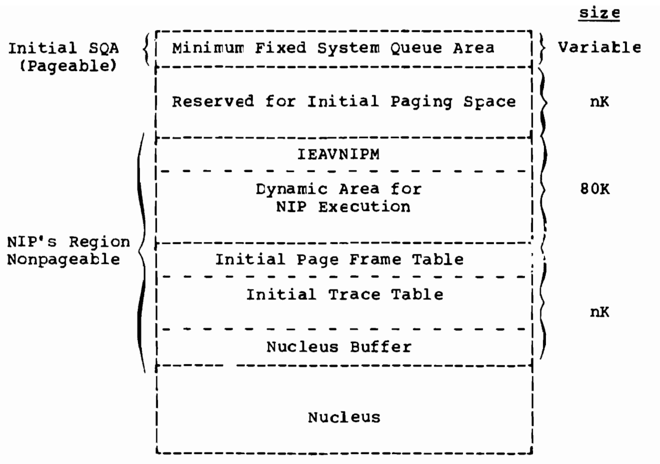

Displacement
CVT
+188
Real Address of System Segment Table
+184
Real Address of User Segment Table
+180
Virtual Address of System Segment Table
+17C
Virtual Address of User Segment Table
Control Blocks Initialized by Step ⑦
System Segment Table
Space Reserved for User Segment Table
Page Table 1 (SQA)
Page Table 2 (locations 0-64K)
Page Table 3 (locations 65K-128K)
Page Table n (last real segment)
Other Control Blocks Initialized by Step ⑦
Available System Queue Area

Figure 6. System segment and page tables



nK = (real storage size - (nucleus size + SQA size + 80K)) / 2

Figure 7. Real storage at completion of IEAVNIPO execution

EC Mode Initialization

IEAVNIPO places the CPU in EC (extended control) mode by loading the PSW described below. This also allows all interruptions from the I/O subsystem. Previously, the system was disabled for all interruptions, including machine checks.

Bit	Value	Description
0	0	Monitor mask -- disabled
1	0	Program event recording mask -- disabled
4	0	24-bit addressing specified
5	1	Dynamic address translation -- enabled
6	1	I/O mask -- enabled
7	1	External mask -- enabled
8-11	0	Supervisor protection key
12	1	EC mode
13	1	Machine-check mask -- enabled
15	0	Supervisor state
24	0	Segment protection -- disabled
40-63		Address of next sequential instruction

Those fields not described are set to 0.

The following multiple control registers are initialized by IEAVNIPO:

Word	Bits	Description
0	1	SSM interruptions
0	8-9	Page size
0	10	Page entry size
0	11-12	Segment size
0	20	Clock comparator
0	21	CPU timer
0	24	Location X'80' timer
1	0-7	Segment table length
1	8-25	Segment table origin address

IEAVNIPM (DIAGRAM 3.0)

IEAVNIPM consists of mainline coding and several subroutines. The subroutines are entered from the mainline processing, from other NIP modules, or asynchronously as a result of unusual conditions that arise during NIP execution. The mainline coding loads, branches to, and deletes the other NIP modules (except IEAVNIPX, which does not return control to IEAVNIPM).

The subroutines within IEAVNIPM are requested by other modules via the IEAPMNIP macro instruction. The expansion of this macro instruction includes a Branch and Link (BALR) instruction to the required subroutine, the address of which is obtained from the NIP vector table. If the subroutine NIPLOAD is requested, the expansion of the macro instruction also includes a Branch and Link to the requested module loaded by NIPLOAD and, for IEAVNP02 subroutine requests, a subsequent SVC 9 (DELETE) instruction. The macro expansion does not include an SVC 9 instruction for IEAVNPA4. This module is deleted after execution of the macro expansion.

NIPLOAD Subroutine (Diagram 3.1)

The NIPLOAD subroutine is entered from various modules to load a specified NIP module into real storage. It creates a BLDL list via the BLDL macro instruction and issues a LOAD macro instruction. If the module is not loaded successfully, IEAVNIPM places the system in a disabled wait state (X'32' or X'33').

NIPSVC and NIPSVCX Subroutines (Diagram 3.2)

The NIPSVCX and NIPSVC subroutines are entered as a result of an XCTL request or a request for a type-3 or type-4 SVC. All SVC table entries for type-3 and type-4 SVC routines are set to the address of NIPSVC; the entry for XCTL is set to the address of NIPSVCX. Because the link pack area (which will eventually contain the required modules) has not been initialized, the NIPSVC and NIPSVCX subroutines must load the module into real storage. Control is then passed to the XCTL module or to the SVC routine, and execution continues normally.

NIPSENSE Subroutine

The NIPSENSE subroutine is entered following an I/O error to write an interpretive error message to the master console. Information concerning the command code of the failing CCW, CSW status bytes, volume

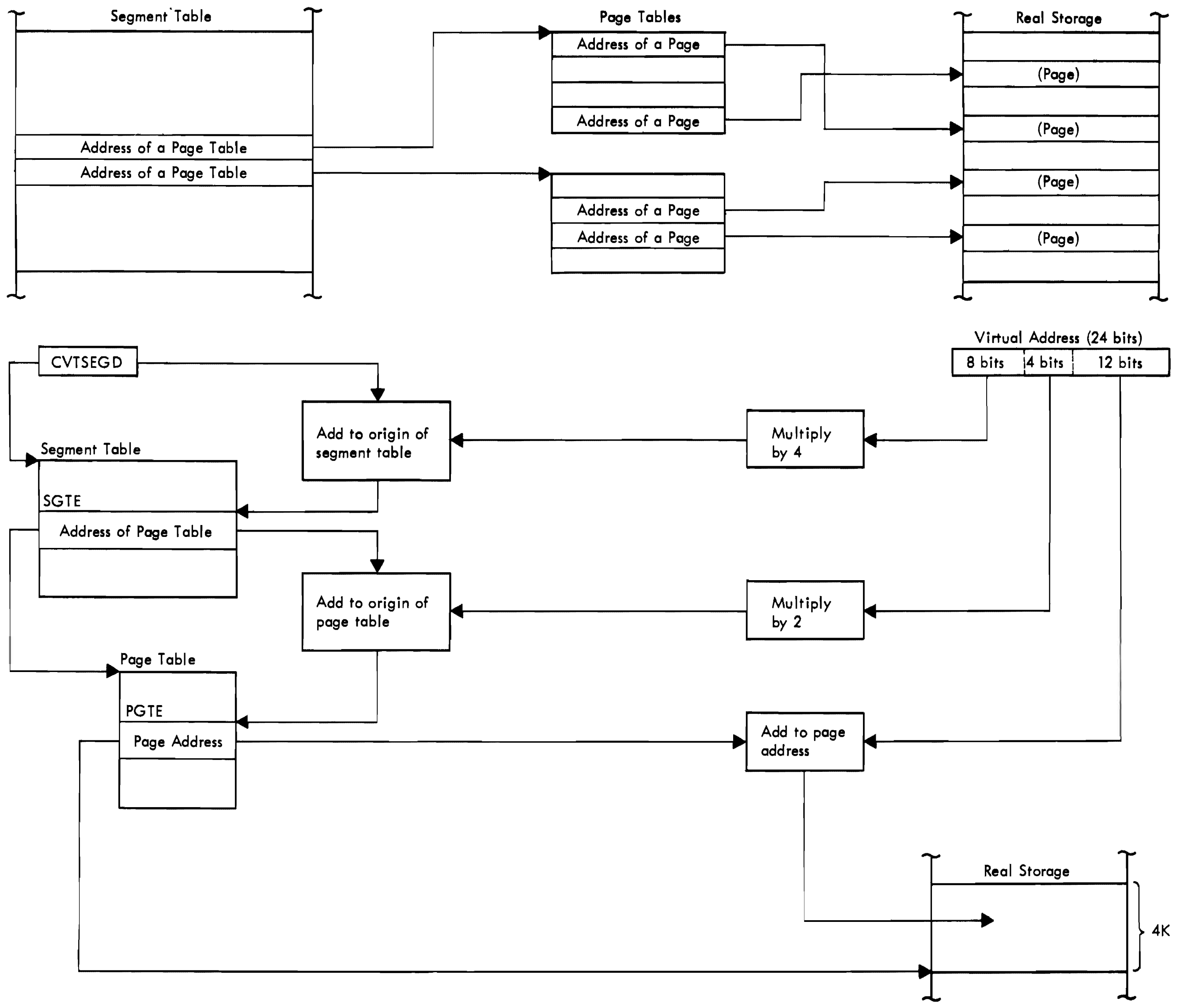


Figure 8. Dynamic address translation

serial number (if direct access storage device), EBCDIC unit address, and (if a unit check) sense bytes are extracted and placed in the error message, which is then sent to the operator via the NIPWTO subroutine (within IEAVNIPM).

NIPUCBFN Subroutine (Diagram 3.3)

The NIPUCBFN subroutine is entered to find a UCB related to a specific device. After converting the input, if necessary, to the hexadecimal unit address, NIPUCBFN issues an IOSGEN macro instruction to link to the IOS UCB Lookup routine. IOS performs the necessary processing and returns the address of the associated UCB, if it was found.

NIPTIME Subroutine (Diagram 3.4)

The NIPTIME subroutine is entered to provide either the time of day in decimal or the binary value representing the time elapsed since IEAVNIPM was entered for this initialization process.

NIPWTO and NIPWTOR Subroutines

The NIPWTO and NIPWTOR Subroutines are entered to build and execute a channel program to simulate WTO and WTOR macro instructions. IEAVNIPM has a temporary reply buffer into which the operator's reply to a WTOR is read. The requesting routine may specify that control is not to be returned until the reply is received. In this case, NIPWTOR passes control to NIPWTOR2 to await the reply.

NIPWTOR2 Subroutine

The NIPWTOR2 subroutine is entered to move an operator's reply into the reply buffer specified by the calling routine. NIPWTOR2 returns control to the requesting routine or, if entered from NIPWTOR, returns control to the routine that requested the WTOR function.

NIPABEND Subroutine

The NIPABEND subroutine is entered asynchronously to process conditions that would normally cause abnormal termination of a task. Upon initial entry from IEAVNIP0, IEAVNIPM replaces the SVC 13 address placed in the SVC table during system generation with the address of the NIPABEND subroutine. For the first entry to process an abnormal condition, NIPABEND issues a message to the operator via NIPWTO indicating the type of error and branches to NIPSWAIT to place the system in a disabled wait state. For a recursive entry, the previous ABEND code is saved and the system is placed in a disabled wait state (X'40'). A recursive entry might be caused by a failure in NIPWTO.

NIPSQEND Subroutine

The NIPSQEND subroutine is entered from GETMAIN when an additional real page for the SQA is requested before the paging subsystem is initialized. The system is placed in a disabled wait state (X'36').

NIPSWAIT Subroutine

The NIPSWAIT subroutine is entered whenever an error condition prevents successful initialization of the system. NIPSWAIT uses NIPWTO to notify the operator of the system wait state code and places the system in a disabled wait state.

IEAVNP01 (DIAGRAM 4.0)

IEAVNP01 receives control from IEAVNIPM to initialize system consoles and to allow the operator to specify system parameters required for NIP execution.

Initializing System Consoles

IEAVNP01 uses several loops to determine the availability of and to initialize the system consoles. The first console examined is always the master console specified in the UCM prefix during system generation. The NP1TESTC subroutine is entered to build a CCW chain for the type of console being tested. An EXCP instruction is issued, and the result of the I/O indicates whether the console is available. If the console is available, it is initialized as described below. After the console is initialized, or if the console is not available, the other consoles in the master alternate queue are examined and initialized in a similar manner.

When the end of the master alternate queue is reached, IEAVNP01 searches, tests, and initializes all UCM entries beginning with that pointed to by UCMVEA. IEAVNP01 then determines whether a master console has been found and initialized. If not, the system is placed in a disabled wait state (X'07').

The NP1INIT subroutine (see Diagram 4.1) initializes available consoles by (1) indicating that the console is active and (2), if the active master console is not the one specified during system generation, by copying the authorization and routing codes from the old master console to the new active master console and adjusting the UCM prefix/base.

Initiating Operator Communication

Immediately after the active master console is initialized (and before the alternate consoles are initialized), the NP1INIT subroutine establishes communications with the operator by issuing message IEA101A SPECIFY SYSTEM PARAMETERS via the NIPWTOR subroutine in IEAVNIPM. The receipt of the reply is determined by the NP1TCOMM subroutine, which is entered from the NP1TESTC subroutine before consoles are tested for availability. If a reply has been received, it is moved by NIPWTOR2 into the reply buffer area, a 2K area previously obtained from subpool 255 of the SQA. If additional parameters are to be entered, the message IEA116A CONTINUE SYSTEM PARAMETERS is issued, and the reply to this is processed as above.

Before returning control to IEAVNIPM, IEAVNP01 ensures that no reply is outstanding to a message issued from this module.

IEAVNP02 (DIAGRAM 5.0)

This module is initially entered from IEAVNIPM to initialize the devices in the system configuration that have not been initialized by IEAVNP01. It also provides system access to the SYS1.LOGREC, SYS1.SVCLIB, and SYS1.LINKLIB data sets. The subroutine NIPOPEN (entry point IEAVNPB2) is used by mainline coding and by other NIP modules to open specified data sets. The subroutine NIPMOUNT (entry point IEAVNPA2) is used only by other NIP modules to request the operator to mount specified volumes on specified devices.

Mainline Coding

IEAVNP02 initializes the unit test DEB and its associated control blocks, which are defined within IEAVNP02. These are used for all I/O testing of devices. Using the UCB address table, each device represented by a UCB that was not examined by IEAVNP01 is tested for availability. For a DASD (direct access storage device), a channel program to read the volume label is created by the subroutine NP2RRCD3 and executed by the subroutine NP2EXCPU. For non-DASD, NP2EXCPU executes an I/O NOP CCW. Devices that are not ready are so marked in their UCBs.

After testing the system device configuration, SYS1.LOGREC and SYS1.SVCLIB are opened using the subroutine NIPOPEN. IEAVNP02 builds the DEBs needed for the open processing. IEAVNP02 also constructs the DEB associated with the SYS1.LINKLIB data set.

NIPMOUNT Subroutine (Diagram 5.1)

This subroutine is entered to request the operator to mount a specified volume on a device. The device may be specified by type or by unit. NIPMOUNT ensures that the correct volume is mounted on the device. Messages notify the console operator of the required action and indicate, where appropriate, incorrect and remedial action. A flowchart of this subroutine is included in Section 3.

NIPOPEN Subroutine (Diagram 5.2)

This subroutine optionally builds the basic DEB and adds DEB extents for the specified data set to be opened.

IEAVNP03 (DIAGRAM 6.0)

This module is entered by IEAVNIPM to initialize the system parameter table according to the specifications given in response to the message SPECIFY SYSTEM PARAMETERS. Two sets of specifications are possible; operator entered, and system programmer entered (via SYS1.PARMLIB). Figure 9 describes system parameter area creation. The IEAVNP03 subroutines:

- Analyze the parameters specified via the SYSP parameter and merge them with the parameters specified by the operator in response to SPECIFY SYSTEM PARAMETERS.

- Open SYS1.LINKLIB and concatenate members specified in the LNKLST00 member of SYS1.PARMLIB with the SYS1.LINKLIB data set.

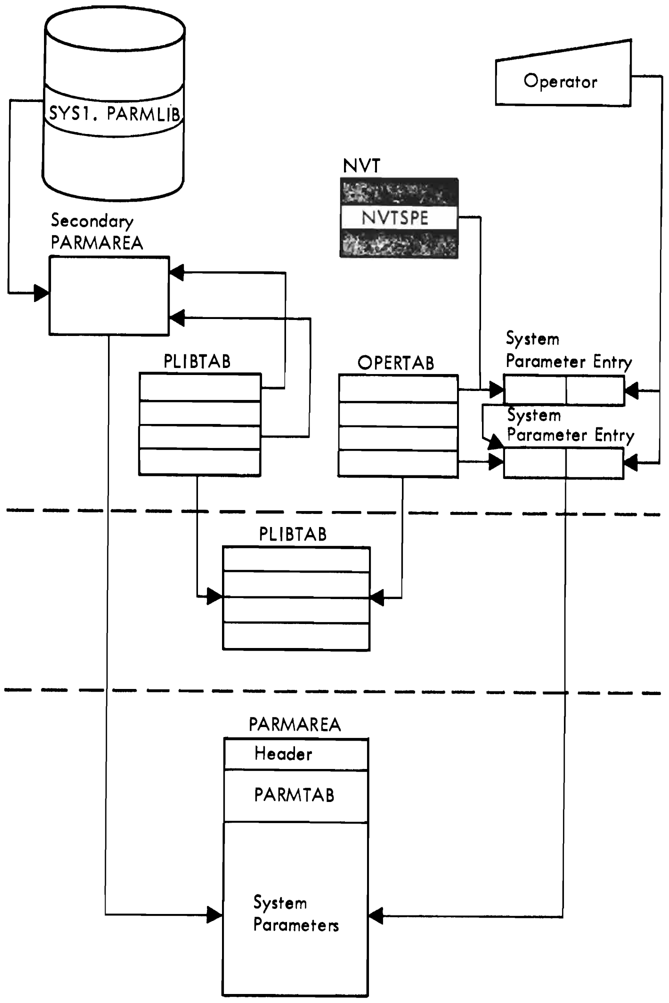


Figure 9. Creating system parameter area

NP3OPSP Subroutine (Diagram 6.1)

This subroutine uses the NP3SCAN subroutine to scan the parameters entered by the operator in response to SPECIFY SYSTEM PARAMETERS. The subroutine moves the addresses of the valid parameters into an area within IEAVNP03 called OPERTAB. If a parameter is invalid, the operator is allowed to respecify or cancel it. Figure 10 illustrates building the OPERTAB.

NP3PBASE Subroutine (Diagram 6.2)

This subroutine allocates a 2K area (PARMAREA) within NIP's dynamic area for later use by IEAVNP03. This area eventually contains the valid system parameters. The lower-address part of PARMAREA is initialized to contain pointers to the next PARMAREA (considered secondary and temporary), to contain the next available area within PARMAREA, and to contain the IOB, DCB, and DEB for the SYS1.PARMLIB. The remainder of the PARMAREA is initialized by the NP3PTAB subroutine described below.

NP3PMLIB Subroutine (Diagram 6.3)

This subroutine finds and opens the SYS1.PARMLIB data set.

NP3SYSP Subroutine (Diagram 6.4)

This subroutine allocates a secondary 2K buffer (chained to the primary buffer, PARMAREA) and reads into it the parameters defined by the SYSP parameter. These parameters are located in the IEASYSxx members of the SYS1.PARMLIB. The first member processed is always IEASYS00, which contains the values used for defaulted parameters.

The NP3SCAN subroutine scans the parameters in the secondary buffer and moves the addresses of the valid parameters into the area within IEAVNP03 called PLIBTAB. When a parameter is specified more than once, the last specification is effective. The entries in both OPERTAB and PLIBTAB contain one address for each parameter (except APG, PAL, and PAGE). Each entry reflects the last valid specification processed for that parameter. Figure 11 illustrates building the PLIBTAB.

NP3PLMRG Subroutine

This subroutine merges the parameters pointed to by the addresses in the OPERTAB and the PLIBTAB. When both tables contain entries for the same parameter, the OPERTAB entry replaces the PLIBTAB entry unless OPI=NO is indicated in the PLIBTAB for that operand. PAGE, PAL, and APG parameters are not merged. The addresses of the valid current system parameters are placed in PLIBTAB, overlaying the address of overridden parameters where necessary. Figure 12 illustrates the merging of specifications into PLIBTAB.

NP3PTAB Subroutine (Diagram 6.5)

This subroutine first removes all secondary PARMAREAs from the queue originating with the initial PARMAREA so that additional space required is pointed to by the initial PARMAREA, instead of the last-acquired secondary PARMAREA. The area within the PARMAREA above the header (initialized by NP3PBASE above) has two parts: (1) the PARMTAB which will contain the addresses of the parameters, and (2) the area into which the parameters will be moved.

NP3PTAB begins moving the specifications for each parameter into the high-address portion of the PARMAREA while updating the corresponding addresses of the parameters in PARMTAB. In moving the parameters into the PARMAREA, NP3PTAB strips out superfluous information, such as the keyword operands. Figure 13 illustrates building the final PARMAREA.

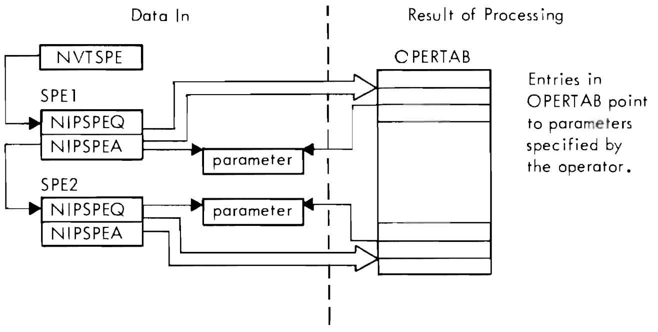


Figure 10. Building OPERTAB

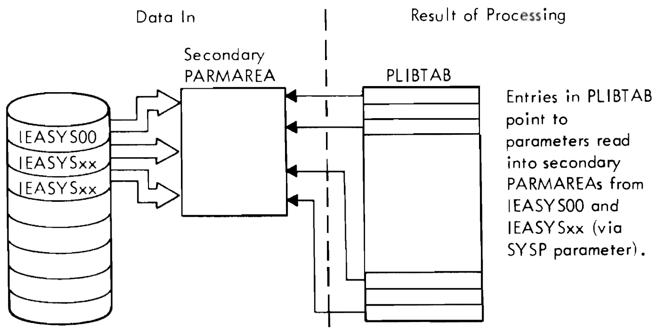

Data In
Result of Processing
Secondary PARMAREA
PLIBTAB
IEASYS00
IEASYSxx
IEASYSxx
Entries in PLIBTAB point to parameters read into secondary PARMAREAs from IEASYS00 and IEASYSxx (via SYSP parameter).

Figure 11. Building PLIBTAB

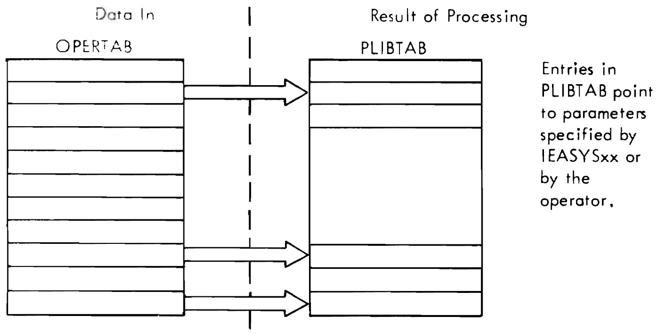

Data In
Result of Processing
OPERTAB
PLIBTAB
Entries in PLIBTAB point to parameters specified by IEASYSxx or by the operator.

Figure 12. Merging OPERTAB into PLIBTAB

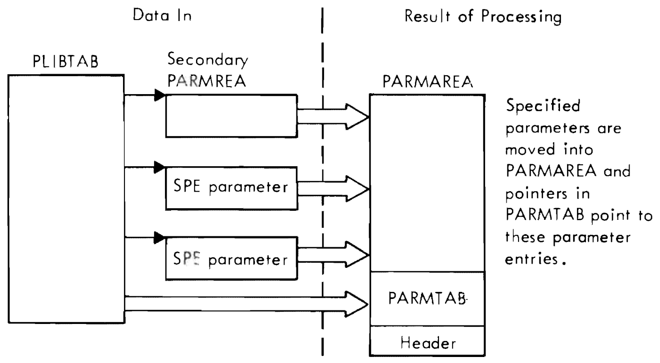

Data In
Result of Processing
PLIBTAB
Secondary PARMREA
PARMAREA
SPE parameter
SPE parameter
PARMTAB
Header
Specified parameters are moved into PARMAREA and pointers in PARMTAB point to these parameter entries.

Figure 13. Building the PARMAREA

NP3LKLIB Subroutine

This subroutine opens the SYS1.LINKLIB data set for later system use.

NP3LCAT Subroutine (Diagram 6.6)

This subroutine concatenates the data set members whose names are specified in the LNKLST00 member of SYS1.PARMLIB data set with the SYS1.LINKLIB data set. A LOCATE macro instruction is issued for each member name found, and the resulting volume identification is passed to the NIPMOUNT subroutine in IEAVNP02. The NIPOPEN subroutine in IEAVNP02 opens the data set and concatenates it with the previously defined portion of the SYS1.LINKLIB. The number of concatenations is limited to 15.

IEAVNP04 (DIAGRAM 7.0)

This module is entered only from IEAVNIPM to process the PAGE, CLPA, CPQE, and SQA parameters and to initialize the control blocks used by the paging supervision routines. IEAVNP04 performs the following functions:

- Interprets the PAGE parameters entered either in SYS1.PARMLIB or by the operator in response to the message SPECIFY SYSTEM PARAMETERS.
- Ensures that the paging devices are online and available for system use.
- Ensures that the SYS1.PAGE data sets are available.
- Expands the definition of the SQA.
- Creates the page device table, page device information table, page frame table, quickstart records, and the channel program queue.

Processing PAGE Parameters

IEAVNP04 uses the system parameter table (PARMTAB) to locate the PAGE parameters included during system generation. A 16-entry matrix is created as a record of the parameters. IEAVNP04 then processes the PAGE parameters specified by the operator in response to SPECIFY SYSTEM PARAMETERS. Any value specified by the operator will override any corresponding specification already in the matrix. After optionally displaying the PAGE parameter values specified, or if no PAGE parameters have been specified, the operator is allowed to change or add any PAGE specifications.

Initializing for Quickstart Records

Because formatting of page data set records requires I/O overhead (a history of a successfully formatted page data set is kept within the data set), IEAVNP04 allocates an 8K buffer in the high-address portion of NIP's region. This buffer is used to build, write, and read records (called quickstart records) that describe the content and extent of each page data set. A quickstart record is built when a page data set is initially formatted. IEAVNP04 attempts to read a quickstart record for each previously allocated page data set. If the read is successful, the data set is not reformatted. If the read is not successful, the data set requires reformatting, and a new quickstart record is created.

NP4PDSEL Subroutine (Diagram 7.1)

This subroutine scans the 16-entry matrix of PAGE parameters to determine specifications for the paging devices. Each specified device is tested for availability, and the operator is allowed to respecify any device that is unacceptable or to cancel the definition.

For each online and ready device, the NP4IPDT subroutine is entered to create the PDIT (page device information table) and the PDTE (page device table entry) for the device. Both entries are created in the upper end of the nucleus buffer.

Upon completion of processing for this device, the PDIT is moved to its permanent position in the low area of the nucleus buffer. However, the PDTE is not moved until all matrix entries have been processed, because the final size of the PDIT is not known until that time. The NIPOPEN subroutine in IEAVNP02 is entered to open the data set represented by the entry in the matrix. The NP4RQSR1 subroutine is entered to read the first quickstart record for the data set. If this data set represents the quickstart LPA data set, and the CLPA parameter is not specified in PARMTAB, IEAVNP04 sets the NVTFLQS field to indicate that the page data set can be defined by the quickstart process. If this read is not successful, page formatting is required.

If formatting is not required, the PDIT is moved from the temporary area to its permanent position in the low area of the nucleus buffer, and additional device-dependent information is added. If formatting is required, the NPA4INTF subroutine in IEAVNPA4 is entered to start formatting, and the ECB for the request (in the PDIT) is added to a list of ECBs that are tested prior to moving the PDTEs.

When all matrix entries have been processed, the list of ECBs is checked to determine whether all formatting has been completed. As formatting for each data set is completed, a quickstart record (NIPQSR1) is built and written for the data set, and its PDIT is moved to its permanent position. When all PDITs have been moved, the PDTEs in the temporary position in the upper end of the nucleus buffer are scanned, and each valid PDTE is moved to its permanent position in the low end of the nucleus buffer above the PDITs.

Figure 14 shows real storage after paging device initialization and initial paging area definition.

NP4BCPQ Subroutine

This subroutine defines the paging supervision channel program queue at the high end of the nucleus. The size of the CPQ (channel program queue) depends on the number of paging devices in the system configuration. Ten CPQ elements are defined for the first paging device; 15 CPQEs are defined if there is more than one device. In addition to the base number of CPQEs defined, the operator may specify an additional number via the CPQE parameter. This number is added to the base number.

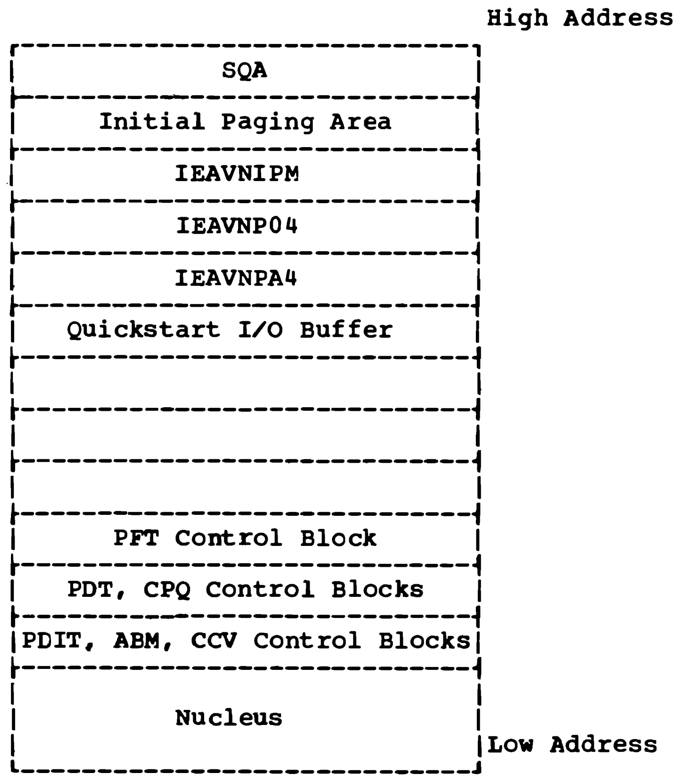


Figure 14. Real storage after paging device initialization and initial paging area definition

NP4APFT and NP4BPFT Subroutines

Space for the page frame table is allocated by NP4APFT in the low end of the nucleus buffer above the area occupied by the PDT. The page frame table is built by NP4BPFT.

NP4RQSR2 Subroutine

The LPA quickstart process is initiated if the NVTFLQS flag has been set by the NP4RQSR1 subroutine. The first and second LPA quickstart records are read into the quickstart buffer by NPA4READ in IEAVNPA4. The second quickstart record is required by NP4BSQA.

NP4BSQA Subroutine

This subroutine processes the operator-specified request to increase the size of the SQA. The value specified by the SQA parameter is added to the basic size of the SQA to determine the new SQA size. If the operator specification would cause the SQA to extend into the quickstart LPA virtual storage, thus causing a coldstart of the LPA, the operator is requested to either approve the coldstart process or to cancel the current SQA definition.

NP4IPVT Subroutine

This subroutine initializes the page vector table to make the paging supervision control blocks available for system use. After using the MODESET instruction to disable interruptions, CVTPAGE1 is set to the address of the DCB in the first PDIT (page device information table) created by NP4PDSEI. The PVTPDT, PVTNPDTE, and PVTCHPGQ fields are initialized. The PVTPSQA field is restored to its original (upon entry to IEAVNIPM) contents.

IEAVNPA4 (DIAGRAM 8.0)

IEAVNPA4 is brought into real storage by IEAVNIPM and entered from the Page Initialization routine in IEAVNP04. It is subsequently entered from IEAVNP04, IEAVNP05, and IEAVNPA5. IEAVNPA4 is deleted by IEAVNP05 upon completion of its execution. IEAVNPA4 contains subroutines that:

- Read and write quickstart records from and to the SYS1.PAGE data sets (NPA4READ, NPA4WRIT).
- Allocate space in NIP's dynamic area for the quickstart I/O buffer (NPA4GBUF).
- Initiate formatting of SYS1.PAGE data sets (NPA4INTF).

- Restart formatting of SYS1.PAGE data sets where a previous I/O error has occurred (NPA4RSTF).
- Initialize the paging tables for initiating the coldstart procedure for the link pack area (NPA4INTC).
- Determine whether a specified load module will fit in the link pack area SYS1.PAGE data set (NPA4LOAD).
- Complete the coldstart process (NPA4CCST).
- Build cylinder count vectors for the SYS1.PAGE data sets and calculate the available pages (NPA4BCCV).
- Release the space allocated for the quickstart I/O buffer (NPA4FREE).

Each of these functions is performed by a subroutine within IEAVNPA4. The fourth and fifth bytes of the parameter list passed to IEAVNPA4 contain bit settings indicating which function is requested.

NPA4READ/NPA4WRIT Subroutines (Diagram 8.1)

These subroutines read and write the quickstart records to and from the SYS1. PAGE data set. Because each quickstart record points to the next higher record (that is, record 1 points to record 2 which, in turn, points to record 3), a request for quickstart record 2 or quickstart record 3 requires that the record pointing to it (quickstart record 1 and quickstart record 2, respectively) already be in the appropriate quickstart record buffer.

NPA4READ and NPA4WRIT rely on common coding to perform the requested I/O. They use a predefined IOB for scheduling I/O requests, but rely on the DCB and its associated DEB, which are defined via the input DCB parameter.

If an I/O error occurs during the execution of the channel program, the NIPSENSE subroutine in IEAVNIPM is entered to write an interpretive error message.

NPA4GBUF Subroutine (Diagram 8.2)

This subroutine obtains an 8K area from subpool 252 of NIP's nonpageable region. Subsequent NIP execution requires that this area remain fixed in real storage. This area is used by NPA4READ and NPA4WRIT as buffers into which quickstart records are read. Quickstart records 1 and 3 are read into the lower end of the area (buffer 1); quickstart record 2 is read into the upper end (buffer 2).

NPA4FREE Subroutine (Diagram 8.3)

This subroutine releases the quickstart record buffer area obtained by NPA4GBUF and resets the pointers to the buffer area and to IEAVNPA4 (in the NVT) to 0.

NPA4INTF/NPA4RSTF Subroutines (Diagram 8.4)

These subroutines initiate formatting, or restart formatting where previous formatting was terminated because of I/O error, of a SYS1.PAGE data set.

There is one channel program built for each paging device supported by VS2. The appropriate build table is determined by using the device type indicator (PDTDT) in the page device table as a displacement to the addresses of the build tables within IEAVNPA4. The build table contains several entries, each of which is used to build a CCW for the channel program executed to format the page data set.

The first two bytes in each build table indicate the number of CCWs required to format one cylinder of the device and indicate the number of count fields required by the channel program. Each of the two-byte entries that follow this header information has the following format:

Bit	Description
0	1 = This CCW is for the last record on the track.
1	1 = This is the last entry in the build table.
2-15	Length of the record to be formatted by this CCW.

There is one entry in the channel program build table for each data set record in one group for the device.

NPA4BCCV Subroutine (Diagram 8.5)

This subroutine scans the auxiliary bit map field in a paging device table entry and totals the number of available pages.

NPA4INTC Subroutine (Diagram 8.6)

This subroutine is entered from IEAVNPA5 to initiate the coldstart process for the link pack area. Pertinent areas in the page vector table that may be modified are saved within this module (it remains resident during NIP execution).

NPA4LOAD Subroutine (Diagram 8.7)

This subroutine determines the number of pages of virtual storage required for the specified module and whether there are enough pages available in the LPA page data set to contain the module.

NPA4CCST Subroutine (Diagram 8.8)

This subroutine reinitializes page vector table fields to their original contents upon entry to the NPA4INTC subroutine. All pages currently in use by LPA modules are forced out of real storage, and the quickstart records are updated.

IEAVNP05 (DIAGRAM 9.0)

This module is also entered from IEAVNIPM. It defines the pageable link pack area either completely via the coldstart process, or uses a previously initialized LPA via the quickstart process. Quickstart records are written to the LPA SYS1.PAGE data set during a coldstart. Pageable LPA modifications, fixed LPA modifications, and the system BLDL table are optionally processed. IEAVNP05 is also responsible for freeing the quickstart buffers obtained by IEAVNPA4 and for deleting IEAVNPA4.

NP5QSLPA Subroutine (Diagram 9.1)

This subroutine uses data in the quickstart records to define the LPA page tables and external page tables. If this subroutine cannot be executed successfully, the coldstart process must be entered.

NP5CSLPA Subroutine (Diagram 9.2)

This subroutine attaches the module IEAVNPA5, which contains subroutines to perform requested coldstart functions. The interaction between IEAVNP05 and IEAVNPA5 is maintained by waiting and posting indications of segment processing completion (ECBs). NP5CSIPA uses internal subroutines and subroutines within IEAVNP05 to open the SYS1.LPALIB, initiate the coldstart process, and process modifications to the fixed link pack area and the system BLDL table.

NP5VTCB Subroutine (Diagram 9.3)

This subroutine is entered to establish and to terminate the relationship between IEAVNP05 and IEAVNPA5. The subroutine NIPLOAD in IEAVNIPM is used to bring IEAVNPA5 into NIP's nonpageable region. After the module is loaded, an ATTACH macro instruction is issued to create IEAVNPA5 as a jobsteptask. Subsequent entries to IEAVNPA5 are made via the NP5POST subroutine, which posts the ECB specified by IEAVNPA5 and issues a WAIT macro instruction specifying an ECB for IEAVNP05. Control returns when the ECB for IEAVNP05 is posted by IEAVNPA5.

When entered to terminate the relationship, NP5POST reenters IEAVNPA5, which releases the previously allocated BLDL save area. Upon return, NP5POST issues an SVC 9 instruction to delete IEAVNPA5, releases storage and control blocks allocated to IEAVNPA5, and causes the attached TCB to be removed from the task queue and the subtask queue for the job step.

NP5LPLIB Subroutine (Diagram 9.4)

This subroutine is entered to open the SYS1.LPALIB data set so that modules can be loaded from it into the link pack area. NP5LPLIB issues a LOCATE macro instruction to determine whether SYS1.LPALIB is defined in the system catalog. If the correct entry is found, the NIPMOUNT subroutine in IEAVNP02 is used to ensure that the volume is mounted and online. Upon return, the NIPOPEN subroutine in IEAVNP02 is used to open the data set. This request is conditional so that any error encountered during the open processing will not cause a system wait. Failure to open SYS1.LPALIB is indicated to the operator via a warning message, and to the calling routine via a return code of 4. Successful opening of the SYS1.LPALIB is indicated by setting NVTCSLIB to the address of the LPALIB DCB.

NP5MLPA Subroutine (Diagram 9.5)

This subroutine is invoked unconditionally for both quickstart and coldstart processing to load LPA modules into the pageable LPA whenever the MLPA parameter in the PARMTAB has a nonzero entry. The MLPA parameter gives the names of the SYS1. PARMLIB members (IEALPAxx) that contain the names of the modules to be added. The NP5VTCB subroutine is entered to attach the pageable task IEAVNPA5, which performs processing common to both quickstart and coldstart. The subroutine NP5MLPRM is used to load the specified modules one at a time. The CDEs for the loaded modules are removed.from the pageable TCB and placed on the active LPA queue by the NP5QLPAQ subroutine.

NP5BLDLP Subroutine (Diagram 9.6)

This subroutine is invoked unconditionally whenever the BLDL parameter in the PARMTAB has a nonzero value. The BLDL parameter gives the name of the SYS1. PARMLIB member that contains the names of the modules to be included in the BLDL table. The valid module names in the parameter string are used to create a BLDL table in the nucleus buffer. NP5BLDLP issues a BLDL macro instruction when the table has been completed. The pageable task (IEAVNPA5) is activated, if not already active, via the subroutine NP5VTCB and entered to move the BLDL table to the area below the pageable LPA. NP5BLDLP places the address of the pageable BLDL table in the nucleus-resident BLDL SVC routine (IGC018).

NP5BLMQ Subroutine

This subroutine builds a load module queue for IEAVNPA5 based on the LPA packing list member of SYS1.PARMLIB.

IEAVNPA5 (DIAGRAM 10.0)

This module is executed as a subtask of IEAVNP05 and performs its functions as a pageable task. Interaction between IEAVNP05 and IEAVNPA5 is governed by the use of WAIT and POST macro instructions specifying ECBs related to the modules. That is, when IEAVNP05 requests a function of IEAVNPA5, the pageable ECB for IEAVNPA5 is posted, and IEAVNP05 issues a WAIT macro instruction specifying the nonpageable ECB. When the requested function has been completed, IEAVNPA5 posts the nonpageable ECB for IEAVNP05 and issues a WAIT macro instruction specifying the pageable ECB. IEAVNPA5 is activated (attached) only if initialization is required for coldstart processing, if the MLPA option was selected, or if the BLDL option was selected.

Upon initial entry, a conditional GETMAIN macro instruction specifying 16 million bytes is issued to allocate all of virtual storage in the pageable region to IEAVNPA5. This allows IEAVNPA5 to control allocation within the pageable region by issuing subsequent FREEMAIN macro instructions for specific storage locations. A second GETMAIN macro instruction allocates 64 bytes from subpool 255 of the SQA for BLDL entries initialized by IEAVNPA5. A skeleton BLDL header, which defines the single BLDL entry as 60 bytes, is moved into this area. After initial processing, control is returned to IEAVNP05 by posting the nonpageable ECB.

NPA5MLPA Subroutine (Diagram 10.1)

This subroutine loads specified modules into the pageable LPA. The BLDL entry pointed to by NVTVRBLD is moved to the BLDL entry in the SQA, which was allocated by the NPA5INIT subroutine within IEAVNPA5. The NPA5ADDR subroutine is used to determine the address at which the specified module will be loaded. The nonpageable ECB is posted so that IEAVNP05 can continue execution. Upon return of control from IEAVNP05, NPA5MLPA uses NPA5LOAD to load the module into the LPA.

NPA5BLDL Subroutine (Diagram 10.2)

This subroutine allocates storage in the pageable area for the system BLDL table and moves the BLDL table from its temporary location in nonpageable storage. The NPA5ADDR subroutine determines the address to which the BLDL table should be moved and moves the table to the newly allocated storage. The fields CVTSHRVM and NVTBLDL are reset to the address of the BLDL table.

NPA5CLPA Subroutine (Diagram 10.3)

This subroutine is entered to perform a coldstart process for the LPA. The NPA4INTC subroutine in IEAVNPA4 is invoked toititialize PVT fields that will be used during the coldstart process.

NPA5CLPA calls the IEAVNPA5 subroutines; NPA5CLIN, NPA5IGRP, NPA5LIND, NPA5ALIS, and NPA5BDIR to build the LPA. NPA5CLPA then calls the IEAVNPA4 subroutine NPA4CCST to complete the coldstart process.

NPA5TERM Subroutine (Diagram 10.4)

This subroutine releases the real storage allocated during the initial execution of IEAVNPA5. The pageable region is not released, since this would release the pages assigned to the pageable LPA and/or BLDL table. The pageable region space is released by IEAVNP05 when the subtask (IEAVNPA5) is removed from the system.

NPA5CLIN Subroutine

This subroutine constructs the BLDL information table and determines the hash value. The BLDL information table (INFODATA) contains flag bits and information extracted from the LPALIB directory. It is used to reconstruct a BLDL entry when loading a module.

NPA5LIND Subroutine

This subroutine loads independent modules (those not loaded by NPA5LGRP). When it can, which is most cases, NPA5LIND tries to load the independent modules into pages which are not completely full. NPA5LIND keeps track of these spaces with a "wasted space" table and a two-page area allocated above the BLDL information table. The wasted space table has the size of the space and a pointer to a 3-word entry in the two-page area. The 3-word entry contains: the size of the space, the address of the space in the LPA, and a pointer to the next largest space entry.

NPA5ALIS Subroutine

This subroutine constructs LPDEs for all module aliases.

NPA5BDIR Subroutine

This subroutine builds a permanent link pack area directory.

NPA5LGRP Subroutine (Diagram 10.5)

This subroutine loads a group of modules specified by the LPA packing list.

IEAVNP06 (DIAGRAM 11.0)

IEAVNP06 contains the initialization routines for the RAS (reliability and serviceability) features of VS2 and processes the DUMP and TRACE system parameters.

IEAVNP06 contains a control routine that branches sequentially to four separate processing routines. The control routine contains sequential calls to the four processor routines. Each of the four processors initializes a separate RAS feature.

The four RAS functions initialized by IEAVNP06 are:

1. DSS (Dynamic Support System) (for planning purposes only)
2. SVCDUMP (SYS1.DUMP data set is opened)
3. RMS (Recovery Management Support), consisting of CCH (Channel-Check Handler) and MCH (Machine-Check Handler)
4. TRACE (system trace table is defined and initialized)

NP6DSS Subroutine (Diagram 11.1)

NP6DSS locates and prepares the DSS task for initialization. It locates and opens the SYS1.DSSVM data set and uses the routine IEAQCDSR and IEAVVMSR to obtain the address of the LPA-resident DSS initialization routine, IQAINI00. NP6DSS places the address of IQAINI00 in the PSW in the PRB pointed to by the DSS TCB. At completion of DSS initialization, NP6DSS returns control to the IEAVNP06 control routine.

NP6DMP Subroutine (Diagram 11.2)

NP6DMP analyzes the DUMP system parameter, and locates and opens the SYS1.DUMP data set for use by the SVCDUMP routine. NP6DMP determines whether the SYS1.DUMP data set is to go on a tape or a direct access device. NP6DMP opens the data set and returns control to the IEAVNP06 control routine.

NP6RMS Subroutine (Diagram 11.3)

NP6RMS issues the RMS macro instruction IGFVNIP, which initializes the MCH and CCH. At completion of RMS initialization, NP6RMS returns control to the IEAVNP06 control routine.

NP6EXCP Subroutine

NP6EXCP is a generalized I/O routine used by NP6DMP to read record 1 and write end-of-file in record 1 of the SYS1.DUMP data set. NP6EXCP initializes the IOB, clears the ECB, and issues an EXCP instruction. A WAIT macro instruction is used to wait for the I/O to complete. When I/O is complete, NP6EXCP returns control to NP6DMP.

NP6TRA Subroutine (Diagram 11.4)

NP6TRA analyzes the TRACE system parameter and defines and initializes the system trace table. The nucleus initialization program defines two trace tables, a temporary trace table defined by IEAVNIP0 and the permanent system trace table defined by IEAVNP06. The permanent system trace table is defined in the last storage allocated in the nucleus buffer by NIP. This takes advantage of space that would otherwise be wasted when the resident nucleus is rounded up to a 4K boundary. The entries in the temporary trace table are moved into the permanent trace table during trace table initialization. Both the temporary and permanent trace tables are constructed similarly and consist of a variable number of 32-byte entries. The number of entries in the temporary table is specified during system generation, while the permanent table size may be specified either during system generation or during nucleus initialization. Each entry contains information about an event (most often an interruption) and is recorded by the system at the time the event occurs.

Three fullword pointers, which occupy the 12 bytes preceding the first entry, define the dimensions of the trace table. The first pointer, which is located through fixed storage location X'54' (FLCTRACE) and a secondary CVT pointer (SCVTRACE), points to the current or most recently recorded entry. The second pointer points to the first (lowest storage address) entry in the table, and the third pointer points to the highest entry in the table.

Entries are made in ascending order in the trace table, beginning with the lowest address. When the highest entry area in the table is filled, the next entry is made in the lowest address space, overlaying a previous entry.

NP6MCVE Subroutine

This is a common routine used to move variable-length areas of storage in increments of 256 bytes or less.

IEAVNP07 (DIAGRAM 12.0)

IEAVNP07 is entered from IEAVNIPM primarily to process the HARDCPY and PAL system parameters. IEAVNP07:

1. Defines the unit that is to be used as the hardcopy log for console messages and/or displays.

2. Defines the location of LPA-resident DCMs (display control modules) that are used by the graphic console support processors to buffer display images for individual consoles (graphic consoles only).

3. Defines the limit and threshold values for the paging supervisor algorithms.

4. Locates the SVC OPEN Router in the LPA.

5. Tests for the presence of the extended-precision floating-point feature (except divide).

NP7HDCPY Subroutine (Diagram 12.1)

NP7HDCPY analyzes the HARDCPY parameter and identifies the console or system log data set to be used for the hardcopy log.

NP7HDCPY alerts the operator if the HARDCPY parameter was specified incorrectly, if the hardcopy console is not available (console was offline during IEAVNP01 console initialization), or if the specified device address does not represent a unit that was defined during system generation as a console with hardcopy capabilities. If the NP7HDCPY routine prompts the operator to specify HARDCPY, the operator is permitted to continue the parameter specification to a second line.

NP7PDCM Subroutine (Diagram 12.2)

NP7PDCM identifies the pageable DCM (display control module) addresses for display consoles (2250, 2260, 3270, 3155, and 3060). The DCM addresses are defined in the nucleus for later use by the communications task. When the last UCM entry has been examined, NP7PDCM returns control to the IEAVNP07 control routine.

NP7LPAFN (Diagram 12.3)

This subroutine is called by NP7PDCM and NP7OTEST to search the LPA for a specified module. Preference is given to a module found in the fixed LPA or the LPA update area (sometimes called LPA modifications).

NP7PAL Subroutine (Diagram 12.4)

NP7PAL analyzes the PAL (paging algorithm limit) system parameters specified via the SYSP parameter or by the operator and establishes paging supervision limits and threshold values. NP7PAL, optionally, displays the PAL subparameter values that result from the analysis and allows the operator to respecify the PAL parameter.

NP7OTEST Subroutine

NP7OTEST determines whether the SVC OPEN Router (IGFOI9RA) is in the link pack area. If found in the LPA, its entry address is stored at CVTDMSR. If not found in the LPA, the operator is notified, and the system is put in a disabled wait state (X'3B').

NP7EPFP Subroutine (Diagram 12.5)

This subroutine tests for the presence of the extended-precision floating-point feature (except divide) in the hardware. The divide capability is not normally present in System/370 CPUs and the Model 145 may lack the other floating-point features. Thus, it is necessary to place in the CVT an indication of the extended-precision floating-point capabilities of the CPU.

IEAVNIPX

IEAVNIPX is the last module to receive control during VS2 initialization. It:

- Defines the upper limit of the area which may be assigned to nonpageable jobs (NPXREAL).
- Defines the task dispatcher's APG (automatic priority group) (NPXAPG).
- Defines the task dispatcher's time-sliced prioity groups (NPXTMSL).
- Defines quickcell areas (NPXQCELL).
- Defines the master scheduler region (NPXMPA).
- Establishes limits on use of auxiliary storage by TSO and background tasks (NPXMPA1).
- Releases NIP buffer space in the SQA (NPXFBUF).
- Releases control blocks for NIP-loaded modules (NPXFJPQ).
- Defines available address space (NPXFAREA).

- Defines the master shceduler's LSQA (NPXMLSQA).
- Defines available page frames (NPXPFTAQ).
- Defines shared subpool 0 for the master scheduler and communications task (NPXMCSP0).
- Releases NIP's trap on ABEND (NPXRTRAP).
- Resets the dynamic address translation tables and exits to the master scheduler initialization program (NPXRDAT).

IEAVNIPX processes the APG, TMSL, MPA, REAL, SQACEL, LSQACEL, TSOAUX, and AUXLIST system parameters.

IEAVNIPX Control Routine (Diagram 13.0)

The Control Routine organizes NIP exit processing by sequentially invoking the various exit subroutines.

NPXREAL Subroutine

NPXREAL interprets the REAL system parameter, defines the nonpageable dynamic area, and ensures that the number of nonpageable pages being defined does not conflict with the previously defined NFX= subparameter of the PAL= parameter.

NPXREAL locates, via PARMTAB, the REAL parameter input and ensures that the value is not too large. NPXREAL determines the new upper boundary of the nonpageable area and stores it at CVTREAL. NPXREAL also stores the index of the PFTE that corresponds to the page containing the high nonpageable address in the PVT at PVTVEQR.

If the REAL parameter is incorrect, NPXREAL informs the operator by calling NIPWTOR to send a message. The operator may respecify or cancel. If he respecifies it, it is processed as above; if he cancels it, NPXREAL attempts to use a system-assigned value, as described below.

If the REAL parameter is not specified, the nonpageable dynamic area is set to 64K. If this area cannot be contained below the minimum system paging/SQA space, the NIPSWAIT routine (in IEAVNIPM) is entered to place the system in a disabled wait state (X'38'). A message is first issued indicating that storage is not available for the nonpageable area.

NPXAPG Subroutine (Diagram 13.1)

NPXAPG interprets the APG system parameter and defines or cancels the task dispatcher's automatic priority group.

NPXTMSL Subroutine (Diagram 13.2)

NPXTMSL interprets the TMSL system parameter and defines or cancels the task dispatcher's time-sliced priority groups.

NPXQCELL Subroutine (Diagram 13.3)

NPXQCELL interprets the SQACEL and LSQACEL system parameters and defines the quickcell areas for SQA or LSQA, or both.

NPXMPA Subroutine (Diagram 13.4)

NPXMPA interprets the MPA system parameter, which allows address space, in 64K segments to be added to the master scheduler region (128K bytes by default) and defines the virtual address space dedicated to the master scheduler region.

NPXMPA1 Subroutine (Diagram 13.5)

NPXMPA1 is called by NPXMPA to interpret the TSOAUX and AUXLIST system parameters and to establish limits on background and TSO use of auxiliary storage.

NPXFBUF Subroutine

NPXFBUF releases the storage occupied by the NIPSPE queue in SQA. NPXFBUF calls FREEMAIN to free each NIPSPE on the NIPSPE queue except the one in the NVT.

NPXFBUF invokes the IEAVNIPM subroutine NIPWTOR2 to release any dynamically acquired SQA reply buffers. NIP buffers in NIP's region space are released when the dynamic area is redefined by the NPXMPA and NPXFAREA routines.

NPXFJPQ Subroutine (Diagram 13.6)

NPXFJPQ purges programs loaded by NIP that still remain on NIP's TCB job pack queue. The purge is logical only, since the CDEs and LLEs representing these programs are dequeued and the storage the CDEs and LLEs, occupy is released while the storage the programs occupy is not released until available storage is redefined by the NPXFAREA routine.

NPXFAREA Subroutine

NPXFAREA defines the pageable and nonpageable address space that is available for later region and LSQA allocation.

NPXFAREA redefines the nonpageable PQE and FBQE to include the address space identified by NPXREAL beginning at the top of the nucleus. NPXFAREA updates the pageable PQE and FBQE to reflect address space beginning at the end of the nonpageable dynamic area and extending to the low address of the master scheduler region identified by NPXMPA.

NPXMLSQA Subroutine (Diagram 13.7)

NPXMLSQA allocates the single-segment LSQA related to the master scheduler region. This is done by means of a special interface to virtual storage supervision that is used only by NIP. The special interface is needed because NIP defines the LSQA without a TCB. This action prevents wasting space for a duplicate and unused TCB.

NPXPFTAQ Subroutine (Diagram 13.8)

NPXPFTAQ scans the paging supervisor's PFT (page frame table) and adds NIP's non-pageable area to the available PFT queue. NPXPFTAQ moves to the available queue all page frames that are assigned to a nonpageable region.

NPXMCSP0 Subroutine (Diagram 13.9)

NPXMCSP0 defines the master scheduler subpool 0 as shared with the communications task. This allows the communications task, which is defined without a region, to acquire storage in the master scheduler region.

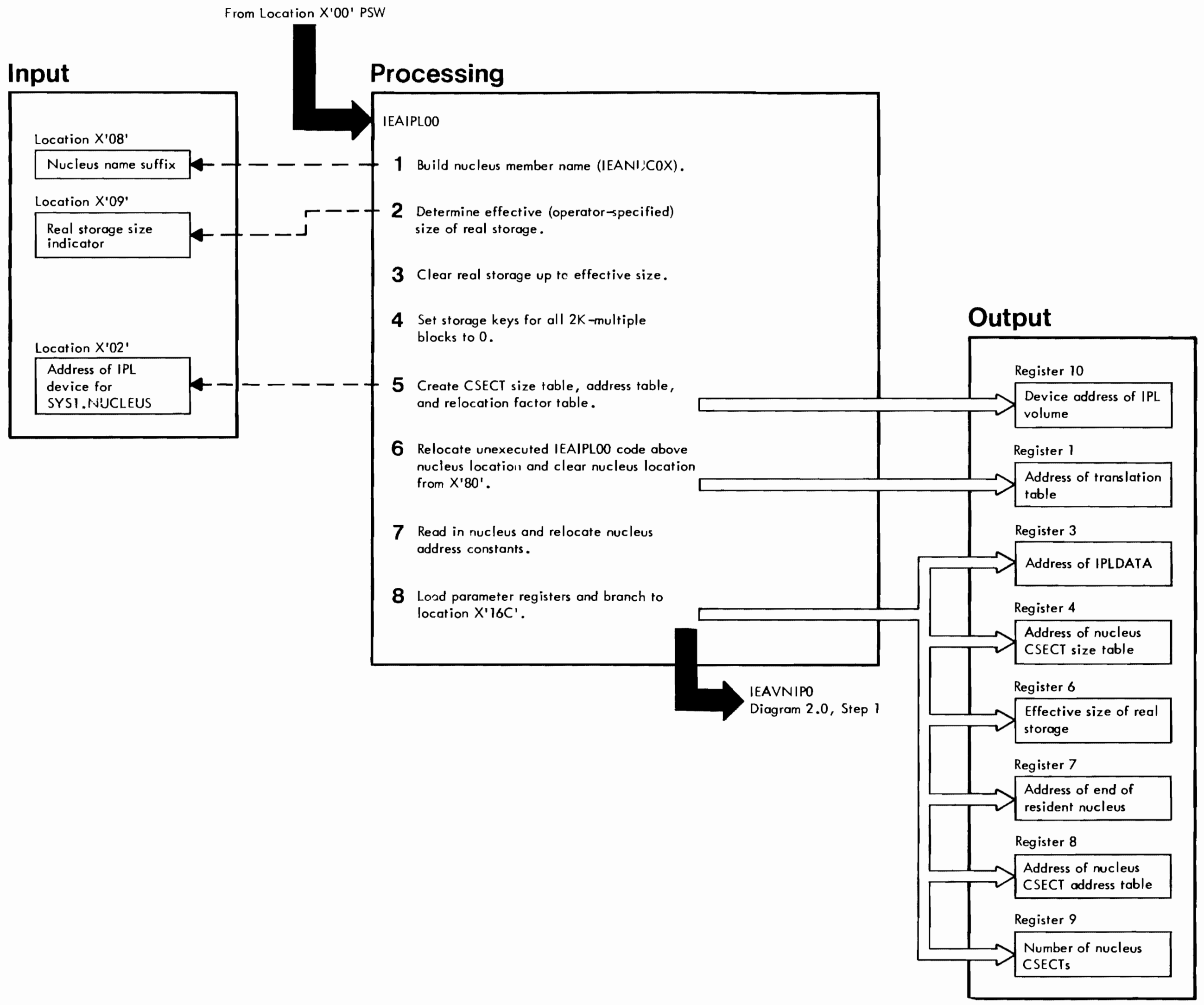
NPXRTRAP Subroutine

NPXRTRAP restores the SVC table entry for ABEND (SVC 13) to its original contents upon entry to NIP.

NPXRDAT Subroutine (Diagram 13.10)

NPXRDAT sets up the input parameters for the master scheduler Initialization program IEEVIPL, resets the dynamic address translation tables, and terminates supervisor initialization by issuing a LINK macro instruction to IEEVIPL.

Diagram 1.0: IEAPIL00
Initial Program Load



1 Build nucleus member name (IEANUC0X).

2 Determine effective (operator-specified) size of real storage.

3 Clear real storage up to effective size.

4 Set storage keys for all 2K-multiple blocks to 0.

5 Create CSECT size table, address table, and relocation factor table.

6 Relocate unexecuted IEAIPL00 code above nucleus location and clear nucleus location from X'80'.

7 Read in nucleus and relocate nucleus address constants.

8 Load parameter registers and branch to location X'16C'.

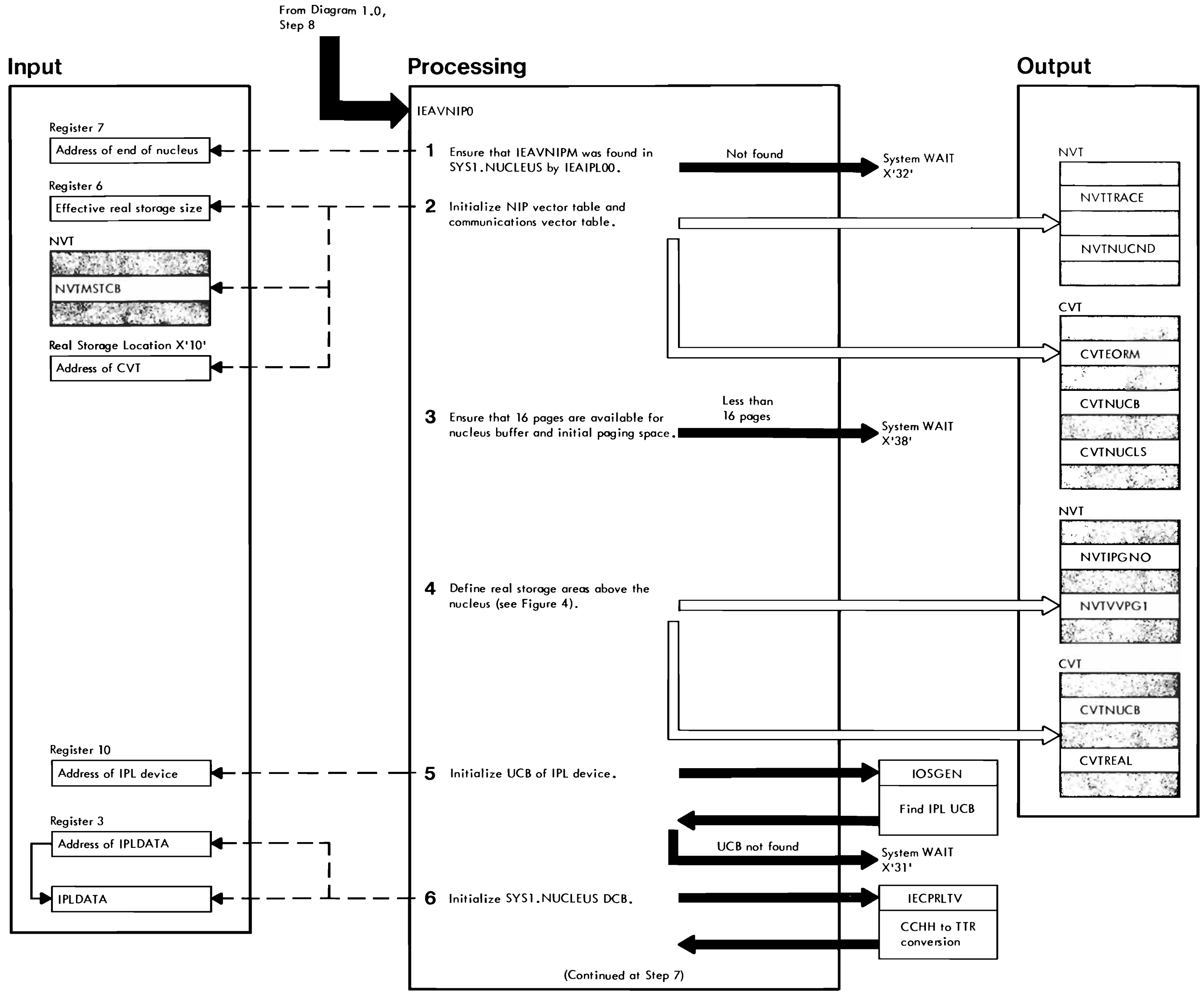
Diagram 1.0: IEAIPL00

Notes on Processing	Label
1. The EBCDIC character at location X'08' is a suffix appended to "IEANUC0" to form the nucleus member name. The system assigned character "1" may be replaced by the operator by setting an address stop at location X'80' before pressing the LOAD button. When the address stop is reached, the operator overlays the "1" with the desired EBCDIC character.	IEASTAR1
2. The contents of location X'09' are used to determine the effective size of real storage. Using the address stop procedure described in Step 1, the operator may specify that effective real storage size is to be less than the real storage size. One of the following hexadecimal values must be inserted in location X'09': X'A7' - 192K, X'A8' - 348K, X'C6' - 64K, X'C7' - 128K, X'C8' - 256K, X'C9' - 512K, X'D0' - 768K, X'D1' - 1024K	
3. The real storage area above IEAIPL00 is set to 0 by a Move Character instruction in a loop.	IEAZRLP3
4. A Set Storage Key instruction is used in a loop to set the storage keys for all 2K-multiple blocks to 0.	IEAMXLOC
5. The halfword at location X'02' is used to determine the location of the device that contains the nucleus data set (SYS1.NUCLEUS).	IEAPCKEY
6. The coding in IEAIPL00 beginning at label IEADATDS is relocated above the area to be occupied by the nucleus. Control is passed to the relocated coding which sets to 0 the area from which it was relocated.	IEAADDRS
7. The text records of IEANUC0x are read into the low end of real storage (see Figure 3).	IEARD1
8. Location X'16C' contains an LPSW instruction that specifies the PSW at location X'170'. This PSW specifies the entry point in IEAVNIP0, the first NIP module.	IEAOUTCD

Diagram 1.0: IEAPIL00 (Cont'd)

Notes on Processing		Label
Error conditions that can occur during execution of IEAIPL00 result in a disabled system wait state with the appropriate wait state code displayed in the lights on the CPU control panel and stored in the address portion of a current PSW. The wait state codes are:		
X'01'	First Test I/O instruction indicates that I/O is not operational.	IEAERR1 (IEAIOTST)
X'02'	An I/O instruction could not be started; a Start I/O instruction resulted in a CSW being stored for reasons other than unit busy (CU busy or CU end).	ERRSIO2 (IEASTRIO)
X'03'	An I/O operation could not be started; the CSW was not stored.	ERRSIO3 (IEASTRIO)
X'04'	The CSW was not stored following an SIO/TIO sequence; channel was not busy.	ERRTIO (IEATSTIO)
X'05'	A unit check occurred following a successfully started I/O operation. The unit check was not caused by a track-condition check, a file mask violation/end of track, or an end of cylinder. The first four bytes of sense data are end of cylinder. The first four bytes of sense data are saved at location X'54'; the address of the failing CCW string is saved at location X'4C'.	ERRUCK (SENSE) (SNSPRCD) (ECFMERFX) (ECFMERFX)
X'06'	The CSW stored after a TIO operation indicates that a program check, channel data check, channel control check, channel chaining check, or interface control check occurred.	ERRCSW (CSWTST)
X'0C'	The active nucleus member IEANUC0x is not edited in scatter format and cannot be loaded by IEAIPL00.	ERRSCTR (IEANONPM)
X'0E'	The SYS1.NUCLEUS data set or the active nucleus member IEANUC0x was not found on the IPL volume on the device specified by the address in bytes X'02' and X'03'. The nucleus cannot be loaded by IEAIPL00.	ERRNUC (IEACOMPR) (SENSE)
X'17'	A unit check occurred following a successfully started I/O operation (either a Sense I/O or an attempt to read the home address and record 0 of a track that previously caused a track condition check).	ERRSNS (SENSE)
X'18'	Available space for nucleus RLD records has been exceeded; there is insufficient space to load IEANUC0x.	IEAERR8 (IEABIDRL)
X'19'	An unexpected program check occurred. IEAIPL00 has not finished loading the resident nucleus. Either IEAIPL00 is in error or the storage in which it resides is failing.	

Diagram 2.0 (1 of 3): IEAVNIPO Initial NIP Processing



1 Ensure that IEAVNIPM was found in SYS1.NUCLEUS by IEAIPL00.

2 Initialize NIP vector table and communications vector table.

3 Ensure that 16 pages are available for nucleus buffer and initial paging space.

4 Define real storage areas above the nucleus (see Figure 4).

5 Initialize UCB of IPL device.

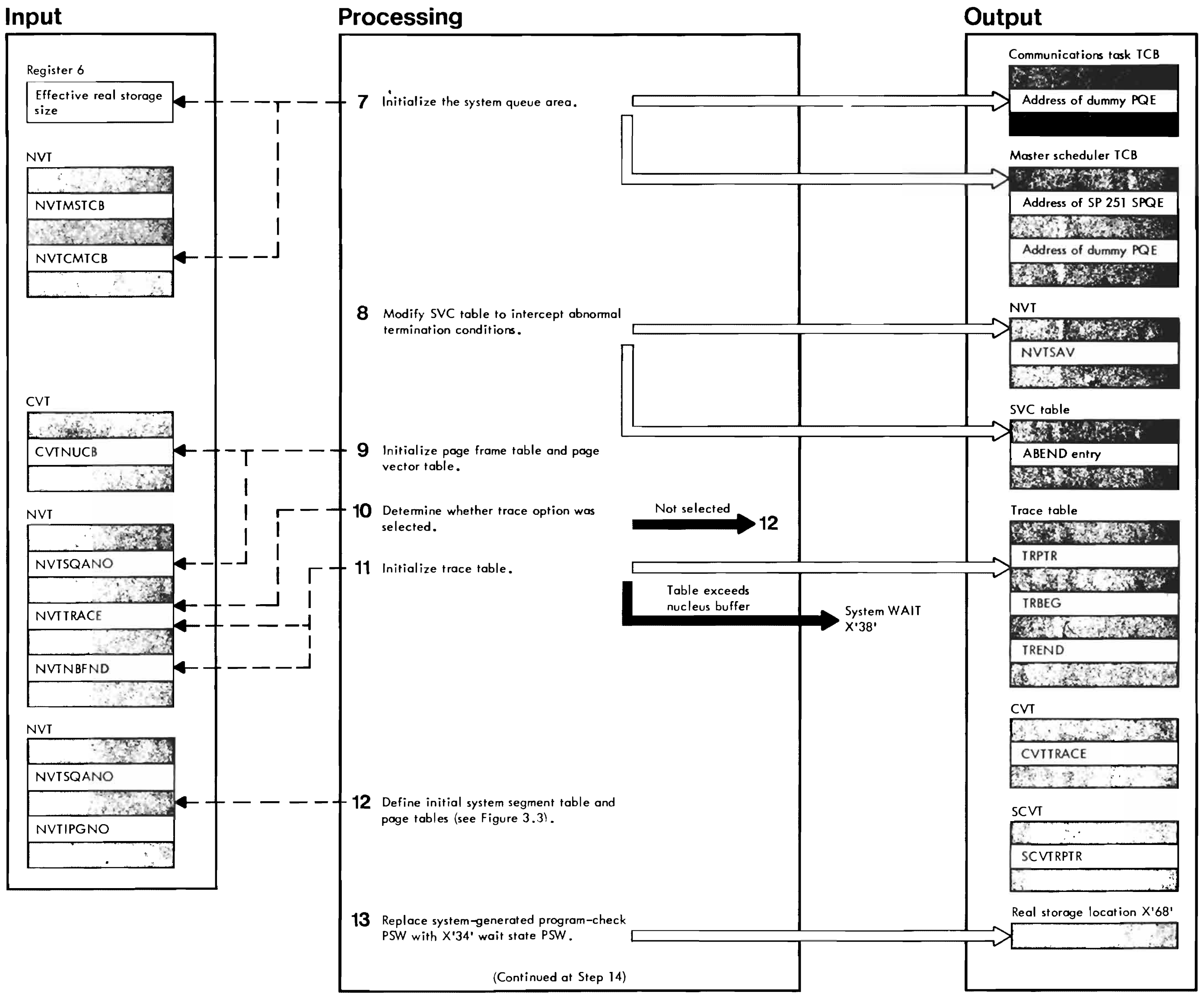
6 Initialize SYS1.NUCLEUS DCB.

(Continued at Step 7)

Diagram 2.0: IEAVNIPO

Notes on Processing	Label
1. IEAVNIPO exists immediately above the nucleus if IEAVNIPM was found in the SYS1.NUCLEUS data set. The highest four pages in real storage are allocated to the system queue area. An 80K portion of the area between the nucleus and the SQA is allocated to NIP as a dynamic execution area. The area not allocated to NIP or SQA is divided equally between the nucleus buffer as subpool 251 in NIP's region (adjacent to the nucleus) and the initial paging space (adjacent to the SQA).	IEAVNIPO
5.	NPOUCBLK
6. IPLDATA is described in Section 5.	
7. The four pages allocated to the SQA in Step 4 are initialized as shown in Figure 5. Until the pageable storage space is defined (by IEAVNP04), all requests for SQA must be satisfied form the minimum four pages. A trap routine and a system wait state are defined in IEAVNIPM to terminate initialization if the SQA is exhausted.	NPOSQINT
8. The entry in the SVC table for SVC 13 is replaced with the address of the ABEND trap routine within IEAVNIPO (label NPOABEND). This routine processes unexpected ABEND conditions and causes a system wait state (X'30').	

Notes on Processing	Label
9. The temporary page frame table is built at the top of the nucleus buffer and contains PFTEs (page frame table entries) for each page in the initial SQA. These PFTEs are initialized with the virtual block numbers; the first and last block numbers are placed in the page vector table in the nucleus. The apparent PFT address is set to the real storage address where the first PFTE would exist if there were PFTEs for real storage page frames.	
10. The number of trace table entries is zero if the TRACE option was not selected during system generation.	
11. The trace table is defined at the high end of the nucleus buffer (beneath the page frame table).	
12. The system segment table and page tables for all of real storage are defined in the upper area of the SQA (see Figure 6). No external page table entries are built because no areas represented by the system segment table are paged. Space is reserved between the system segment table and the first (SQA) page table for the user segment table which will be defined during IEAVNIPX execution. All page table entries for real storage between NIP's region and the SQA are flagged as invalid, and corresponding segment table entries are flagged as page-table-not-defined.	NPOATINT NPOSTELP NPOVRUN



7 Initialize the system queue area.

8 Modify SVC table to intercept abnormal termination conditions.

9 Initialize page frame table and page vector table.

10 Determine whether trace option was selected.

11 Initialize trace table.

12 Define initial system segment table and page tables (see Figure 3.3).

13 Replace system-generated program-check PSW with X'34' wait state PSW.

(Continued at Step 14)

Diagram 2.0: IEAVNIP0 (Cont'd)

Notes on Processing	Label
13. This causes a system wait state (X'34') when a System/370 instruction or an instruction that requires the dynamic address translation feature is executed on a system that does not include this feature.	
Error Processing	
Abnormal error conditions result in a disabled system wait state with one of the following codes being stored in the address field of the current PSW (bits 56-63):	
X'30' An unexpected ABEND request has occurred. The system completion code is found in bits 36-47 of the current (wait state) PSW.	
X'31' The IPL volume resides on a unit for which no UCB was found.	NP0ERR31

Notes on Processing	Label
X'32' IEAVNIPM was not found in the SYS1.NUCLEUS data set. Bits 36-47 of the current (wait state) PSW contain X'7D4' (the last 12 bits of the EBCDIC name "IEAVNIPM").	NP0ERR32
X'33' An I/O error occurred during BLDL processing. Bits 36-47 of the current (wait state) PSW contain X'7D4' (the last 12 bits of the EBCDIC name "IEAVNIPM").	NP0ERR33
X'34' An instruction that requires the dynamic address translation feature has been executed on a CPU that does not include this feature.	
X'38' There is insufficient real storage for VS/2 to be initialized by NIP.	NP0ERR38

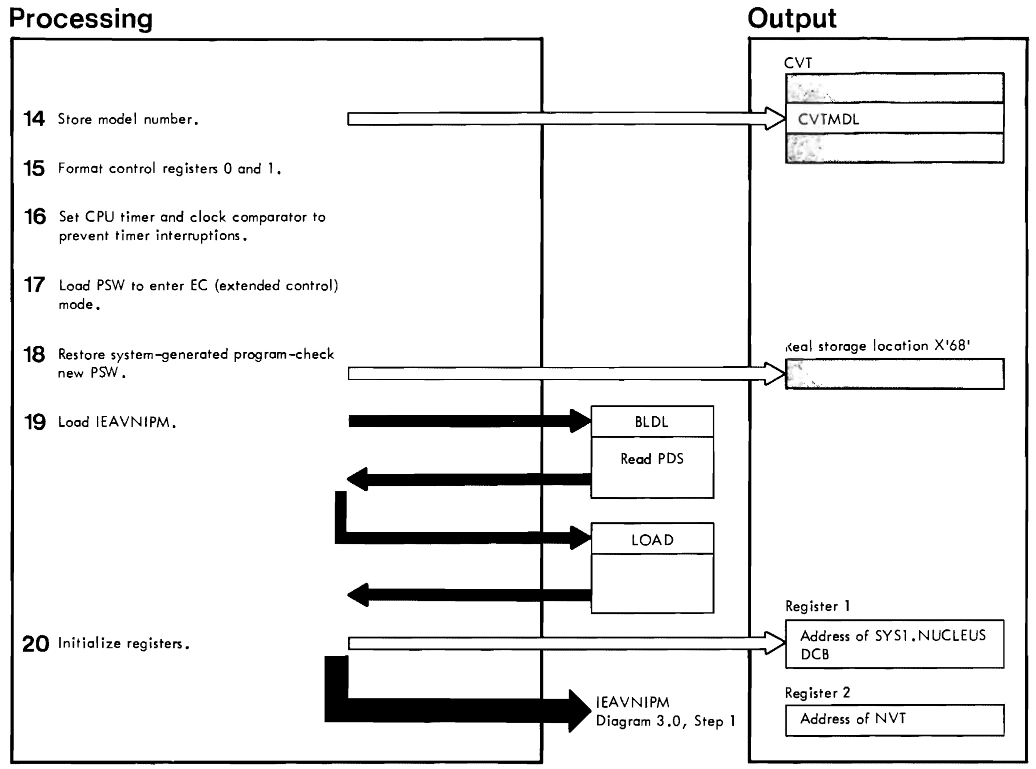
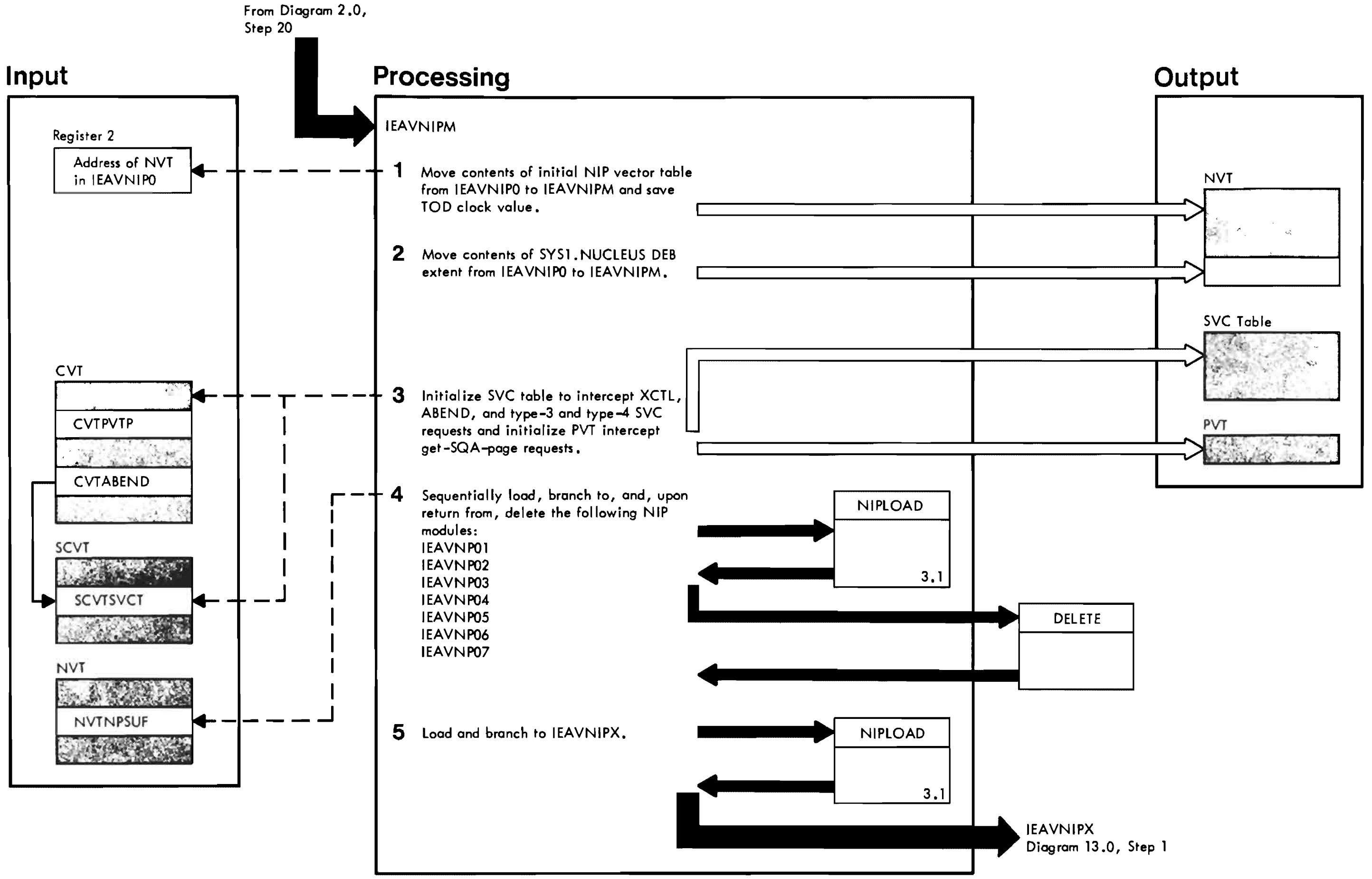

Processing
Output
14 Store model number.
15 Format control registers 0 and 1.
16 Set CPU timer and clock comparator to prevent timer interruptions.
17 Load PSW to enter EC (extended control) mode.
18 Restore system-generated program-check new PSW.
19 Load IEAVNIPM.
20 Initialize registers.
CVT
CVTMDL
Real storage location X'68'
BLDL
Read PDS
LOAD
Register 1
Address of SYS1.NUCLEUS DCB
Register 2
Address of NVT
IEAVNIPM
Diagram 3.0, Step 1

Diagram 2.0: IEAVNIP0 (Cont'd)

Notes on Processing	Label
17.	NP0ELPSW



1 Move contents of initial NIP vector table from IEAVNIP0 to IEAVNIPM and save TOD clock value.

2 Move contents of SYS1.NUCLEUS DEB extent from IEAVNIP0 to IEAVNIPM.

3 Initialize SVC table to intercept XCTL, ABEND, and type-3 and type-4 SVC requests and initialize PVT intercept get-SQA-page requests.

4 Sequentially load, branch to, and, upon return from, delete the following NIP modules:
IEAVNP01
IEAVNP02
IEAVNP03
IEAVNP04
IEAVNP05
IEAVNP06
IEAVNP07

5 Load and branch to IEAVNIPX.

Diagram 3.0: IEAVNIPM

Notes on Processing	Label
1. The initial NIP vector table in IEAVNIPO is moved to corresponding fields in IEAVNIPM, thus relocating the NVT.	IEAVNIPM
3. The addresses of the Intercept routine for ABEND (NIPABEND), XCTL (NIPSVCX), type-3 and type-4 SVC routines (NIPSVC), and the Get-SQA-Page routine (NIPSQEND) are placed in the appropriate locations in the SVC table and the page vector table (NIPSQEND). Entry to NIPSQEND is from virtual storage management; other entries are from the SVC Second-Level Interruption Handler. NIPSQEND causes a system wait state (X'36'). NIPABEND causes a system wait state (X'40'). NIPSVCX and NIPSVC are described by Diagram 3.3.	
4. IEAVNIPM uses a series of suffixes and an index to these suffixes in the NVT that are appended to the base name "IEAVNP" to determine which NIP module is to get control next. Each module is loaded by the subroutine NIPLOAD, branched to, and deleted. The last value, 07, is followed by X'FF', which indicates that the last NIP module, IEAVNIPX, is to be entered. This module is loaded and branched to similarly; but, because control does not return, IEAVNIPX is not deleted by IEAVNIPM.	NPMNEXT NPMLOAD

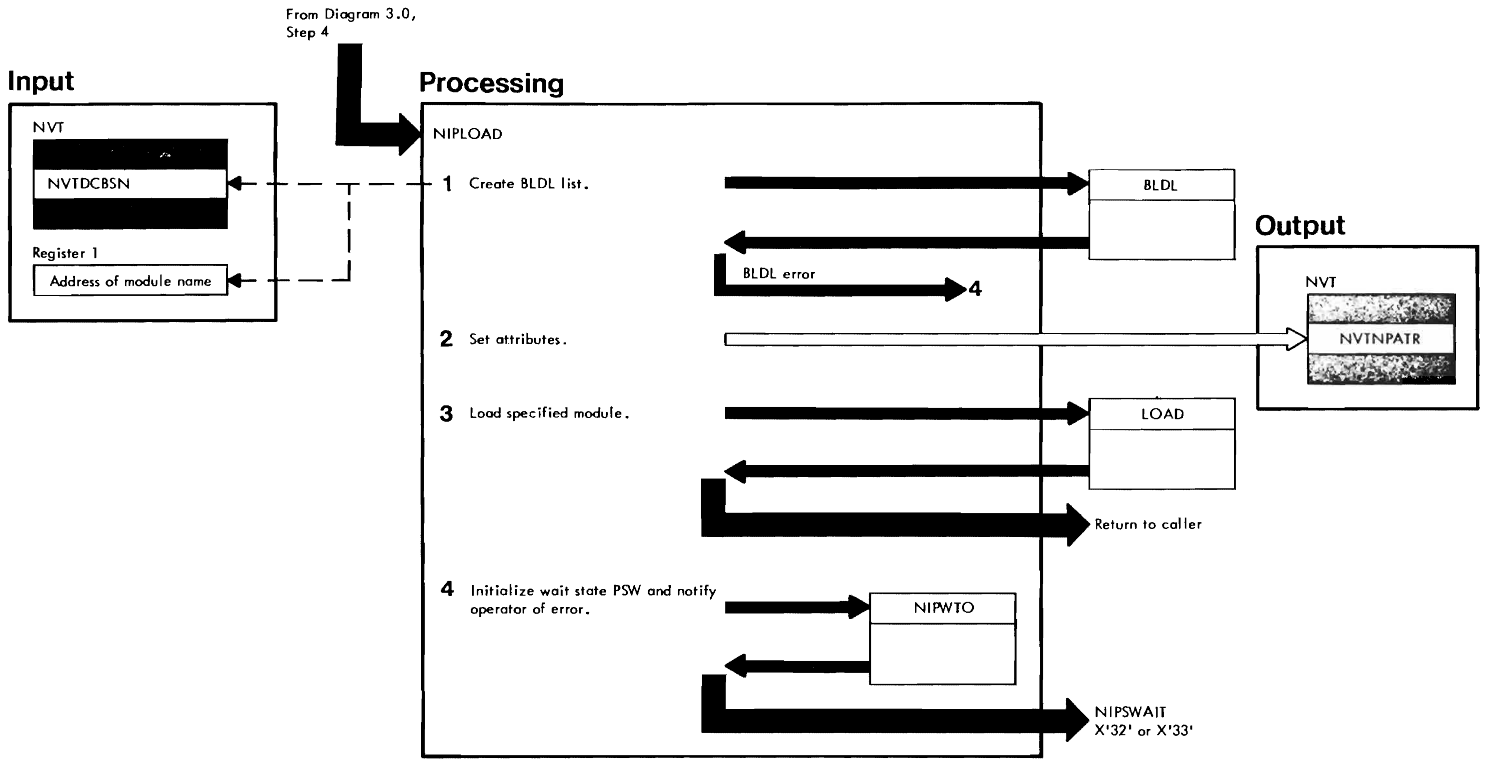

From Diagram 3.0, Step 4
Input
NVT
NVTDCBSN
Register 1
Address of module name
Processing
NIPLOAD
1 Create BLDL list.
BLDL
BLDL error
4
2 Set attributes.
3 Load specified module.
LOAD
Return to caller
4 Initialize wait state PSW and notify operator of error.
NIPWTO
NIPSWAIT X'32' or X'33'
Output
NVT
NVTNPATR

Diagram 3.1 NIPLOAD

Notes on Processing	Label
4. If the module was not found by BLDL, the NIPWTO subroutine is entered to issue the message IEA3-01I module NOT FOUND IN dsn. If an I/O error occurred during BLDL processing, NIPWTO issues the message IEA300I I/O ERROR DURING BLDL FOR module IN dsn. In both cases, the NIPSWAIT subroutine is entered to place the system in a disabled wait state (X'32' or X'33').	NPMLXIT

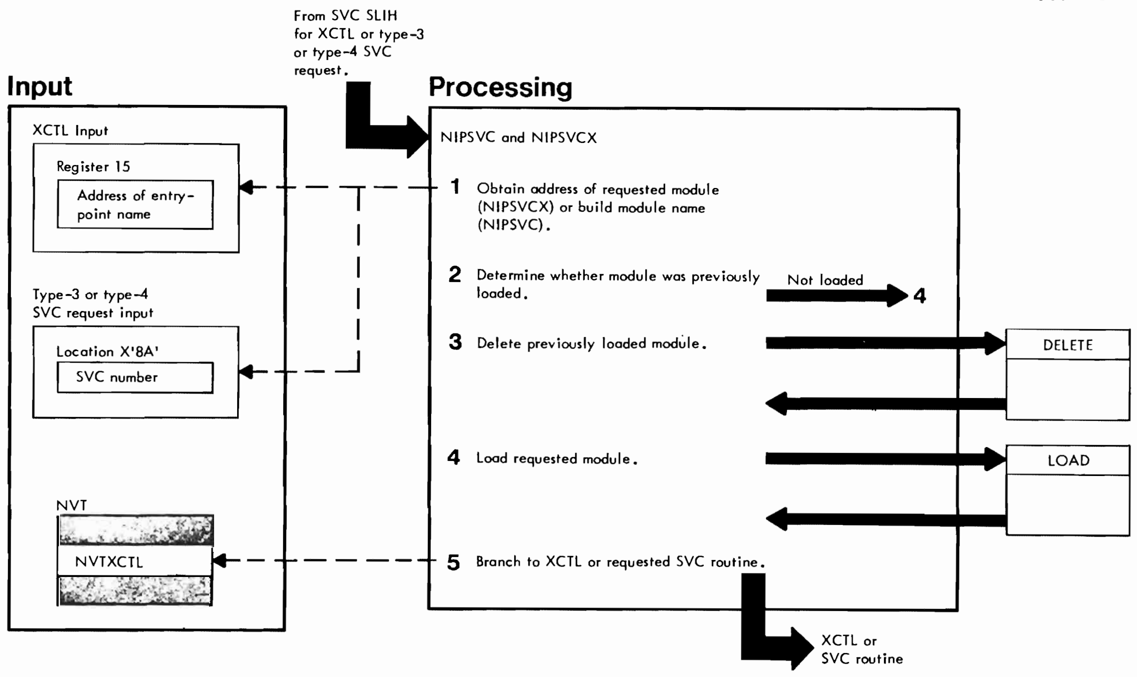

From SVC SLIH for XCTL or type-3 or type-4 SVC request.
Input
XCTL Input
Register 15
Address of entry-point name
Type-3 or type-4 SVC request input
Location X'8A'
SVC number
NVT
NVTXCTL
Processing
NIPSVC and NIPSVCX
1 Obtain address of requested module (NIPSVCX) or build module name (NIPSVC).
2 Determine whether module was previously loaded.
Not loaded
4
3 Delete previously loaded module.
DELETE
4 Load requested module.
LOAD
5 Branch to XCTL or requested SVC routine.
XCTL or SVC routine

Diagram 3.2 NIPSVC & NIPSVCX (IEAVNIPM)

Notes on Processing	Label
2. The name of a module loaded for a previous XCTL request is retained by IEAVNIPM until a subsequent XCTL is issued.	NPMCDSRH
4.	NPMCDSR2
5.	NPMSVCEX

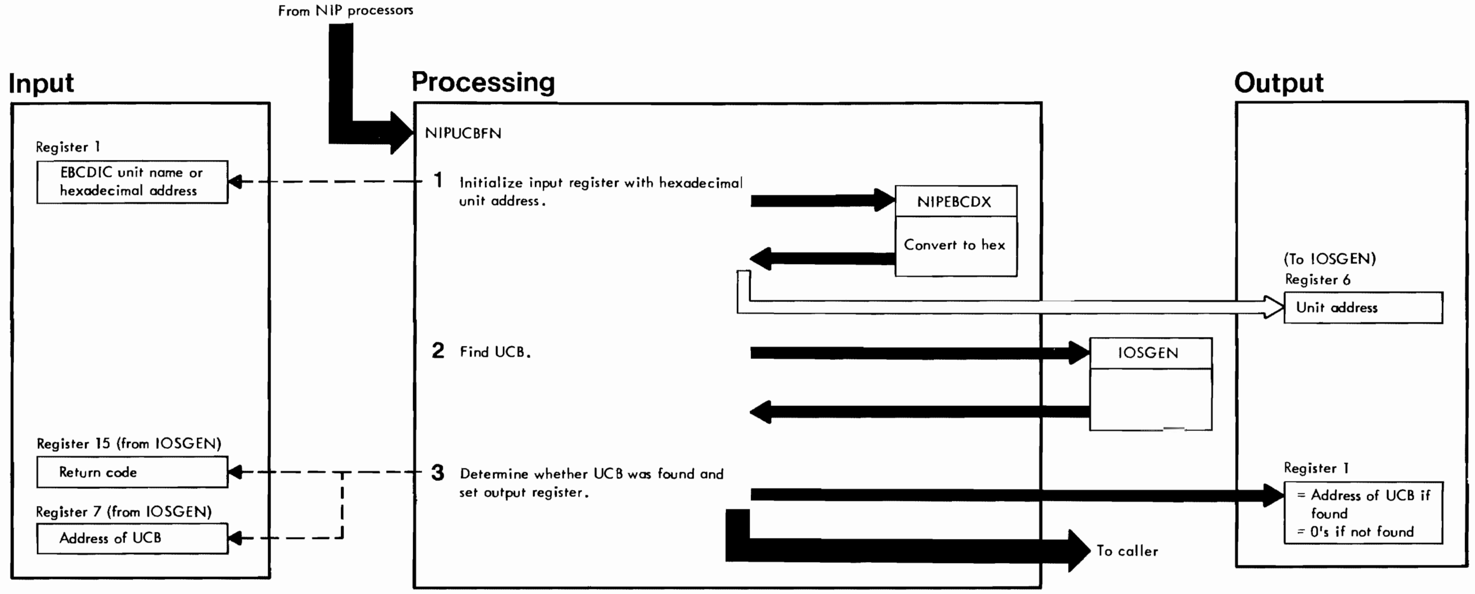
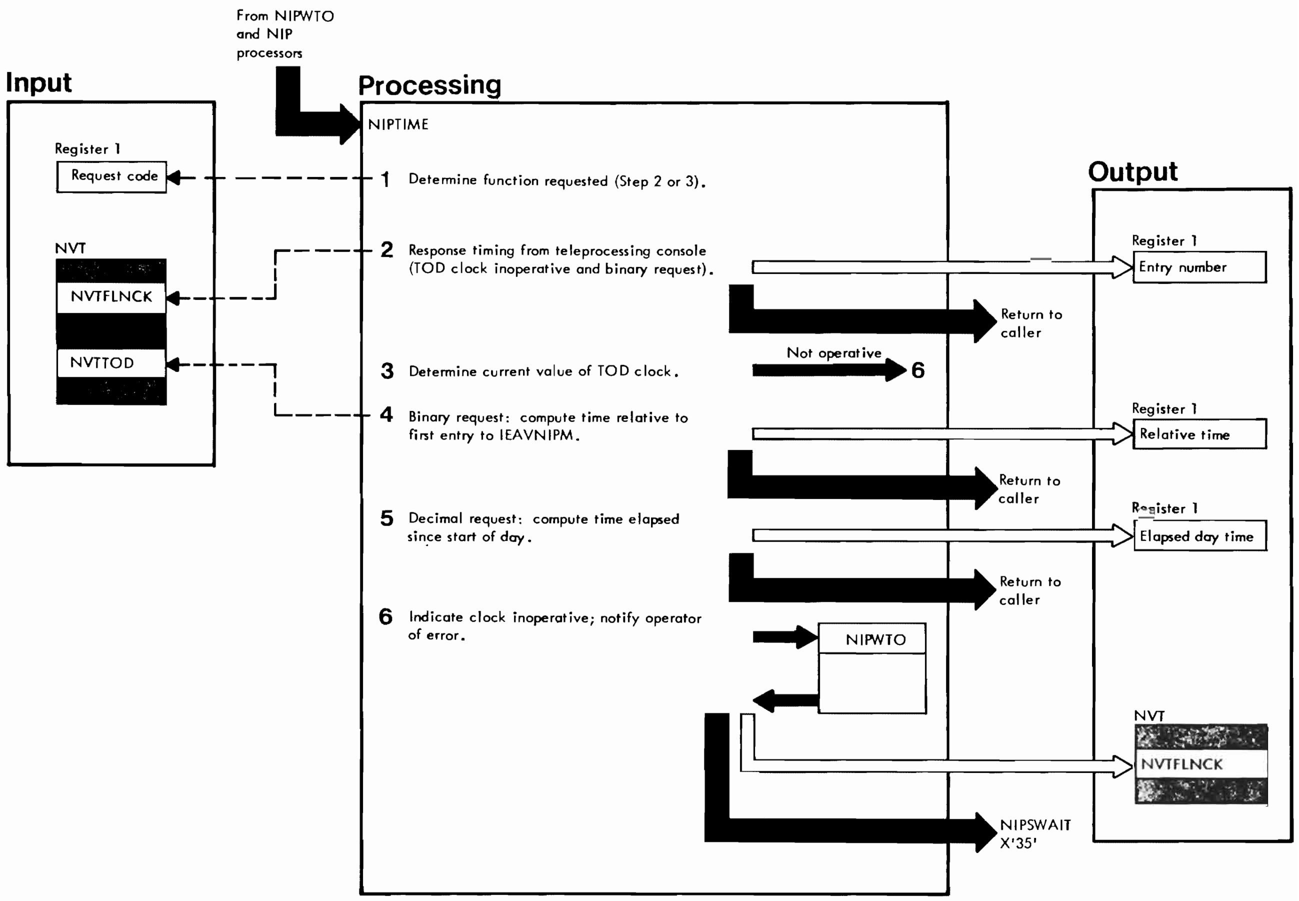

From NIP processors
Input
Register 1
EBCDIC unit name or hexadecimal address
Register 15 (from IOSGEN)
Return code
Register 7 (from IOSGEN)
Address of UCB
Processing
NIPUCBFN
1 Initialize input register with hexadecimal unit address.
NIPEBCDX
Convert to hex
2 Find UCB.
IOSGEN
3 Determine whether UCB was found and set output register.
To caller
Output
(To IOSGEN)
Register 6
Unit address
Register 1
= Address of UCB if found
= 0's if not found

Diagram 3.3: NIPUCBFN (IEAVNIPM)

Notes on Processing	Label
1. If register 1 is negative, the register contains the EBCDIC unit name and requires conversion by NIPEBCDX to the hexadecimal unit address.	NIPUCBFN
2.	NPMUCBLK
3.	NPMHVUCB

Diagram 3.4: NIPTIME
Determine Absolute Time or Relative Time Since IEAVNIPM was Entered



1 Determine function requested (Step 2 or 3).

2 Response timing from teleprocessing console (TOD clock inoperative and binary request).

3 Determine current value of TOD clock.

4 Binary request: compute time relative to first entry to IEAVNIPM.

5 Decimal request: compute time elapsed since start of day.

6 Indicate clock inoperative; notify operator of error.

Diagram 3.4: NIPTIME (IEAVNIPM)

Notes on Processing	Label
1. A request code X'04' indicates a binary request for time relative to first entry to IEAVNIPM. However, if the TOD clock has been previously indicated as inoperative, this is a request from NIPWTO to time teleprocessing response in writing messages prior to a system wait state. The entry number is (n+1)/10000.	
3. A Store Clock instruction is issued to determine the value of the TOD clock.	
5. A decimal request is indicated by X'00' in register 1.	NIPTOP
6. NIPTIME uses the NIPWTO subroutine to issue the message IEA302I TOD CLOCK INOPERATIVE and passes control to NIPSWAIT to place the system in a disabled wait state (X'35').	NIPTDEC

Diagram 4.0: IEAVNP01 Initialize System Consoles

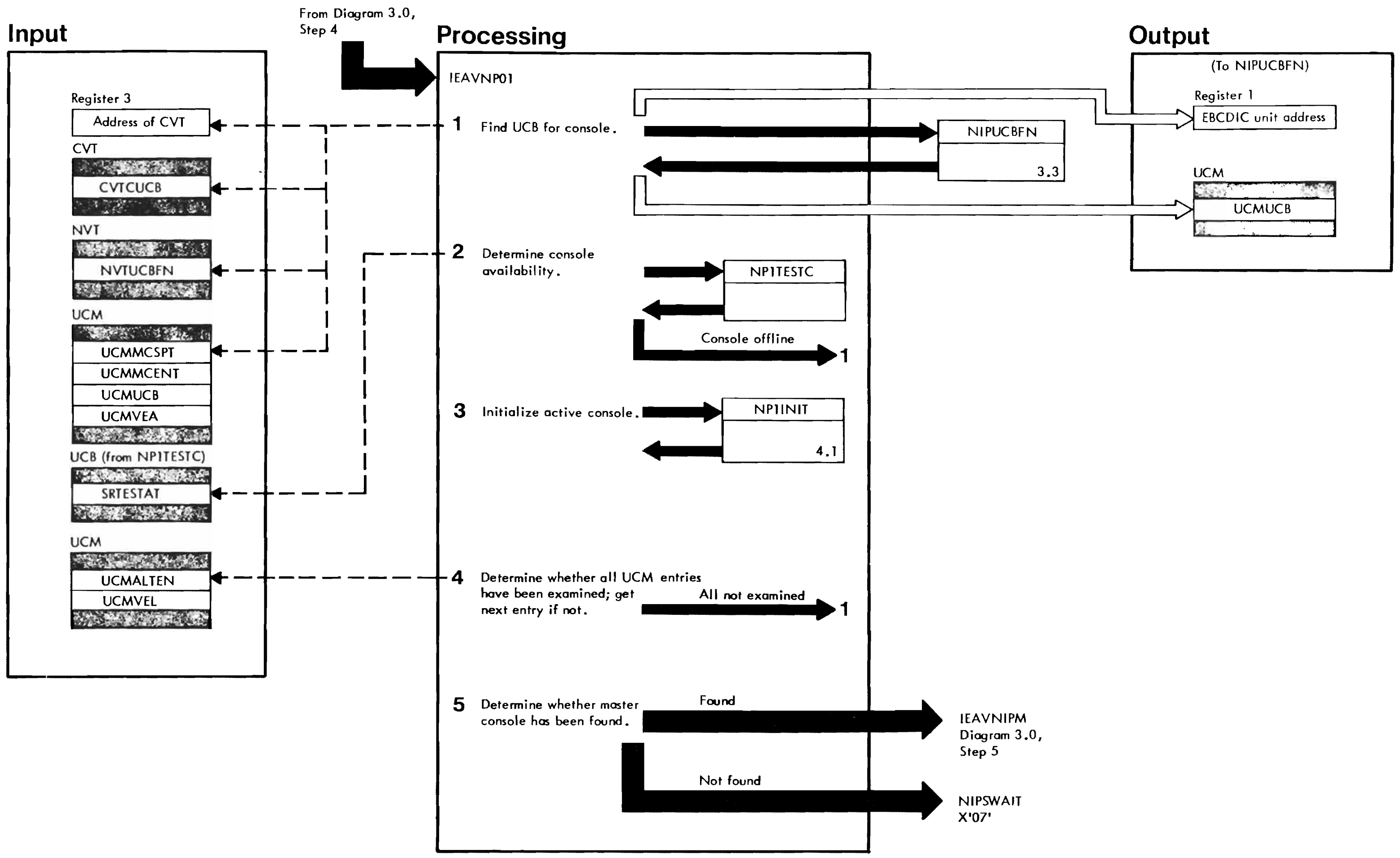
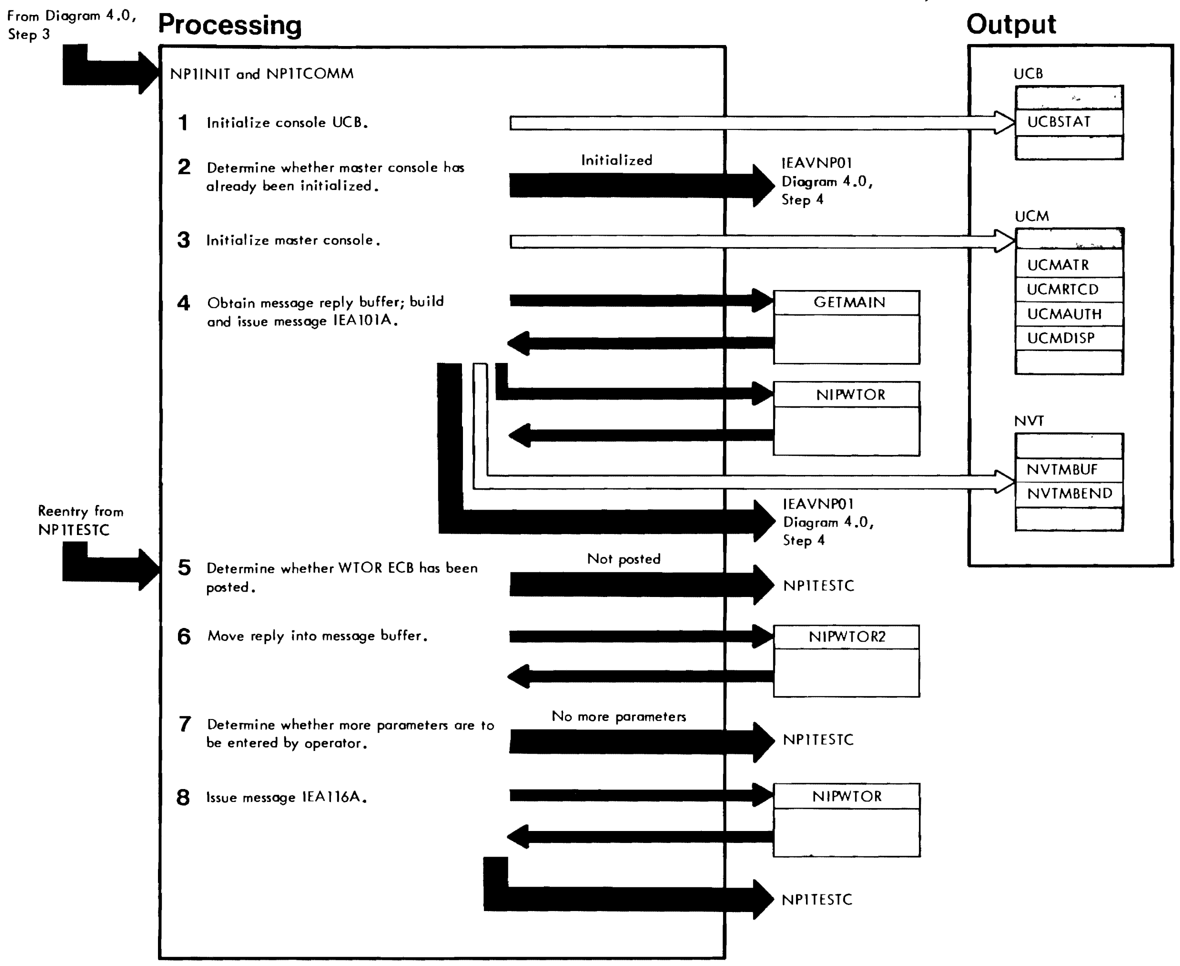


Diagram 4.0: IEAVNP01

Notes on Processing	Label
1. The initialization of consoles is performed in a loop, always beginning with the master console specified at system generation, then the alternates to the master console, and finally the other consoles represented in the UCM not yet tested, beginning with UCMVEA.	NP1ALT
2. The subroutine NP1TESTC executes a channel program to the specified console and indicates the results in SRTESTAT.	NP1TESTC
3. The subroutine NP1INIT flags the console as active. If the master console has not been initialized, the routing and authorization codes and the UCM master pointers are set, and initial operator communications are established (see Diagram 4.1).	NP1INIT
5. If more than one console is online or if the single console is graphic, HARDCPY is required (referred to by IEAVNP07).	NP1CMNT

Diagram 4.1: NP1INIT and NP1TCOMM
Establish Operator Communications



NP1INIT and NP1TCOMM

1 Initialize console UCB.

2 Determine whether master console has already been initialized.

3 Initialize master console.

4 Obtain message reply buffer; build and issue message IEA101A.

5 Determine whether WTOR ECB has been posted.

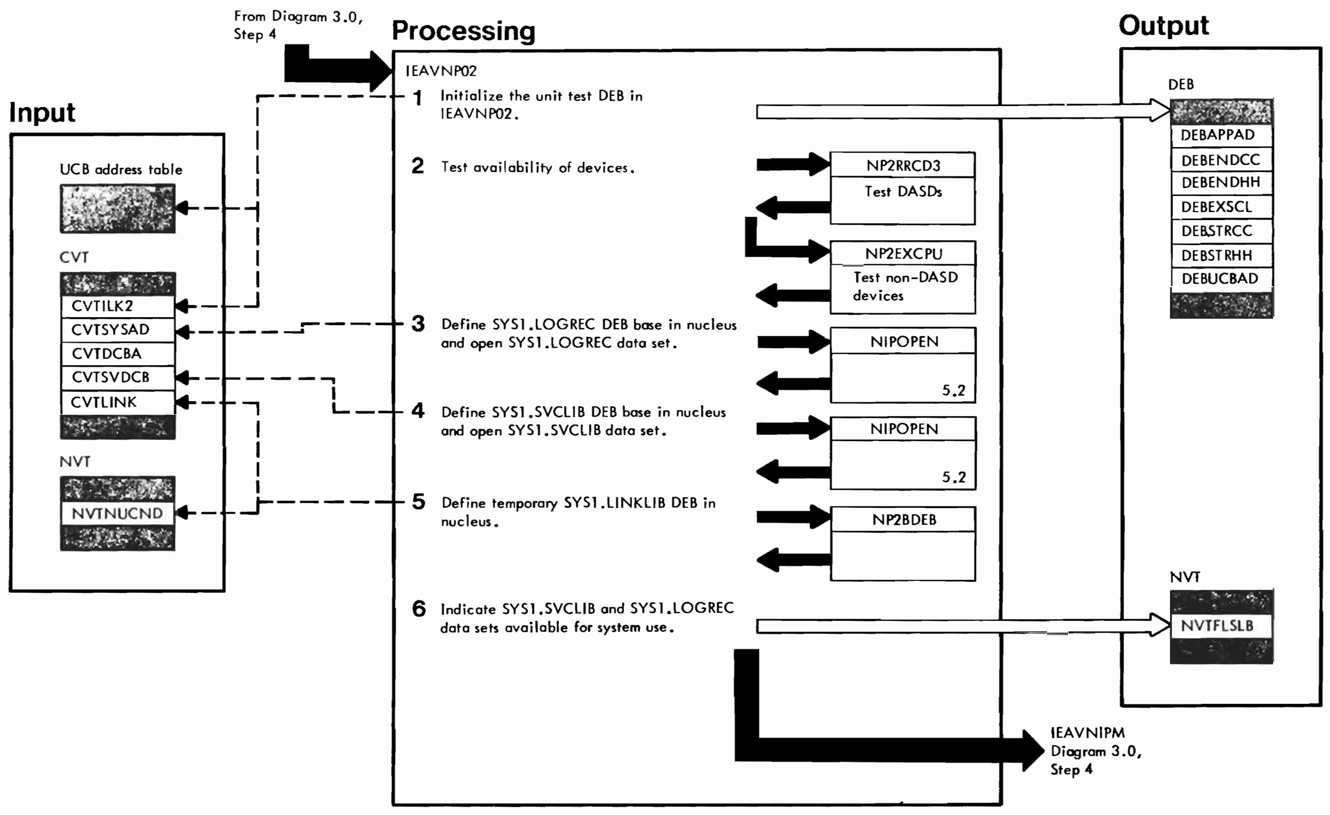
6 Move reply into message buffer.

7 Determine whether more parameters are to be entered by operator.

8 Issue message IEA116A.

Diagram 4.1: NP1INIT & NP1TCOMM (IEAVNP01)

Notes on Processing	Label
Establishing communications with the console operator is a two-part process in which the operator is requested to specify system parameters (NP1INIT) and the reply is moved into a message buffer for later processing by IEAVNP03.	NP1INIT
1. Both halves of a composite console are initialized before proceeding with Step 2.	NP1ACT
3. If the active master console is not the console specified during system generation, the routing and authorization codes are moved to the UCM for the new active master console, and the UCM prefix is updated.	
4. A 2K reply buffer is obtained from subpool 255 of the SQA. This buffer is deleted by IEECMWTL after NIP execution.	NP1TCOMM NP1COMM
7. Additional parameters are indicated by the characters 'CONT' followed by a quotation mark at the end of the parameter string. A quotation mark following anything but 'CONT' indicates the end of the parameter string.	NP1SCAN
8. The reply to this message is processed in the same way as the reply to IEA101A.	


From Diagram 3.0, Step 4
Input
UCB address table
CVT
CVTILK2
CVTSYSAD
CVTDCBA
CVTSVDCB
CVTLINK
NVT
NVTNUCND
Processing
IEAVNP02
1 Initialize the unit test DEB in IEAVNP02.
2 Test availability of devices.
NP2RRCD3
Test DASDs
NP2EXCPU
Test non-DASD devices
3 Define SYS1.LOGREC DEB base in nucleus and open SYS1.LOGREC data set.
NIPOPEN
5.2
4 Define SYS1.SVCLIB DEB base in nucleus and open SYS1.SVCLIB data set.
NIPOPEN
5.2
5 Define temporary SYS1.LINKLIB DEB in nucleus.
NP2BDEB
6 Indicate SYS1.SVCLIB and SYS1.LOGREC data sets available for system use.
IEAVNIPM Diagram 3.0, Step 4
Output
DEB
DEBAPPAD
DEBENDCC
DEBENDHH
DEBEXSCL
DEBSTRCC
DEBSTRHH
DEBUCBAD
NVT
NVTFLSLB

Direct Access Device Class

Unit Type	SIO/TIO cc=3	Unit Check (Intervention Required)	Unit Check (Other)	I/O Error (Other)	Normal I/O Status
2301	Offline	Offline	Offline[1]	Offline[1]	Cnline/Ready
2305	Offline	Offline	Offline[1]	Offline[1]	Online/Ready
2314	Offline	Not Ready*	Offline[1]	Offline[1]	Cnline/Ready
2319	Offline	Not Ready*	Offline[1]	Offline[1]	Online/Ready
3330	Offline	Not Ready*	Offline[1]	Offline[1]	Cnline/Ready
(Other)	Offline	Not Ready	Offline[1]	Offline[1]	Cnline/Ready

*Offline status is set if the DEVSTAT option was selected during system generation.

[1]An interpretive error message identifies the device status.

Magnetic Tape Device Class

Unit Type	SIO/TIO cc=3	Unit Check (Intervention Required) With Status "A" and "B" off	Unit Check (Intervention Required) With Status "A" and/or "B" on	Other
2400				
3400				
2420	Offline	Offline	NotReady	Cnline/Ready
(Other)	Offline	Offline	NotReady	Cnline/Ready

Unit Record, Communications, and Graphics Device Classes

Unit Type	SIO/TIO cc=3	Other
All	Offline	Online/Ready

Diagram 5.0: IEAVNP02

Notes on Processing	Label
1. IEAVNP02 contains the DEB used for testing devices. It is defined to reflect a maximum data set size of X'7FFF' tracks extending from CCHH 0 to X'FFFFFFFF.'	IEAVNP02
2. Each UCB in the UCB address table that has not been examined by IEAVNP01 is tested to determine the status of the associated device. For direct access devices, the subroutine NP2RRCD3 builds a channel program to read the volume label. For nondirect access devices, a NOP CCW is specified. The subroutine NP2EXCPU executes the channel program. The subroutine NP2TERR is entered following all non-DASD I/O operations and from NP2RRCD3 for failing DASD I/C. For a failing I/O operation, the UCB is set to indicate that the device is offline. A UCB for a DASD is set to indicate that the device is not ready. This processing occurs in a loop for each untested ECB in the UCB address tatle. Device availability testing is described in the adjoining tatle.	NP2SEARC NP2NDASD
3. IEAVNP02 passes the address of the system residence UCB (CVTSYSAD) and the address of the SYS1.LOGREC DCB (CVTDCBA) to NIPOPEN.	NP2DDEFS
4. The address of the SYS1.SVCLIB DCB (CVTSVDCB) is passed to NIPOPEN.	
5. The address of the SYS1.LINKLIB DCB (CVTLINK) is passed to NIPOPEN.	

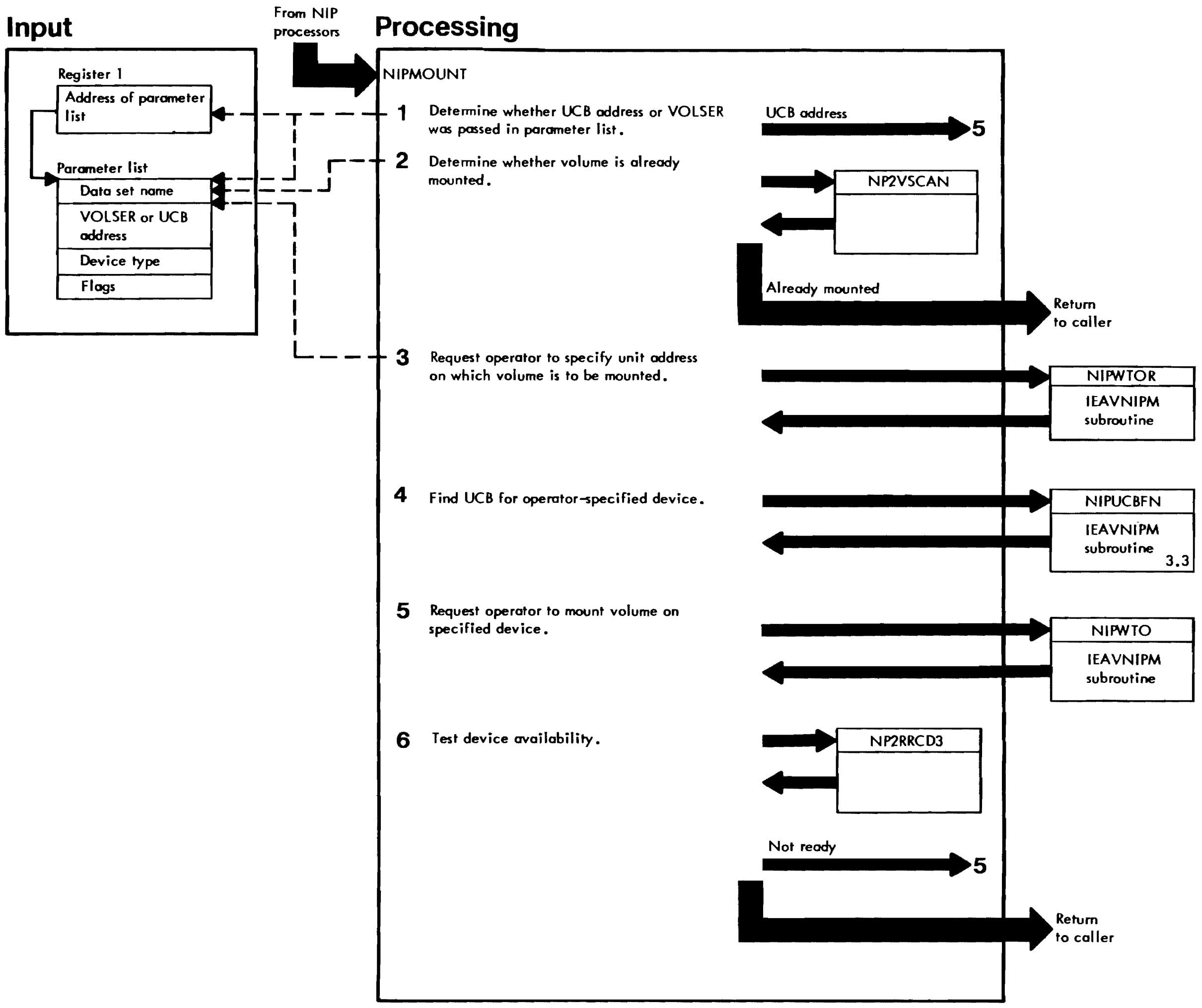
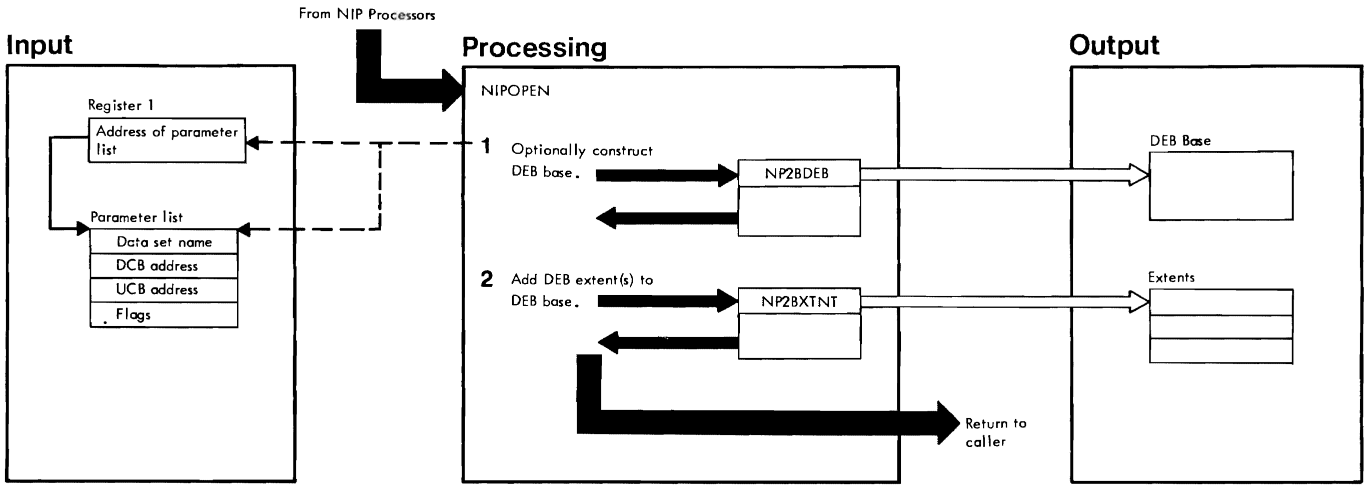

Input
Register 1
Address of parameter list
Parameter list
Data set name
VOLSER or UCB address
Device type
Flags
From NIP processors
Processing
NIPMOUNT
1 Determine whether UCB address or VOLSER was passed in parameter list.
UCB address
5
2 Determine whether volume is already mounted.
NP2VSCAN
Already mounted
Return to caller
3 Request operator to specify unit address on which volume is to be mounted.
NIPWTOR
IEAVNIPM subroutine
4 Find UCB for operator-specified device.
NIPUCBFN
IEAVNIPM subroutine
3.3
5 Request operator to mount volume on specified device.
NIPWTO
IEAVNIPM subroutine
6 Test device availability.
NP2RRCD3
Not ready
5
Return to caller

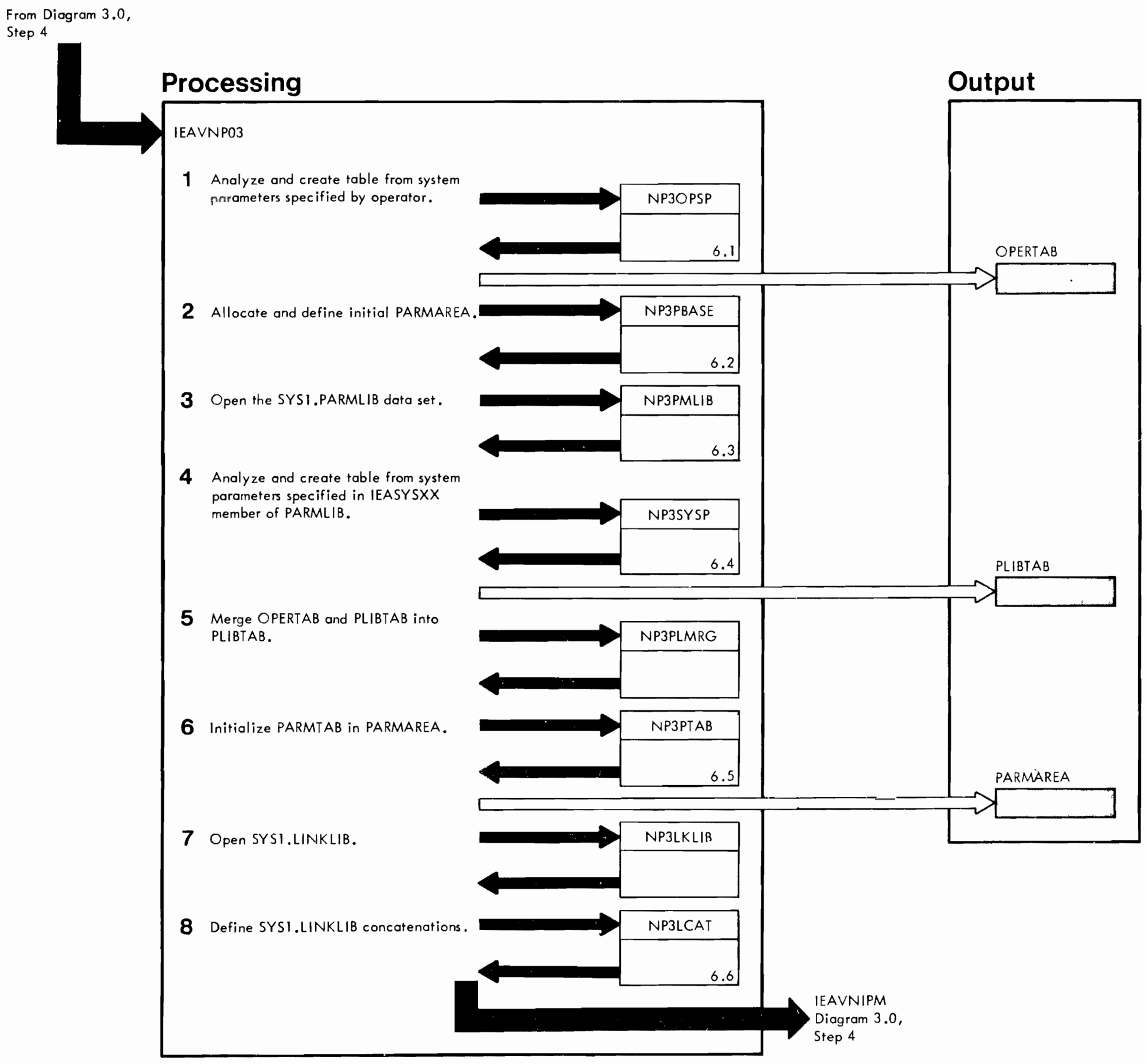
Diagram 5.1: IEAVNPA2 (IEAVNP02)

Notes on Processing	Label
1.	IEAVNPA2
2. The NP2VSCAN usbroutine compares the volume serial passed with each entry in the UCB address table. If a duplicate volume serial specification is found, NP2VSCAN returns the address of the duplicate UCB address table entry. If no device type is specified in the parameter list or if the UCB found by NP2VSCAN is for the specified device type, the volume is considered mounted.	NPA2VOL
3. The operator may be allowed, if so specified by the parameter list, to cancel the mount request. If the request is canceled, NIPMOUNT sets register 1 to 0 and returns control to the caller. If an invalid reply is received, the operator is requested to respecify the unit address.	NPA2SPEC NPA2EOB NPA2INV

Notes on Processing	Label
4. If the unit is invalid, the operator is requested to respecify the unit address. If a UCB was found for the specified volume but an unacceptable device type is found by NP2VSCAN in Step 2, and the volume is permanently resident, and the request cannot be canceled, the system is put in a disabled wait state by NIPSWAIT in IEAVNIPM (X'39'). If the volume is not permanently resident, the UCB found by NP2VSCAN is marked unavailable by NP2TERR, the UCB found by NIPUCBFN is marked online and not ready, and the operator is requested to mount the volume on the operator-specified device.	NPA2UNAC NPA2MNT
6. The subroutine NP2RRCD3 builds a channel program to read the volume label. If the channel program executed by NP2EXCPU fails but the record was found, the NIPSENSE subroutine in IEAVNIPM is entered to write an interpretive error message. A flowchart of this subroutine is included in Section 3.	

Diagram 5.2: NIPOPEN
Open Specified Data Set




From Diagram 3.0, Step 4
Processing
Output
IEAVNP03
1 Analyze and create table from system parameters specified by operator.
NP3OPSP
6.1
OPERTAB
2 Allocate and define initial PARMAREA.
NP3PBASE
6.2
3 Open the SYS1.PARMLIB data set.
NP3PMLIB
6.3
4 Analyze and create table from system parameters specified in IEASYSXX member of PARMLIB.
NP3SYSP
6.4
PLIBTAB
5 Merge OPERTAB and PLIBTAB into PLIBTAB.
NP3PLMRG
6 Initialize PARMTAB in PARMAREA.
NP3PTAB
6.5
PARMAREA
7 Open SYS1.LINKLIB.
NP3LKLIB
8 Define SYS1.LINKLIB concatenations.
NP3LCAT
6.6
IEAVNIPM Diagram 3.0, Step 4

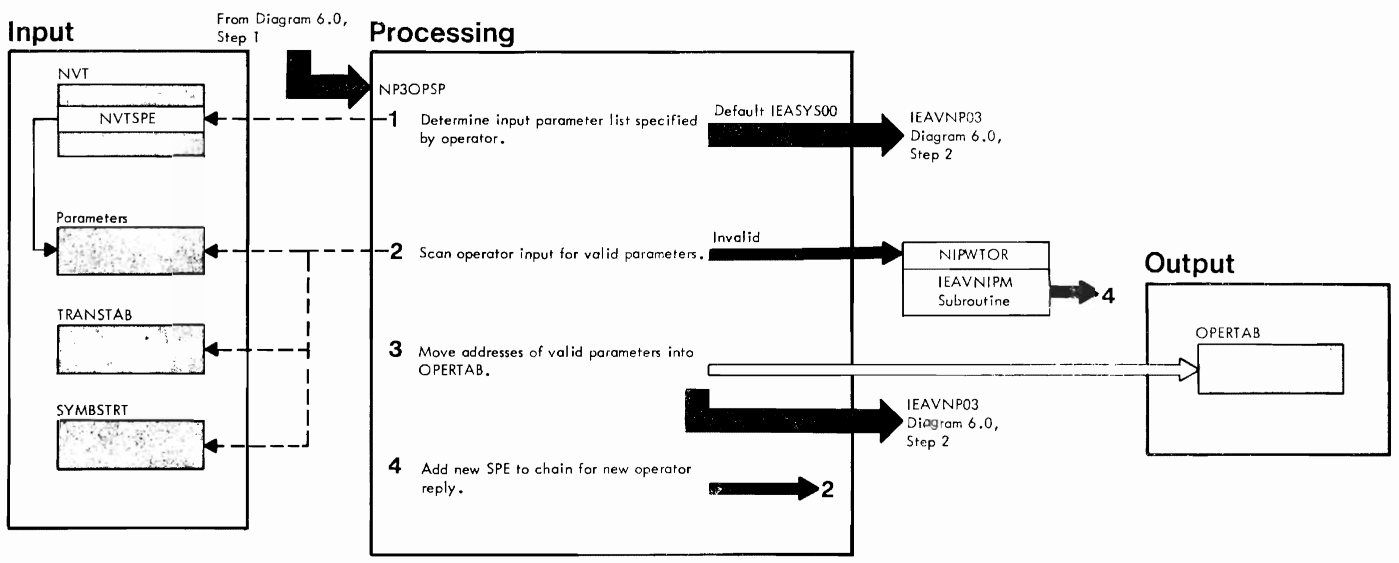

Input
From Diagram 6.0, Step 1
Processing
NVT
NVTSPE
Parameters
TRANSTAB
SYMBSTRT
NP3OPSP
1 Determine input parameter list specified by operator.
Default IEASYS00
IEAVNP03 Diagram 6.0, Step 2
2 Scan operator input for valid parameters.
Invalid
NIPWTOR
IEAVNIPM Subroutine
4
Output
3 Move addresses of valid parameters into OPERTAB.
OPERTAB
IEAVNP03 Diagram 6.0, Step 2
4 Add new SPE to chain for new operator reply.
2

Diagram 6.1: NP3OPSP (IEAVNP03)

Notes on Processing	Label
1. If the first operator-specified parameter is "U" or "U,L" the system parameter list IEASYS00 is selected as the source for all parameter input.	NP3OPSP
2. IEAVNP03 uses two tables within IEAVNP03 to verify the parameters. TRANSTAB contains an index value for each character which may validly begin a parameter specification. This index value is used as a displacement into SYMBSTRT. SYMBSTRT contains all of the valid parameters that may be specified by the operator. Each parameter entry is in character format and is followed by a hexadecimal constant indicating the number of valid characters (including the equal sign where necessary) in the parameter. The value from TRANSTAB indicates the position within SYMBSTRT where the comparison of the operator-specified parameter with valid parameters begins.	NP3SCAN
The operator is notified of errors, and is allowed to respecify invalid parameters.	NP3SCINV NP3OPRES
3. The TRANSTAB index value is used as the displacement into OPERTAB at which the address of the parameter is stored.	NP3ENTER
4. An eight-byte SPE is added to the chain of SPEs that contain operator-specified parameters. The address of the operator reply (to respecify or cancel) is moved into the new SPE.	NP3GETMN

Diagram 6.2: NP3PBASE
Allocate and Initialize PARMAREA

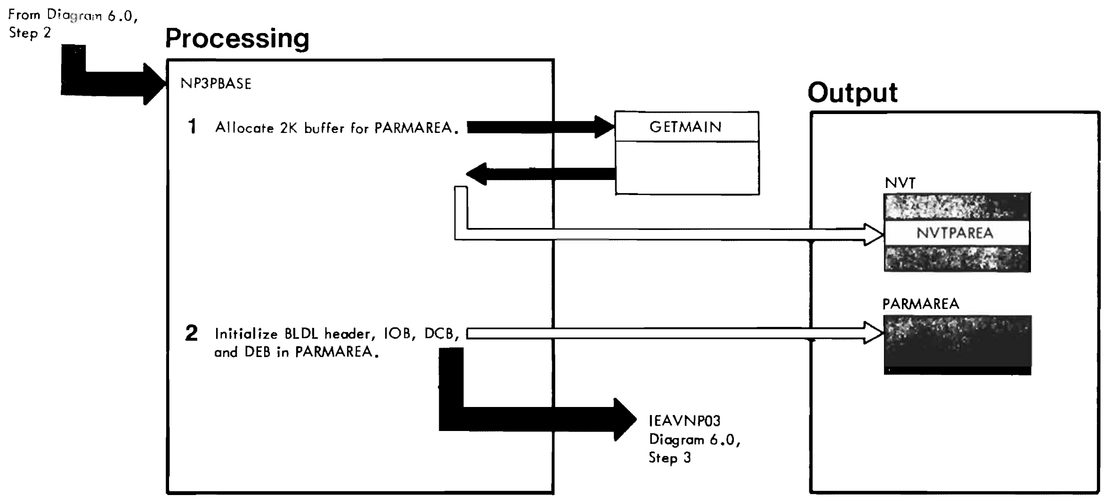
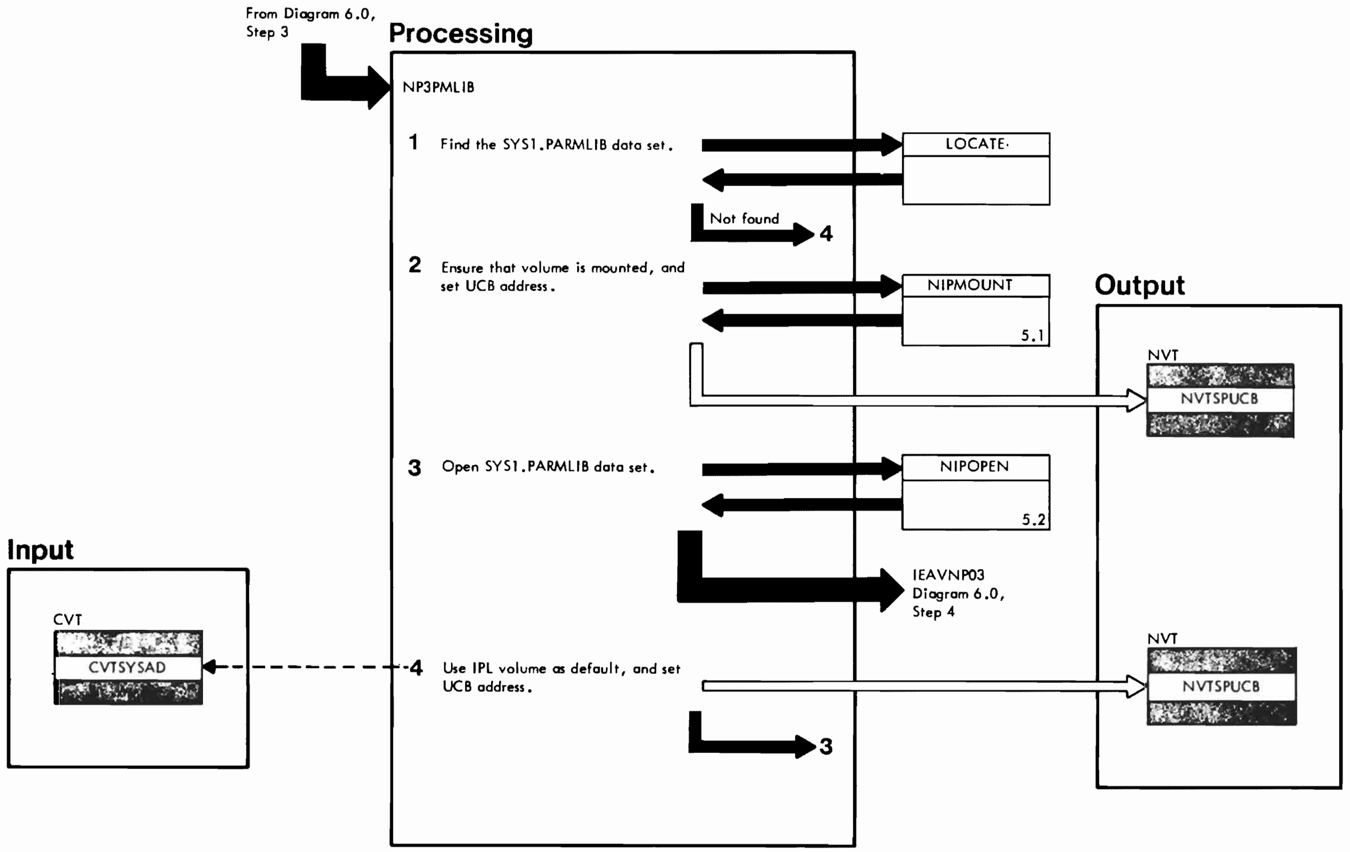

From Diagram 6.0, Step 2
Processing
NP3PBASE
1 Allocate 2K buffer for PARMAREA.
GETMAIN
2 Initialize BLDL header, IOB, DCB, and DEB in PARMAREA.
IEAVNP03 Diagram 6.0, Step 3
Output
NVT
NVTPAREA
PARMAREA

Diagram 6.2: NP3PBASE (IEAVNP03)

Notes on Processing	Label
1. The PARMAREA will contain the operator-specified parameters pointed to by OPERTAB and the parameters specified via SYSP and pointed to by PLIBTAB. These parameters will be merged by NP3PLMRG.	NP3PBASE

Diagram 6.3: NP3PMLIB
Open SYS1.PARMLIB Data Set



1 Find the SYS1.PARMLIB data set.

2 Ensure that volume is mounted, and set UCB address.

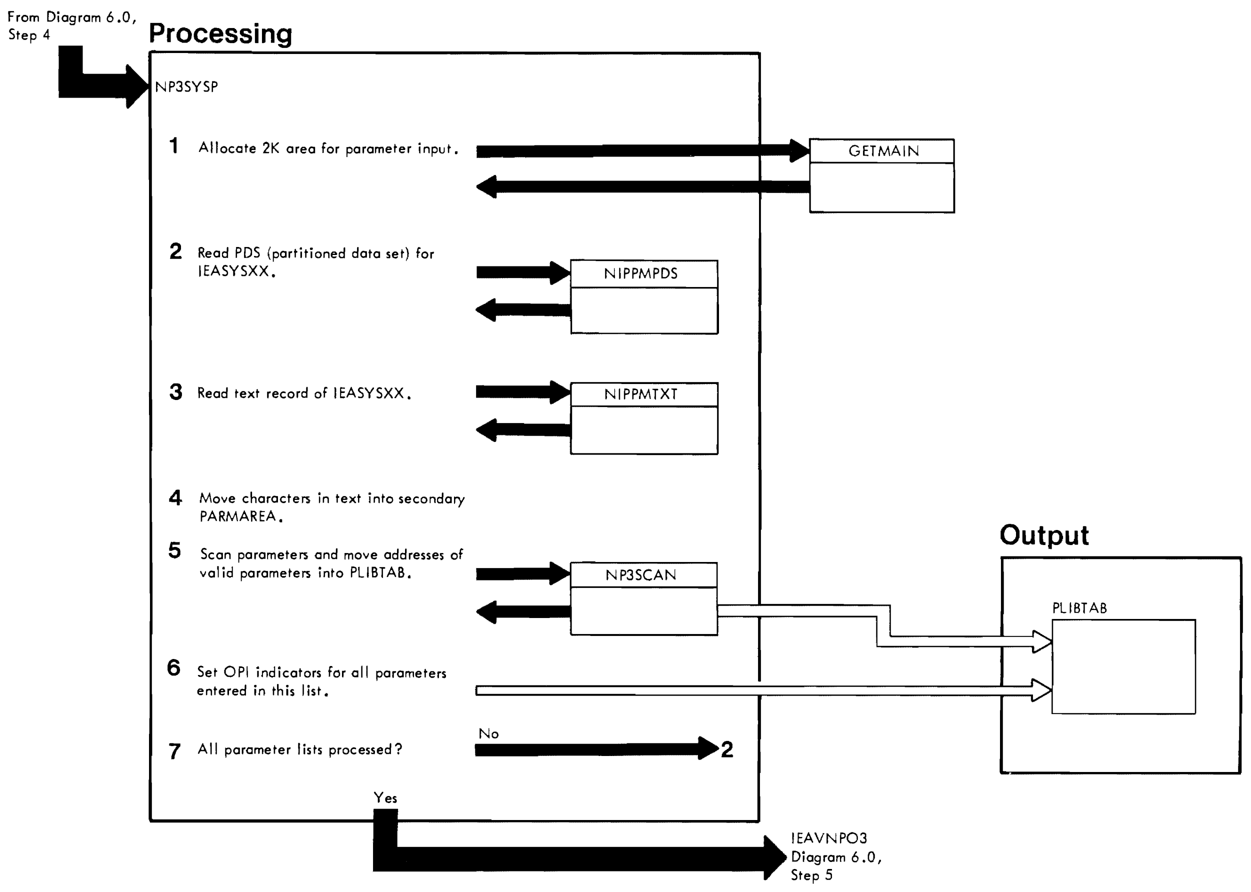
3 Open SYS1.PARMLIB data set.

4 Use IPL volume as default, and set UCB address.

Diagram 6.3: NP3PMLIB (IEAVNP03)

Notes on Processing	Label
1.	NP3PMLIB
4.	NP3PMDEF

Diagram 6.4: NP3SYSP Build the PLIBTAB



1 Allocate 2K area for parameter input.

2 Read PDS (partitioned data set) for IEASYSXX.

3 Read text record of IEASYSXX.

4 Move characters in text into secondary PARMAREA.

5 Scan parameters and move addresses of valid parameters into PLIBTAB.

6 Set OPI indicators for all parameters entered in this list.

7 All parameter lists processed?

Diagram 6.4: NP3SYSP (IEAVNP03)

Notes On Processing	Label
1. The secondary PARMAREA is used as a buffer to contain all of the input parameters from IEASYSxx members of SYS1.PARMLIB.	NP3SYSP
2. IEASYS00 is always read first. Steps 2 through 6 are executed in a loop for the specified input strings.	NP3MOVNM
3. The text of the input string is read into an 80-byte buffer by a channel program within the initial PARMAREA.	NP3SYTXT
4. The text is moved, one character at a time, into the secondary buffer. An additional secondary buffer is obtained if the present buffer becomes filled.	NP3MOVST
5. The subroutine NP3SCAN moves the addresses of the valid parameters into the PLIBTAB within IEAVNP03. If duplicate specifications are encountered, the PLIBTAB entry for the duplicate parameter contains the address of the last-examined specification for that entry.	NP3SYSFT

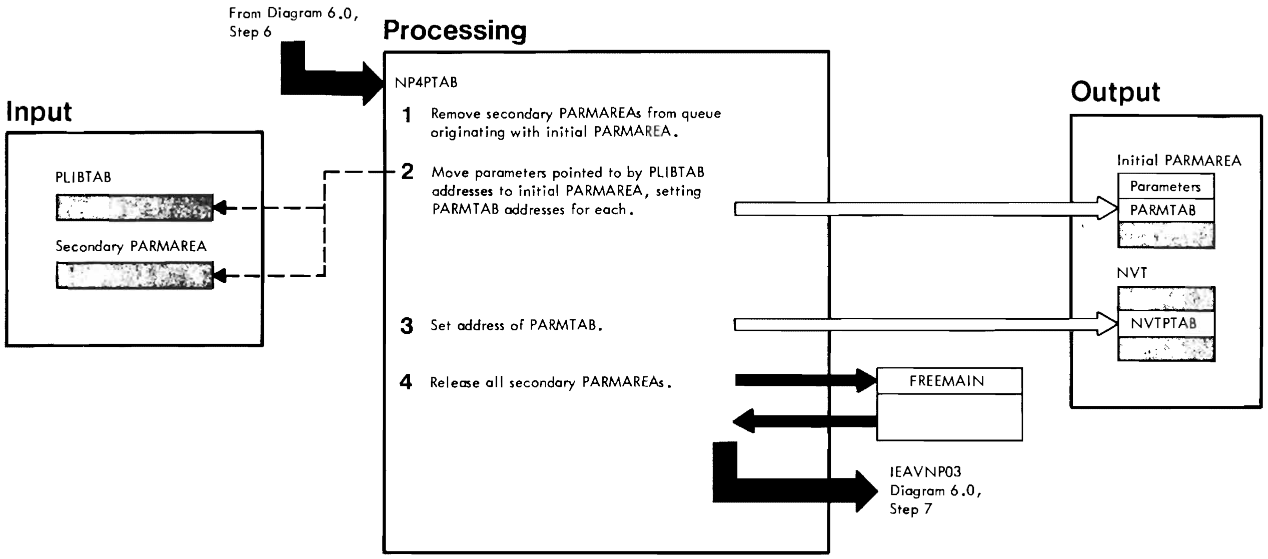

From Diagram 6.0, Step 6
Processing
NP4PTAB
1 Remove secondary PARMAREAs from queue originating with initial PARMAREA.
2 Move parameters pointed to by PLIBTAB addresses to initial PARMAREA, setting PARMTAB addresses for each.
3 Set address of PARMTAB.
4 Release all secondary PARMAREAs.
Input
PLIBTAB
Secondary PARMAREA
Output
Initial PARMAREA
Parameters
PARMTAB
NVT
NVTPTAB
FREEMAIN
IEAVNP03 Diagram 6.0, Step 7

Diagram 6.5: NP3PTAB (IEAVNP03)

Notes on Processing	Label
1. The secondary PARMAREAs are removed so that the initial PARMAREA can be expanded as necessary. The PARMAREAs are chained so that any additional PARMAREAs obtained are queued to the last PARMAREA obtained. Removing the secondary PARMAREAs leaves the initial PARMAREA as the last one obtained.	NP3PTAB
2.	NP3NULL
3.	NP3TINT
4.	NP3PAFM

From Diagram 6.0, Step 8

Processing

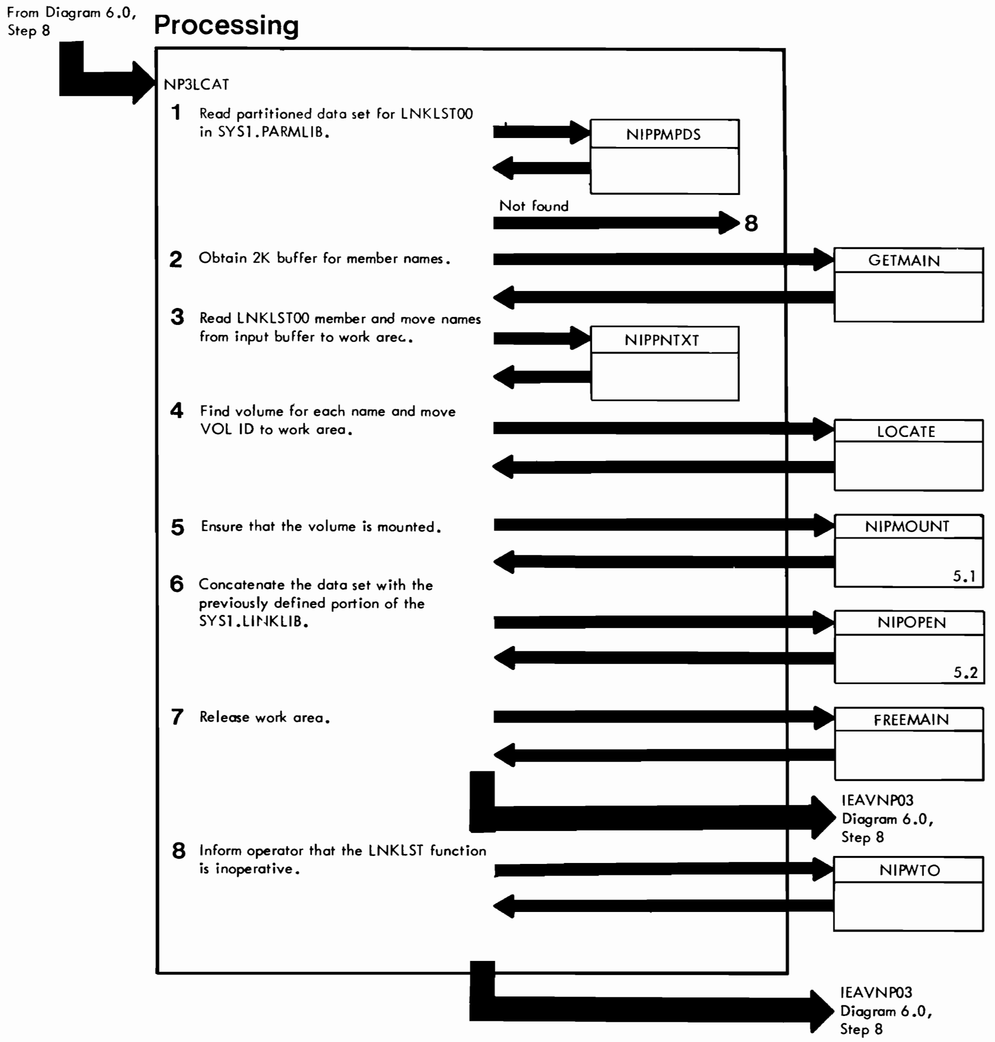
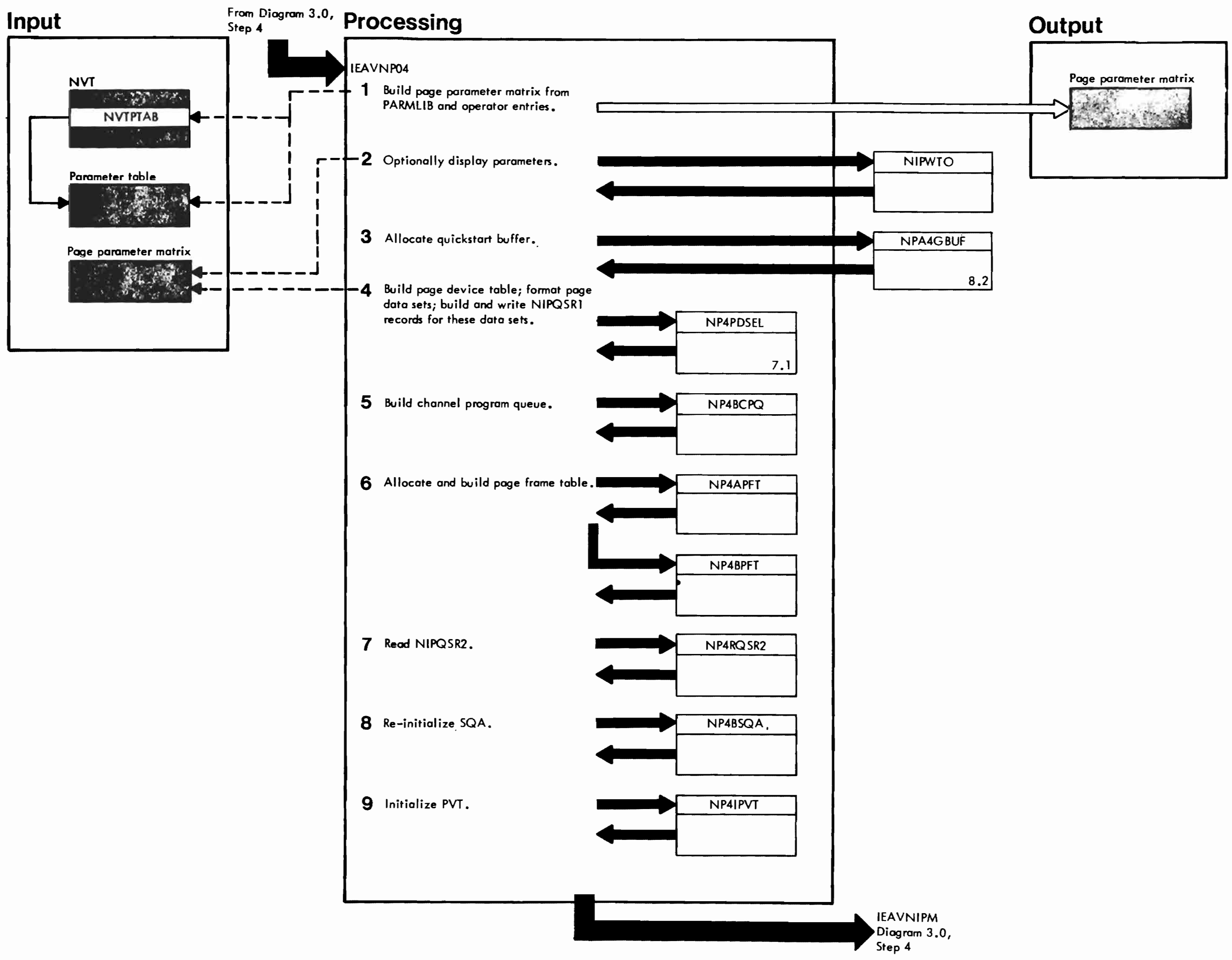

NP3LCAT
1 Read partitioned data set for LNKLST00 in SYS1.PARMLIB.
NIPPMPDS
Not found
8
2 Obtain 2K buffer for member names.
GETMAIN
3 Read LNKLST00 member and move names from input buffer to work area.
NIPPNTXT
4 Find volume for each name and move VOL ID to work area.
LOCATE
5 Ensure that the volume is mounted.
NIPMOUNT
5.1
6 Concatenate the data set with the previously defined portion of the SYS1.LINKLIB.
NIPOPEN
5.2
7 Release work area.
FREEMAIN
IEAVNP03 Diagram 6.0, Step 8
8 Inform operator that the LNKLST function is inoperative.
NIPWTO
IEAVNP03 Diagram 6.0, Step 8

Diagram 6.6: NP3LCAT (IEAVNP03)

Notes on Processing	Label
2. The 2K buffer is assumed to be large enough to contain all of the member names found in LNKLST00. If it is too small, the member name list is truncated following the last name entered in the 2K buffer.	NP3LCAT
3. The first byte of the name in the work area is a flag byte, the next 44 bytes are the name of the member, padded to the right with blanks if necessary, and the last 6 bytes contain the VOL ID.	SCANRCD NP3LCNAM
4. Steps 4 through 6 are executed in a loop for each member name placed in the work area, or until 15 members are successfully concatenated.	NP3LCLOC
6.	NP3LCVOL
7.	NP3LCEND
8.	NP3LCMFL

Diagram 7.0: IEAVNP04
Open and Format Page Data Sets



1 Build page parameter matrix from PARMLIB and operator entries.

2 Optionally display parameters.

3 Allocate quickstart buffer.

4 Build page device table; format page data sets; build and write NIPQSR1 records for these data sets.

5 Build channel program queue.

6 Allocate and build page frame table.

7 Read NIPQSR2.

8 Re-initialize SQA.

9 Initialize PVT.

Diagram 7.0: IEAVNP04

Notes on Processing	Label
1. All entries from the IEASYSxx list of SYS1.PARMLIB are scanned first. Operator-specified entries are entered next, overlaying (if permissible) previous duplicate entries. If no entries are found in either list, the operator is requested to enter PAGE parameters via NIPWTOR. If a format error is encountered, the PAGE parameters are truncated with the last valid parameter specification, and the operator is requested to respecify the remainder of the PAGE parameters or accept the truncated PAGE definition.	IEAVNP04 NP4PSCAN NP4PINV
2. If any parameters are displayed, the operator is allowed to respecify any of the PAGE parameters.	NP4PDOPT
3. The quickstart buffer is used for reading and writing the quickstart records (NIPQSR1, NIPQSR2, and NIPQSR3). A NIPQSR1 will be created for each data set formatted by NP4PDSEL (Step 4).	NP4PDEV
4. NP4PDSEL formats only the page data sets that have not been formatted during a previous entry to NIP, or that have been respecified.	NP4PDSEL
5. The channel program queue is built in the high end of the nucleus buffer and contains ten or more skeleton channel programs. The number of channel programs (entries) is determined as follows: 1 paging device = 10 CPQEs; more than one paging device = 15 CPQEs; plus the number of operator-specified CPQEs.	NP4BCPQ
6. The area for the page frame table is allocated above the channel program queue.	NP4APFT
7. If the NVTFLQS flag has been set by NP4PDSEL, the quickstart buffer is refreshed with NIPQSR1 and NIPQSR2 for the LPA page data set. NIPQSR2 is required by NP4BSQA (Step 8).	NP4RQSR2
8. NP4BSQA processes the SQA parameter. If the redefined SQA size would overlap the high LPA address, and a quickstart is in process, the operator is given the choice of canceling the SQA specification or terminating the quickstart process (in which case the NVTFLQS flag is set to 0).	NP4BSQA

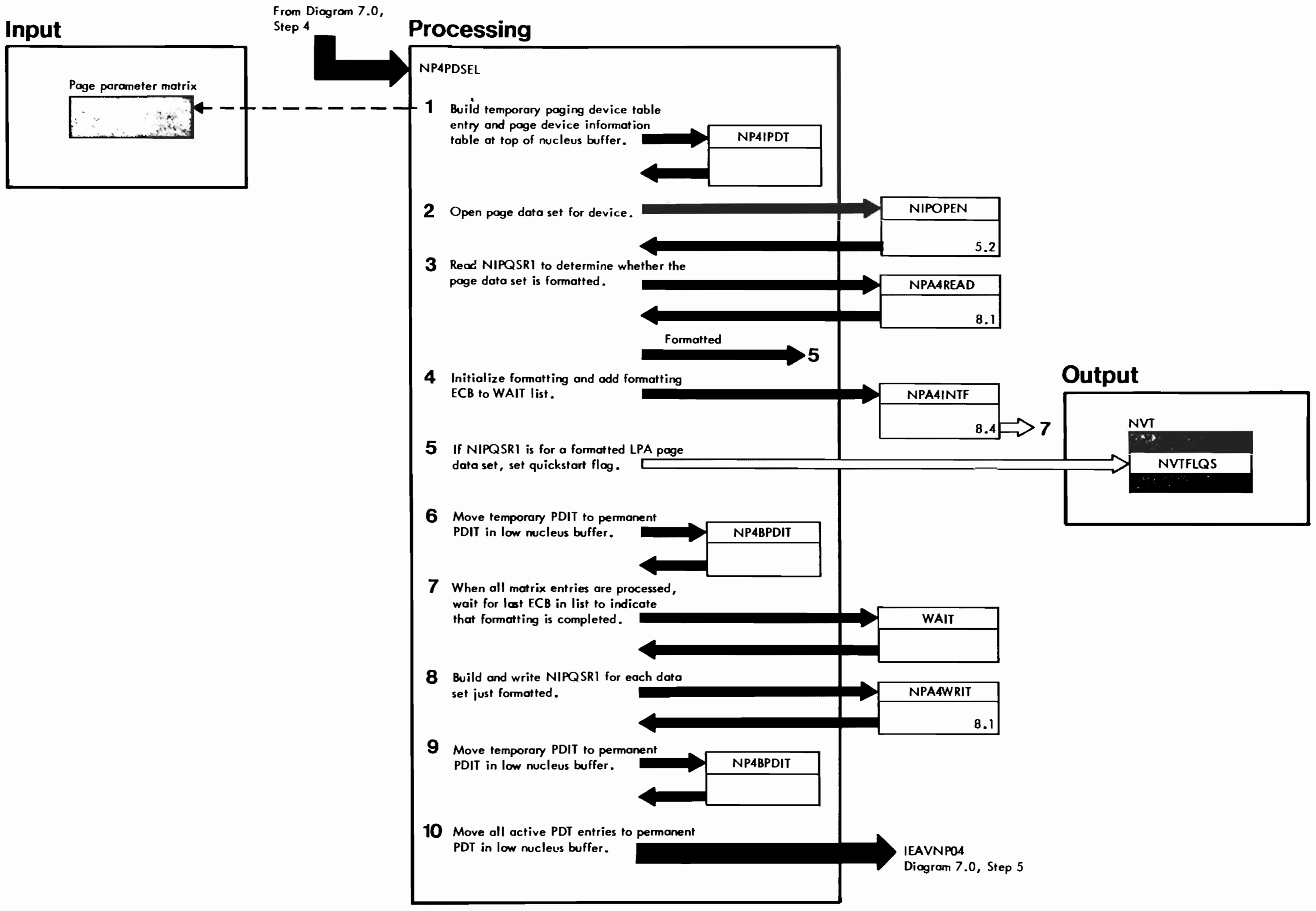
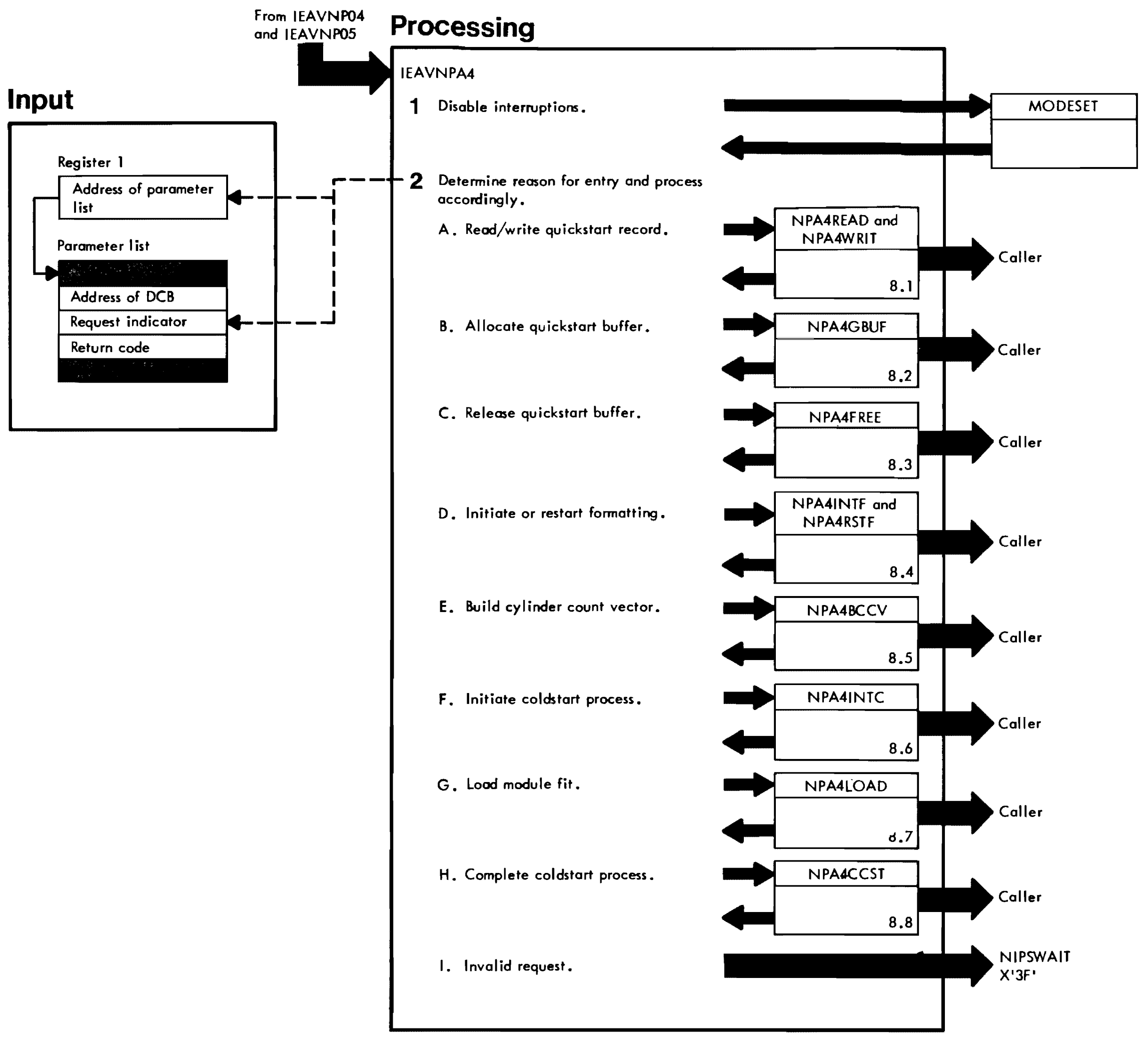

Input
Page parameter matrix
From Diagram 7.0, Step 4
Processing
NP4PDSEL
1 Build temporary paging device table entry and page device information table at top of nucleus buffer.
NP4IPDT
2 Open page data set for device.
NIPOPEN
5.2
3 Read NIPQSR1 to determine whether the page data set is formatted.
NPA4READ
8.1
Formatted
5
4 Initialize formatting and add formatting ECB to WAIT list.
NPA4INTF
8.4
7
5 If NIPQSR1 is for a formatted LPA page data set, set quickstart flag.
6 Move temporary PDIT to permanent PDIT in low nucleus buffer.
NP4BPDIT
7 When all matrix entries are processed, wait for last ECB in list to indicate that formatting is completed.
WAIT
8 Build and write NIPQSR1 for each data set just formatted.
NPA4WRIT
8.1
9 Move temporary PDIT to permanent PDIT in low nucleus buffer.
NP4BPDIT
10 Move all active PDT entries to permanent PDT in low nucleus buffer.
IEAVNP04
Diagram 7.0, Step 5
Output
NVT
NVTFLQS

Diagram 7.1: NP4PDSEL (IEAVNP04)

Notes on Processing	Label
The first six steps are executed in a loop, processing each page data set specified in the PAGE parameter matrix created by IEAVNP04 (Step 1).	NP4PDSEL
1. NP4PDSEL verifies the parameter specifications; the operator is allowed to respecify invalid devices. The PDIT and PDTE for the data set are created at the upper end of the nucleus because the permanent PDIT will contain only active entries, and the number of these will not be known until all data sets have been processed.	NP4PDTI
2. If the data set cannot be found by NIPOPEN, the NP4ALLOC subroutine attempts to allocate the page data set. If the data set cannot be allocated, the operator is requested to respecify the parameter or cancel the specification.	NP4POPEN
3. An NIPQSR1 record exists for each data set that has been formatted during a previous entry to IPL/NIP.	NP4RQSR1
4.	NP4BPBAL
5. The NVTFLQS flag is set if coldstart processing was not specified (CLPA parameter) or if the "F" parameter was not specified in the PAGE parameter.	NP4RQSR1
8.	NP4BBQS1
10.	NP4PDEND



1 Disable interruptions.

2 Determine reason for entry and process accordingly.

A. Read/write quickstart record.

B. Allocate quickstart buffer.

C. Release quickstart buffer.

D. Initiate or restart formatting.

E. Build cylinder count vector.

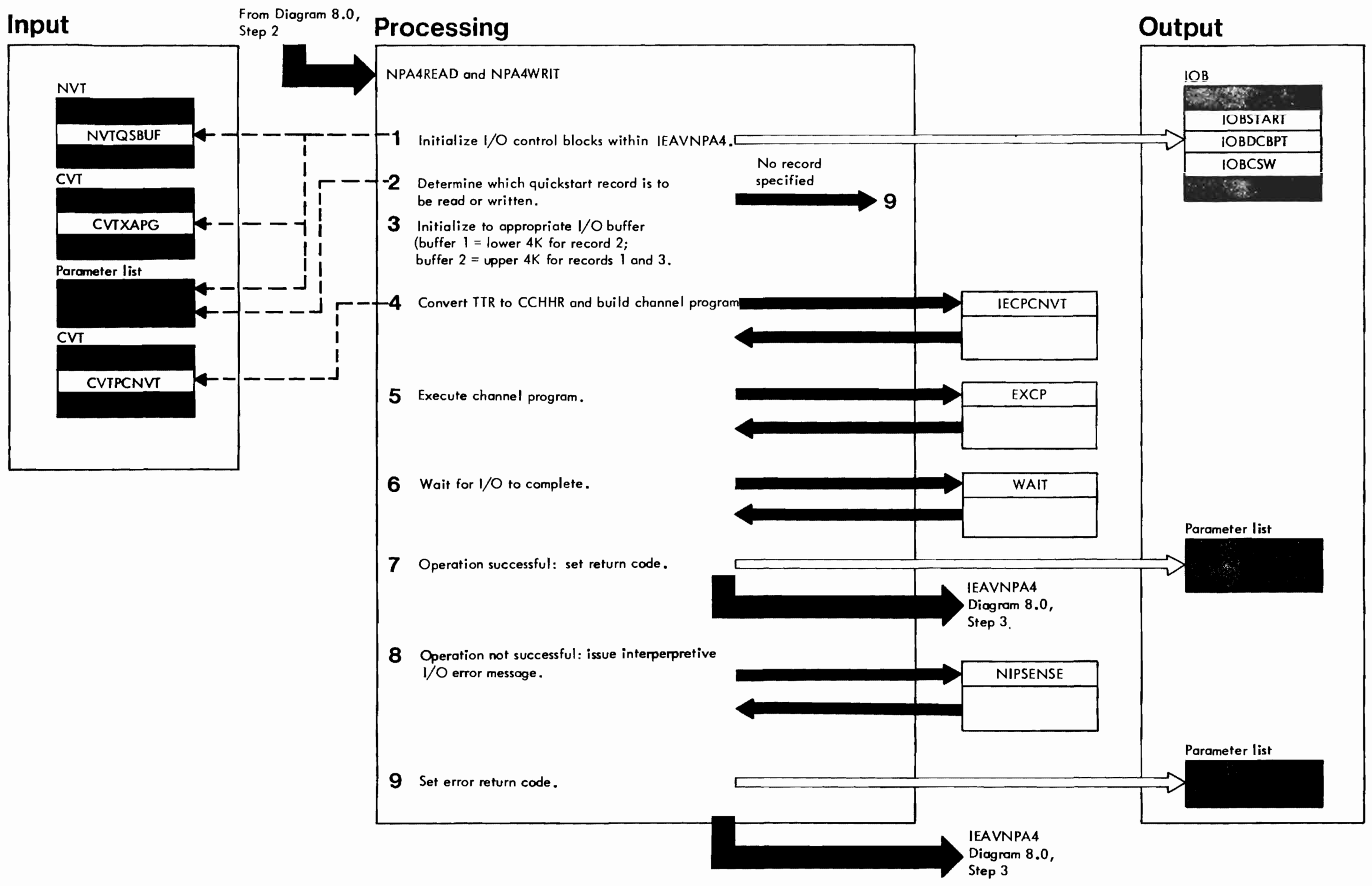
F. Initiate coldstart process.

G. Load module fit.

H. Complete coldstart process.

I. Invalid request.

Diagram 8.1: NPA4READ and NPA4WRIT Read and Write Quickstart Records



1 Initialize I/O control blocks within IEAVNPA4.

2 Determine which quickstart record is to be read or written.

3 Initialize to appropriate I/O buffer (buffer 1 = lower 4K for record 2; buffer 2 = upper 4K for records 1 and 3.

4 Convert TTR to CCHHR and build channel program

5 Execute channel program.

6 Wait for I/O to complete.

7 Operation successful: set return code.

8 Operation not successful: issue interperpretive I/O error message.

9 Set error return code.

Diagram 8.1: NPA4READ/NPA4WRIT (IEAVNPA4)

Notes on Processing	Label
1.	NPA4RDWR
3. The lower buffer (buffer 1) is pointed to by NVTQSBUF and the upper buffer (buffer 2) is equal to buffer 1 plus 4096. NIPQSR2 is read into and written from buffer 1; NIPQSR1 and NIPQSR3 are read into and written from buffer 2. because NIPQSR1 contains the TTR of NIPQSR2 and NIPQSR2 contains the TTR of NIPQSR3, a read or write request for NIPQSR2 or NIPQSR3 requires that the record (NIPQSR1 or NIPQSR2, respectively) that contains the TTR be in the other buffer.	NPA4QSR1 NPA4QSR2 NPA4QSR3
4.	NPA4QSIO
7.	NPA4EXIT
8.	NPA4FAIL
9. Error return codes are: X'00' No error X'04' Insufficient space X'08' I/O error X'0C' Insufficient DASD	

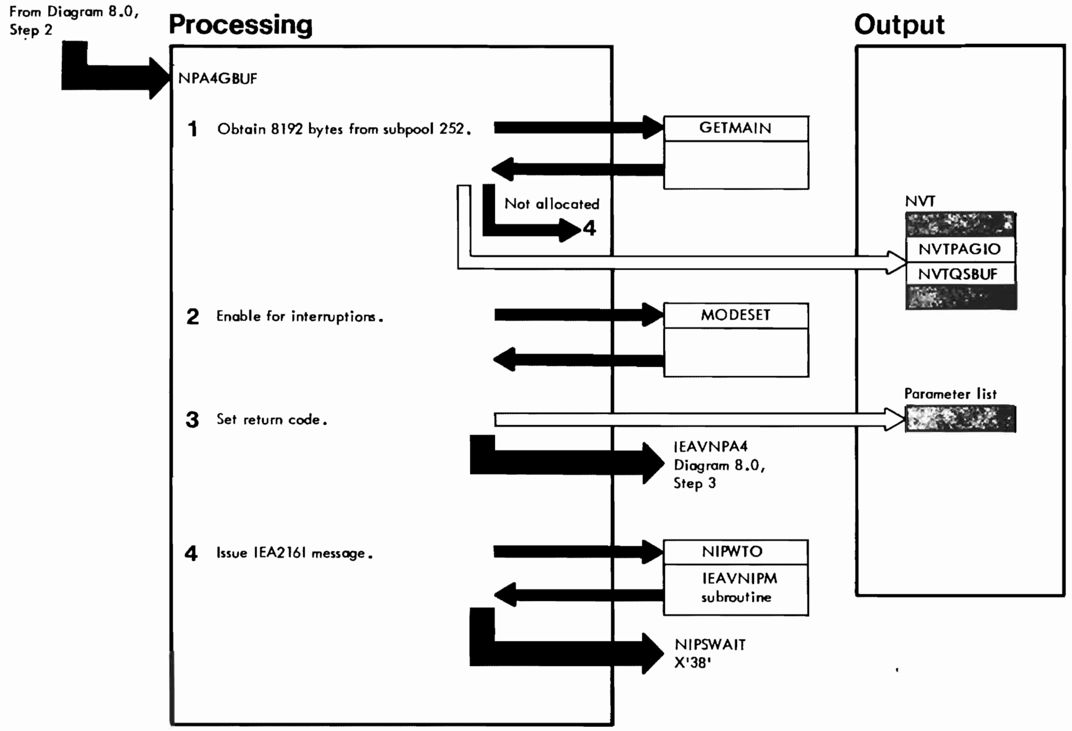

From Diagram 8.0, Step 2
Processing
Output
NPA4GBUF
1 Obtain 8192 bytes from subpool 252.
GETMAIN
Not allocated
4
NVT
NVTPAGIO
NVTQSBUF
2 Enable for interruptions.
MODESET
Parameter list
3 Set return code.
IEAVNPA4 Diagram 8.0, Step 3
4 Issue IEA216I message.
NIPWTO
IEAVNIPM subroutine
NIPSWAIT X'38'

Diagram 8.2: NPA4GBUF (IEAVNPA4)

Notes on Processing	Label
1.	NPA4GBUF
2.	NPA4PXIT
3.	NPA4NXIT
4.	NPA4NCOR

Diagram 8.3: NPA4FREE
Free Quickstart Buffer

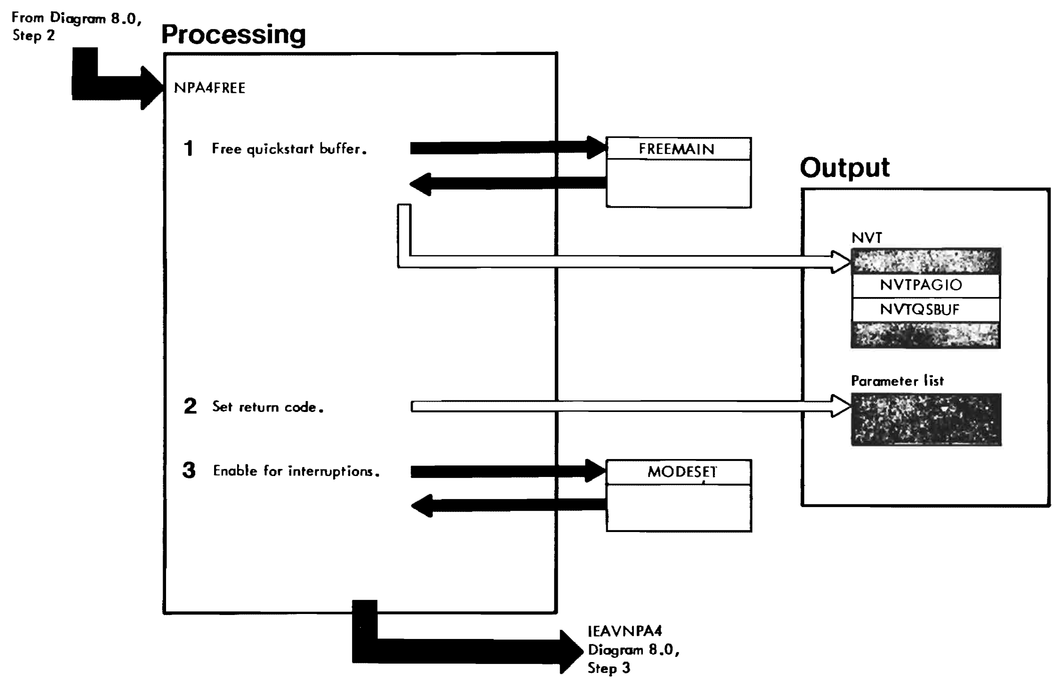


Diagram 8.3: NPA4FREE

Notes on Processing	Label
1.	NPA4FREE
2.	NPA4NXIT
3.	NPA4PXIT

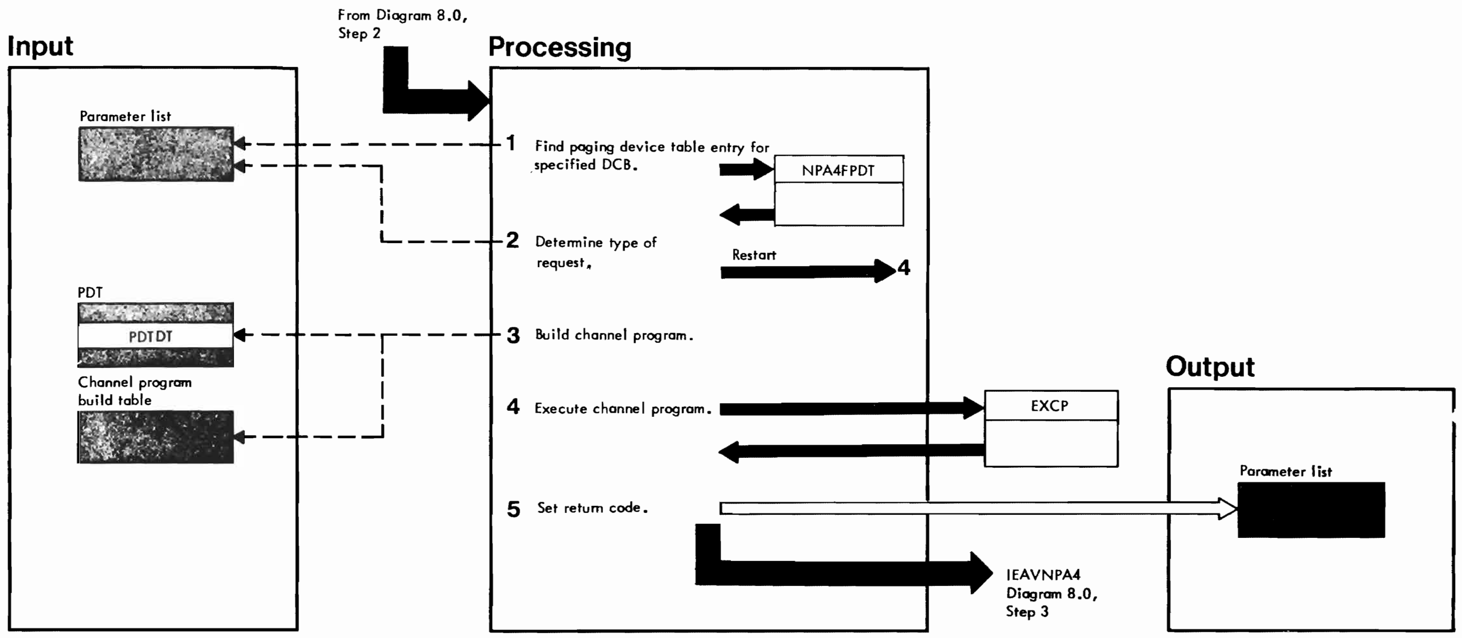

From Diagram 8.0, Step 2
Input
Processing
Output
Parameter list
PDT
PDTDT
Channel program build table
1 Find paging device table entry for specified DCB.
NPA4FPDT
2 Determine type of request.
Restart
4
3 Build channel program.
4 Execute channel program.
EXCP
5 Set return code.
Parameter list
IEAVNPA4 Diagram 8.0, Step 3

Diagram 8.4: NPA4INTF/NPA4RSTF (IEAVNPA4)

Notes on Processing	Label
This subroutine formats page data sets for which IEAVNP04 did not find a NIPQSR1 record or for which reformatting is necessary.	
1.	NPA4INTF
3. The channel program build table is within IEAVNPA4 and contains device-dependent information concerning the format of the CCW string required to write the count, key and data to the paging devices.	
4. The channel program formats one cylinder of the paging device.	NPA4EXCP
5. The return code X'04' indicates that the channel program was too long to fit in the nucleus buffer. A return code X'00' indicates that the I/O operation was initiated successfully.	NPA4AXIT NPA4NXIT

Diagram 8.5: NPA4BCCV
Determine Number of Available Pages

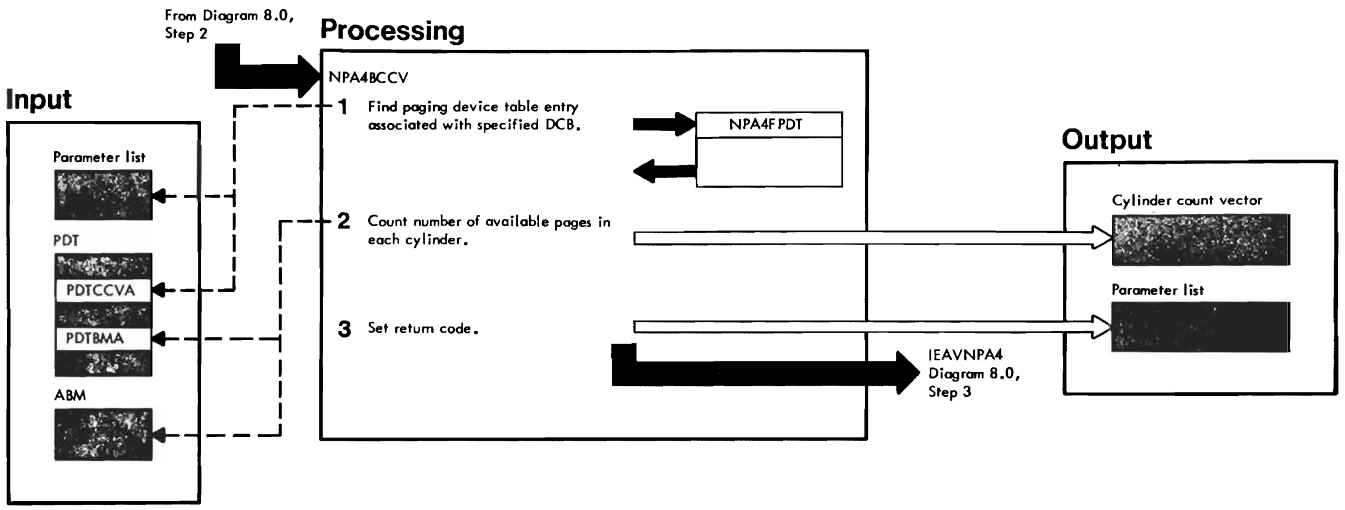


Diagram 8.5: NPA4BCCV (IEAVNPA4)

Notes on Processing	Label
2. IEAVNPA4 uses loops to examine slots and page groups within the cylinder to determine from the auxiliary bit map the number of available pages.	NPA4BC01 NPA4MXIT

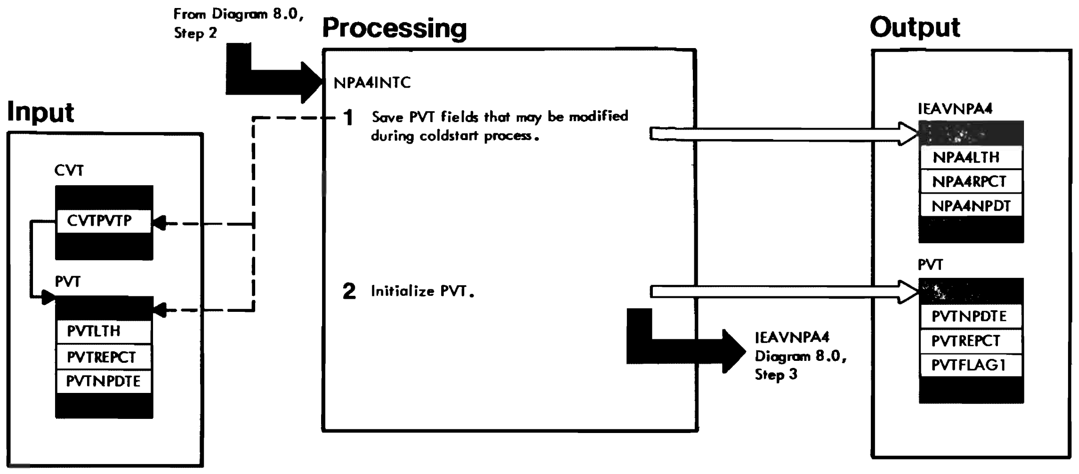

From Diagram 8.0, Step 2
Input
CVT
CVTPVTP
PVT
PVTLTH
PVTREPCT
PVTNPDTE
Processing
NPA4INTC
1 Save PVT fields that may be modified during coldstart process.
2 Initialize PVT.
IEAVNPA4 Diagram 8.0, Step 3
Output
IEAVNPA4
NPA4LTH
NPA4RPCT
NPA4NPDT
PVT
PVTNPDTE
PVTREPCT
PVTFLAG1

Diagram 8.6: NPA4INTC (IEAVNPA4)

Notes on Processing	Label
1. The field PVTNPDTE is set to one to force all page-outs to go to the paged set for the link pack area. The PVTFLAG1 field is set to prevent migration during the coldstart process.	NPA4INTC

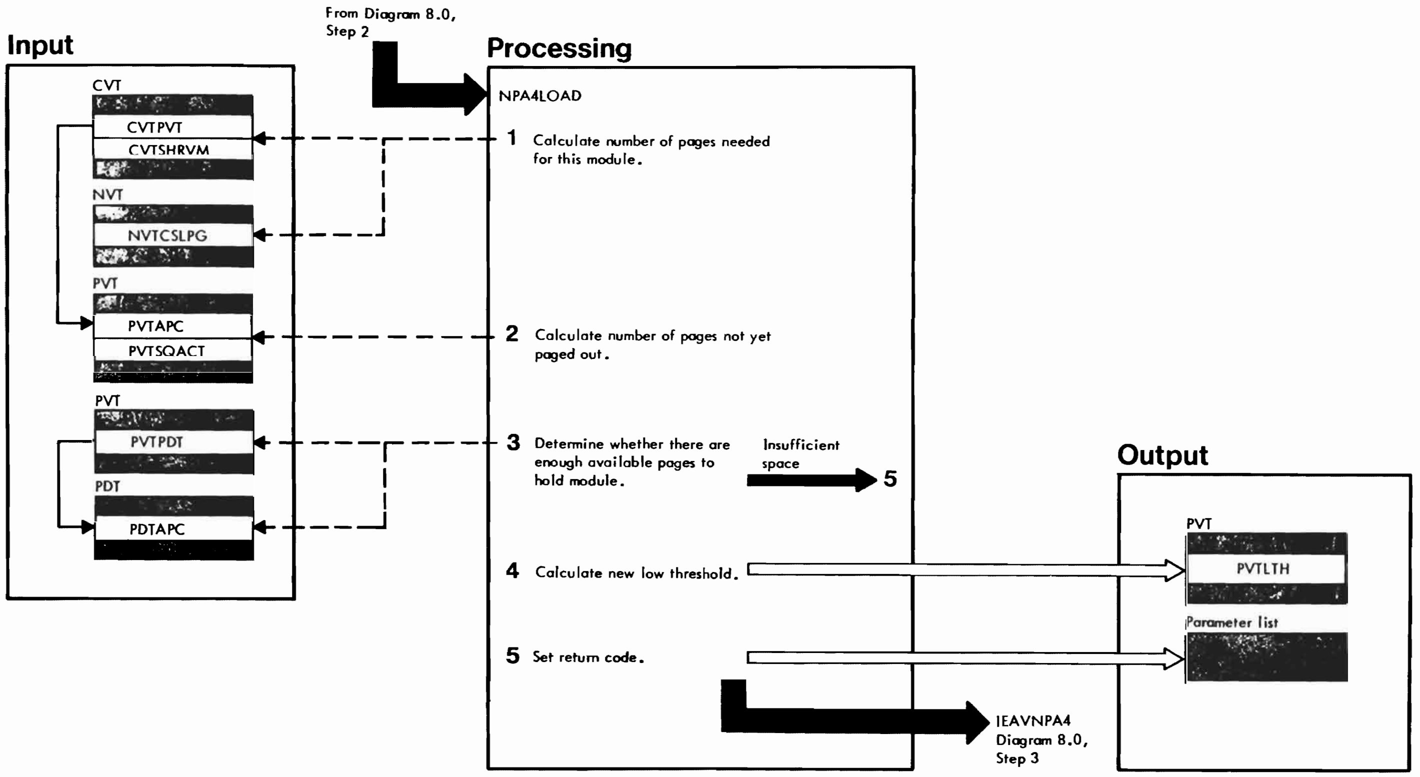

Input
CVT
CVTPVT
CVTSHRVM
NVT
NVTCSLPG
PVT
PVTAPC
PVTSQACT
PVT
PVTPDT
PDT
PDTAPC
From Diagram 8.0, Step 2
Processing
NPA4LOAD
1 Calculate number of pages needed for this module.
2 Calculate number of pages not yet paged out.
3 Determine whether there are enough available pages to hold module.
Insufficient space
5
4 Calculate new low threshold.
5 Set return code.
IEAVNPA4 Diagram 8.0, Step 3
Output
PVT
PVTLTH
Parameter list

Diagram 8.7: NPA4LOAD (IEAVNPA4)

Notes on Processing	Label
1.	NPA4LOAD
3.	NPA4LM01
5. A return code of X'0C' indicates insufficient space for the module.	NPA4AXIT NPA4NXIT

Diagram 8.8: NPA4CCST Complete LPA Coldstart Process

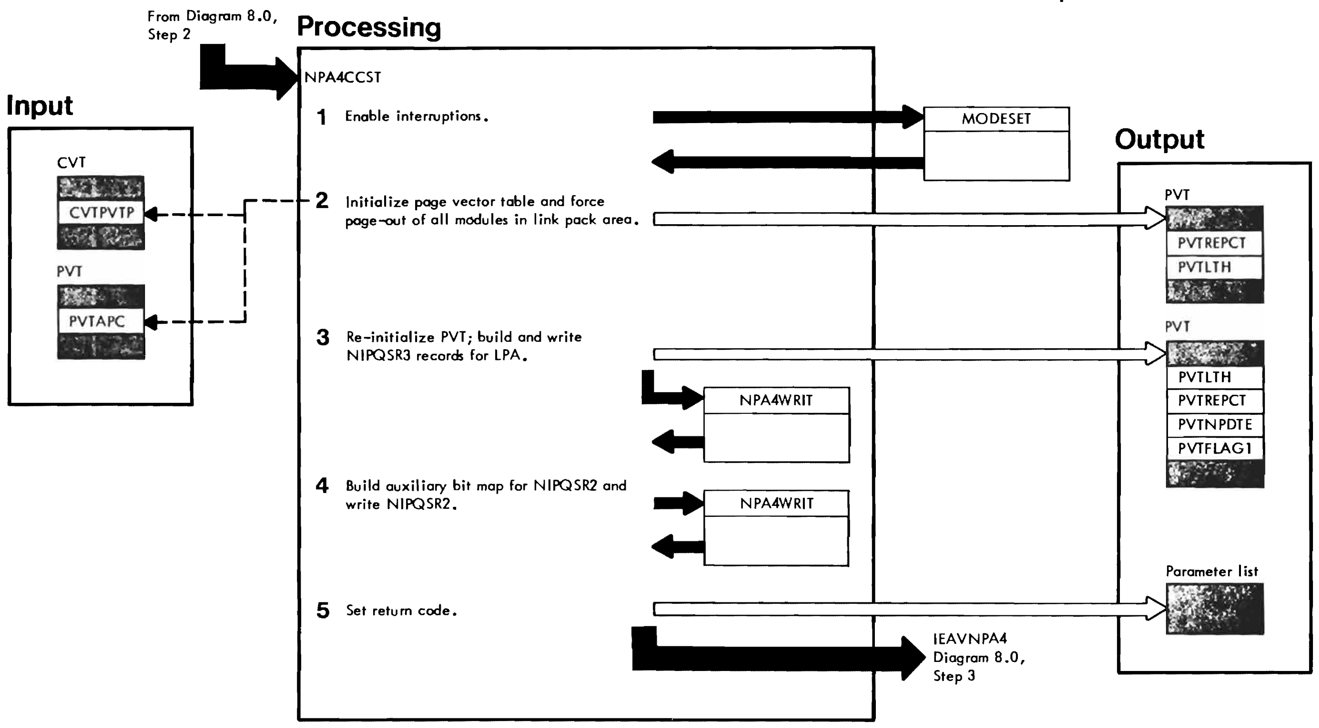
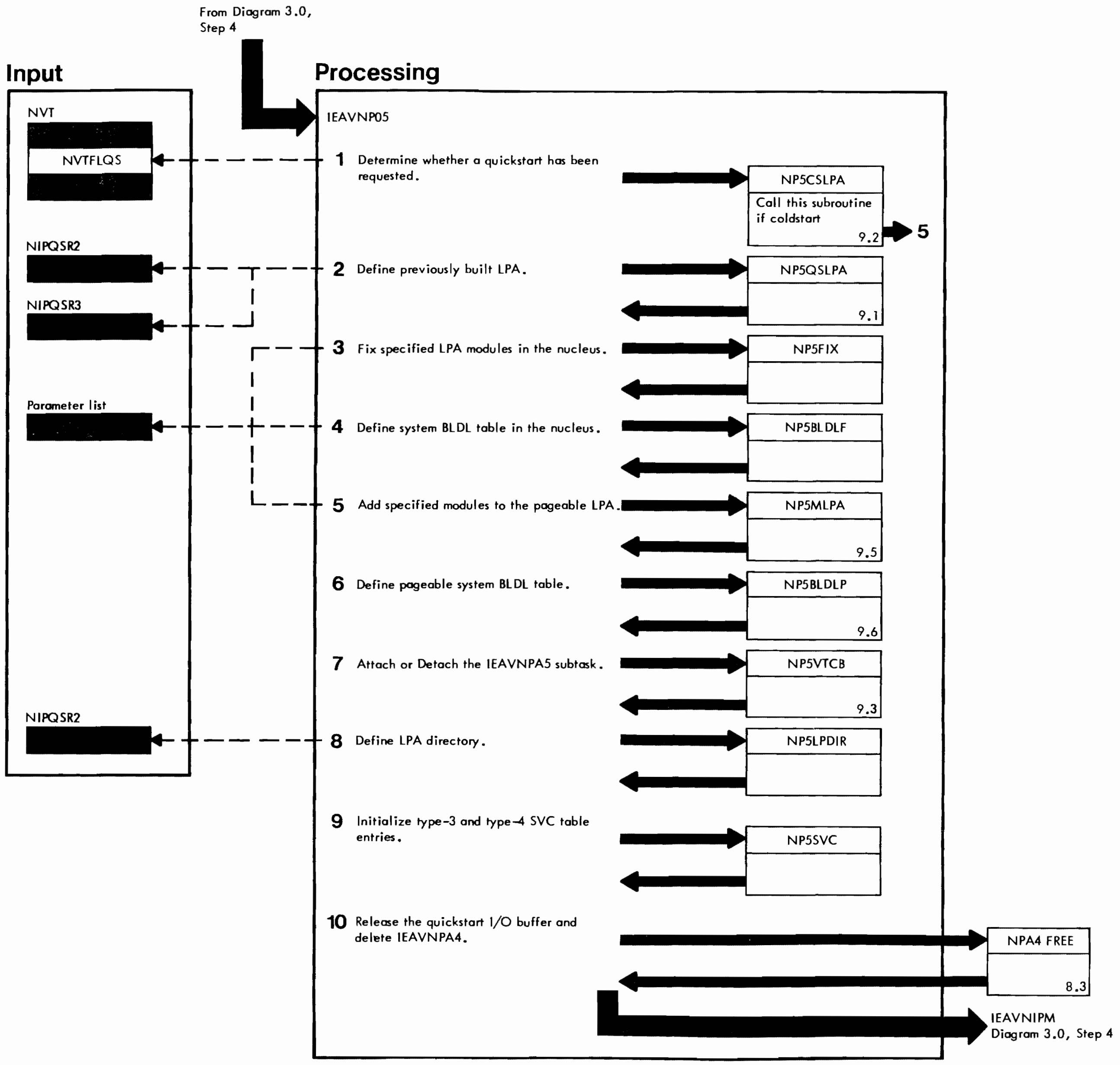

From Diagram 8.0, Step 2
Processing
Input
CVT
CVTPVTP
PVT
PVTAPC
NPA4CCST
1 Enable interruptions.
MODESET
2 Initialize page vector table and force page-out of all modules in link pack area.
3 Re-initialize PVT; build and write NIPQSR3 records for LPA.
NPA4WRIT
4 Build auxiliary bit map for NIPQSR2 and write NIPQSR2.
NPA4WRIT
5 Set return code.
IEAVNPA4 Diagram 8.0, Step 3
Output
PVT
PVTREPCT
PVTLTH
PVT
PVTLTH
PVTREPCT
PVTNPDTE
PVTFLAG1
Parameter list

Diagram 8.8: NPA4CCST (IEAVNPA4)

Notes on Processing	Label
2. The paging low threshold is set to the number of page frames available. The paging supervisor is entered by refering to a page that is not in real storage. Discovering that the low threshold has been reached (forced in Step 2), the paging supervisor initiates page replacement.	NPA4CC01
3. The PVT fields are restored to their original condition upon entry to the Initiate Coldstart subroutine (NPA4INTC).	NPA4CC02 NPA4CC03
5. A return code of X'08' indicates I/O error while reading or writing of a quickstart record by the NPA-4READ or NPA4WRIT subroutine. A return code of X'0C' indicates that no slot was found by the sub-routine NPA4FSLT.	NPA4CCRS NPA4FSLT

Diagram 9.0: IEAVNP05
Initialize Link Pack Area and BLDL Table



1 Determine whether a quickstart has been requested.

2 Define previously built LPA.

3 Fix specified LPA modules in the nucleus.

4 Define system BLDL table in the nucleus.

5 Add specified modules to the pageable LPA.

6 Define pageable system BLDL table.

7 Attach or Detach the IEAVNPA5 subtask.

8 Define LPA directory.

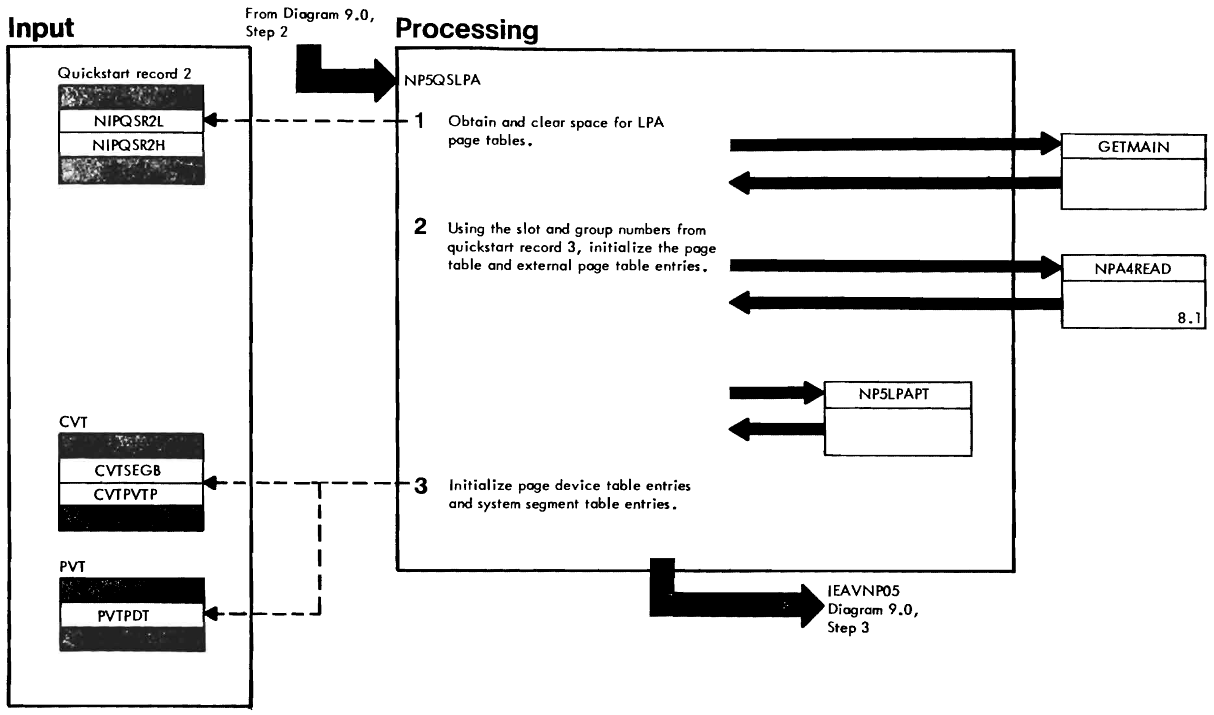
9 Initialize type-3 and type-4 SVC table entries.

10 Release the quickstart I/O buffer and delete IEAVNPA4.

Diagram 9.0: IEAVNP05

Notes on Processing	Label
1. The NVTFLQS flag was set by IEAVNP04 (label NP4RQSR1) if the LPA page data was previously formatted and the CLPA parameter was not specified. NP5CSLPA builds the LPA page data set.	IEAVNP05
2. NIPQSR2 contains the start and end virtual addresses of the LPA. NIPQSR3 contains the slot and group numbers for the records within the LPA page data set.	NP5QSTRT
10. IEAVNP05 uses the IEAPMNIP macro instruction to gain access to the NPA4FREE subroutine in IEAVNPA4. Upon return, IEAVNP05 issues an SVC 9 (DELETE) instruction to remove IEAVNPA4.	NP5RLBUF

Diagram 9.1: NP5QSLPA Initialize Table Entries



1 Obtain and clear space for LPA page tables.

2 Using the slot and group numbers from quickstart record 3, initialize the page table and external page table entries.

3 Initialize page device table entries and system segment table entries.

Diagram 9.1: NP5QSLPA (IEAVNP05)

Notes on Processing	Label
1. Space is obtained for page tables and external page tables with a GETMAIN macro instruction specifying subpool 255 of the SQA. All page table entries from the beginning of the page table up to the entry for the first page for the LPA are set invalid.	NP5QSLPA NP5PGTE
2. The quickstart 3 records are read and processed in a loop. Each record is processed by the subroutine NP5LPAPT. If the quickstart process cannot continue because an error occurred while reading a quickstart record 3 (in Step 2), execution resumes at Step 4 (9.0) for coldstart.	NP5RDQSR
3. Auxiliary page records that contain LPA pages are indicated as unavailable. The virtual addresses of LPA page tables are converted to real addresses and the segment table entries are set to indicate maximum size.	NP5LPDTE NP5LPAST

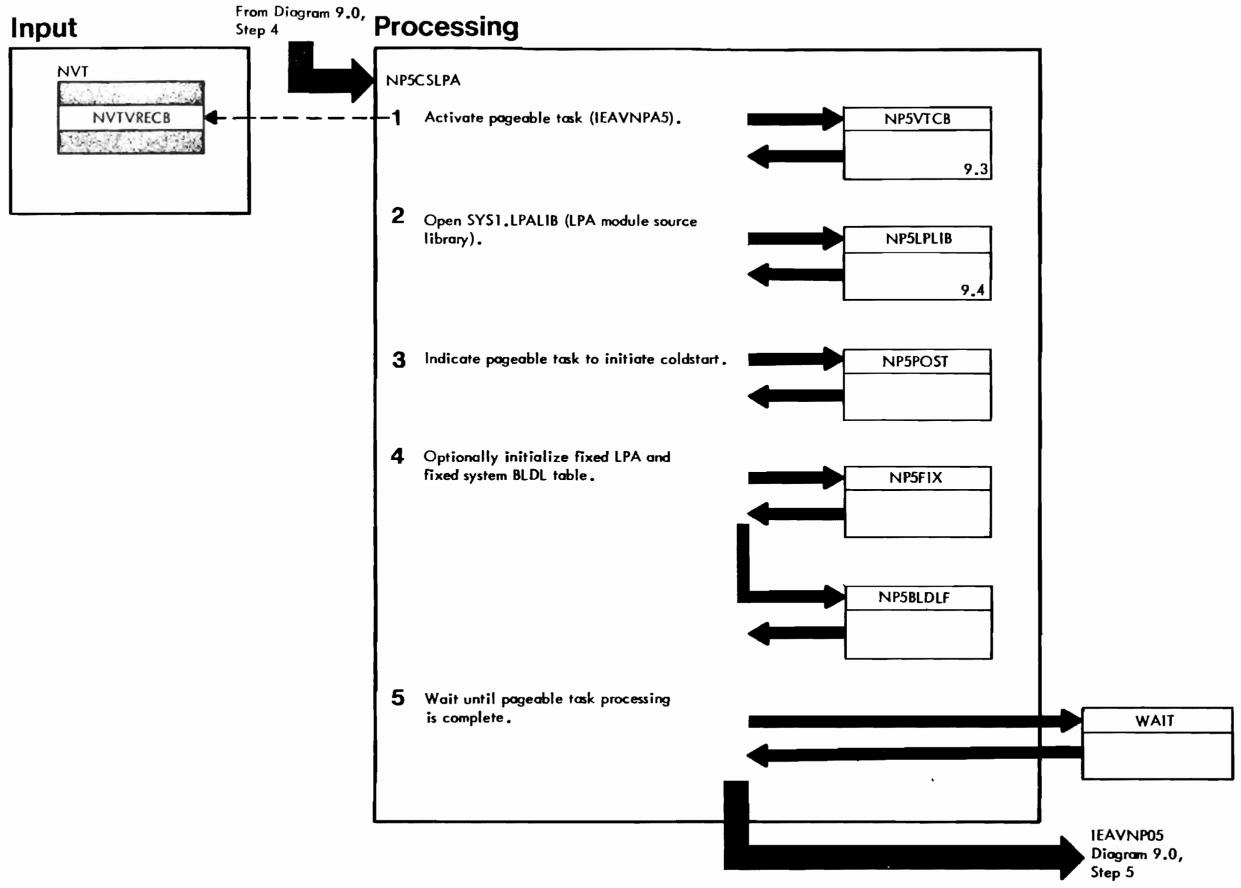
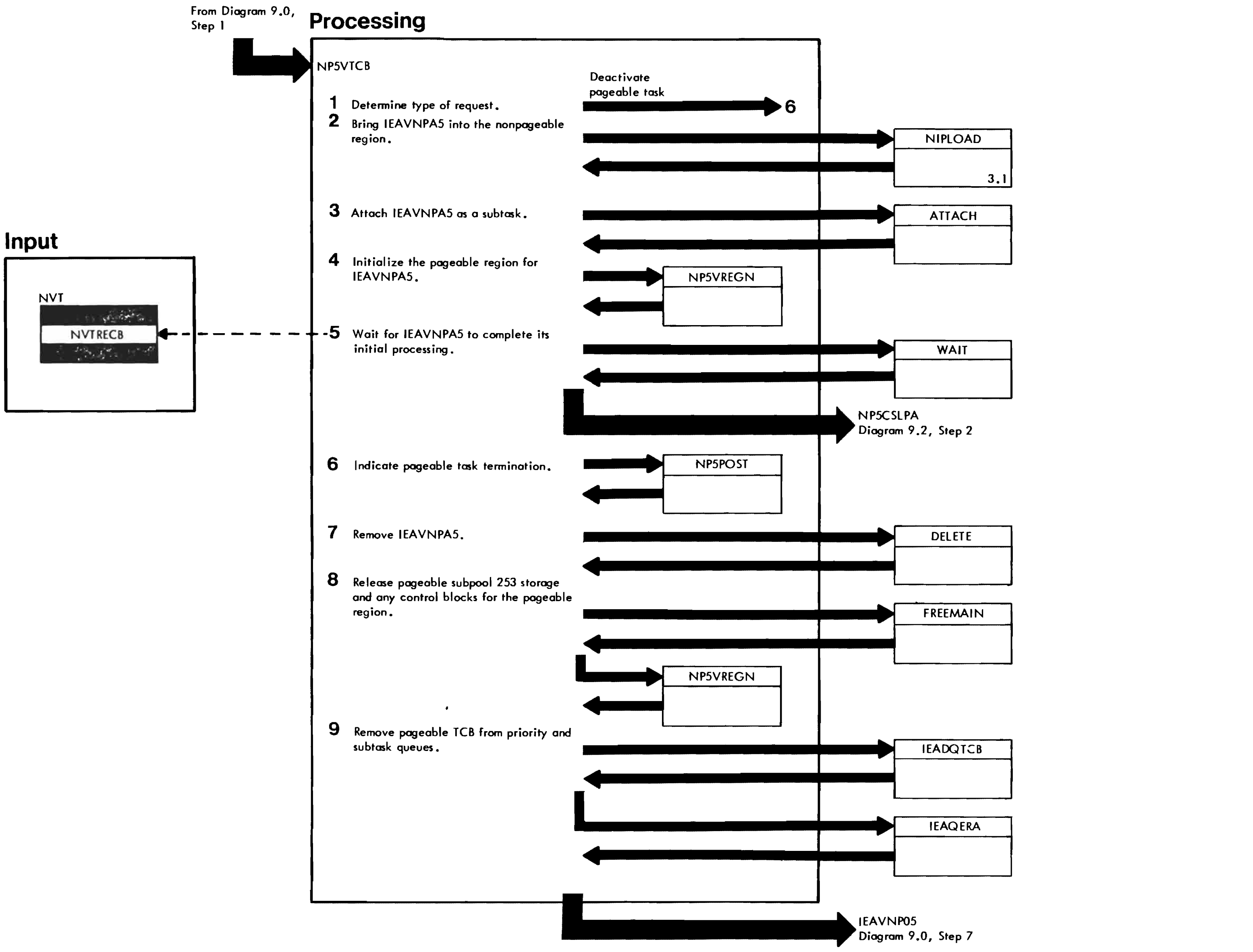

Input
NVT
NVTVRECB
From Diagram 9.0, Step 4
Processing
NP5CSLPA
1 Activate pageable task (IEAVNPA5).
NP5VTCB
9.3
2 Open SYS1.LPALIB (LPA module source library).
NP5LPLIB
9.4
3 Indicate pageable task to initiate coldstart.
NP5POST
4 Optionally initialize fixed LPA and fixed system BLDL table.
NP5FIX
NP5BLDLF
5 Wait until pageable task processing is complete.
WAIT
IEAVNP05 Diagram 9.0, Step 5

Diagram 9.2: NP5CSLPA (IEAVNP05)

Notes on Processing	Label
1. Because module IEAVNPA5 performs certain coldstart functions (and other functions common to coldstart and quickstart), IEAVNP05 attaches IEAVNPA5 as a subtask. The interaction between IEAVNP05 and IEAVNPA5 is governed by event control blocks, one for each module. When IEAVNP05 needs processing by IEAVNPA5, it posts the pageable ECB (for IEAVNPA5) and issues a WAIT macro instruction specifying its own ECB (nonpageable). When the required processing is complete, IEAVNPA5 posts the nonpageable ECB and issues a WAIT macro instruction specifying its pageable ECB.	NP5CSLPA
The initial interface with IEAVNPA5 is provided by the subroutine NP5VTCB; subsequent interfaces are provided by the subroutine NP5POST.	
2. The SYS1.LPALIB is the source of the modules to be loaded into the link pack area.	NP5LPLIB
3. See Diagram 10.3 for a description of the coldstart processing. IEAVNP05 is executed asynchronously from IEAVNPA5 until Step 5; IEAVNPA5 performs part of the coldstart process and indicates that IEAVNP05 can continue its processing asynchronously. However, IEAVNPA5 must complete its coldstart processing before subroutine NP5CSLPA returns control to mainline coding.	
NP5POST calls NP5BLMQ which returns control to NP5POST which calls NP5BLMQ again. This loop continues until NP5BLMQ indicates the end of the list to NP5POST.	
4. The NP5FIX subroutine and its subroutine NP5MLPRM loads the specified modules into subpool 251 in the nonpageable region. Both subroutine functions are optional (user requested).	NP5FIX
The NP5BLDLF subroutine and its subroutine NP5BLPRM create the fixed BLDL table in the nucleus buffer.	NP5BLDLF



1 Determine type of request.
2 Bring IEAVNPA5 into the nonpageable region.
3 Attach IEAVNPA5 as a subtask.
4 Initialize the pageable region for IEAVNPA5.
5 Wait for IEAVNPA5 to complete its initial processing.
6 Indicate pageable task termination.
7 Remove IEAVNPA5.
8 Release pageable subpool 253 storage and any control blocks for the pageable region.
9 Remove pageable TCB from priority and subtask queues.

Diagram 9.3: NP5VTCB (IEAVNP05)

Notes on Processing	Label
2. For a description of the interface provided by NP5VTCB between IEAVNP05 and IEAVNPA5, see 9.2, Notes on Processing, Step 1. Once IEAVNPA5 is activated, subsequent requests to activate IEAVNPA5 are ignored.	
6.	NP5VTDET

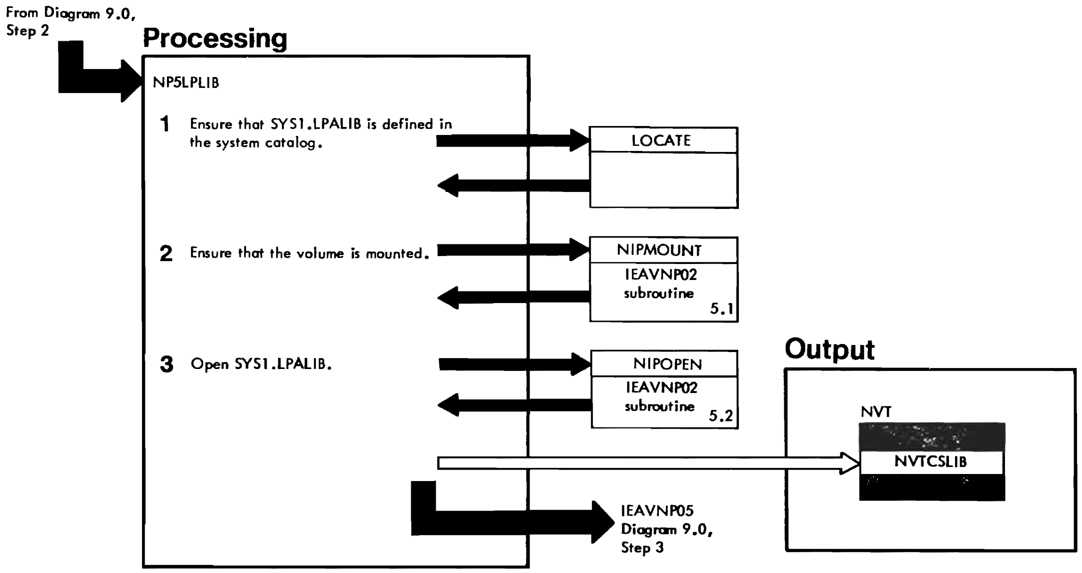

From Diagram 9.0, Step 2
Processing
NP5LPLIB
1 Ensure that SYS1.LPALIB is defined in the system catalog.
2 Ensure that the volume is mounted.
3 Open SYS1.LPALIB.
LOCATE
NIPMOUNT
IEAVNP02 subroutine 5.1
NIPOPEN
IEAVNP02 subroutine 5.2
IEAVNP05 Diagram 9.0, Step 3
Output
NVT
NVTCSLIB

Diagram 9.4: NP5LPLIB (IEAVNP05)

Notes on Processing	Label
1.	NP5LPLOC
4. The DCB address for SVC1.LPALIB is put in NVTCSLIB.	

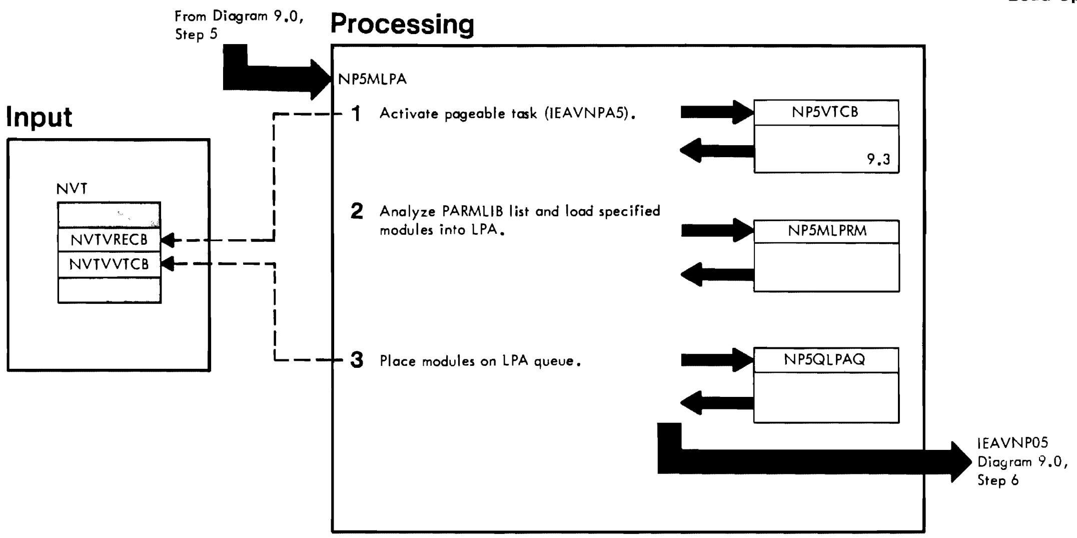

From Diagram 9.0, Step 5
Input
Processing
NVT
NVTVRECB
NVTVVTCB
NP5MLPA
1 Activate pageable task (IEAVNPA5).
2 Analyze PARMLIB list and load specified modules into LPA.
3 Place modules on LPA queue.
NP5VTCB
9.3
NP5MLPRM
NP5QLPAQ
IEAVNP05 Diagram 9.0, Step 6

Diagram 9.5 NP5MLPA (IEAVNP05)

Notes on Processing	Label
1. If IEAVNPA5 has previously been activated, this request is ignored.	NP5MLPA

Diagram 9.6: NP5BLDLP
Include Modules in BLDL Table

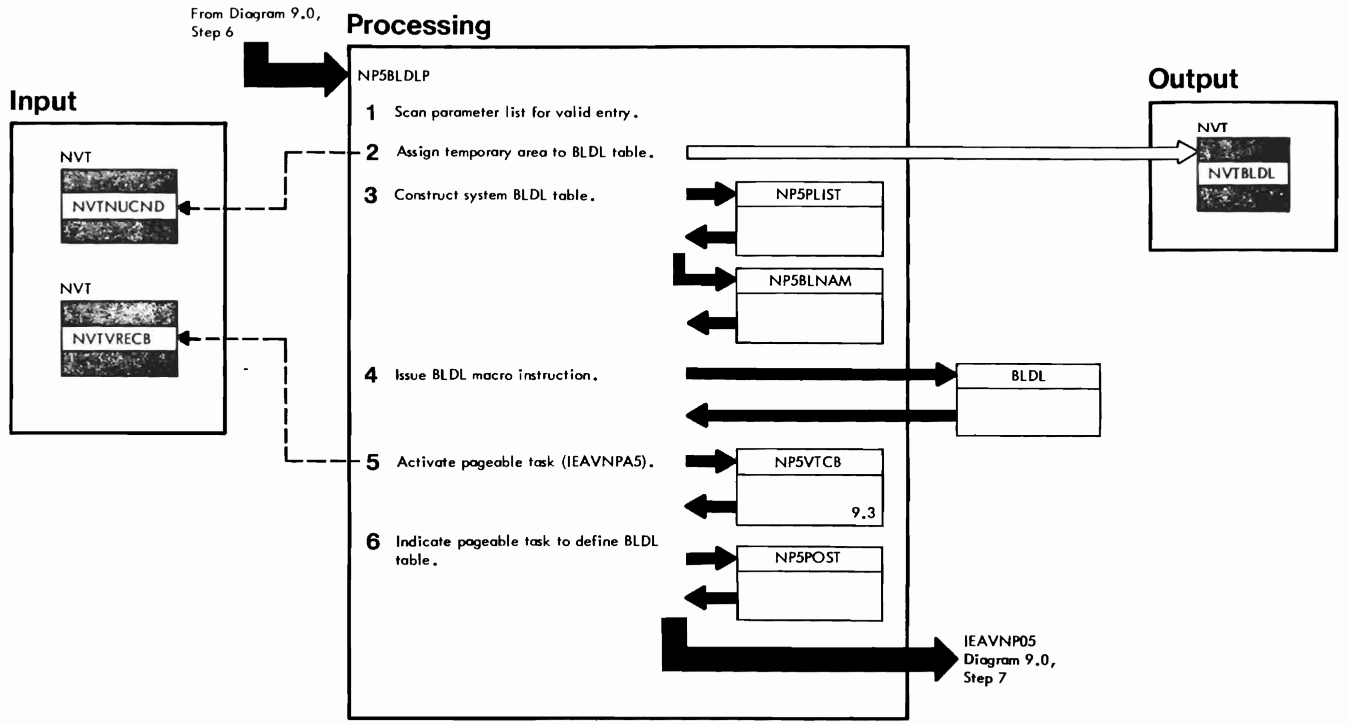
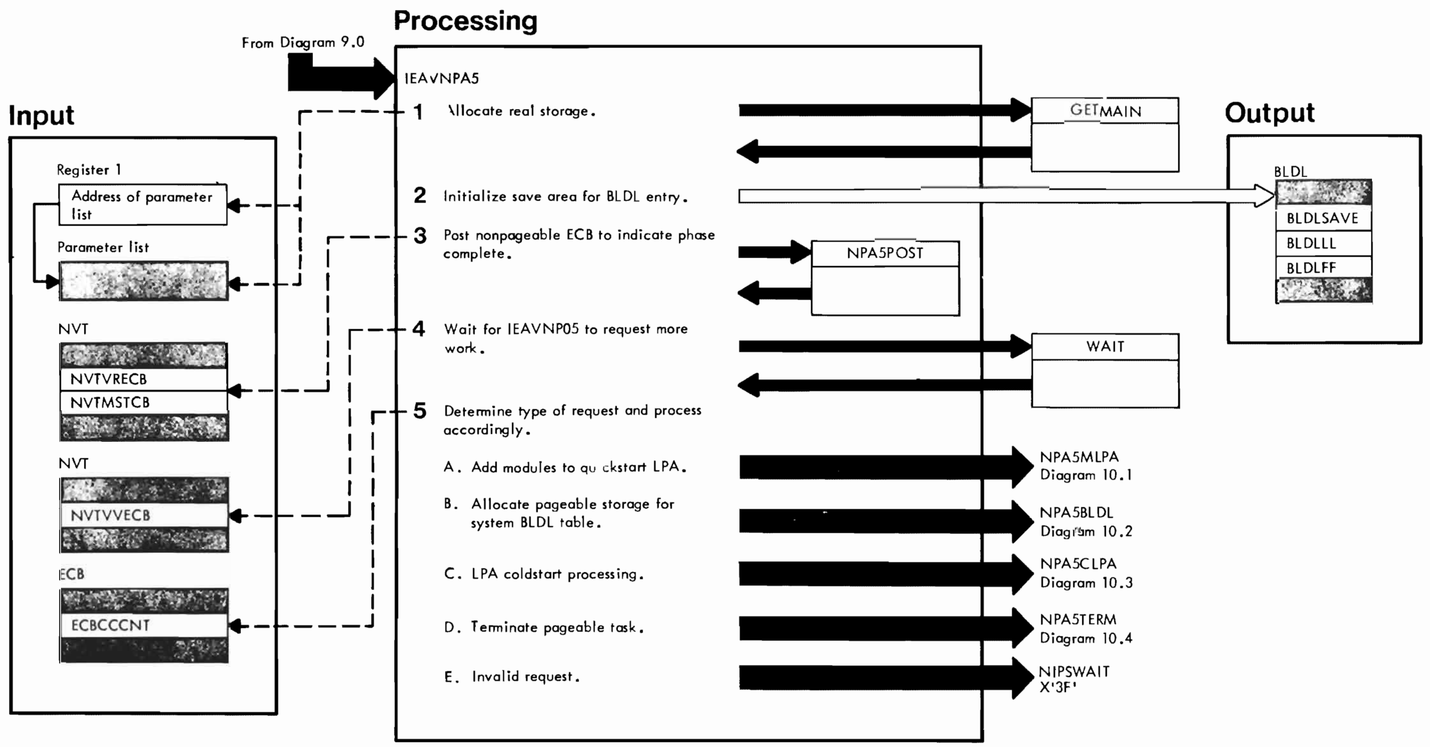

From Diagram 9.0, Step 6
Input
NVT
NVTNUCND
NVT
NVTVRECB
Processing
NP5BLDLP
1 Scan parameter list for valid entry.
2 Assign temporary area to BLDL table.
3 Construct system BLDL table.
NP5PLIST
NP5BLNAM
4 Issue BLDL macro instruction.
BLDL
5 Activate pageable task (IEAVNPA5).
NP5VTCB
9.3
6 Indicate pageable task to define BLDL table.
NP5POST
IEAVNP05 Diagram 9.0, Step 7
Output
NVT
NVTBLDL

Diagram 10.0: IEAVNPA5 Control Routine



1 Allocate real storage.

2 Initialize save area for BLDL entry.

3 Post nonpageable ECB to indicate phase complete.

4 Wait for IEAVNP05 to request more work.

5 Determine type of request and process accordingly.

A. Add modules to quickstart LPA.

B. Allocate pageable storage for system BLDL table.

C. LPA coldstart processing.

D. Terminate pageable task.

E. Invalid request.

Diagram 10.0: IEAVNPA5

Notes on Processing	Label
1. All of the region is allocated from subpool 252. Sixty-four bytes are allocated from subpool 255 for the BLDL save area.	NPA5INIT
3. The nonpageable ECB is posted to indicate completion of the initialization process in IEAVNPA5. The POSTCD field is set by the subroutine that performed the processing.	NPA5SYNC
4. The pageable ECB is specified in the WAIT macro instruction. IEAVNP05 posts this ECB when additional work is required.	
5. The type of request is indicated by a bit pattern in the pageable ECB. Steps 1 through 4 are executed only on the first entry to IEAVNPA5. Subsequent entries are to Step 5.	

Diagram 10.1: NPA5MLPA Load Specified Modules into LPA

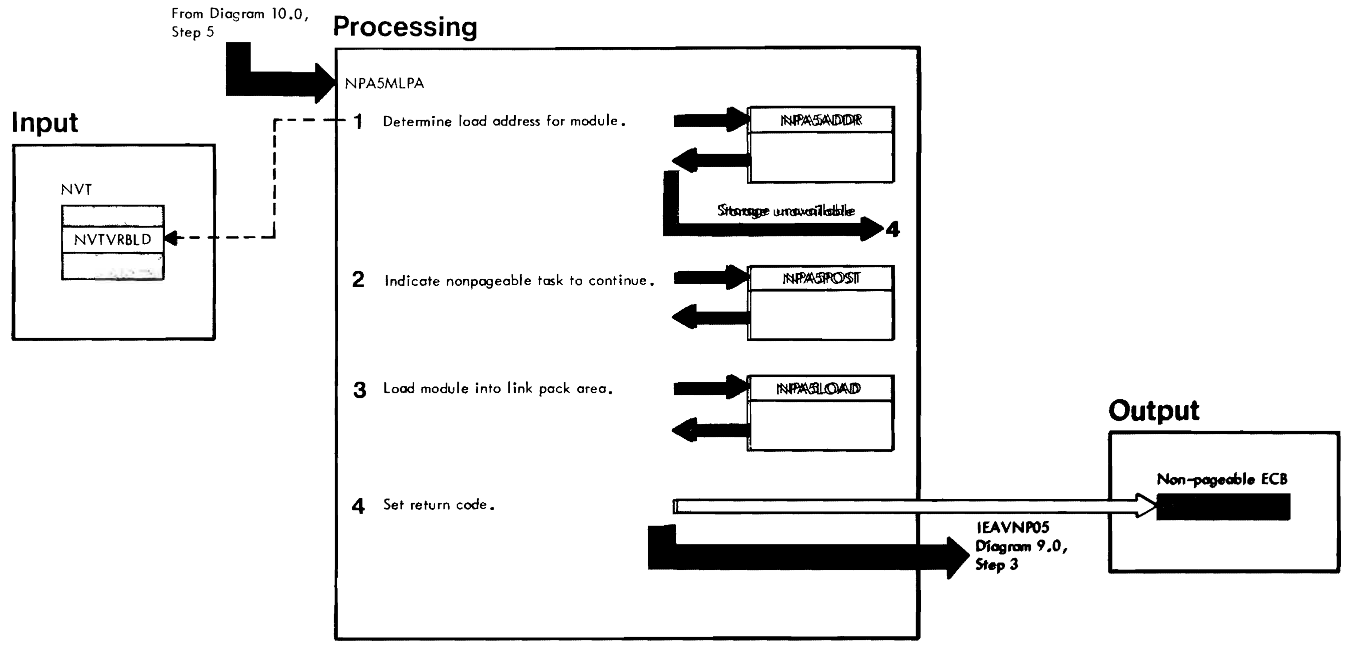

From Diagram 10.0, Step 5
Input
NVT
NVTVRBLD
Processing
NPA5MLPA
1 Determine load address for module.
NPA5ADDR
Storage unavailable
4
2 Indicate nonpageable task to continue.
NPA5POST
3 Load module into link pack area.
NPA5LOAD
4 Set return code.
IEAVNP05 Diagram 9.0, Step 3
Output
Non-pageable ECB

Diagram 10.1: NPA5MLPA (IEAVNPA5)

Notes on Processing	Label
1. NPA5MLPA is entered from the NP5MLPRM subroutine in IEAVNP05.	NPA5MLPA
4. Return is via the WAIT/POST interface.	NPA5MSER NPA5MXIT

Diagram 10.2: NPA5BLDL Allocate Storage for, and Move, BLDL Table

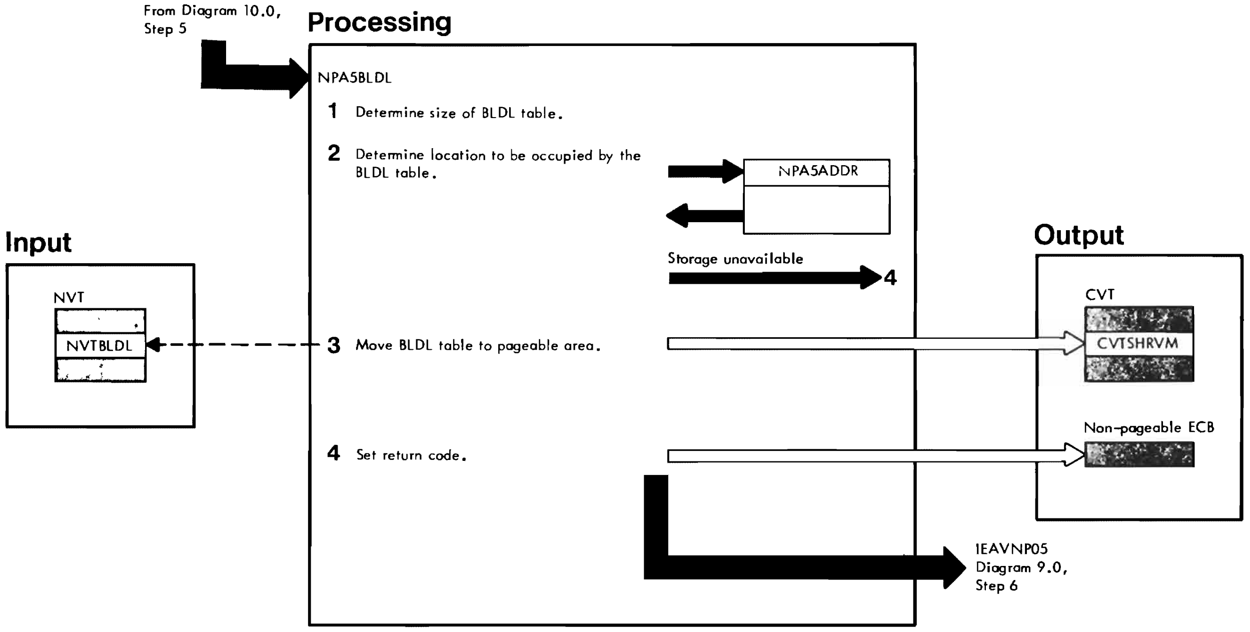
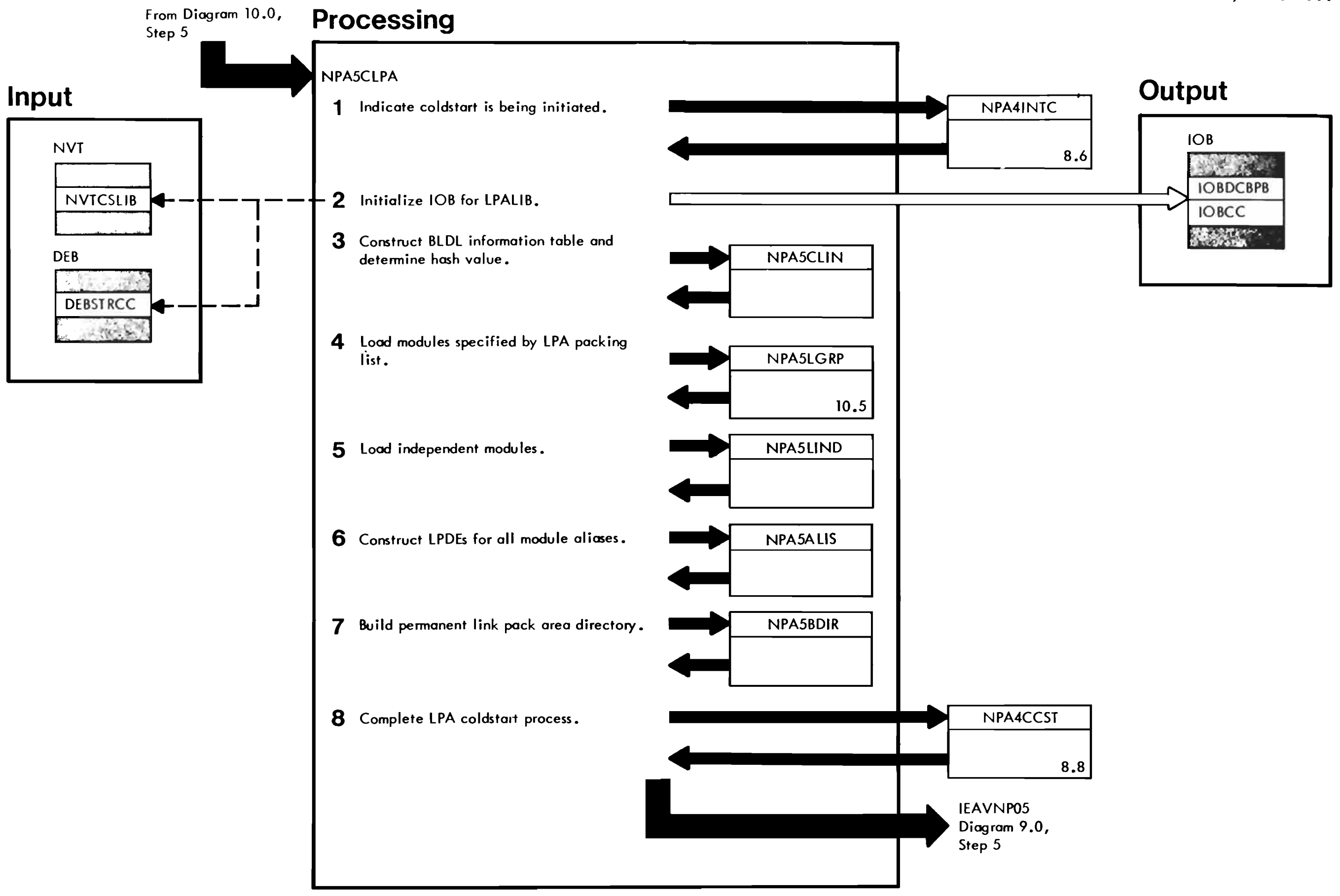

From Diagram 10.0, Step 5
Processing
NPA5BLDL
1 Determine size of BLDL table.
2 Determine location to be occupied by the BLDL table.
NPA5ADDR
Storage unavailable
4
Input
NVT
NVTBLDL
3 Move BLDL table to pageable area.
Output
CVT
CVTSHRVM
Non-pageable ECB
4 Set return code.
IEAVNP05 Diagram 9.0, Step 6

Diagram 10.2: NPA5BLDI (IEAVNPA5)

Notes on Processing	Label
1. NPA5BLDL is entered from the NP5BLDLP suborutine in IEAVNP05.	NPA5BLDL
4. The return code X'04' indicates storage was unavailable. X'0C' indicates end of normal processing.	NPA5BSER NPA5BXIT

Diagram 10.3: NPA5CLPA
Coldstart Process; Build LPA



1 Indicate coldstart is being initiated.

2 Initialize IOB for LPALIB.

3 Construct BLDL information table and determine hash value.

4 Load modules specified by LPA packing list.

5 Load independent modules.

6 Construct LPDEs for all module aliases.

7 Build permanent link pack area directory.

8 Complete LPA coldstart process.

Diagram 10.3: NPA5CLPA (IEAVNPA5)

Notes on Processing	Label
1. NPA5CLPA is entered at the request of the NP5CSLPA subroutine in IEAVNP05.	NPA5CLPA
2. The IOB is within IEAVNPA5 and is used for the I/O operations that read the SYS1.LPALIB directory records.	

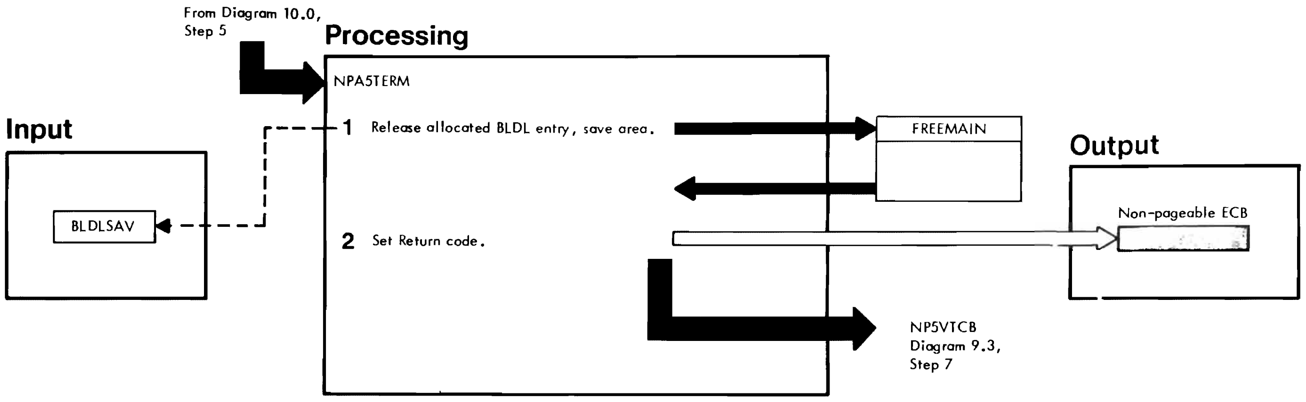
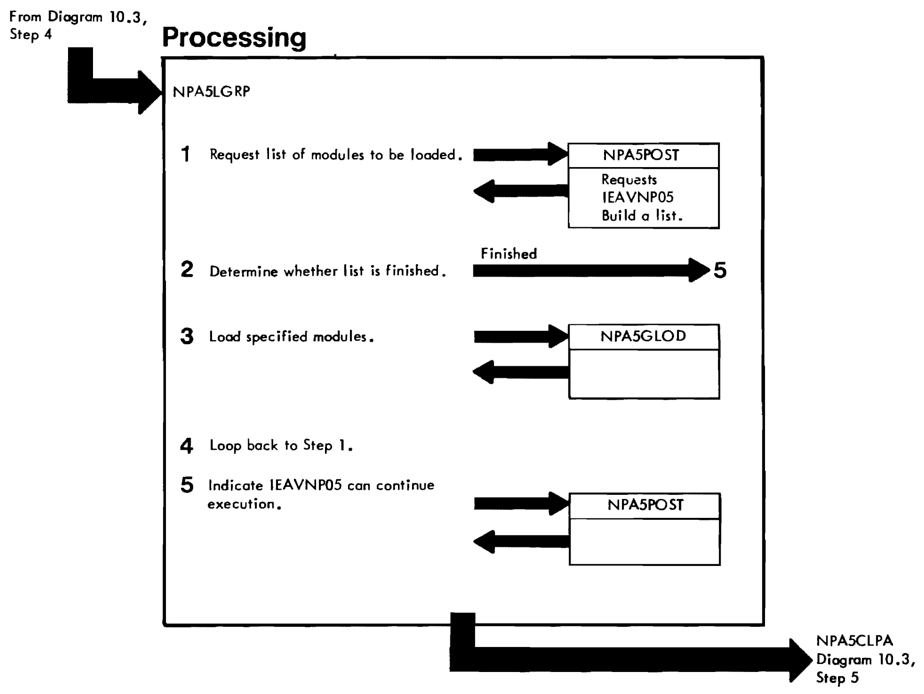

From Diagram 10.0, Step 5
Processing
NPA5TERM
1 Release allocated BLDL entry, save area.
2 Set Return code.
Input
BLDLSAV
FREEMAIN
Output
Non-pageable ECB
NP5VTCB Diagram 9.3, Step 7

Diagram 10.4: NPA5TERM (IEAVNPA5)

Notes on Processing	Label
1. The V=V space allocated by Step 1 in IEAVNPA5-0 will be released by IEAVNP05.	NPA5TERM
2. NPA5TERM relinquishes control by setting the return code to IEAVNP05 as X'0C' and by posting the V=R ECB.	

Diagram 10.5: NPA5LGRP Load Groups of Modules Specified by LPA Packing List


From Diagram 10.3, Step 4

Processing

NPA5LGRP

1 Request list of modules to be loaded.

2 Determine whether list is finished.

3 Load specified modules.

4 Loop back to Step 1.

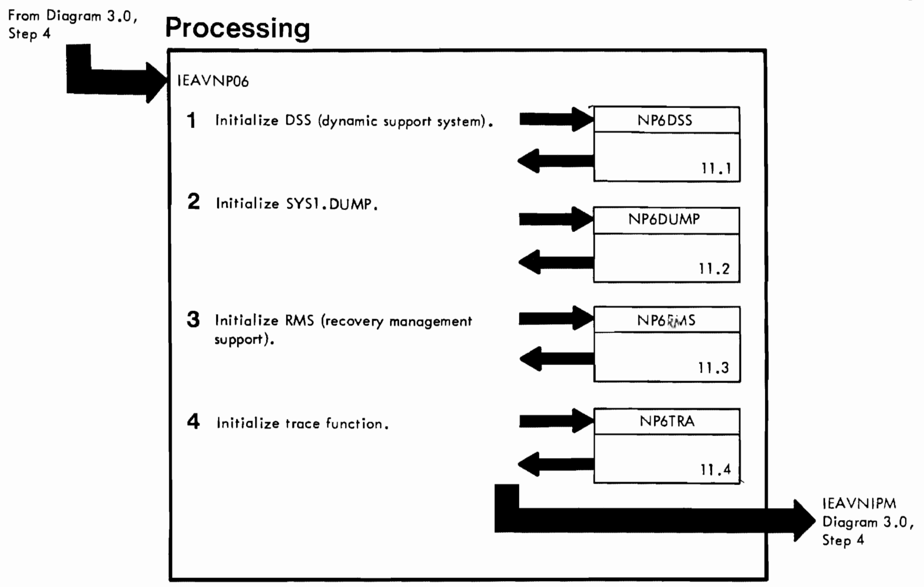
5 Indicate IEAVNP05 can continue execution.

NPA5POST
Requests IEAVNP05 Build a list.
Finished
5
NPA5GLOD
NPA5POST
NPA5CLPA Diagram 10.3, Step 5

Diagram 10.5: NPA5LGRP (IEAVNPA5)

Notes on Processing	Label
1. NPA5LGRP issues a Wait while awaiting return from NPA5POST.	
2. Determined by return code issued by NPA5POST.	

Processing



1 Initialize DSS (dynamic support system).

2 Initialize SYS1.DUMP.

3 Initialize RMS (recovery management support).

4 Initialize trace function.

Diagram 11.1: NP6DSS
Initialize DSS (Dynamic Support System)

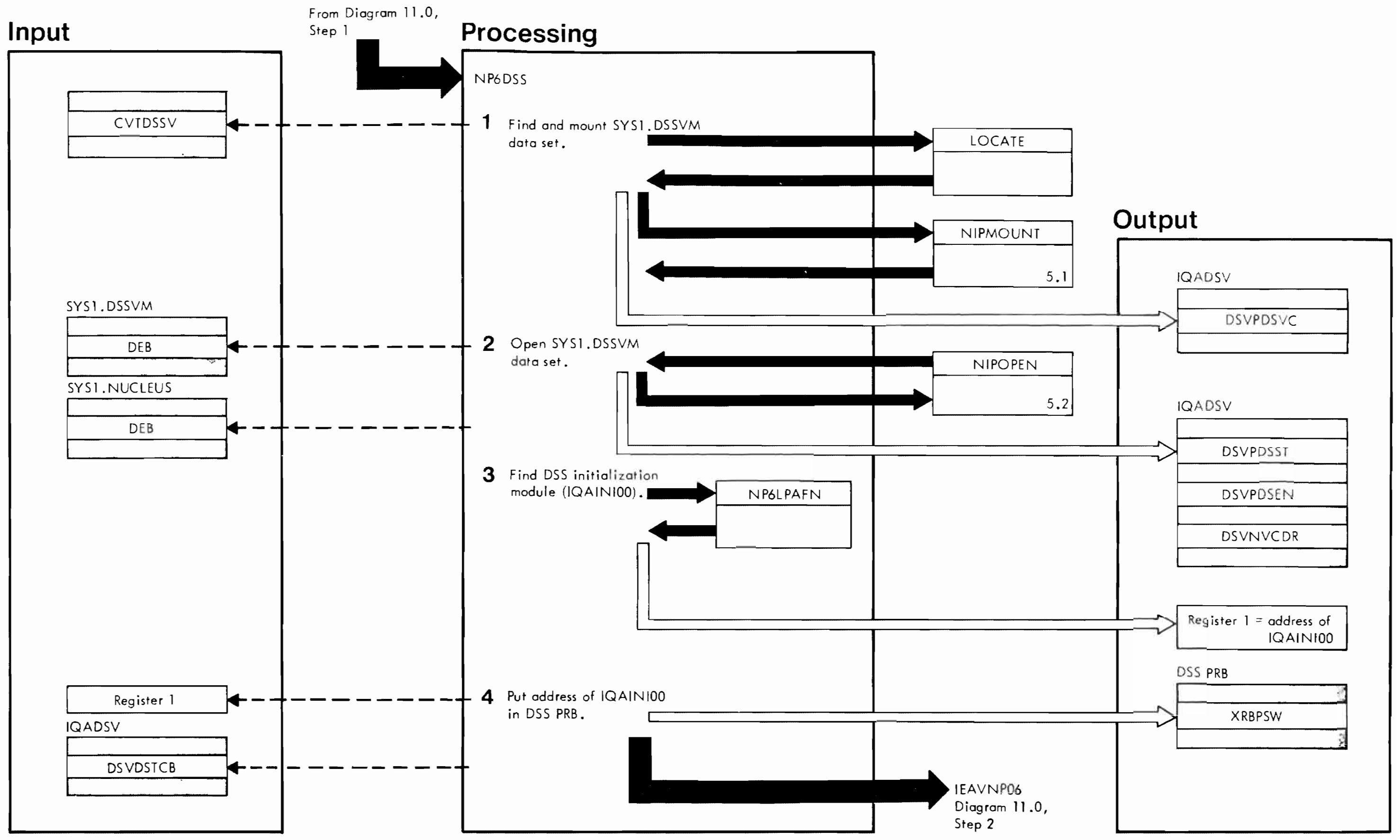

Input
From Diagram 11.0, Step 1
Processing
NP6DSS
CVTDSSV
1 Find and mount SYS1.DSSVM data set.
LOCATE
NIPMOUNT
5.1
Output
IQADSV
DSVPDSVC
SYS1.DSSVM
DEB
2 Open SYS1.DSSVM data set.
NIPOPEN
5.2
SYS1.NUCLEUS
DEB
IQADSV
DSVPDSST
DSVPDSEN
DSVNVCDR
3 Find DSS initialization module (IQAINI00).
NP6LPAFN
Register 1 = address of IQAINI00
DSS PRB
Register 1
4 Put address of IQAINI00 in DSS PRB.
XRBPSW
IQADSV
DSVDSTCB
IEAVNP06 Diagram 11.0, Step 2

Diagram 11.1: NP6DSS (IEAVNP06)

Notes on Processing	Label
1. If the data set is not found, the beginning and ending CCHH fields in the DSS vector table (IQADSV) are set to all F's. NP6DSS calls NIPWTO to write the messages LOCATE FAILED FOR SYS1.DSSVM and DSS INOPERATIVE to the operator. CVTDSSV is the CVT pointer used to locate the DSS vector table.	NP6DSER1
2. If NP6DSS cannot open the data set, the beginning and ending CCHH fields in the DSS vector table (IQADSV) are set to all F's. NP6DSS calls NIPWTO to write the message DSS INOPERATIVE to the operator.	NP6DSER2
3. NP6LPAFN calls IEAQCDSR to search the LPA active queue for IQAINI00. If IQAINI00 is not found, it then calls IEAVVMSR to search the LPA directory for IQAINI00. If IQAINI00 is found, NP6LPAFN sets register 1 to the address of IQAINI00 from the CDE or LPDE.	
If the module is not found, NP6DSS calls NIPWTO to write the message IQAINI00 NOT FOUND IN LPA to the operator. NP6DSS puts the address of the DSS Mask Out routine (IQAPFXRT) in the restart new PSW.	NP6DSER3
4. Register 1 contains the address of IQAINI00. DSVDSTCB is a pointer to the DSS TCB.	

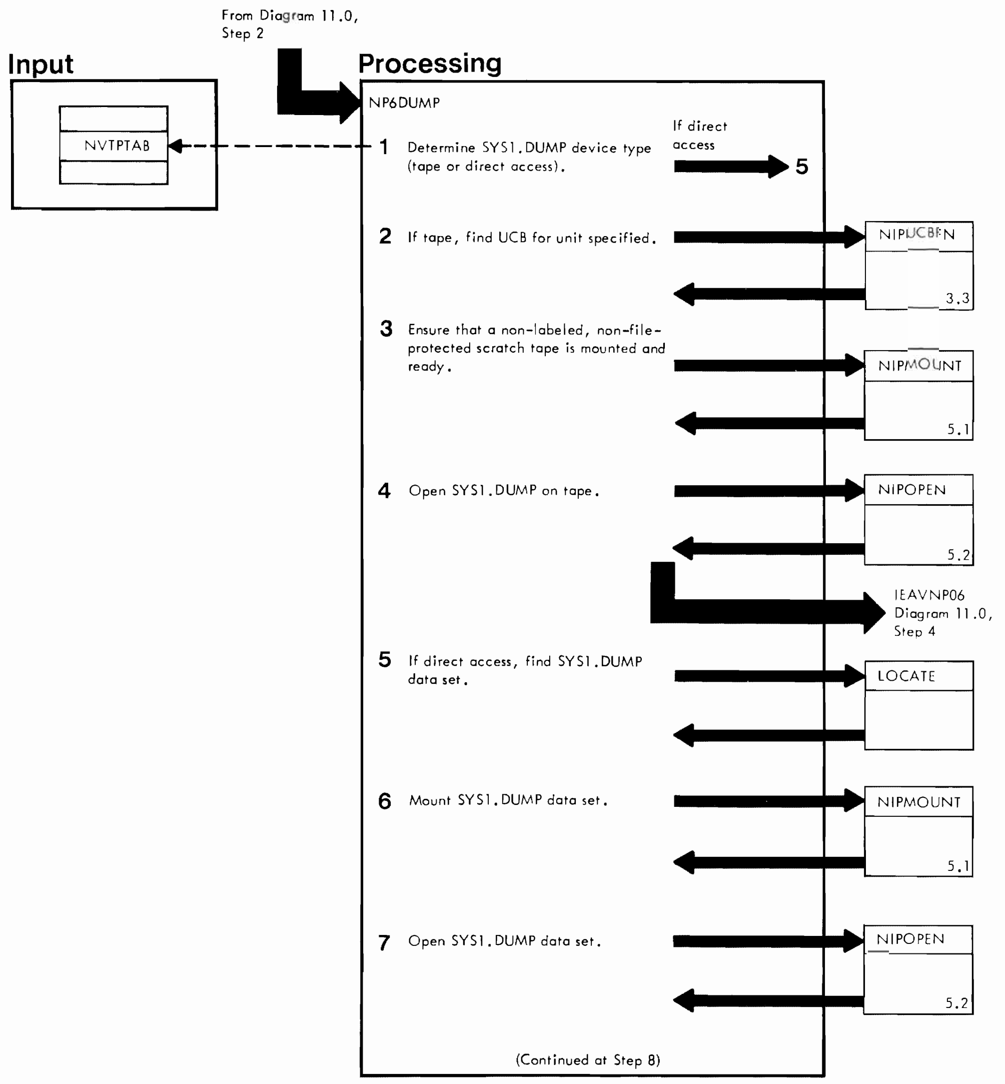

From Diagram 11.0, Step 2
Input
NVTPTAB
Processing
NP6DUMP
1 Determine SYS1.DUMP device type (tape or direct access).
If direct access
5
2 If tape, find UCB for unit specified.
NIPUCBFN
3.3
3 Ensure that a non-labeled, non-file-protected scratch tape is mounted and ready.
NIPMOUNT
5.1
4 Open SYS1.DUMP on tape.
NIPOPEN
5.2
IEAVNP06 Diagram 11.0, Step 4
5 If direct access, find SYS1.DUMP data set.
LOCATE
6 Mount SYS1.DUMP data set.
NIPMOUNT
5.1
7 Open SYS1.DUMP data set.
NIPOPEN
5.2
(Continued at Step 8)

Diagram 11.2: NP6DMP (IEAVNP06)

Notes on Processing	Label
1. NP6DMP locates and verifies the DUMP parameter.	NP6DMP
If the parameter is not specified or is not valid, NP6DMP checks whether there is enough space in the nucleus buffer for a DCB for SYS1.DUMP. If there is enough space, the operator is prompted to specify the DUMP parameter or cancel the specification. If he cancels, SYS1.DUMP is disabled by setting the pointer to the DCB (CVTDAR) to 0 and NVTNUCND to its value on entry to NP6DMP. Control goes to Diagram 11.1, Step 4.	NP6DMP2
If there is not enough space for the DCB, the operator is notified, and SYS1.DUMP is disabled as above.	NP6DUMP2
If DUMP=NO is specified, SYS1.DUMP is disabled as above. If DUMP=NO is not specified, NP6DMP checks whether there is enough space for the SYS1.DUMP DCB. If there is not, the operator is notified, and SYS1.DUMP is disabled as above.	NP6DMP1
NP6DMP also checks the validity of the unit address parameter. If it is invalid, NP6DMP notifies the operator via NIPWTOR and prompts him to respecify the parameter.	NP6INVLD

Notes on Processing	Label
2. On return of control from NIPUCBFN, the UCB is checked to ensure that the device specified is a 2400-compatible tape drive and is online.	NP6DMPTA
3. If the device is not 2400-compatible and online, or if NIPMOUNT was unable to have the required tape mounted, the operator is notified that this device is unacceptable and is prompted to respecify or cancel the parameter.	NP6UNACC
4. If the OPEN operation is unsuccessful, SYS1.DUMP is disabled as in Step 1.	NP6NODMP
5. If the data set is not found, the operator is notified with the LOCATE FAILED message and is prompted to respecify the DUMP parameter.	NP6LOCNG
6. If the volume containing the data set cannot be mounted, the operator is prompted to respecify the DUMP parameter.	NP6RESPE

Processing

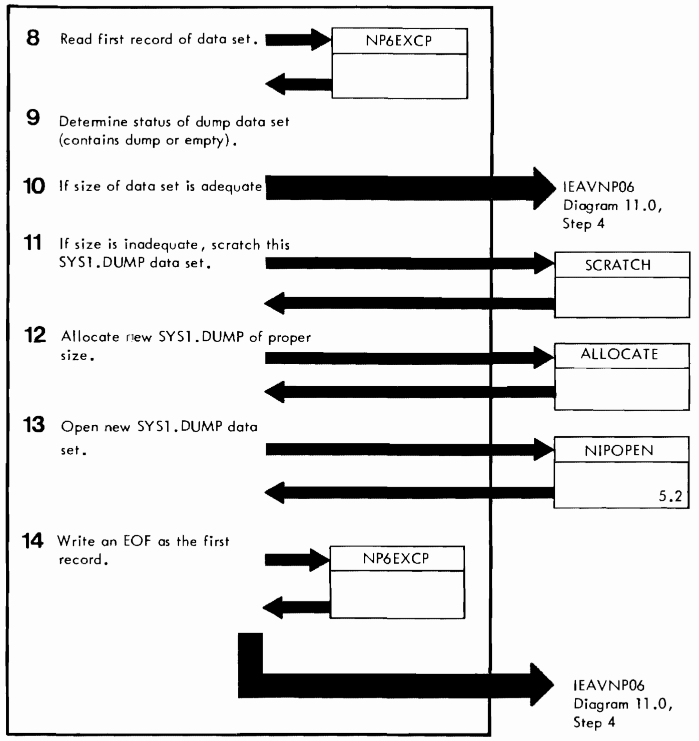

8 Read first record of data set.
NP6EXCP
9 Determine status of dump data set (contains dump or empty).
10 If size of data set is adequate
IEAVNP06 Diagram 11.0, Step 4
11 If size is inadequate, scratch this SYS1.DUMP data set.
SCRATCH
12 Allocate new SYS1.DUMP of proper size.
ALLOCATE
13 Open new SYS1.DUMP data set.
NIPOPEN
5.2
14 Write an EOF as the first record.
NP6EXCP
IEAVNP06 Diagram 11.0, Step 4

Diagram 11.2: NP6DMP (IEAVNP06) (Cont'd)

Notes on Processing	Label
7. If OPEN processing fails because the data set is not found, an attempt is made to allocate and open a new data set. If OPEN processing fails because of insufficient storage, SYS1.DUMP is disabled as in Step 1. If OPEN processing fails for any other reason, the operator is prompted to respecify the parameter.	NP6OPNNG
8. If there is an I/O error, NP6DMP informs the operator and prompts him to respecify or cancel the DUMP parameter.	NP6IOEOF
9. If the data set contains a dump, the operator is prompted with the message SYS1.DUMP CONTAINS DATA.	NP6LOCAT
If the data set is empty and the size is not specified or is adequate, control returns to Step 4.	NP6EMTY
10. If the size specified in the parameter is greater than the size of the data set, the operator is notified with the message SYS1. DUMP TOO SMALL and prompted to respecify or cancel the parameter.	NP6SMAL

Notes on Processing	Label
11. The operator is notified with the message SYS1.DUMP SCRATCHED or UNABLE TO SCRATCH SYS1.DUMP and prompted to respecify the DUMP parameter.	NP6EMTY NP6NOSCR
12. If allocation fails, the operator is notified and prompted to respecify the DUMP parameter.	NP6FAIL
13. If OPEN processing fails because of insufficient space, SYS1.DUMP is disabled as in Step 1. If it fails for any other reason, the operator is prompted to respecify the DUMP parameter.	NP6OPNBD
14. If there is an I/O error, NIPSENSE is called, and the operator is prompted to respecify the DUMP parameter.	NP6BADIC

Processing

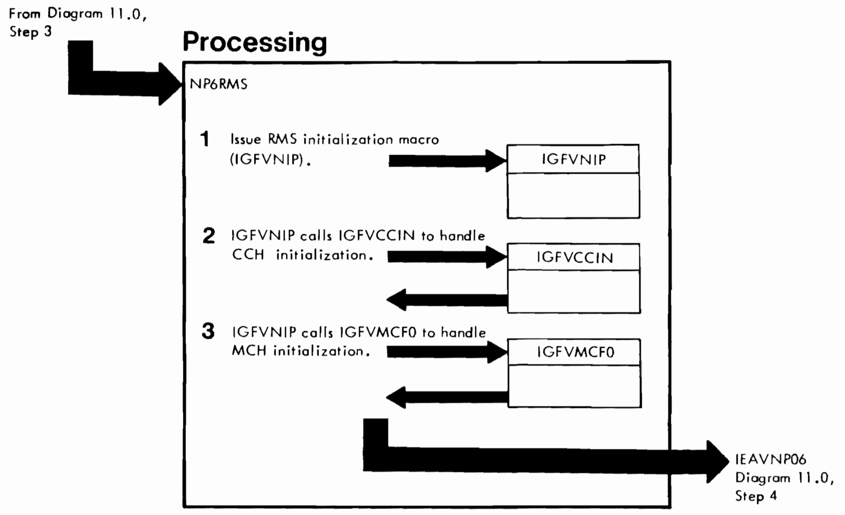
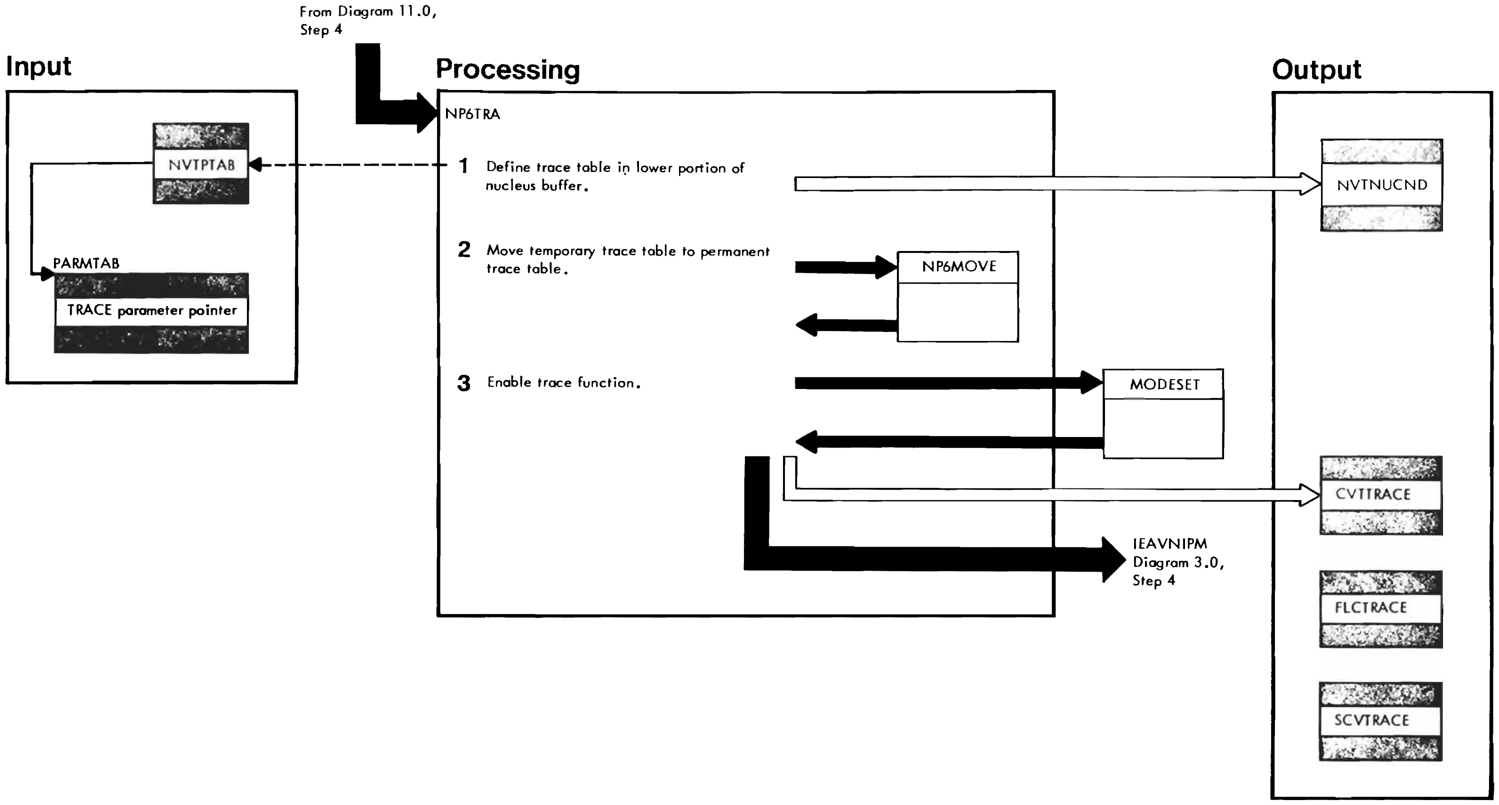

From Diagram 11.0, Step 3
NP6RMS
1 Issue RMS initialization macro (IGFVNIP).
IGFVNIP
2 IGFVNIP calls IGFVCCIN to handle CCH initialization.
IGFVCCIN
3 IGFVNIP calls IGFVMCF0 to handle MCH initialization.
IGFVMCF0
IEAVNP06 Diagram 11.0, Step 4

Diagram 11.3: NP6RMS (IEAVNP06)

Notes on Processing	Label
2. Using BLDL, IGFVNIP loads, calls, and deletes IGFVCCIN. If an error occurs, a message is written to the operator via NIPWTOR.	NP6RMS
3. Using BLDL, IGFVNIP loads, calls, and deletes IGFVMCF0. If an error occurs, a message is written to the operator via NIPWTOR.	NP6RMS



1 Define trace table in lower portion of nucleus buffer.

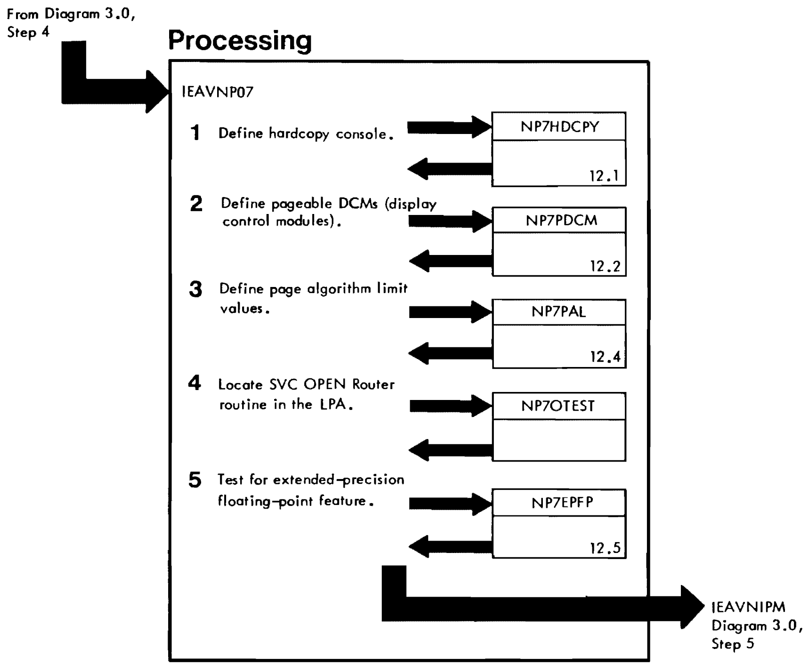
2 Move temporary trace table to permanent trace table.

3 Enable trace function.

Diagram 11.4: NP6TRA (IEAVNP06)

Notes on Processing	Label
1. If the TRACE parameter is not specified or is equal to 0, the trace function is disabled by putting a BR 11 in the CVT field CVTTRACE and putting 0's in FLCTRACE and SCVTRPTR.	NP6NOTRA
If the parameter is invalid, the operator is prompted with the message INVALID TRACE PARAMETER and given the option to respecify or cancel the parameter. If he cancels, NVTNUCND is rounded to a 4K boundary and CVTNUCB is set equal to NVTNUCND. The trace function is then disabled as above.	NP6INVAL
If space is not available in the nucleus buffer, NVTNUCND is set equal to CVTNUCB, and the trace function is disabled as above. The operator is notified with the message INSUFFICIENT SPACE, and control is returned to the caller.	NP6NOSPA
If there is not enough space in the nucleus buffer, the last entry pointer is set to CVTNUCB-32, and NVTNUCND is set to CVTNUCB. The operator is notified that the trace table size has been decreased.	NP6SMALL
2. If the two tables are equal in size or the PTT (permanent trace table) is larger, the TTT (temporary trace table) is moved to the PTT beginning with the lowest address in each table (first entry pointer). Any remaining area is set to 0's.	NP6COPY
If the PTT is smaller than the TTT, as much of the TTT (starting with the most recent entry) as will fit is moved to the PTT.	NP6COPY

Diagram 12.0: IEAVNP07 Call Routines Sequentially



1 Define hardcopy console.

2 Define pageable DCMs (display control modules).

3 Define page algorithm limit values.

4 Locate SVC OPEN Router routine in the LPA.

5 Test for extended-precision floating-point feature.

Diagram 12.0: IEAVNP07

Notes on Processing	Label
2 and 4. NP7PDCM and NP7OTEST call NP7LPAFN to search the LPA. See Diagram 12.3 for a description of NP7LPAFN.	

Diagram 12.1: NP7HDCPY
Define the Console or System Log Data Set to be Used for Message Hardcopy

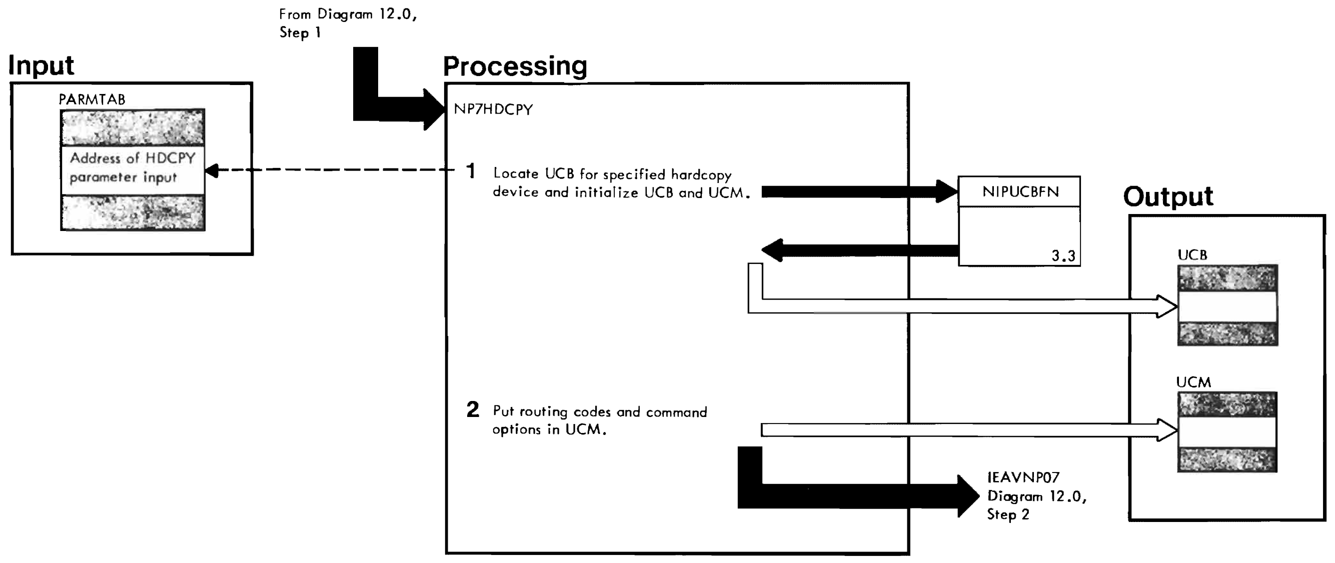
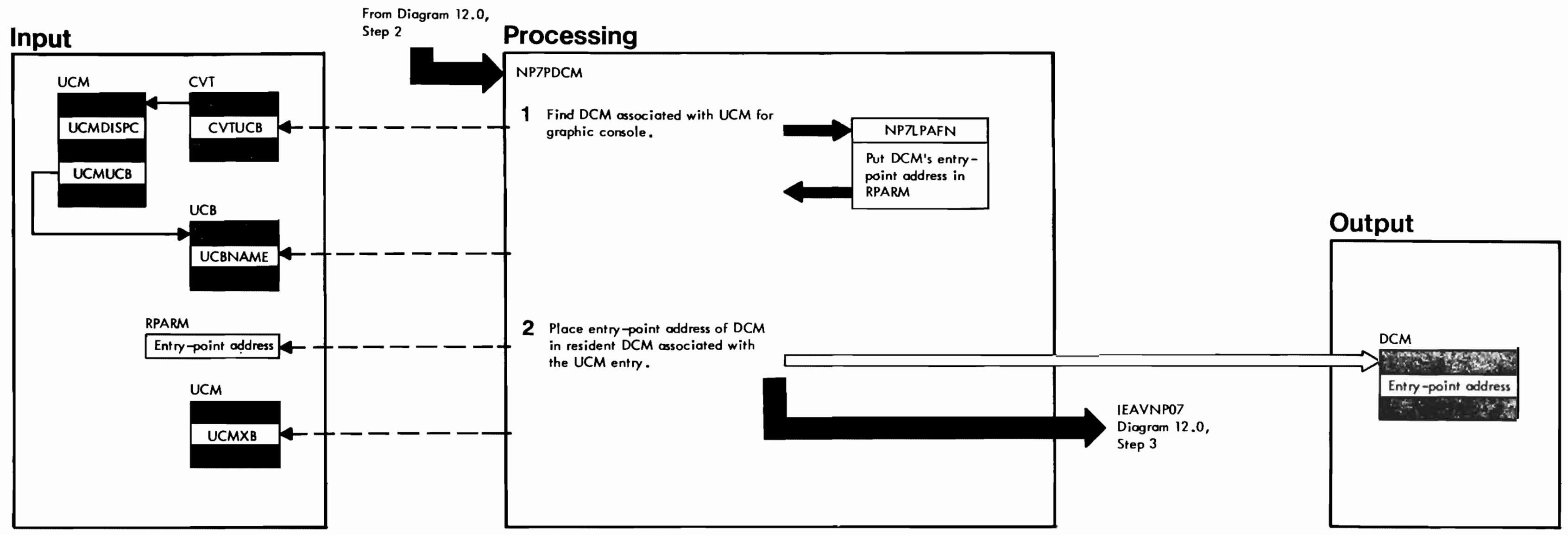

From Diagram 12.0, Step 1
Input
PARMTAB
Address of HDCPY parameter input
Processing
NP7HDCPY
1 Locate UCB for specified hardcopy device and initialize UCB and UCM.
NIPUCBFN
3.3
2 Put routing codes and command options in UCM.
IEAVNP07 Diagram 12.0, Step 2
Output
UCB
UCM

Diagram 12.1: NP7HDCPY (IEAVNP07)

Notes on Processing	Label
1. NP7HDCPY locates and verifies the HARDCPY parameter input.	NP7HDCPY
IEAVNP01 determined whether a hardcopy console was required.	NP7HCINV
If SYSLOG is specified, NP7HDCPY identifies the system log as the hardcopy device.	NP7HCSLG
If HARDCPY is not specified and it is required, or if the specification is in error, or if the console is unavailable, NIPWTOR is called to prompt the operator to respecify the parameter.	NP7HCREQ
If HARDCPY is not specified and is not required, control goes to Diagram 12.0, Step 2.	NP7HCXIT

Diagram 12.2: NP7PDCM Relate Each Graphic Device to its LPA-resident DCM



1 Find DCM associated with UCM for graphic console.

2 Place entry-point address of DCM in resident DCM associated with the UCM entry.

Diagram 12.2: NP7PDCM (IEAVNP07)

Notes on Processing	Label
1. NP7PDCM examines UCM entries in the nucleus-resident UCM and constructs the name of the associated DCM for each graphic console by adding the EBCDIC unit name from the UCB to the DCM.	NP7PDLST
If NP7LPAFN cannot find the DCM in the LPA, it sets the active console flag in the UCB to 0 and writes an error message to the operator, using NIPWTOR, identifying the missing DCM name and the unit address of the associated console. NP7PDCM returns control to Diagram 12.0, Step 3.	NP7PDNFN
If the DCM is not found for the master console (a graphic console), NIPSWAIT is called to put the system in a wait state (code X'3B').	

**Diagram 12.3: NP7LPAFN
Find a Module in the LPA**

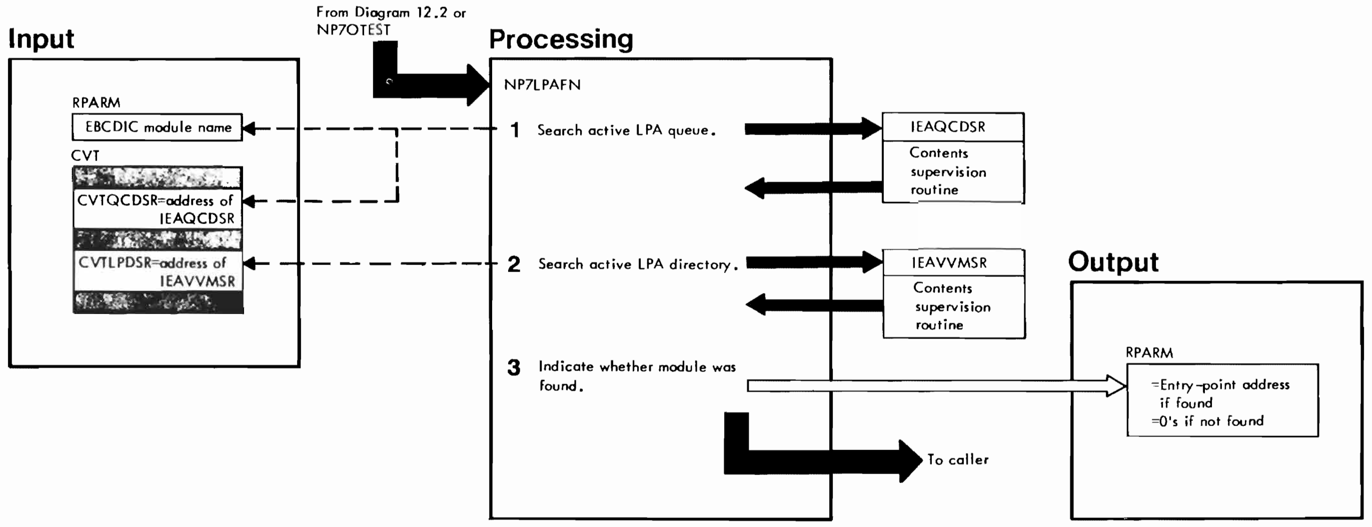

Input
RPARM
EBCDIC module name
CVT
CVTQCDSR=address of IEAQCDSR
CVTLPDSR=address of IEAVVMSR
From Diagram 12.2 or NP7OTEST
Processing
NP7LPAFN
1 Search active LPA queue.
2 Search active LPA directory.
3 Indicate whether module was found.
IEAQCDSR
Contents supervision routine
IEAVVMSR
Contents supervision routine
To caller
Output
RPARM
=Entry-point address if found
=0's if not found

Diagram 12.3: NP7LPAFN (IEAVNP07)

Notes on Processing	Label
1. If the module is found, control goes to Step 3.	
2. The IEAQCDSR Search routine is called before the IEAVVMSR routine to give preference to a module found in the fixed LPA or the LPA update area.	NP7LPAFN

Diagram 12.4: NP7PAL
Define Page Algorithm Limit Values

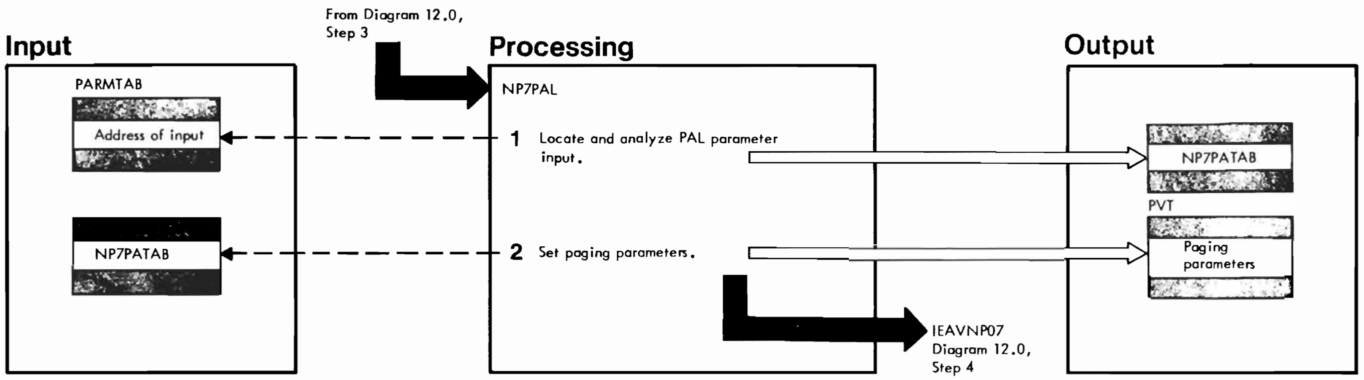


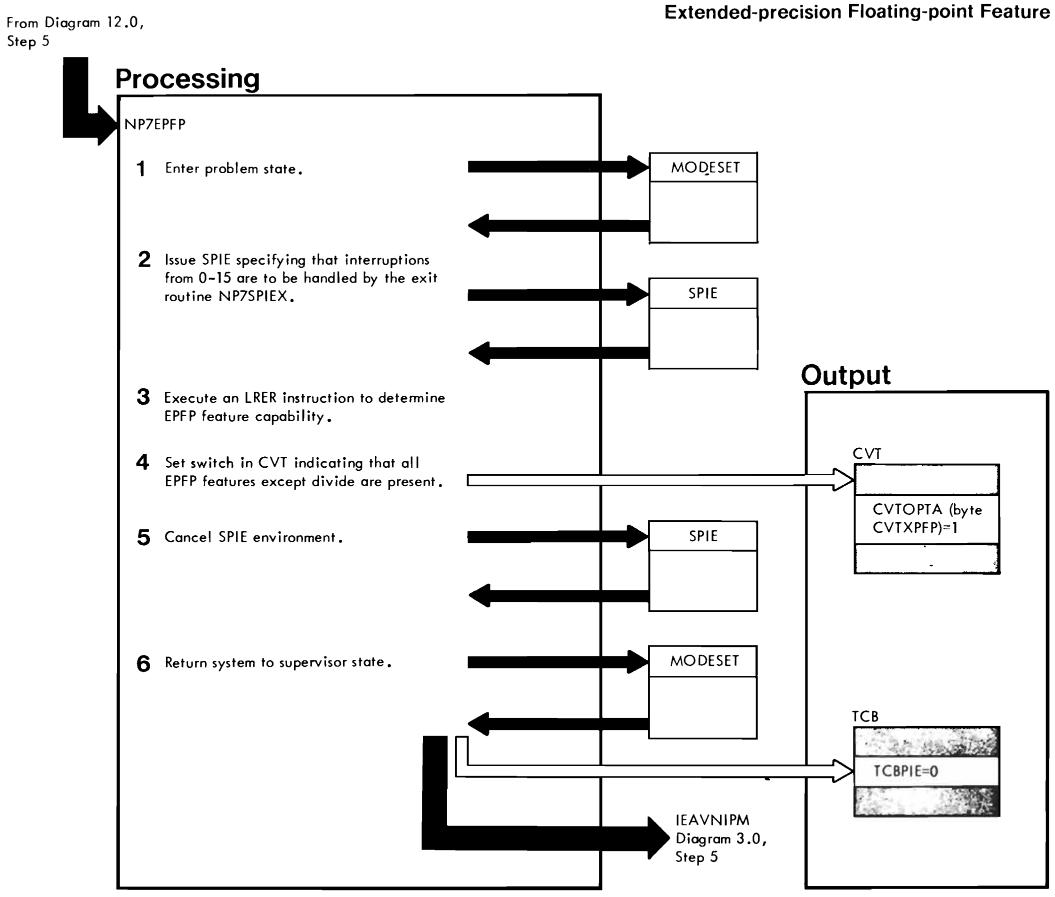
Diagram 12.4: NP7PAL (IEAVNP07)

Notes on Processing	Label
1. NP7PAL uses the system parameter table, PARMTAB, to locate the PAL parameter input. The first input source that is analyzed is the one from the IEASYSxx list of SYS1. PARMLIB. When the specifications from this source have been analyzed and put in a table, NP7PAL analyzes the operator-supplied PAL input. The subparameters are merged with duplicate specification of a subparameter, resulting in a parameter override; the last-supplied definition (in the normal case, the operator overriding an IEASYSxx definition) for such a value will be the only effective specification. NP7PAL optionally displays the PAL subparameter values that result from its analysis and allows the operator to respecify the PAL parameter. NP7PAL verifies that the PAL parameter input is in the correct format by examining each of the subparameter definitions in an EBCDIC matrix. If an error is encountered, analysis of the PAL input from the operator or PARMLIB is terminated; any subparameters that are correctly and completely defined up to the point of the error are accepted. In the case of an error, the operator is informed via the INVALID PAL PARM message written by the IEAVNIPM NIPWTOR routine. The message identifies the input source as being either the PARMLIB or the operator, and the operator is permitted either to respecify the remainder of the PAL parameter or to cancel his option and thereby accept the truncated PAL definition, together with any system-assigned values.	NP7PALA

Notes on Processing	Label

2. Paging supervisor fields to be set, together with their limit/ assumed values are:
 	
PVTTBASE =(CVTEORM - CVTNUCB)/4096
PVTLTH (LTH=) = 1 - 99;
    assumed value = 5
PVTREPCT (REPCT=) = 1 - 99;
    assumed value = 3
PVTBFXTF (NFX=) = 8 - 9999;
    assumed value = .25 PVTTBASE
PVTSFXTF = 1.25 PVTBFXTF
PVTLFXTF = 1.5 PVTBFXTF
PVTLFXL = (PVTTBASE - PVTLFXTF);
    maximum = 255
PVTSFXL = (PVTTBASE - PVTSFXTF);
    maximum = 255
PVTBFXL = (PVTTBASE - PVTBFXTF);
    maximum = 255
PVTLRLIM (LRC=) = 0 - 9999;
    assumed value = 5
PVTHRLIM (HRC=) = 0 - 9999;
    assumed value = 20
PVTLALIM (LRD=) = 0 - 9999;
    assumed value = 9999
PVTHALIM (HRD=) = 0 - 9999;
    assumed value = 0
PVTIMEAD (MTIM=) = 1 - 9;
    assumed value = 1
 	
Assumed values are assembled into the respective NVT fields. The assumed value for NFX= is expressed as a fractional value, in hundredths, to be multiplied times PVTTBASE.
	NP7PTBLD

Diagram 12.5: NP7EPFP Test for the Presence of the Extended-precision Floating-point Feature



From Diagram 12.0, Step 5
Processing
NP7EPFP
1 Enter problem state.
MODESET
2 Issue SPIE specifying that interruptions from 0-15 are to be handled by the exit routine NP7SPIEX.
SPIE
3 Execute an LRER instruction to determine EPFP feature capability.
4 Set switch in CVT indicating that all EPFP features except divide are present.
5 Cancel SPIE environment.
SPIE
6 Return system to supervisor state.
MODESET
IEAVNIPM Diagram 3.0, Step 5
Output
CVT
CVTOPTA (byte CVTXPFP)=1
TCB
TCBPIE=0

Diagram 12.5: NP7EPFP (IEAVNP07)

Notes on Processing	Label
1. NP7EPFP must enter problem-program state, since a program check cannot be tolerated in supervisor state.	
3. If no program interruption occurs, EPFP hardware is present. If a program interruption occurs, the SPIE Exit routine, NP7SPIEX, receives control via the control program. The register contents upon receiving control are: Register 0 Unpredictable Register 1 Address of PIE Register 2-13 Same as at time of program check Register 14 Return address (to control program) Register 15 Address of NP7SPIEX The registers 0, 1, 2, 14, and 15 have been saved in the PIE and will be restored from that location when the SPIE Exit routine is completed. Registers 3-13 will reflect any alterations made by NP7SPIEX. The PSW in the PIE (PIEPSW) is examined to determine what type of interruption occurred. The resume address in PIEPSW is then set accordingly (note that this is a BC (basic control) mode PSW). For an interruption code other than X'01' (operation), the extended-precision floating-point feature is assumed to exist and PIEPSW is set to Step 4. Otherwise, the feature is assumed to be absent and PIEPSW is set to Step 5.	NP7EPFP
4. The divide-only switch (bit CVTXPFP in byte CVTOPTA) is set to 1.	NP7FPYES
5. This restores the program mask to its original value.	NP7FPNO
6. Since the PIE is not automatically freed, TCBPIE is set to 0 to avoid leaving a useless and possibly misleading pointer in the master scheduler TCB. Actual freeing of the PIE is done at the time the final master scheduler region is established.	NP7FPNO

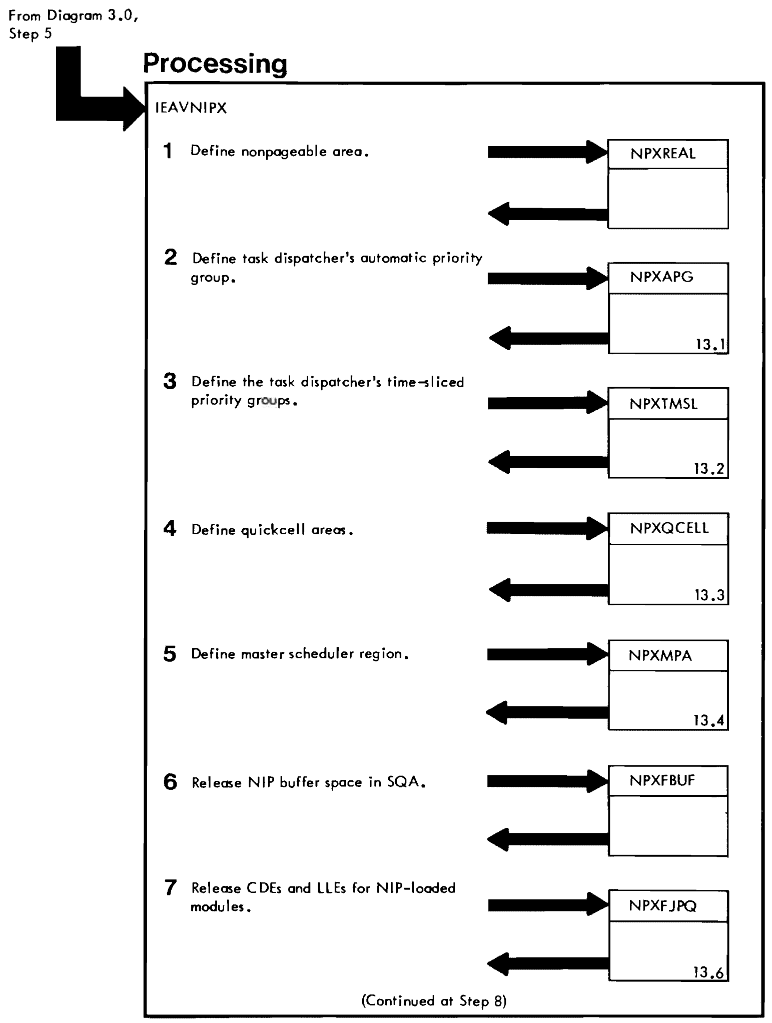
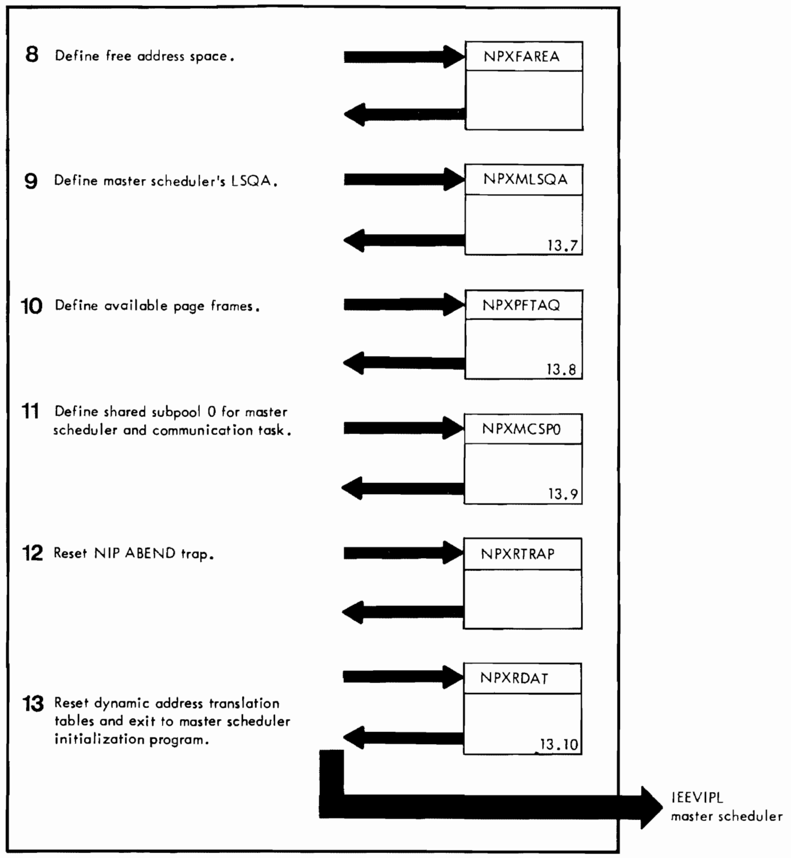

From Diagram 3.0, Step 5
Processing
IEAVNIPX
1 Define nonpageable area.
NPXREAL
2 Define task dispatcher's automatic priority group.
NPXAPG
13.1
3 Define the task dispatcher's time-sliced priority groups.
NPXTMSL
13.2
4 Define quickcell areas.
NPXQCELL
13.3
5 Define master scheduler region.
NPXMPA
13.4
6 Release NIP buffer space in SQA.
NPXFBUF
7 Release CDEs and LLEs for NIP-loaded modules.
NPXFJPQ
13.6
(Continued at Step 8)

Diagram 13.0: IEAVNIPX

Notes on Processing	Label
1. There is no diagram for NPXREAL.	
5. NPXMPA calls NPXMPA1 to limit background and TSO use of auxiliary storage (see Diagram 13.5).	
6. There is no diagram for NPXBUF.	

Processing



8 Define free address space.

9 Define master scheduler's LSQA.

10 Define available page frames.

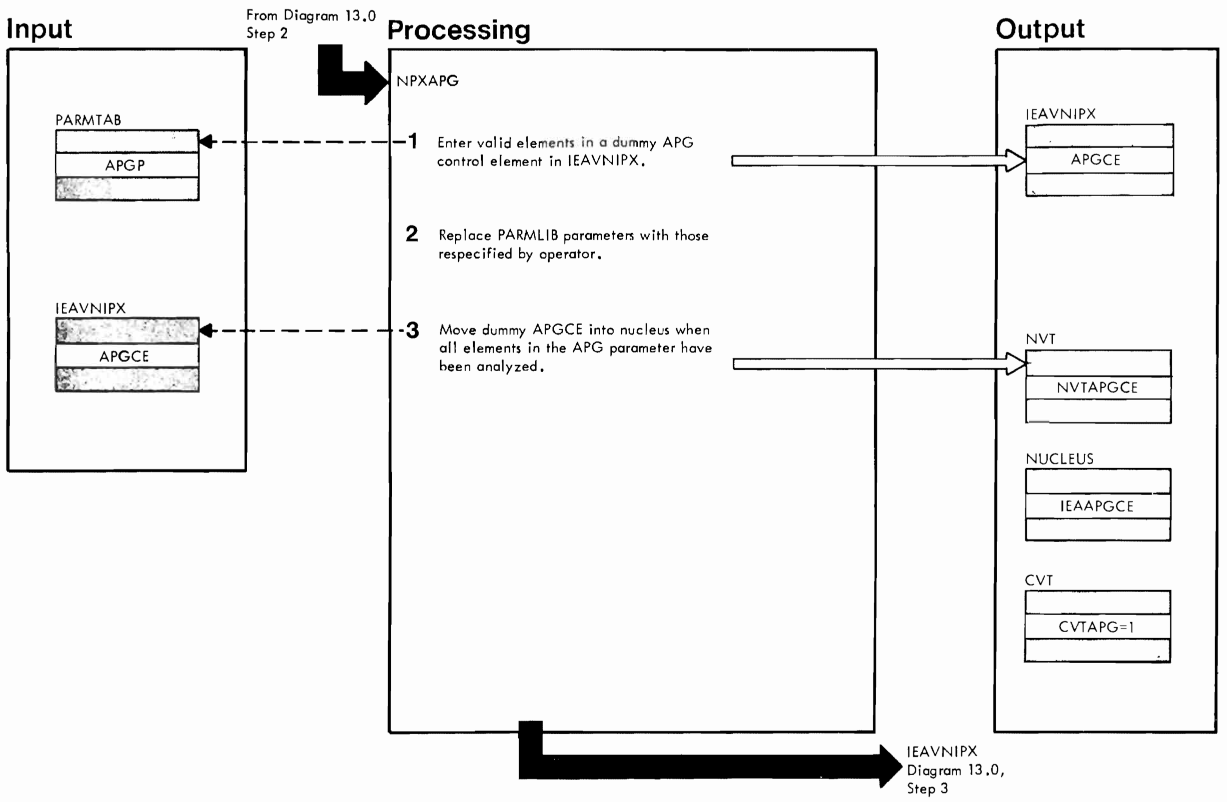
11 Define shared subpool 0 for master scheduler and communication task.

12 Reset NIP ABEND trap.

13 Reset dynamic address translation tables and exit to master scheduler initialization program.

Diagram 13.0: IEAVNIPX (Cont'd)

Notes on Processing	Label
8. There is no diagram for NPXFAREA.	
12. There is no diagram for NPXRTRAP.	



1 Enter valid elements in a dummy APG control element in IEAVNIPX.

2 Replace PARMLIB parameters with those respecified by operator.

3 Move dummy APGCE into nucleus when all elements in the APG parameter have been analyzed.

Diagram 13.1: NPXAPG (IEAVNIPX)

Notes on Processing	Label
1. NPXAPG locates and examines the APG parameter.	NPXAPRTY
	NPXAPNXT
If the APG parameter is not defined or is invalid, NPXAPG does not modify the IPL values of CVTAPG and the nucleus-resident APGCE.	
2. If there is an error in the definition of the parameter, NPXAPG issues the INVALID APG PARM message to the master console via the NIPWTOR routine (in IEAVNIPM) and allows the operator to respecify or cancel APG. If the operator cancels APG, NPXAPG examines the next input source or exits (if the operator's input was in error).	NPXAPIOP
3. NPXAPG sets the APG active flag (CVTAPG) to 1. NPXAPG moves the dummy APGCE after the scheduling priority (S.P.) is converted to a dispatching priority (D.P.) by the formula: APG D.P. = 251 - ((15 - S.P.) x 16).	NPXAPEND

Diagram 13.2: NPXTMSL
Build the Required Number of TSCEs (Time Slice Control Elements) in the SQA

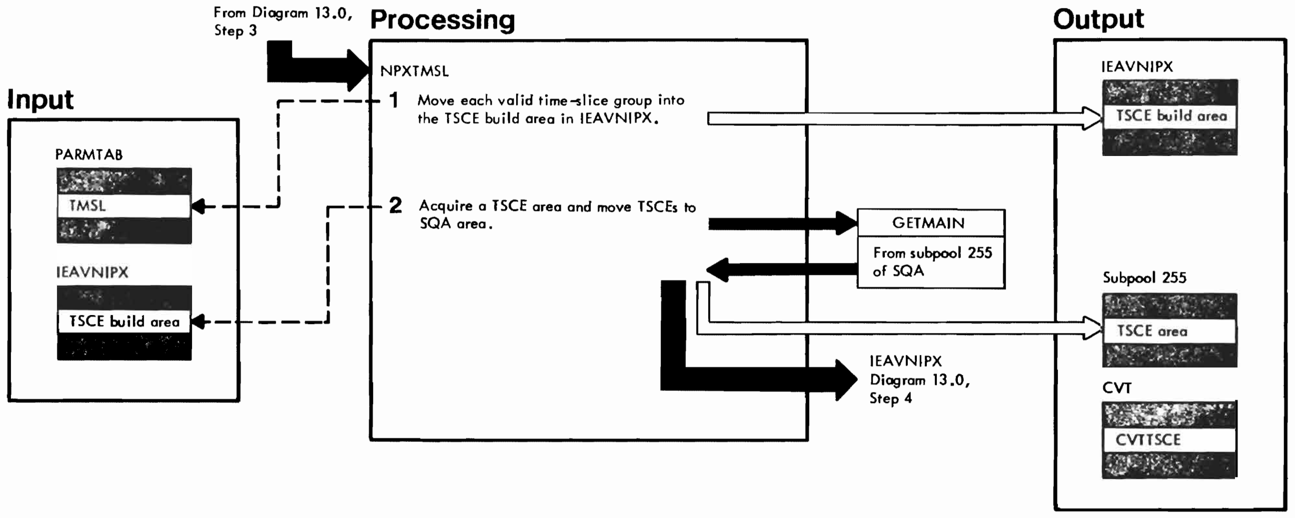

From Diagram 13.0, Step 3
Input
PARMTAB
TMSL
IEAVNIPX
TSCE build area
Processing
NPXTMSL
1 Move each valid time-slice group into the TSCE build area in IEAVNIPX.
2 Acquire a TSCE area and move TSCEs to SQA area.
GETMAIN
From subpool 255 of SQA
IEAVNIPX Diagram 13.0, Step 4
Output
IEAVNIPX
TSCE build area
Subpool 255
TSCE area
CVT
CVTTSCE

Diagram 13.2: NPXTMSL (IEAVNIPX)

Notes on Processing	Label
1. NPXTMSL builds the required number of TSCEs (time slice control elements) in the SQA and defines the time-slice intervals in units of 16 microseconds.	NPXTMSL
If the TMSL parameter is invalid, NPXTMSL terminates its scan at that point and allows the operator to cancel or respecify the TMSL definition. If he cancels it, NPXTMSL returns to the caller. If he respecifies, he need not respecify the time-slice groups defined previous to that in error since NPXTMSL retains the previous definitions in the IEAVNIPX TSCE build area and reuses this area for further TMSL definitions.	NPXTMRTY
NPXTMSL moves each into the build area relative to its priority -- priority 0 in the first slot, priority 1 in the second, etc.	NPXTMLPN
The priority value placed in the TSCE is the dispatching priority (D.P.). This is derived from the scheduling priority (S.P.), 0-13, by the formula:	
Time-Slicing D.P. = 251 - ((15 - S.P.) x 16)	
When the last valid time-slice group has been defined, the TSCEs are packed together in the build area to eliminate zero-value time-slice TSCEs.	

Notes on Processing	Label
2. NPXTMSI requests an area equal in size to the packed TSCEs in the IEAVNIPX build area.	NPXTMEND
NPXTMSL sets a flag in the last TSCE to identify to the dispatcher the end of the chain. NPXTMSL also places the low address of the SQA-resident TSCE chain in the CVT at CVTTSCE.	

Diagram 13.3: NPXQCELL (IEAVNIPX)

Notes on Processing	Label
1. If the quickcell definition is too big for the work area, NPXQCELL terminates the scan at the point of error and allows the operator to cancel or respecify the SQACEL or LSQACEL. If the operator cancels SQACEL, NPXQCELL sets up to process LSQA-CEL. If he cancels LSQACEL, it returns to the caller. If the operator respecifies the parameter, he need not respecify quickcell definitions previous to the one in error since previous definitions are retained and the area is reused for further quick-cell definitions.	NPXQCNXT
When the last valid quickcell has been defined, the definitions are packed together in the work area to eliminate zero-value quick-cells. Any respecification of a given quickcell size results in the last such specification being effective. NPXQCELL calculates the total required area for quick-cells while packing the work area. The first unused halfword after the packed data is set to 0. NPXQCELL also calculates the size of the QCDBLK (quickcell descrip-tion block). QCDBLK contains the requirements for each quickcell.	NPXQCEND
2. NPXQCELL requests an area big enough to contain both the QCDBLK and the quickcells.	NPQCLND
4. NPXQCELL requests an area equal in size to the packed data work area (including the extra halfword of 0's).	NPQCLND

Diagram 13.4: NPXMPA
Define Virtual Address Space Dedicated to Master Scheduler Region

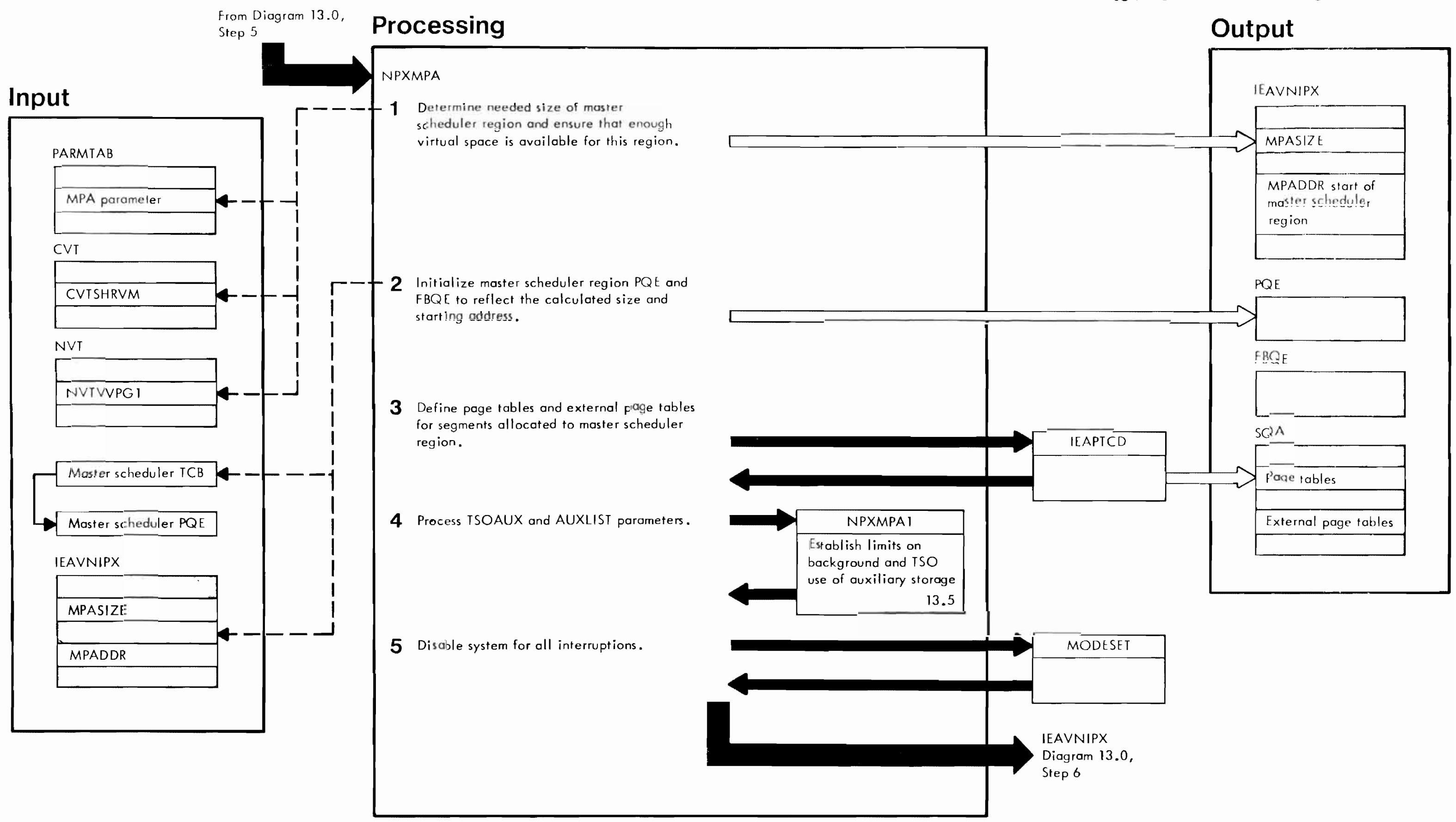

From Diagram 13.0, Step 5
Input
PARMTAB
MPA parameter
CVT
CVTSHRVM
NVT
NVTVVPG1
Master scheduler TCB
Master scheduler PQE
IEAVNIPX
MPASIZE
MPADDR
Processing
NPXMPA
1 Determine needed size of master scheduler region and ensure that enough virtual space is available for this region.
2 Initialize master scheduler region PQE and FBQE to reflect the calculated size and starting address.
3 Define page tables and external page tables for segments allocated to master scheduler region.
4 Process TSOAUX and AUXLIST parameters.
5 Disable system for all interruptions.
IEAPTCD
NPXMPA1
Establish limits on background and TSO use of auxiliary storage
13.5
MODESET
IEAVNIPX Diagram 13.0, Step 6
Output
IEAVNIPX
MPASIZE
MPADDR start of master scheduler region
PQE
FBQE
SQA
Page tables
External page tables

Diagram 13.4: NPXMPA (IEAVNIPX)

Notes on Processing	Label
1. NPXMPA locates the pointer to the MPA parameter. If the pointer has a value of 0, MPA has not been specified, and the master scheduler region is defined as two segments (or 128K bytes) of virtual storage.	NPXNOMPA
If MPA has been specified, NPXMPA ensures that enough virtual space is available for the master scheduler region. NPXMPA calculates the low region address by subtracting the region size from the low address of the "shared virtual area" (CVTSHRVM), which contains the SQA and the pageable LPA. It determines whether this address overlaps the high NIP page (NVTVVPG1). If these address spaces overlap, the master scheduler region cannot be defined as specified. If there are less than 128K bytes, the operator is informed that storage is unavailable, and NPXMPA branches to the NIPSWAIT routine (in IEAVNIPM) to place the system in a disabled wait state (X'38'). If there are at least 128K bytes, the operator is informed that address space has been exceeded, and is given the opportunity to redefine or cancel MPA. If he cancels, the	NPXMPDEF
master scheduler region is defined as 128K bytes by default.	NPXMPINV

Notes on Processing	Label
2. If there are multiple FBQEs defined for this PQE, the more recently created FBQEs (at the front of the FBQE queue) are dequeued, and the SQA space for those FBQEs is released by FREEMAIN (subpool 255). The last FBQE on the queue, created by IEAVNIP0, is retained rather than one of the later-created FBQEs in an attempt to decrease SQA fragmentation.	NPXMPDEF
5. Since the remainder of NIP execution occurs outside of any region boundaries, IEAVNIPX protects its existence by not reenabling the system and not issuing non-type-1 SVCs, which might reenable the system. The exception is in the case of a system wait state. If normal NIP execution continues, the system is not reenabled until control is passed to the master scheduler via the LINK SVC.	DISABLE

Diagram 13.5: NPXMPA1 Establish Limits on Background and TSO use of Auxiliary Storage

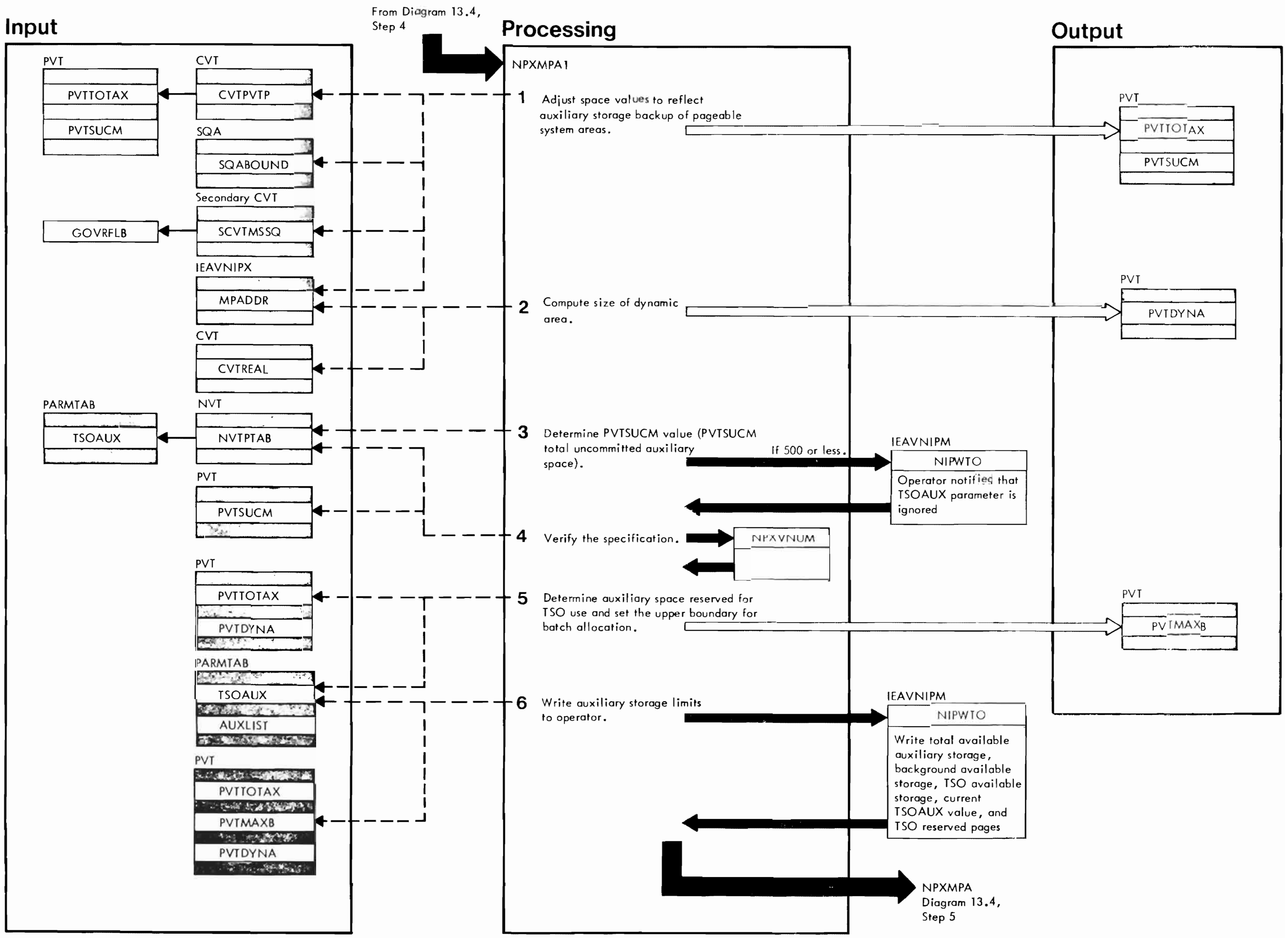
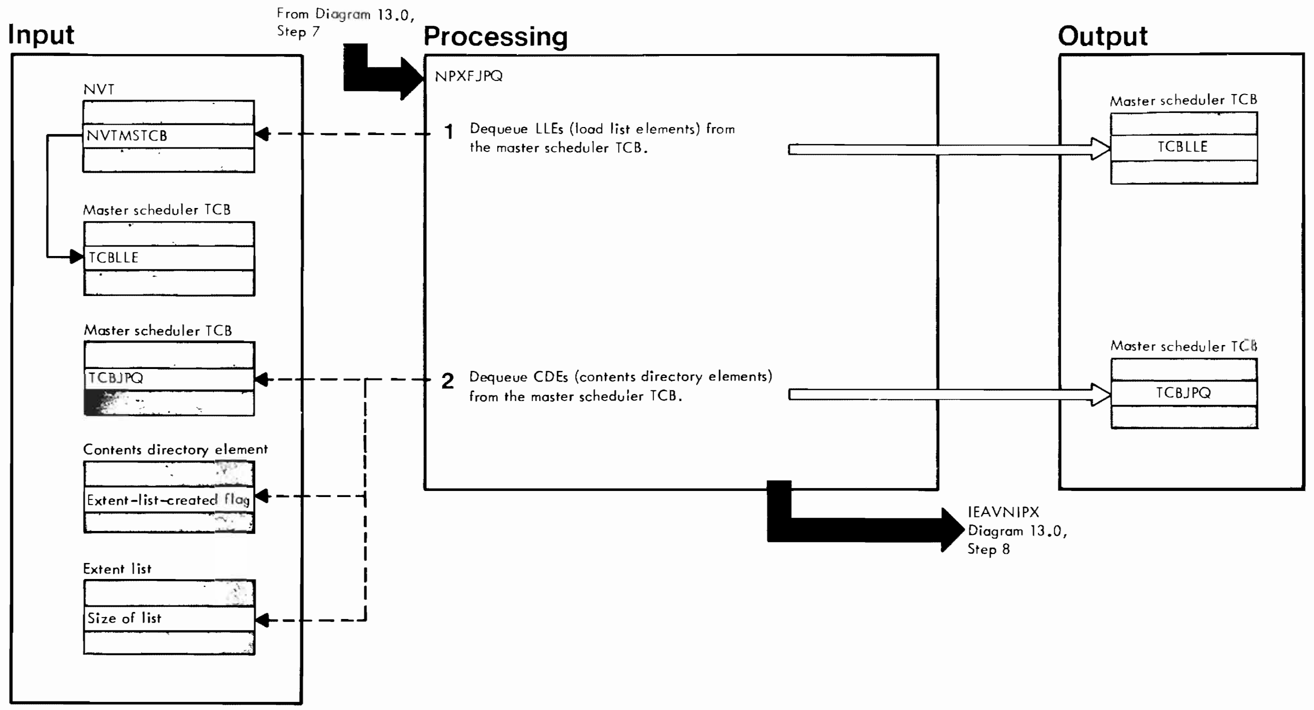

Input
From Diagram 13.4, Step 4
Processing
Output
PVT
PVTTOTAX
PVTSUCM
CVT
CVTPVTP
SQA
SQABOUND
Secondary CVT
GOVRFLB
SCVTMSSQ
IEAVNIPX
MPADDR
CVT
CVTREAL
PARMTAB
TSOAUX
NVT
NVTPTAB
PVT
PVTSUCM
PVT
PVTTOTAX
PVTDYNA
PARMTAB
TSOAUX
AUXLIST
PVT
PVTTOTAX
PVTMAXB
PVTDYNA
NPXMPA1
1 Adjust space values to reflect auxiliary storage backup of pageable system areas.
2 Compute size of dynamic area.
3 Determine PVTSUCM value (PVTSUCM total uncommitted auxiliary space).
If 500 or less.
IEAVNIPM
NIPWTO
Operator notified that TSOAUX parameter is ignored
4 Verify the specification.
NPXVNUM
5 Determine auxiliary space reserved for TSO use and set the upper boundary for batch allocation.
6 Write auxiliary storage limits to operator.
IEAVNIPM
NIPWTO
Write total available auxiliary storage, background available storage, TSO available storage, current TSOAUX value, and TSO reserved pages
NPXMPA Diagram 13.4, Step 5
PVT
PVTTOTAX
PVTSUCM
PVT
PVTDYNA
PVT
PVTMAXB

Diagram 13.5: NPXMPA1 (IEAVNIPX)

Notes on Processing	Label
1. PVTTOTAX and PVTSUCM must be decreased using the formula: PVTTOTAX = PVTSUCM = PVTTOTAX - (Beginning address of SQA - Beginning address of Master Scheduler Region) /4096 Also must subtract pages between the beginning address of the LPA directory and the beginning address of the SQA.	NPXMPA1
2. Computed in 4K pages; should be a multiple of 64K. It is decreased by 16 pages (64K) to reflect the master LSQA. PVTDYNA represents the maximum address space available for region allocation.	
3. If PVTSUCM is not greater than 500 pages, no auxiliary storage is available for TSO use. The operator is notified that the TSOAUX parameter has been ignored due to auxiliary storage restraints. The TSOAUX pointer in PARMTAB is set to 0 to show that the parameter is inoperative (processing for AUXLIST will need to refer to this value). The maximum value for batch allocation (PVTMAXB) is set to PVTTOTAX or PVTDYNA (whichever is smaller), and control goes to Diagram 13.5, Step 5. A null pointer for TSOAUX in PARMTAB causes the upper boundary for background allocation (PVTMAXB) to be set to the smaller of PVTDYNA or PVTTOTAX. Control goes to Diagram 13.5, Step 5.	

Notes on Processing	Label
The value specified must be 0-99. If improper, the TSOAUX entry is set to 0, and the operator is notified of the invalid specification. The operator may enter EOB, canceling the parameter, or he may respecify the parameter. If he cancels it, PVTMAXB is set to the lesser of PVTDYNA or PVTTOTAX. Control goes to Diagram 13.5, Step 5.	
5. This is necessary, since the coding may be entered from multiple paths, and TSOAUX may be present due to operator specification, PARMLIB specification, respecification of an invalid value, or respecification following AUXLIST. AUXLIST always uses PARMTAB to extract the current TSOAUX value. PVTTOTAX and PVTDYNA are compared, and the smaller is multiplied by TSOAUX to find TSO reserved space. The space reservation for TSO must include at least 500 pages for potential batch use. If not, the operator is notified, and PVTMAXB is set to 500 pages. NPXMPA1 tries to locate the AUXLIST parameter. If the parameter is 0, its control is returned to the caller. The operator can accept the auxiliary storage limits as displayed (EOB), or he may respecify TSOAUX. When PVTTOTAX is equal to or less than 500 pages, an operator response is bypassed. (A respecification of TSOAUX automatically implies an AUXLIST). If TSOAUX is respecified, and the operator response is in error and canceled; the auxiliary storage limits are left as previously computed and displayed.	

Diagram 13.6: NPXFJPQ
Release Control Blocks for
NIP-loaded Modules

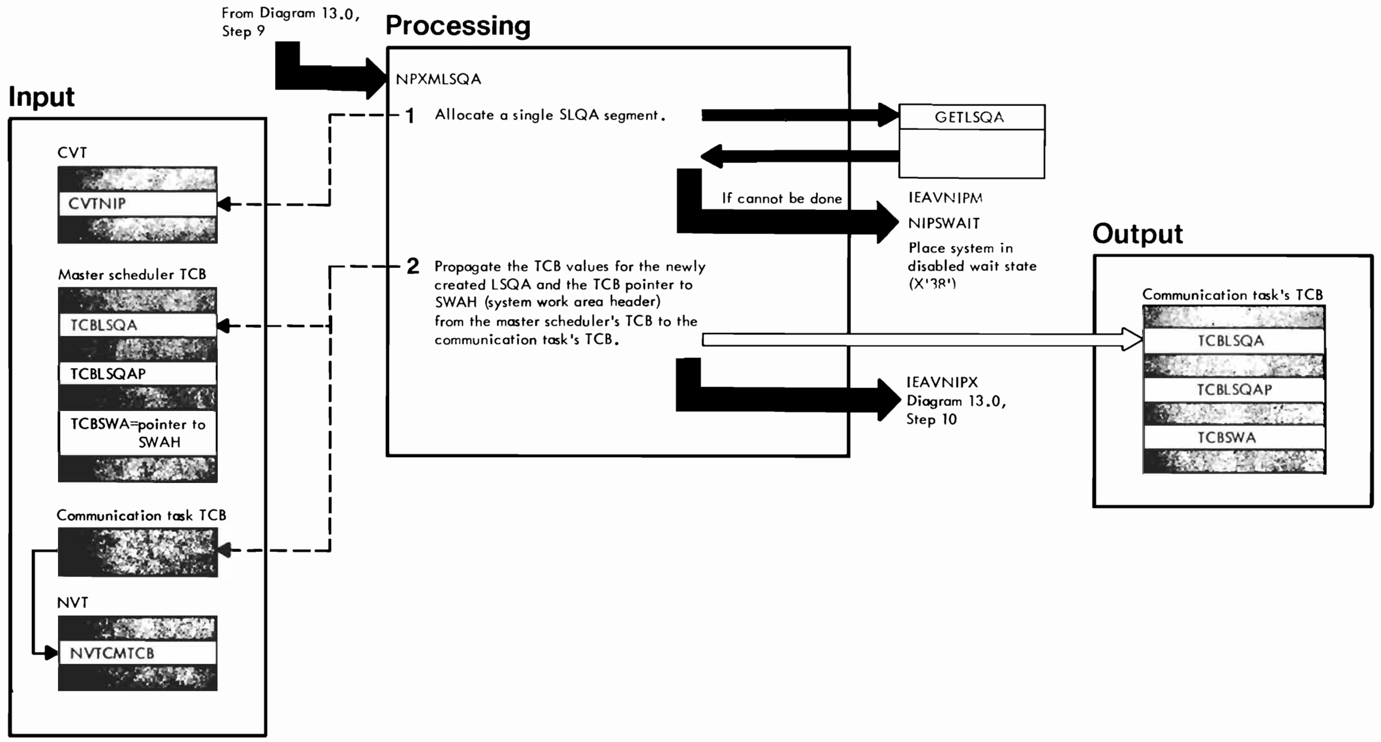


Input
From Diagram 13.0, Step 7
Processing
Output
NPXFJPQ
NVT
NVTMSTCB
Master scheduler TCB
TCBLLE
Master scheduler TCB
TCBJPQ
Contents directory element
Extent-list-created flag
Extent list
Size of list
1 Dequeue LLEs (load list elements) from the master scheduler TCB.
2 Dequeue CDEs (contents directory elements) from the master scheduler TCB.
Master scheduler TCB
TCBLLE
Master scheduler TCB
TCBJPQ
IEAVNIPX Diagram 13.0, Step 8

Diagram 13.6: NPXFJPQ (IEAVNIPX)

Notes on Processing	Label
1. Each LLE is dequeued starting at the TCBLLE queue origin in the master scheduler TCB. Before the storage containing each LLE is released, the TCBLLE pointer is updated with the chain pointer to the next LLE. The processing is completed when all LLEs on the queue have been released, and the TCBLLE pointer has a value of 0.	NPXFJLLE
2. Before dequeueing each CDE, NPXFJPQ determines whether an extent list is associated with the CDE by testing the extent-list-created flag in the CDE. If an extent list exists, its size is extracted from the extent list, and the storage containing the extent list is released. The routine updates the TCBJPQ pointer with the chain pointer to the next CDE and releases storage containing the dequeued CDE. This process continues until TCBJPQ has a value of 0.	NPXFJCDE
3. Storage occupied by released programs is not released until available storage is redefined by the NPXFAREA routine.	NPXFJXIT

Diagram 13.7: NPXMLSQA
Allocate the Single-segment LSQA
Related to Master Scheduler Region



1 Allocate a single SLQA segment.

2 Propagate the TCB values for the newly created LSQA and the TCB pointer to SWAH (system work area header) from the master scheduler's TCB to the communication task's TCB.

Diagram 13.7: NPXMLSQA (IEAVNIPX)

Notes on Processing	Label
1. The CVT NIP-in-process flag (CVTNIP) is set, indicating to the GETLSQA routine that the special NIP interface for creating an LSQA without a TCB is in effect. NPXMLSQA invokes the GETLSQA routine via an SVC request.	NPXMLSQA
Before the wait state, NIPWTO sends a message stating that storage was unavailable for initialization of MPA.	NPXSW38
2. The master scheduler's TCB will have been initialized by GETMAIN with the LSQA definition parameters. The NPXMLSQA routine propagates the values to permit region sharing.	NPXMLSQA

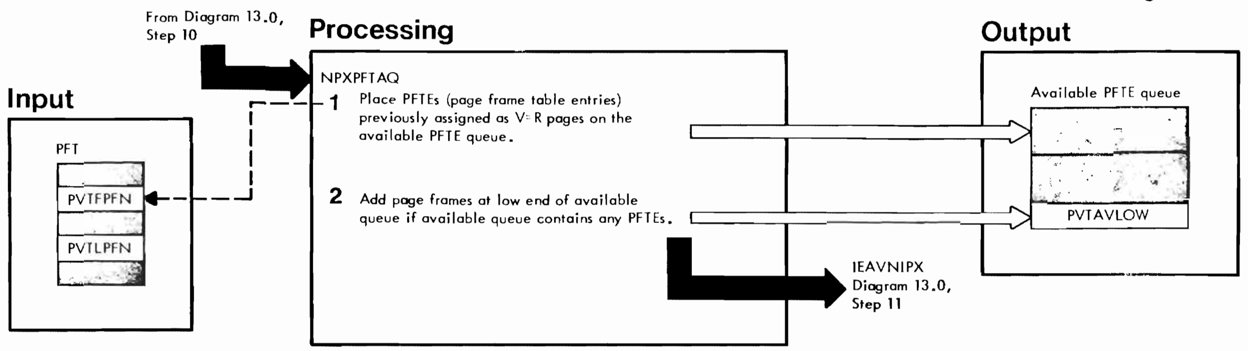

From Diagram 13.0, Step 10
Input
PFT
PVTFPFN
PVTLPFN
Processing
NPXPFTAQ
1 Place PFTEs (page frame table entries) previously assigned as V=R pages on the available PFTE queue.
2 Add page frames at low end of available queue if available queue contains any PFTEs.
IEAVNIPX Diagram 13.0, Step 11
Output
Available PFTE queue
PVTAVLOW

Diagram 13.8: NPXPFTAQ (IEAVNIPX)

Notes on Processing	Label
1. NPXPFTAQ locates the PFT and examines all entries in reverse order (beginning with the last).	NPXPFTAQ
NPXPFTAQ initializes these PFTEs with PFTE indexes for the previously found available PFTE (PFTBQPTR) and the next such PFTE to be found (PFTFQPTR).	NPXPFCUR
2. NPXPFTAQ sets the PFTE index of the first available PFTE (PVTAVFST) to the index of the first PFTE placed on the available queue, if this queue is empty, and sets the backward queue pointer (PFTBQPTR) to 0.	NPXQPFTE
NPXPFTAQ sets the PFTE index of the last available PFTE (PVTAVLOW) to the index of the last PFTE placed on the available queue and sets the forward queue pointer (PFTFQPTR) to 0.	NPXPFCUR
NPXPFTAQ sets the PFTE fields not initialized by the NPXPFTAQ routine to an initial value of 0 and increases the available page frame count (PVTAPC) by the number of PFTEs added to the available queue.	NPXQPFTE

Diagram 13.9: NPXMCSP0 Define Shared Subpool 0 for Master Scheduler and Communication Task

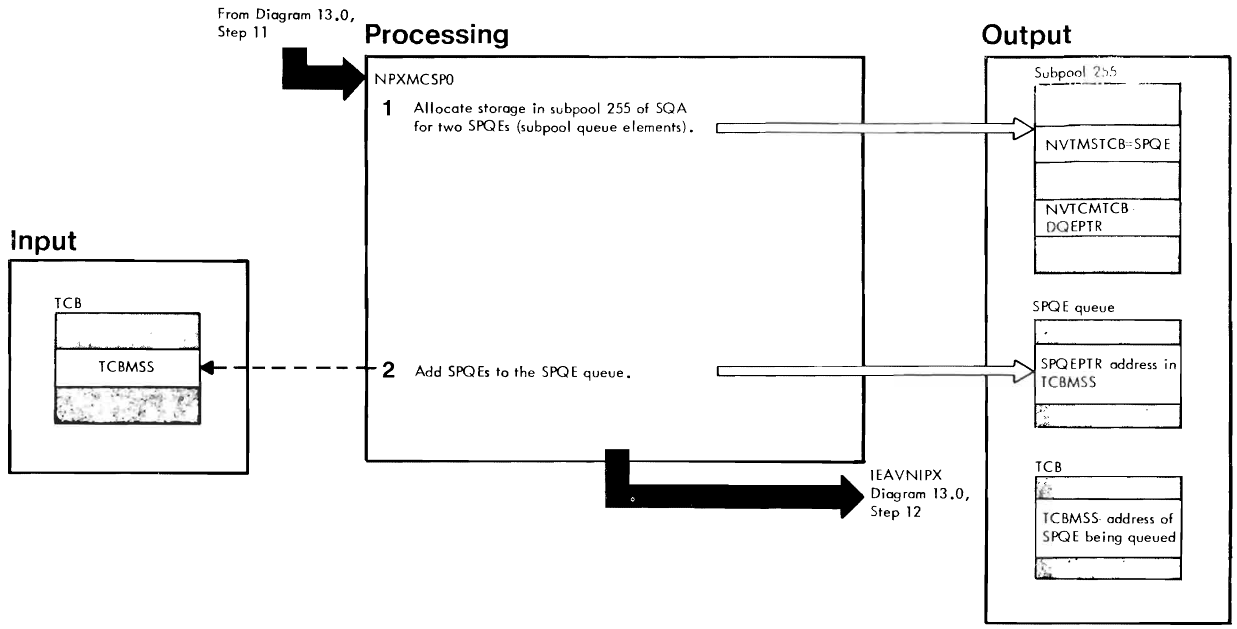
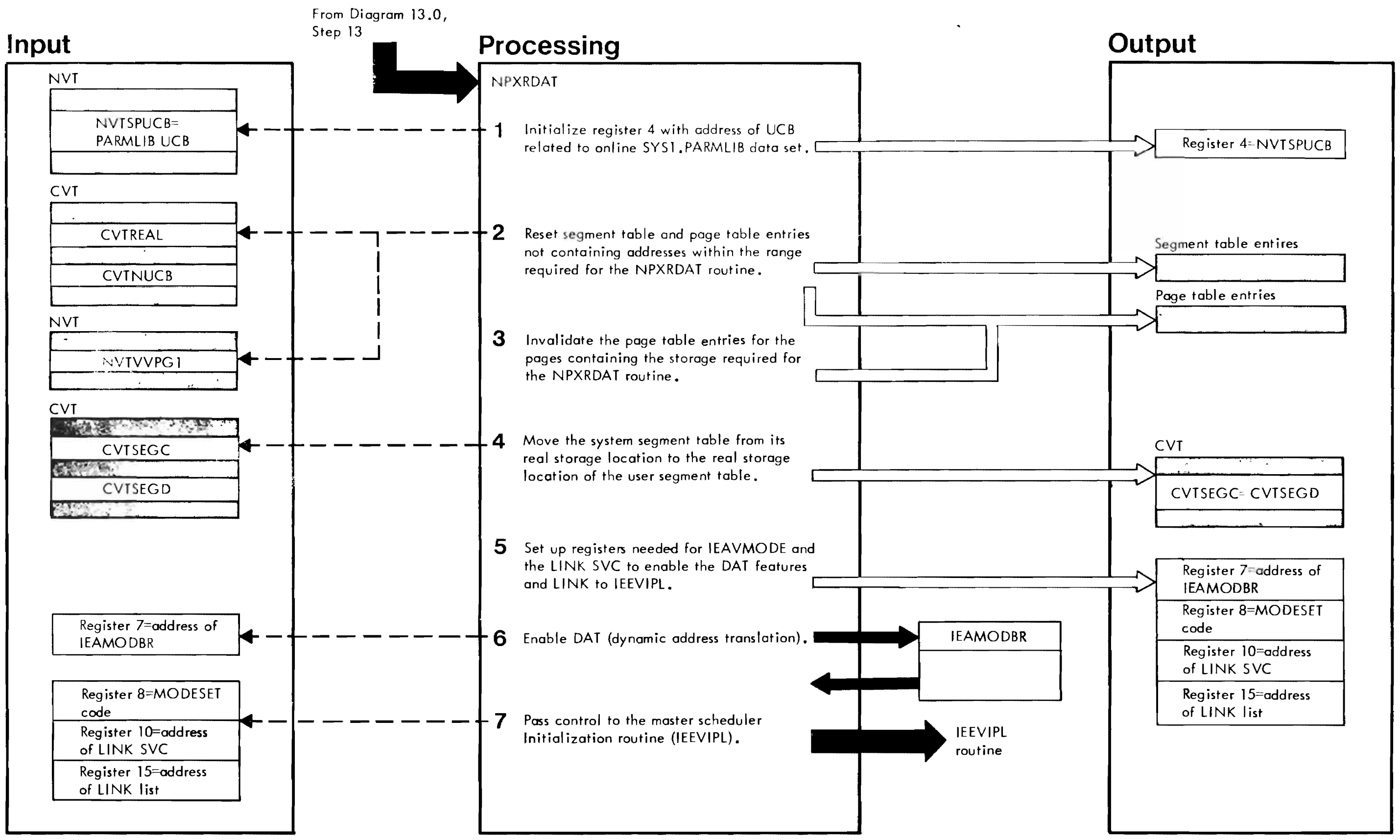

From Diagram 13.0, Step 11
Processing
Output
Input
NPXMCSP0
1 Allocate storage in subpool 255 of SQA for two SPQEs (subpool queue elements).
2 Add SPQEs to the SPQE queue.
TCB
TCBMSS
Subpool 255
NVTMSTCB-SPQE
NVTCMTCB-DQEPTR
SPQE queue
SPQEPTR address in TCBMSS
TCB
TCBMSS-address of SPQE being queued
IEAVNIPX Diagram 13.0, Step 12

Diagram 13.9: NPXMCSP0 (IEAVNIPX)

Notes on Processing	Label
1. The first SPQE area is defined for an owned SPQE for subpool 0 and is added at the head of the SPQE queue for the master scheduler TCB (NVTMSTCB). The second SPQE area is defined for a shared SPQE for subpool 0 and is added at the head of the SPQE queue for the communication task TCB (NVTCMTCB). This SPQE points to the owning, or master scheduler, SPQE (DQEPTR).	NPXMCSP0
2. If the TCBMSS field is 0, the SPQE is flagged as the last SPQE on the SPQE queue.	NPXMCSP0

Diagram 13.10: NPXRDAT
Reset DAT Tables and Exit to Master Scheduler Initialization Routine



1 Initialize register 4 with address of UCB related to online SYS1.PARMLIB data set.

2 Reset segment table and page table entries not containing addresses within the range required for the NPXRDAT routine.

3 Invalidate the page table entries for the pages containing the storage required for the NPXRDAT routine.

4 Move the system segment table from its real storage location to the real storage location of the user segment table.

5 Set up registers needed for IEAVMODE and the LINK SVC to enable the DAT features and LINK to IEEVIPL.

6 Enable DAT (dynamic address translation).

7 Pass control to the master scheduler Initialization routine (IEEVIPL).

Diagram 13.10: NPXRDAT (IEAVNIPX)

Notes on Processing	Label
1. This is the single parameter passed to the IEEVIPL program.	NPXRDAT
2. The DAT tables were created by IEAVNIP0 for all real storage addresses. The pages between the top of the nucleus (CVTNUCB) and the end of NIP's nonpageable region (NVTVVPG1) are invalidated. Page table entries that are below the nonpageable line and are on the available PFTE queue are set to page-unassigned and not GETMAINed status (PGTPAM,PFTPVM). Page table entries that are above the nonpageable line and were previously assigned for NIP's use are set to the same status; however, the page table itself will be released for any complete segment of available storage above the nonpageable line. Whenever an entire page table is released, the corresponding segment table entry is set to a page-table-unassigned status (SGTPAM). With those pages required for NPXRDAT, the associated page tables may not be released during this process, and both the segment table entries and the page table entries remain valid and assigned.	NPXRDAT

Notes on Processing	Label
3. This process operates with the DAT feature disabled. Only real addresses are used for the remainder of NPXRDAT storage references including segment and page table invalidation. NPXRDAT locates the PTEs (page table entries) via the real address of the system segment table (CVTSEGD) by simulating the address translation process to isolate the affected PTEs. The PTEs are set to a page-unassigned-by-GETMAIN status. If this process results in an entire segment invalid condition, the associated segment table entry is set to a page-table-unassigned status.	DATOFF
4. NPXRDAT purges the pre-invalidation values in the translation look-aside buffer by issuing the PTLB instruction. The buffer is used by the hardware to optimize the DAT process.	NPXRDXRS
7. IEAVMODE exits to the LINK SVC in the CVT, which causes NIP to pass control to the master scheduler's IEEVIPL routine. The normal LINK SVC processing ensures that the IEEVIPL routine receives control in key-0 supervisor state.	CVTLINKSC

This section shows modules and subroutines that compose IPL/NIP. It also contains flowcharts for the NIPMOUNT subroutine of IEAVNP02. These flowcharts are included to help explain the levels of subroutines used in NIPMOUNT.

Initial Program Loader

IEAIPL00

Nucleus Initialization Program

IEAVNIP0								
IEAVNIPM	NIPABEND	NIPSQEND	NIPLOAD	NIPSWAIT	NIPSENSE	NIPUCBFN	NIPTIME	NIPWTO
	NIPWTOR	NIPWTOR2	NPMHDCPY	NPMEXCPC	NIPSVC	NIPSVCX		
IEAVNP01	NP1INIT	NP1TESTC	NP1TCOMM					
IEAVNP02	IEAVNPA2	IEAVNPB2						
IEAVNP03	NP3OPSP	NP3PBASE	NP3PMLIB	NP3SYSP	NP3PLMRG	NP3PTAB	NP3LKLIB	NP3LCAT
IEAVNP04	NP4PSCAN	NP4PDEV	NP4PDSEL	NP4IPDT	NP4RQSR1	NP4ALLOC	NP4BPDIT	NP4TLPA
	NP4BCPQ	NP4BPFT	NP4APFT	NP4RQBR2	NP4BSQA	NP4IPVT		
IEAVNPA4	NPA4GBUF	NPA4FREE	NPA4READ	NPA4WRIT	NPA4INTF	NPA4RSTF	NPA4BCCV	NPA4INTC
	NPA4LOAD	NPA4CCST						
IEAVNP05	NP5QSLPA	NP5CSLPA	NP5LPAPT	NP5LPLIB	NP5POST	NP5VTCB	NP5FIX	NP5BLDLF
	NP5MLPA	NP5BLDLP	NP5LPDIR	NP5SVC				
IEAVNPA5	NPA5INIT	NPA5CLPA	NPA5MLPA	NPA5BLDL	NPA5TERM	NPA5ADDR	NPA5POST	NPA5LOAD
	NPA5DIR	NPA5SDIR	NPA5ALPD	NPA5LGRP	NPA5CLIN	NPA5LIND	NPA5ALIS	NPA5BDIR
IEAVNP06	NP6DSS	NP6RMS	NP6DMP	NP6EXCP	NP6TRA	NP6MOVE		
IEAVNP07	NP7HDCPY	NP7PDCM	NP7PAL	NP7OTEST	NP7EPFP	NP7LPAFN		
IEAVNIPX	NPXAPG	NPXTMSL	NPXQCELL	NPXMPA	NPXMPA1	NPXBUF	NPXFJPQ	NPXFAREA
	NPXMLSQA	NPXPFTAQ	NPXMCSP0	NPXRTRAP	NPXRDAT			

Figure 15. Organization of IPL/NIP modules and subroutines

Chart AA. NIPMOUNT Processing (Page 1 of 3)

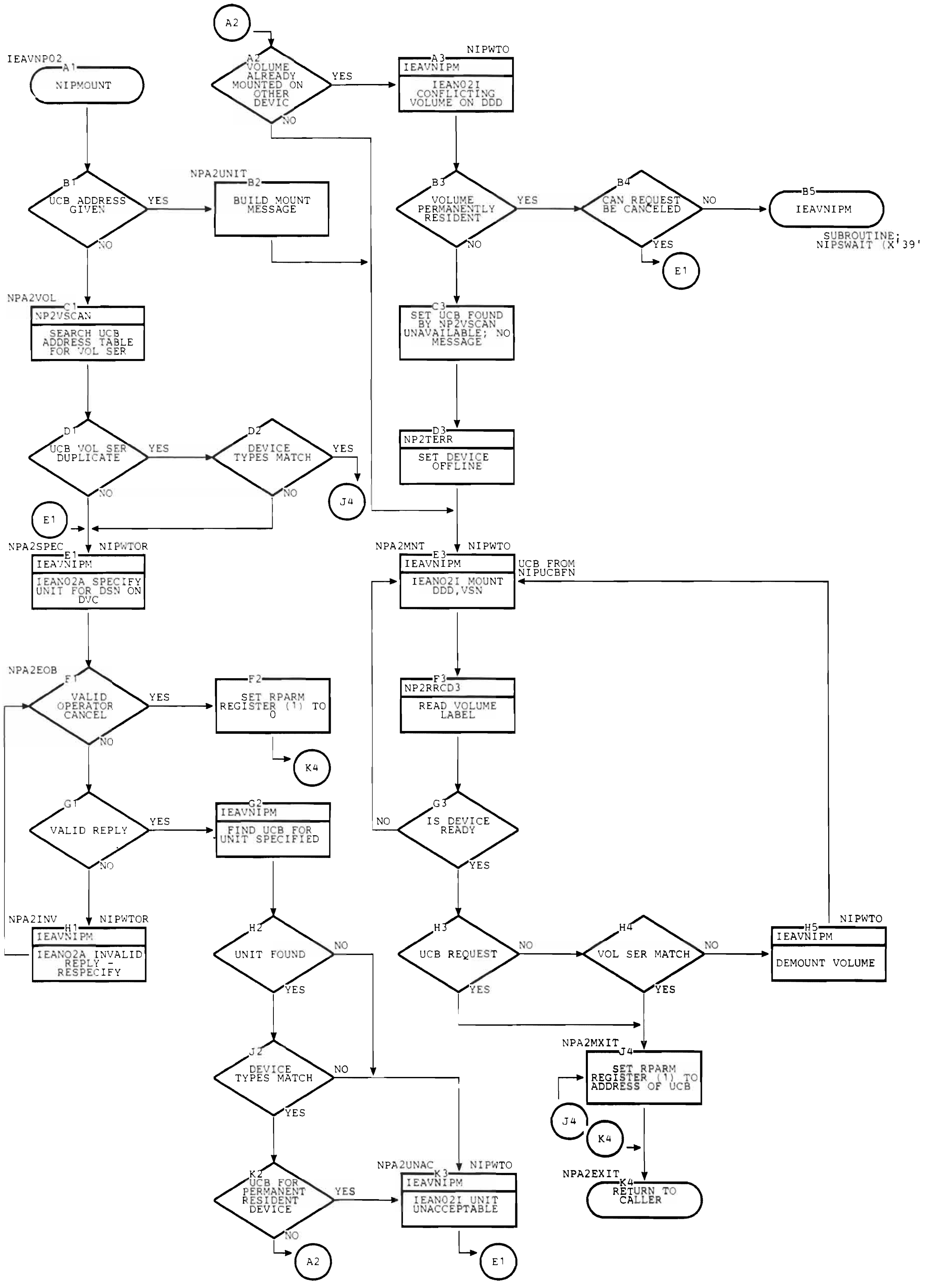


Chart AA. NIPMOUNT Processing (Page 2 of 3)

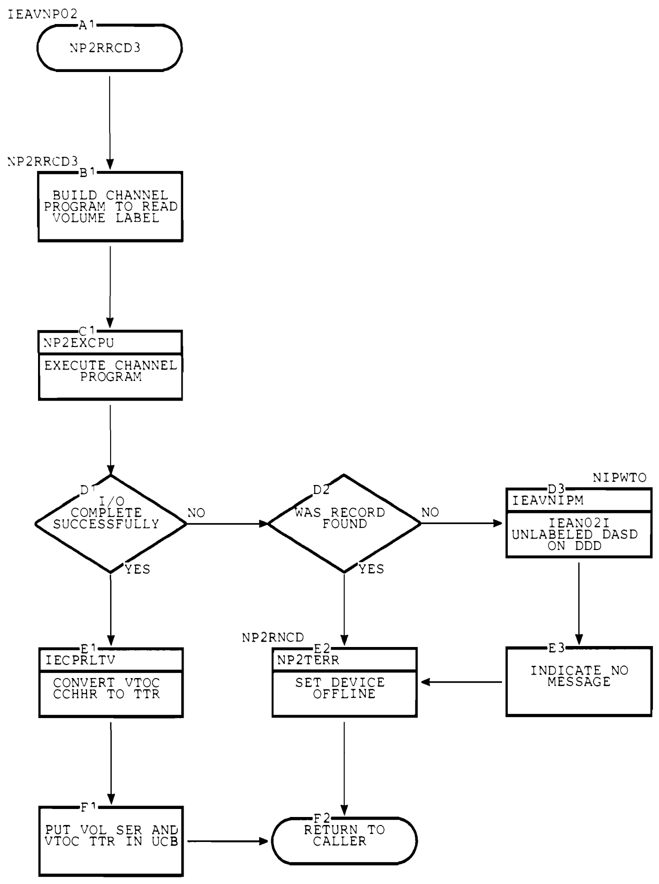
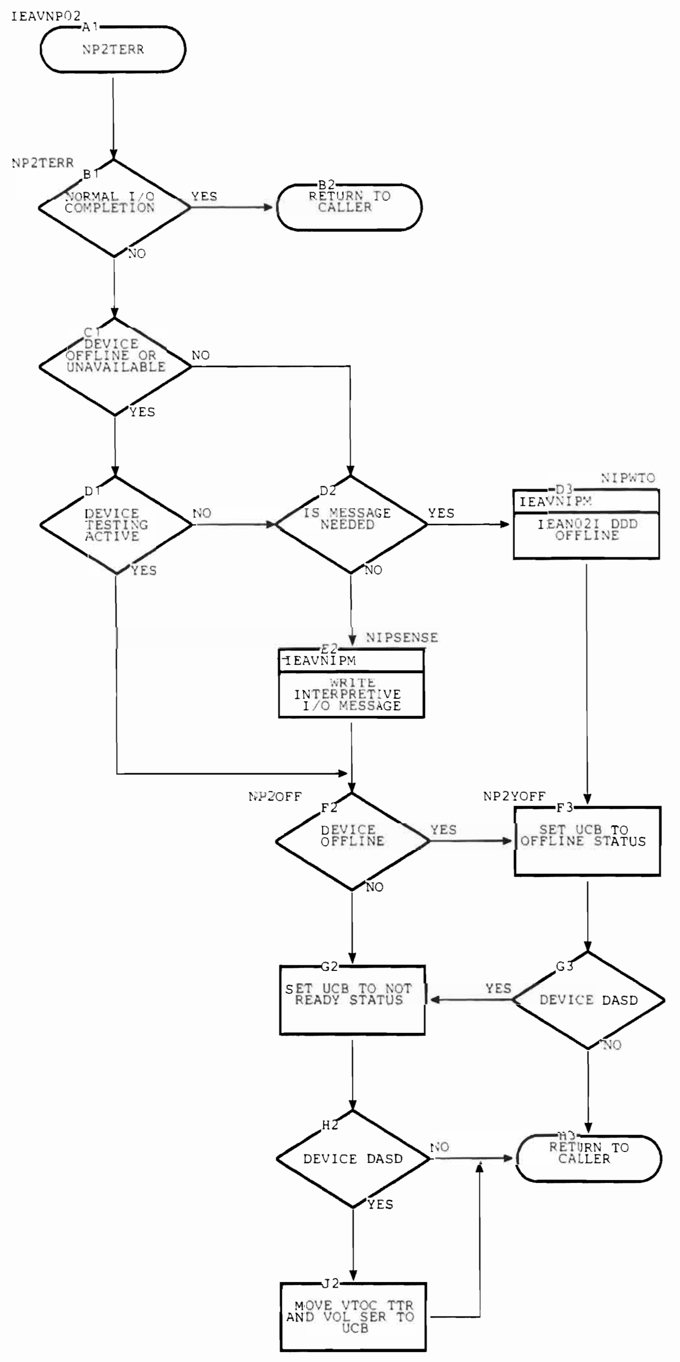


Chart AA. NIPMOUNT Processing (Page 3 of 3)



This section contains a directory of IPL/NIP modules, arranged by entry-point name.

Entry Point	Function of Routine	Module	Diagram No.	Library
IEAIPL00	Unused entry point	IEAIPL00	1.0	Nuc
IEAANIP0	Unused entry point	IEAVNIP0	2.0	Nuc
IEAVNIPM	Normal entry to control routines	IEAVNIPM	3.0	Nuc
IEAVNIPX	Finishes nucleus initialization	IEAVNIPX	13.0	Nuc
IEAVNIP0	Begins nucleus initialization	IEAVNIP0	2.0	Nuc
IEAVNPA2	Mounting a specific DASD or tape	IEAVNP02	5.1	Nuc
IEAVNPA4	I/O requests to SYS1.PAGE data sets	IEAVNPA4	8.0	Nuc
IEAVNPA5	Defines LPA modules for quickstart	IEAVNPA5	10.0	Nuc
IEAVNPB2	Opens a specific data set	IEAVNP02	5.2	Nuc
IEAVNP01	Initializes system consoles	IEAVNP01	4.0	Nuc
IEAVNP02	System library initialization	IEAVNP02	5.0	Nuc
IEAVNP03	Analyzes system parameters	IEAVNP03	6.0	Nuc
IEAVNP04	Initializes paging control blocks	IEAVNP04	7.0	Nuc
IEAVNP05	Initializes shared storage	IEAVNP05	9.0	Nuc
IEAVNP06	Initializes reliability and serviceability features	IEAVNP06	11.0	Nuc
IEAVNP07	Processes HARDCPY and PAL parameters	IEAVNP07	12.0	Nuc
NIPABEND	From SVC SLIH for ABEND	IEAVNIPM	3.0	Nuc
NIPLOAD	Loads NIP processor modules	IEAVNIPM	3.1	Nuc
NIPSENSE	Interprets sense data after I/O failure	IEAVNIPM	3.0	Nuc
NIPSQEND	Abnormal entry from GETMAIN	IEAVNIPM	3.0	Nuc
NIPSVC	Loads requested type 3 or 4 SVC routine into real storage	IEAVNIPM	3.2	Nuc
NIPSWAIT	Stores system completion code, places system in disabled wait state	IEAVNIPM	3.0	Nuc
NIPTIME	Determines correct time from TOD clock	IEAVNIPM	3.4	Nuc
NIPUCBFN	Locates UCB associated with device	IEAVNIPM	3.3	Nuc
NIPWTO	Writes messages to operator	IEAVNIPM	3.0	Nuc

Entry Point	Function of Routine	Module	Diagram No.	Library
NIPWTOR	Writes messages to operator and receives reply	IEAVNIPM	3.0	Nuc
NIPWTOR2	Moves operator reply to specified buffer	IEAVNIPM	3.0	Nuc
NPA4FCEA	Channel end appendage for formatting SYS1. PAGE data sets	IEAVNPA4	8.0	Nuc
NPA4FXCA	Abnormal end appendage for formatting SYS1. PAGE data sets	IEAVNPA4	8.0	Nuc
NPA4NOAP	Dummy I/O appendage used as SIO, PCI, and end of extent appendage	IEAVNPA4	8.0	Nuc
START	Begin initial program load	IEAIPL00	1.0	Nuc

SECTION 5: DATA AREAS

This section contains information about, and descriptions of, the following Data Areas:

IPLDATA
NIPMOUNT Parameter List
NIPOPEN Parameter List
NIPWTO Message Header
NVT (NIP Vector Table)
PDIT (Page Device Information Table)
Page Device Table
NIP PARMAREA
PARMTAB (Parameter Address Table)
Parameter Table Entry
Quickstart Record 1 (NIPQSR1)
Quickstart Record 2 (NIPQSR2)
Quickstart Record 3 (NIPQSR3)
Slot Queue
SPE (NIP System Parameter Entry)

IPLDATA

Pointed to by:	Register 3 on entry to IEAVNIP0
Size:	20 bytes
Initialized by:	IEAIPL00

Displacement	Bytes and Bit Pattern	Field Name	Description
0(0)	6		IPL unit volume serial
6(6)	5		IPL unit VTOC CCHHR
11(B)	1		Nucleus member ID
12(C)	4		SYS1.NUCLEUS data set start CCHH
16(10)	4		SYS1.NUCLEUS data set end CCHH

NIPMOUNT PARAMETER LIST

Pointed to by:	Register 1 (RPARM)
Size:	21 bytes
Initialized by:	Module calling NIPMOUNT (IEAVNPA2)

Displacement	Bytes and Bit Pattern	Field Name	Description
0(0)	12	NMNTDS	The left-justified data set name in EBCDIC
0(0)	4	NMNTDSA	When NMNTFLI flag is set, the address of a 44-byte data set name
12(C)	6	NMNTVS	The volume serial (EBCDIC) of the direct access device
18(12)	2	NMNTDT	The device type of the volume to be mounted or 0 if any device type is acceptable
20(14)	1	NMNTFL	NIPMOUNT flags:
	.... .1..	NMNTFLI	The parameter list contains the address of a 44-byte data set name
	.... ..1.	NMNTFLC	Operator may cancel the mount request via an EOB reply
	1	NMNTFLB	UCB address given

NIPOPEN PARAMETER LIST

Pointed to by:	Register 1 (RPARM)
Size:	24 bytes
Initialized by:	Module calling NIPOPEN

Displacement	Bytes and Bit Pattern	Field Name	Description
0(0)	12	NOPNDS	The left-justified data set name in EBCDIC, or
0(0)	4	NOPNDSA	Address of a 44-byte data set name if NOPNFLI flag is set
12(C)	4	NOPNDCB	Address of the related DCB
16(10)	4	NOPNUCB	Address of the UCB for the device on which the volume is to be mounted
20(14)	1	NOPNFL	NIPOPEN flags:
	..1.	NOPNFLM	Data set not found message is to be suppressed (valid only for conditional requests)
	...1	NOPNFLI	The parameter list contains a 44-byte data set name
	.... 1...	NOPNFLNB	Construct entire DEB (DCB must point to build area or DEB will be built at end of nucleus)
	.... .1..	NOPNFLB	DEB to be added to end of nucleus
	.... ..1.	NOPNFLC	Conditional request; if NIPOPEN fails, register 1 (RPARM) will be set to two's complement
	1	NOPnFLSX	Data set description to be limited to first extent
21(15)	1	NOPNRC	NIPOPEN return code:
	0000 0000		Successful
	0000 0100		Data set not found
	0000 1000		I/O error

NIPWTO MESSAGE HEADER

Pointed to by:	Register 1 (RPARM) or, if NIPWTCR request, by NIPWTOR parameter list
Size:	4 bytes
Initialized by:	Module calling NIPWTO

Displacement	Bytes and Bit Pattern	Field Name	Description
0(0)	2	NWTOLNG	Message length including header
2(2)	1	NWTOFL	Message flag:
	1...	NWTOFLNH	Message is not to be hardcopied
3(3)	1		Reserved
4(4)	Variable	NWTOMSG	Message text

NIPWTOR PARAMETER LIST

Pointed to by:	Register 1 (RPARM)
Size:	12 bytes
Initialized by:	Module calling NIPWTOR

Displacement	Bytes and Bit Pattern	Field Name	Description
0(0)	4	NWTORRPA	Address of the area to contain the reply
4(4)	4	NWTORECB	Address of the reply ECB
8(8)	2	NWTORRDL	The length in bytes of the reply area
10(A)	1	NWTORFL	NIPWTOR flags:
	1...	NWTORFLA	NIPWTOR is not to wait for reply completion (asynchronous request)
	.... 1...	NWTORFLB	NIPWTOR2 is to provide an SQA reply buffer
11(B)	1		Reserved

NIP VECTOR TABLE (NVT)

Pointed to by: Register 2 (RNVT)
Size: 310 bytes (approx.)
Initialized by: IEAVNIP0

Displacement	Bytes and Bit Pattern	Field Name	Description
0(0)	8	NVTNPSUF	List of suffixes to be appended to the base IEAVNP to form the name of the NIP modules to be used next
8(8)	1	NVTNPSFX	Index to be used in determining which suffix in NVTNPSUF is to be appended next
9(9)	1	NVTPXIT	Suffix to be appended to form the name of the exit module IEAVNIPX
10(A)	1	NVTNPATR	Attributes of executing module:
	.1..	NVTNPREN	Reenterable
	..1.	NVTNPREU	Reusable
11(B)	1	NVTFLLB	Library status indicator:
	1...	NVTFLSLB	SVCLIB and LOGREC defined
12(C)	4	NVTMSTCB	Address of master scheduler TCB
16(10)	4	NVTCMTCB	Address of communications task TCB
20(14)	4	NVTAPGCE	Address of APG table
24(18)	4	NVTVBLDL	Address of BLDL table
28(1C)	4	NVTIGCER	Address of SVC error routine
32(20)	4	NVTPTCD	Address of Create/Destroy Page Table routine
36(24)	4	NVTVVMDI	Address of LPA hash value
40(28)	4	NVTMSLNK	Address of link parameter List
44(2C)	4	NVTPQCI	Address of Quickcell Initialization routine
48(30)	4	NVTDSSNG	Address of DSS Mask-Out routine
52(34)	12		Reserved
64(40)	1	NVTSQANO	Number of initial SQA pages
65(41)	1	NVTNRANO	Number of available NIP pages
66(42)	2	NVTIPGNO	Number of initial buffer pages
68(44)	4	NVTABSAV	Save area for original SVC 13 entry in SVC table
72(48)	4	NVTPQSAV	Save area for original get-SQA-page entry in page vector table
76(4C)	16		Reserved
92(5C)	4	NVTNUCND	Address of next available byte in nucleus buffer
96(60)	4	NVTNBFND	Address of end of nucleus buffer
100(64)	4	NVTVVPG1	Address of first pageable page
104(68)	16		Reserved
120(78)	2	NVTTRACE	Number of trace table entries
122(7A)	1	NVTFLSG	Reserved
123(7B)	1	NVTFLCN	Message handling indicators:
	1...	NVTFLAC	Active master console
	.1..	NVTFLIOC	Composite master console
	.... 1...	NVTFLNHC	Hardcopy discontinued
	.... .1..	NVTFLNCK	TOD clock inoperative
	.... ..1.	NVTFLRAC	WTOR reply outstanding
124(7C)	4		Reserved
128(80)	8	NVTWTPSW	System wait state PSW NIP modifies the following 4 bytes of the PSW
132(84)	2	NVTIDPSW	PSW ID (NIP module identifier)
134(86)	2	NVTFLWS1	System wait state code
136(88)	4	NVTLOAD	Address of NIPLOAD in IEAVNIPM
140(8C)	4	NVTSENSE	Address of NIPSENSE in IEAVNIPM
144(90)	4	NVTSWAIT	Address of NIPSWAIT in IEAVNIPM
148(94)	4	NVTTIME	Address of NIPTIME in IEAVNIPM
152(98)	4	NVTUCBFN	Address of NIPUCBFN in IEAVNIPM
156(9C)	4	NVTWTO	Address of NIPWTO in IEAVNIPM
160(A0)	4	NVTWTOR	Address of NIPWTOR in IEAVNIPM
164(A4)	4	NVTWTOR2	Address of NIPWTOR2 in IEAVNIPM
168(A8)	4	NVTPAGIO	Address of IEAVNPA4
172(AC)	8	NVTNIPM	Save area for IEAVNIPM base register
180(B4)	4	NVTNMBLD	NIPM BLDL entry
184(B8)	16		Reserved
200(C8)	4	NVTDCBIC	Address of input console DCB

204(CC)	4	NVTDCBOC	Address of output console DCB
208(D0)	4	NVTDCBSN	Address of SYS1.NUCLEUS DCB
212(D4)	4	NVTMBUF	Address of next available byte in message buffer
216(D8)	4	NVTMBEND	Address of end of NIP message buffer
220(DC)	4	NVTSPE	Address of first SPE in queue
224(E0)	4		Reserved
228(E4)	4	NVTTOD	Save area for TOD clock readings
232(E8)	2	NVTABCD1	Save area for first ABEND code processed
234(EA)	1	NVTABWS1	Save area for wait state code upon entry to NIPABEND in IEAVNIPM
235(EB)	1		Reserved
236(EC)	4	NVTIPDT	Address of initial page device table
240(F0)	4	NVTPAREA	Address of first PARMAREA obtained (by IEAVNP03)
244(F4)	4	NVTPTAB	Address of PARMTAB
248(F8)	4	NVTQSBUF	Address of quickstart buffer
252(FC)	2		Reserved
254(FE)	2	NVTSPUCB	Address of SYS1.PARMLIB UCB
256(100)	4	NVTVVTCB	Address of IEAVNPA5 TCB (pageable TCB)
260(104)	4	NVTVVECB	Address of ECB for IEAVNPA5
264(108)	4	NVTVRECB	Address of ECB for IEAVNP05
268(10C)	4	NVTVRBLD	Address of LPA BLDL entry
272(110)	4	NVTBLDL	Address of BLDL table build area
276(114)	4	NVTCSLIB	Address of SYS1.LPALIB DCB
280(118)	4	NVTCSLNM	Address of current LPA name
284(11C)	4	NVTCSIOB	Address of IOB for failing coldstart I/O requests
288(120)	16		Reserved
304(130)	1	NVTFLPO	Parameter option indicators:
	1...	NVTFLLST	Display PARMLIB lists
	.... 1...	NVTFLQS	LPA may be quickstarted

PAGE DEVICE INFORMATION TABLE

Displacement	Bytes and Bit Pattern	Field Name	Description
0(0)	40	PDITIOB	IOB for this device
40(28)	1	PDITFLG1	Flags:
	1...	PDITSSDV	This is a set sector device
	.1..	PDITMHDV	Moveable head device
	..1.	PDITXCP	Device needs to be restarted via EXCP
	...1	PDITPON	Slot queues are being searched for a primary slot
	.... 1...	PDITOAPF	Channel program appended
	.... .1..	PDITMIOB	Multiple-exposure device
	.... ..1.	PDITNOPS	No primary or secondary slot found on slot queue
41(29)	1	PDITINDX	Displacement used to determine base IOB for this device; base IOB = address of IOB for this exposure - (4 X PDITINDX)
42(2A)	2	PDITIOBP	Displacement to find IOB for next exposure (current IOB + PDITIOBP)
44(2C)	1	PDITCPCT	Number of channel programs chained to this IOB after a PCI had been set
45(2D)	3	PDITLACP	Address of the last channel program on chain for IOB
48(30)	4	PDITECB	ECB for this device
52(34)	48	PDITDEB	DEB for this device
100(64)	4	PDITDCB	DCB for this device
104(68)	1	PDITSCNT	Number of slot queues for this device
105(69)	3	PDIT1SQA	Address of slot queue for first slot for this device
108(6C)	1	PDITSQCT	Number of channel programs on slot queue containing the most channel programs
109(6D)	3	PDITHSQA	Address of slot queue containing the most channel programs

PAGE DEVICE TABLE

Displacement	Bytes and Bit Pattern	Field Name	Description
0(0)	1	PDTNO	Device number used to index into the PDT
1(1)	1	PDTLSN	Last assigned slot number
2(2)	2	PDTAPC	Number of available pages on this device
4(4)	2	PDTLGN	Last assigned group number
6(6)	2	PDTSEL	Slot entry length
8(8)	1	PDTALI	Alternate slot increment
9(9)	1	PDTTG	Number of tracks per group
10(A)	1	PDTFL1	Flags:
	1...	PDTDEVT1	Primary device
	.1..	PDTDEVT2	Fixed-head device
	..1.	PDTLAST	Last PDTE
11(B)	5	PDTBA	Beginning CCHHR of paging data set on this device
16(10)	2	PDTR1	Reserved
18(12)	2	PDTGC	Number of groups per cylinder
20(14)	1	PDTSG	Number of slots per group
21(15)	3	PDTCCVA	Address of cylinder count vector
24(18)	1	PDTR2	Reserved
25(19)	3	PDTBMA	Address of bit map for this device
28(1C)	1	PDTDT	Device type field from UCB
29(1D)	3	PDTIOB	Address of IOB for this device

NIP PARMAREA

Pointed to by:	NVTPAREA
Size:	Variable
Initialized by:	IEAVNP03

Displacement	Bytes and Bit Pattern	Field Name	Description
0(0)	4	NIPPAQ	Address of next 4K block in parameter area
4(4)	4	NIPPABYT	Address of next available byte in PARMTAB
	****************PARMLIB BLDL Entry****************		
8(8)	4	NIPPABDH	BLDL header
12(C)	8	NIPPANAM	PARMLIB member name
20(14)	3	NIPPATTR	Member TTR entry
23(17)	3		Reserved for BLDL entry
26(1A)	6		Reserved
	****************I/O Data Area****************		
32(20)	32	NIPPATXT	CCWs for reading text record
64(40)	5	NIPPASID	Search ID for next text record
69(45)	3		Reserved
72(4C)	40	NIPPAIOB	PARMLIB IOB
112(70)	4	NIPPAIOB	PARMLIB DCB
116(74)	4	NIPPAECB	PARMLIB ECB
120(78)	48	NIPPADEB	PARMLIB DEB
168(A8)	80	NIPPARCD	Read area for PARMLIB records
248(F8)	232	NIPPAPTB	Parameter address table
480(1E0)	1568	NIPPABUF	Initial PARMAREA buffer

PARAMETER ADDRESS TABLE (PARMTAB)

Pointed to by:	Within PARMAREA
Size:	232 bytes
Initialized by:	IEAVNP03

Displacement	Bytes and Bit Pattern	Field Name	Description
248(F8)	8	PTABSTRT	Dummy entry
256(100)	4	APGP	Address of automatic priority group parameters specified via SYSP (PLIBTAB)
260(104)	2		APGP flags
262(106)	2		Source identification:
	0000 0000		Operator (CPERTAB)
	....		IEASYSxx via SYSP
264(108)	4	APGO	Address of automatic priority group parameters specified by operator
268(10C)	2		APGO flags
270(10E)	2		Source ID
272(110)	4	AUXLIST	Not applicable
276(114)	2		AUXLIST flags
278(116)	2		Source ID
280(118)	4	BLDL	Address of PARMLIB member for BLDL
284(11C)	2		BLDI flags
286(11E)	2		Source ID
288(120)	4	BLDLF	Fixed BIDI list
292(124)	2		BLDLF flags
294(126)	2		Source ID
296(128)	4	CLPA	Address of create link pack area parameter string
300(12C)	2		CLPA flags
302(12E)	2		Source ID
304(130)	4	CONT	Line continuation
308(134)	4		Reserved
312(138)	4	CPQE	Address of channel program queue extension parameters
316(13C)	2		CPQE flags
318(13E)	2		Source ID
320(140)	4	DUMP	Address of tape for SYS1.DUMP
324(144)	2		DUMP flags
326(146)	2		Source ID
328(148)	4	FIX	Address of list of routines in fixed LPA
332(14C)	2		FIX flags
334(14E)	2		Source ID
336(150)	4	HARDCPY	Address of hardcopy log parameters
340(154)	2		HARDCPY flags
342(156)	2		Source ID
344(158)	4		Reserved
348(15C)	2		
350(15E)	2		
352(160)	4	LSQACEL	LSQA quickcell parameters
356(164)	2		LSQACEL flags
358(166)	2		Source ID
360(168)	4	MLPA	Address of list of routines in LPA extension
364(16C)	2		MLPA flags
366(16E)	2		Source ID
368(170)	4	MPA	Master scheduler region virtual space
372(174)	2		MPA flags
374(176)	2		Source ID
376(178)	4	OPI	Operator intervention
380(17C)	2		OPI flags
382(17E)	2		Source ID
384(180)	4	PAGEP	Address of page data set parameters from IEASYSxx via SYSP (PIIBTAB)
388(184)	2		PAGEP flags
390(186)	2		Source ID
392(188)	4	PAGEO	Address of page data set parameters from OPERTAB
396(18C)	2		PAGEO flags
398(18E)	2		Source ID

400(190)	4	PALP	Address of paging algorithm limits from IEASYSxx via SYSP (PLIBTAB)
404(194)	2		PALP flags
406(196)	2		Source ID
408(198)	4	PALO	Address of paging algorithm limits from operator (OPERTAB)
412(19C)	2		PALO flags
414(19E)	2		Source ID
416(1A0)	4	REAL	Nonpageable address limit
420(1A4)	2		REAL flags
422(1A6)	2		Source ID
424(1A8)	4	SQA	Address of SQA virtual space
428(1AC)	2		SQA flags
430(1AE)	2		Source ID
432(1B0)	4	SQACEL	SQA quickcell
436(1B4)	2		SQACEL flags
438(1B6)	2		Source ID
440(1B8)	4	SYSP	Address of list of IEASYSxx member names
444(1BC)	2		SYSP flags
446(1BE)	2		Source ID
448(1C0)	4	TMSL	Address of time-slice group parameters
452(1C4)	2		TMSL flags
454(1C6)	2		Source ID
456(1C8)	4	TRACE	Address of trace table entry parameters
460(1CC)	2		TRACE flags
462(1CE)	2		Source ID
464(1D0)	4	TSOAUX	TSOAUX parameter list
468(1D4)	2		TSOAUX flags
470(1D6)	2		Source ID
472(1D8)	4	PTABEND	Dummy entry for end

PARAMETER TABLE ENTRY

Each Parameter Table Entry is eight bytes long and has the following format:

Displacement	Bytes and Bit Pattern	Field Name	Description
0(0)	4	NIPPTADR	Address of parameter specifications
4(4)	1	NIPPTOPF	Entry flags:
	1...	NIPPTOPI	OPI option
	.1..	NIPPTLST	List option
5(5)	1	NIPPTATF	Attribute flags:
	1...	NIPPTMRG	Merge
6(6)	2	NIPPTSID	Source ID:
	X'0000'		Operator
	C'xx'		PLIB member suffix

QUICKSTART RECORD 1 (NIPQSR1)

Pointed to by:	None
Size:	4096 bytes
Initialized by:	IEAVNP04

Displacement	Bytes and Bit Pattern	Field Name	Description
0(0)	8	NIPQSR1N	"PAG1" EBCDIC record identifier
8(8)	1	NIPQSR1F	Device identification indicators:
	1...		Primary page device
	.1..		Page device with LPA quickstart records NIPQSR2 and NIPQSR3
9(9)	1		Reserved
10(A)	2	NIPQSR1T	Number of formatted tracks
12(C)	2	NIPQSR1A	Number of available pages (APC)
14(E)	2	NIPQSR1L	Number of halfwords in bit map beginning at NIPQSR1M
16(10)	4		Reserved
20(14)	3	NIPQSR1P	TTR of NIPQSR2
23(17)	1		Reserved
24(18)	Variable	NIPQSR1M	Bit map of page availability
	0	NIPQSR1A	Page record available
	1	NIPQSR1U	Page record unavailable due to I/O error on initial page write

QUICKSTART RECORD 2 (NIPQSR2)

Pointed to by:	NIPQSR1P
Size:	4096 bytes
Initialized by:	IEAVNP04

Displacement	Bytes and Bit Pattern	Field Name	Description
0(0)	8	NIPQSR2N	"PAG2" EBCDIC record identifier
8(8)	4	NIPQSR2L	Virtual address assigned to the start of the LPA directory
12(C)	4	NIPQSR2H	Highest virtual address assigned to the LPA
16(10)	4	NIPSWR2D	Hash value for LPA directory
20(14)	3	NIPQSR2P	TTR of first NIPQSR3 record
23(17)	1		Reserved
24(18)	Variable	NIPQSR2M	Bit map of pages
	0	NIPQSR2A	Page record available
	1	NIPQSR2U	Page record unavailable (assigned to quickstart LPA)

QUICKSTART RECORD 3 (NIPQSR3)

Pointed to by:	NIPQSR2P or NIPQSR3P
Size:	4096 bytes (There will be 1 to 3 NIPQSR3 Records depending on the size of the LPA to be described.)
Initialized by:	IEAVNP04

Displacement	Bytes and Bit Pattern	Field Name	Description
0(0)	8	NIPQSR3N	"PAG3" EBCDIC record identifier
8(8)	1	NIPQSR3D	EBCDIC last record identifier:
	0		Not last record
	X		Last NIPQSR3 record
9(9)	3	NIPQSR3P	TTR of next NIPQSR3 record (if any)
12(C)	Variable	NIPQSR3M	Slot/Group map of the sequential LPA address space; each entry is three bytes long; the first entry is for the lowest address LPA page

SLOT QUEUE

Displacement	Bytes and Bit Pattern	Field Name	Description
0(0)	1	SQCHPGNO	Number of channel programs for this queue
1(1)	3	SQSEQSQA	Address of the slot queue for the next sequential slot on device
4(4)	1	SQSECNO	For a slot sector device, sector number for this slot
5(5)	3	SQSECSQA	Address of the slot queue for the next secondary slot on device
8(8)	1		
	xxxx	SQHHDEL	HH Delta
	 xxxx	SQRECNO	Record number within a track for this slot
9(9)	3	SQ1CHPGA	Address of first channel program on slot queue
12(C)	1	SQINDX	Index used to calculate the address of this slot queue
13(D)	3	SQLCHPGA	Address of last channel program on slot queue

NIP SYSTEM PARAMETER ENTRY (SPE)

Pointed to by:	NVTSPE field in NVT or NIPSPEQ
Size:	8 bytes
Initialized by:	Operator reply processing routines in IEAVNP01 and IEAVNP03

Displacement	Bytes and Bit Pattern	Field Name	Description
0(0)	4	NIPSPEQ	Address of the next NIPSPE on the queue (0 in last SPE)
4(4)	4	NIPSPEA	Address of the operator's reply (the reply prefix, REPLY 00 or R 00, is not included in the reply text; end of reply test is indicated by a quotation mark character)

This section contains a listing of the messages and wait state codes, a register usage table, and a module/control block cross-reference table.

Wait State Code

Code	Issued by	Description
X'01'	IEAIPL00	I/O not operational.
X'02'	IEAIPL00	I/O could not be started.
X'03'	IEAIPL00	I/O could not be started.
X'04'	IEAIPL00	CSW not stored.
X'05'	IEAIPL00	Unit check.
X'06'	IEAIPL00	Program, channel data, channel control, channel chaining, or interface control check.
X'07'	IEAVNP01	No active master console found.
X'0A'	IEAVNP03	Catalog entry for the SYS1.LINKLIB could not be found or successfully retrieved.
X'0C'	IEAIPL00	IEANUC0X not edited in scatter format.
X'0E'	IEAIPL00	SYS1.NUCLEUS or IEANUC0X not found.
X'17'	IEAIPL00	Unit check.
X'18'	IEAIPL00	Insufficient space to load IEANUC0X.
X'19'	IEAIPL00	Unexpected program check.
X'21'	IEAVNIPM	I/O error to system console.
X'30'	IEAVNIP0	Unexpected ABEND.
X'31'	IEAVNIP0	IPL device has no UCB.
X'32'	IEAVNIP0	IEAVNIPM not found.
	IEAVNIPM	A required NIP module is not found in SYS1.NUCLEUS.
X'33'	IEAVNIP0 IEAVNIPM	I/O error during BIDL.
X'34'	IEAVNIP0	DAT feature required but not included.
X'35'	IEAVNIPM	Time-of-day clock is inoperative.
X'36'	IEAVNIPM	Attempted SQA expansion prior to initialization of paging subsystem.
X'37'	IEAVNP02	DSCB for SYS1.LOGREC, SYS1.SVCLIB, SYS1.PARMLIB, or SYS1. LINKLIB could not be read because the data set does not exist; or an I/O error occurred.
X'38'	IEAVNIP0 IEAVNP03 IEAVNP04 IEAVNPA4	Insufficient real storage for initialization.
X'39'	IEAVNP02	A required DASD volume cannot be successfully mounted.
X'3A'	IEAVNP05	Coldstart process failed: OPEN failed for SYS1.LPALIB, I/O error, LPALIB empty, DASD pages unavailable, or storage unavailable.
X'3F'	IEAVNIPM	System error (indicated by bits 36-48 in wait state PSW).
	IEAVNPA5	Invalid request code specified via POST.
X'40'	IEAVNIPM	Unexpected ABEND.

	Issuing Module, IEAVxxxx										
Message ID	NIPM	NP01	NP02	NP03	NP04	NPA4	NP05	NPA5	NP06	NP07	NIPX
IEA101A		X									
IEA107I				X			X				
IEA108I							X				
IEA109I							X				
IEA116A		X		X							
IEA118I											X
IEA120A			X								
IEA152I										X	
IEA153I										X	
IEA154I	X										
IEA205I					X				X		
IEA206I					X				X		
IEA207I					X						
IEA208I				X					X		
IEA209I					X				X		
IEA210I					X				X		
IEA211I			X								
IEA212A			X								
IEA216I				X		X					
IEA300I	X			X			X				
IEA301I	X			X			X				
IEA302I	X										
IEA303I	X										
IEA304W	X										
IEA305A	X										
IEA306I	X										
IEA310I			X		X						
IEA311I			X								
IEA312I			X								
IEA313I			X								
IEA314I			X								

	Issuing Module, IEAVxxxx										
Message ID	NIPM	NP01	NP02	NP03	NP04	NPA4	NP05	NPA5	NP06	NP07	NIPX
IEA315A			X								
IEA316A			X								
IEA317A			X								
IEA318I			X								
IEA319I			X								
IEA320A				X							
IEA321I				X							
IEA322A				X	X		X		X	X	X
IEA323I				X							
IEA324I				X							
IEA325I				X							
IEA326I				X			X		X		
IEA327I				X							
IEA328I				X							
IEA330A					X					X	X
IEA331A					X						X
IEA332A					X				X	X	
IEA333A					X						
IEA334A					X						
IEA335A					X						
IEA336A					X						
IEA337I					X						
IEA338A					X					X	X
IEA340I					X		X		X		X
IEA342I					X						
IEA343A					X						
IEA344I					X						
IEA345I					X						
IEA350I							X				
IEA351I							X				
IEA352I							X				

Message ID	Issuing Module, IEAVxxxx										
	NIPM	NP01	NP02	NP03	NP04	NPA4	NP05	NPA5	NP06	NP07	NIPX
IEA353I							X				
IEA354I							X				
IEA355A							X		X	X	
IEA356I								X			
IEA357I							X				
IEA360I									X		
IEA361I									X		
IEA362I									X		
IEA363I							X		X	X	
IEA370I										X	
IEA371I										X	
IEA380I											X
IEA381I											X
IEA382I											X
IEA383I											X
IEA384I											X
IEA385I											X
IEA386I											X

REGISTER USAGE TABLE

This table lists the entry points for each module, the name of the routine to which each module normally exits, and the register contents upon entry and exit. Beginning with the exit from IEAVNIP0, the following registers are part of standard linkage conventions and have the same contents upon entry to each module. The contents of these registers are noted in the table only where significant. The contents of other omitted registers are irrelevant.

1 (RPARM)	Address of parameter list
2 (RNVT)	Address of NIP vector table
3 (RCVT)	Address of CVT
14 (REXIT)	Address of next entry point to receive control
15 (RENTRY)	Address of entry point within this module.

Entry	Exit	Registers	Contents
IEAIPL00 (not used)		0-15	Irrelevant
	to X'16C'	1	Address of the IEANUC0x translation table
		3	Address of the nucleus data set/volume data area
		4	Address of the nucleus CSECT size table (SIZTABLE)
		6	Size of real storage
		7	Address of the end of the resident nucleus
		8	Address of the nucleus CSECT address table (ADRTABLE)
		9	Number of nucleus CSECTs (SIZTABLE entries + ADRTABLE entries)
		10	Device address of the IPL volume
IEAVNIP0		3	Address of the IPLDATA parameter area
		6	Real storage size
		7	Address of the end of the resident nucleus
		10	Device address of the IPL volume
	IEAVNIPM	1 (RPARM)	Address of the SYS1.NUCLEUS DCB
		2 (RNVT)	Address of the NVT
		3 (RCVT)	Address of the CVT
		15 (RENTRY)	Address of IEAVNIPM entry
IEAVNIPM		1 (RPARM)	Address of the SYS1.NUCLEUS DCB
	IEAVNIPX	15 (RENTRY)	Address of IEAVNIPX entry
NIPLOAD		1 (RPARM)	Address of the 8-character module name to be loaded from SYS1.NUCLEUS
	Caller via REXIT	1 (RPARM)	Address of entry point of loaded module
NIPSENSE		1 (RPARM)	Address of the IOB for the failing I/O operation
	Caller via REXIT		
NIPSQEND			
	No exit; disabled wait		
NIPSWAIT	No exit; disabled wait		

Entry	Exit	Registers	Contents
NIPSVC (NIPSVCX)		0, 1, 13, 15	SVC parameter registers
		4	Address of TCB
		5	Address of RB
		14	Address of SVC 3 instruction
		15	XCTL entry - address of parameter list
	Type-3, Type-4, or XCTL routine	0, 1, 4, 5, 13, 14, 15	Unchanged
NIPTIME		1 (RPARM)	Request indication: X'00' - TCD clock value X'04' - relative time since IEAVNIPM first entered
	Caller via REXIT	1 (RPARM)	Current value of TOD clock or relative time
NIPUCBFN		1 (RPARM)	EBCDIC unit name or device address
	Caller via REXIT	1 (RPARM)	Address of UCB if found or 0 if not found
NIPWTO		1 (RPARM)	Address of message to be written to console
	Caller via REXIT	1	Unchanged
NIPWTOR		1 (RPARM)	Address of WTOR parameter list
	NIPWTOR2	15 (RENTRY)	Address of NIPWTOR2 entry
	Caller via REXIT	1	Unchanged
NIPWTOR2		1 (RPARM)	Positive - address of NIPWTOR parameter list Negative (two's complement) address of SQA-buffered reply to be released
		14 (REXIT)	Return address of caller of NIPWTOR if wait specified or address of caller of NIPWTOR2
	Caller of NIPWTOR or NIPWTOR2	1 (RPARM)	Address of the operator's reply (irrelevant if called to release SQA reply buffer)
IEAVNP01		Standard	
	IEAVNIPM	Standard	Unchanged
IEAVNP02		Standard	
	IEAVNIPM	Standard	Unchanged
IEAVNPA2		1 (RPARM)	Address of the NIPMOUNT parameter list
	Caller via REXIT	1	Address of the UCB for the mounted volume, or 0 if request canceled
IEAVNPB2		1 (RPARM)	Address of the parameter list of the data set to be opened
	Caller	1	Unchanged if open is successful, or two's complement of address if open failed
IEAVNP03		Standard	
	IEAVNIPM	Standard	Unchanged
IEAVNP04		Standard	
	IEAVNIPM	Standard	Unchanged
IEAVNPA4		1 (RPARM)	Address of parameter list indicating function to be performed
	Caller via REXIT	1	Read requests - address of quickstart record Other successful requests - unchanged Unsuccessful request - two's complement of address

Entry	Exit	Registers	Contents
IEAVNP05		Standard	
	IEAVNIPM	Standard	Unchanged
IEAVNPA5		1(RPARM)	Address of parameter list containing RCVT and RNVT values
	IEAVNP05	Standard	Unchanged
IEAVNP06		Standard	
	IEAVNIPM	Standard	
IEAVNP07		Standard	
	IEAVNIPM	Standard	
IEAVNIPX		Standard	
	IEEVIPL	4	Address of SYS1.PARMLIB UCB

Module/Control Block Cross Refernce

The following table is a module/control block cross-reference table. The letter M means the module modifies the control block (the module may also refer to the control block). The letter R means the module refers to the control block (the module does not modify the control block).

Control Block	NIP0	NIPM	NP01	NP02	NP03	NP04	NPA4	NP05	NPA5	NP06	NP07	NIPX
CVT	M	R	R	R	R	M	R	M	M	M	M	M
DEB	M	M	M	M			M			R		
NVT	M	M	M	M	M	M	M	M	M	M	M	R
PVT	M	M				M	M	M			M	M
SCVT	M	R				R		R		M		R
Master Scheduler TCB	M							M				M
Communications Task TCB										R		M

active page: A page in real storage that can be addressed.

active page queue: A queue of pages in real storage that are currently assigned to tasks. Pages on this queue are eligible for placement on the available page queue.

address translation: The process of changing the address of an item of data or an instruction from its virtual address to its real storage address. See also dynamic address translation.

automatic priority group: In VS2, a group of tasks at a single priority level that are dispatched according to a special algorithm that attempts to provide optimum use of CPU and I/O resources by these tasks. See also dynamic dispatching.

available page frame count: In VS2, a count of page frames that are ready for reassignment.

available page queue: A queue of the pages whose real storage is currently available for allocation to any task. See also active page queue, hold page queue.

basic control (BC) mode: A mode in which the features of a System/360 computing system and additional System/370 features, such as new machine instructions, are operational on a System/370 computing system. See also extended control (EC) mode.

DAT: Dynamic address translation.

demand paging: Transfer of a page from external page storage to real storage at the time it is needed for execution.

device number: In VS2, a part of an external page address that refers to a particular paging device; together with a group number and a slot number, it identifies the location of a page in external page storage.

disabled page fault: A page fault that occurs when I/O and external interruptions are disallowed by the CPU.

dynamic address translation (DAT): (1) The change of a virtual storage address to a real storage address during execution of an instruction. See also address translation. (2) A hardware feature that performs the translation.

dynamic area: The portion of virtual storage that is divided into regions or partitions that are assigned to job steps and system tasks. See also pageable dynamic area, nonpageable dynamic area. Contrast with nondynamic area.

dynamic dispatching: In VS2, a facility that assigns priorities to tasks within an automatic priority group to provide optimum use of CPU and I/O resources.

EC mode: Extended control mode.

enabled page fault: A page fault that occurs when I/O and external interruptions are allowed by the CPU.

extended control (EC) mode: A mode in which all the features of a System/370 computing system, including dynamic address translation, are operational. See also basic control (BC) mode.

external page address: An address that identifies the location of a page in a page data set. In VS2, the address consists of a relative device number, a relative group number, and a relative slot number.

external page storage: The portion of auxiliary storage that is used to contain pages.

external page storage management: A set of routines in the paging supervisor that control external page storage.

external page table (XPT): In VS2, an extension of a page table that identifies the location on external page storage of each page in that page table.

fixed: In OS/VS, not capable of being paged out.

fixed BLDL table: A BLDL table that the user has specified to be fixed in the lower portion of real storage.

fixed link pack area: In VS2, an extension of the link pack area that occupies fixed pages in the lower portion of real storage.

fixed page: A page in real storage that is not to be paged out.

group number: In VS2, a part of an external page address that refers to a slot group; together with a device number and a slot number, it identifies the location of a page in external page storage.

invalid page: A page that cannot be directly addressed by the dynamic address translation feature of the central processing unit.

link pack area (LPA): In VS2, an area of virtual storage containing selected reenterable and serially reusable routines that are loaded at IPL time and can be used concurrently by all tasks in the system.

link pack area directory: In VS2, a directory that contains an entry for each entry point in link pack area modules.

link pack area library: In VS2, a partitioned data set that contains the modules specified to be in the link pack area.

link pack area queue: In VS2, a queue that contains a contents directory entry for each link pack area module currently in use, for each module in the link pack update area, and for each module in the fixed link pack area.

link pack update area: In VS2, an area in virtual storage containing modules that are additions to or replacements for link pack area modules for the current IPL.

local system queue area (LSQA): In VS2, one or more segments associated with each virtual storage region that contains job-related system control blocks.

LPA: Link pack area.

LSQA: Local system queue area.

memory: See real storage, virtual storage.

nondynamic area: The area of virtual storage occupied by the resident portion of the control program (the nucleus and the link pack area). Contrast with dynamic area.

nonpageable dynamic area: An area of virtual storage whose virtual addresses are identical to real addresses; it is used for programs or parts of programs that are not to be paged during execution. Synonymous with V=R dynamic area.

nonpageable region: In VS2, a subdivision of the nonpageable dynamic area that is allocated to a job step or system task that is not to be paged during execution. In a nonpageable region, each virtual address is identical to its real address. Synonymous with V=R region.

OS/VS: A compatible extension of the System/360 Operating System that supports relocation hardware and the extended control facilities of System/370.

page: (1) A fixed-length block of instructions, data, or both, that can be transferred between real storage and external page storage. (2) To transfer instructions, data, or both between real storage and external page storage.

pageable dynamic area: An area of virtual storage whose addresses are not identical to real addresses; it is used for programs that can be paged during execution. Synonymous with V=V dynamic area.

pageable region: In VS2, a subdivision of the pageable dynamic area that is allocated to a job step or system task that can be paged during execution. Synonymous with V=V region.

page control block (PCB): A control block that indicates the status of a paging request.

page data set: A data set in external page storage, in which pages are stored.

page fault: A program interruption that occurs when a page that is marked "not in real storage" is referred to by an active page. Synonymous with page translation exception.

page fixing: Marking a page as nonpageable so that it remains in real storage.

page frame: A block of real storage that can contain a page. Synonymous with frame. See also storage block.

page frame table: In VS2, a table that contains an entry for each frame. Each frame table entry describes how the frame is being used.

page migration: In VS2, the transfer of pages from a primary paging device to a secondary paging device to make more space available on the primary paging device.

page-in: The process of transferring a page from external page storage to real storage.

page number: The part of a virtual storage address needed to refer to a page. See also frame number.

page-out: The process of transferring a page from real storage to external page storage.

page reclamation: The process of making addressable the contents of a page in real storage that has been marked invalid. Page reclamation can occur after a page fault or after a request to fix or load a page.

page table (PGT): A table that indicates whether a page is in real storage and correlates virtual addresses with real storage addresses.

page translation exception: A program interruption that occurs when a virtual address cannot be translated by the hardware because the invalid bit in the page table entry for that address is set. Synonymous with page fault, program interruption code 17.

page wait: A condition in which the active request block for a task is placed in a wait state while a requested page is located in real storage or is brought into real storage.

paging: The process of transferring pages between real storage and external page storage.

paging device: A direct access storage device on which pages (and possibly other data) are stored.

paging supervisor: A part of the supervisor that allocates and releases real storage space (page frames) for pages, and initiates page-in and page-out operations.

PFT: Page frame table.

PGT: Page table.

primary paging device: In VS2, an auxiliary storage device that is used in preference to secondary paging devices for paging operations. Portions of a primary paging device can be used for purposes other than paging operations.

quick cell: In VS2, a reserved space in the system queue area or a local system queue area that can be used to reduce the time required to allocate space for a control block.

quickstart pages: Pages in a single page data set that contain the link pack area modules and link pack area directory that are to be made available to the system during a quickstart.

quickstart records: One or more records, located at the beginning of a formatted page data set, that contain information that makes it possible to use the data set over successive IPLs. The quickstart records identify all usable pages in the data set and, for a page data set containing quickstart pages, identify the location of those pages in the data set.

real address: The address of a location in real storage.

real storage: The storage of a System/370 computing system from which the central processing unit can directly obtain instructions and data, and to which it can directly return results.

reference bit: A bit associated with a page in real storage; the reference bit is turned "ON" by hardware whenever the associated page in real storage is referred to (read or stored into). In VS2, there is a reference bit in each of two storage keys associated with each page frame.

relocate hardware: See dynamic address translation.

secondary paging device: In VS2, an auxiliary storage device that is not used for paging operations until the available space on primary paging devices falls below a specified minimum. Portions of a secondary paging device can be used for purposes other than paging operations.

segment: A continuous 64K area of virtual storage, which is allocated to a job or system task.

segment table (SGT): A table used in dynamic address translation to control user access to virtual storage segments. Each entry indicates the length, location, and availability of a corresponding page table.

segment table entry (STE): An entry in the segment table that indicates the length, location, and availability of a corresponding page table.

slot: In VS2, a continuous area on a paging device in which a page can be stored.

slot group: In VS2, a set of slots on one or more tracks within a cylinder on a paging device.

slot number: In VS2, a part of an external page address that refers to a slot; together with a device number and a group number, it identifies the location of a page in external page storage.

SQA: System queue area.

swapping: In VS2 with TSO, a paging technique that writes the active pages of a job to external page storage and reads pages of another job from external page storage into real storage.

system queue area (SQA): An area of virtual storage reserved for system-related control blocks.

thrashing: A condition in which the system can do little useful work because of excessive paging.

virtual address: An address that refers to virtual storage and must, therefore, be translated into a real storage address when it is used.

virtual equals real (V=R) storage: An area of virtual storage that has the same range of addresses as real storage and is used for a program or part of a program that cannot be paged during execution.

V=R dynamic area: Same as nonpageable dynamic area.

V=R partition: Same as nonpageable partition.

V=R region: Same as nonpageable region.

virtual storage: Addressable space that appears to the user as real storage, from which instructions and data are mapped into real storage locations. The size of virtual storage is limited by the addressing scheme of the computing system and by the amount of auxiliary storage available, rather than by the actual number of real storage locations.

virtual storage region: In VS2, a subdivision of the dynamic area that is allocated (in segment-size blocks) to a job step or a system task.

V=V dynamic area: Same as pageable dynamic area.

V=V region: Same as pageable region.

XPT: External page table.

XPTE: External page table entry.

INDEX

SY27-7243-0

OS/VS IPL/NIP Printed in U.S.A. SY27-7243-1

IBM

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
(U.S.A. only)

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
(International)