File No. S360-36
GY28-6661-5

Program Logic

OS Release 21

IBM System/360 Operating System Initial Program Loader and Nucleus Initialization Program

Program Number 360S-CI-535

This publication presents the internal logic of the IBM System/360 Operating System Initial Program Loader and Nucleus Initialization Program. The operation of the Nucleus Initialization Program in each of the control program environments (MFT, MVT) is described in the section dealing with the Nucleus Initialization Program. Special initialization procedures for MVT with Model 65 multiprocessing systems are included in Appendix A. The tables, work areas, and control blocks are illustrated in the publication, as well as flowcharts illustrating the logic flow of the Initial Program Loader and the Nucleus Initialization Program.

This program logic manual is directed to personnel responsible for program maintenance. It can be used to locate specific areas of the program, and it enables the reader to relate these areas to the corresponding program listings. Program logic information is not necessary for program operation and use.

Sixth Edition (March, 1972)

This is a major revision of, and makes obsolete, GY28-6661-4 and its associated TNL GN27-1353. This edition applies to Release 21 of the System/360 Operating System and to all subsequent releases until otherwise indicated in new editions or Technical Newsletters. A change is indicated by a vertical line to the left of the change. Changes are continually being made to the specifications herein; before using this publication in connection with the operation of IBM systems, consult the latest System/360 SRL Newsletter, GN20-0360, for the editions that are applicable and current.

This publication was prepared for production using an IBM computer to update the text and to control the page and line format. Page impressions for photo-offset printing were obtained from an IBM 1403 Printer using a special print train.

Copies of this and other IBM publications can be obtained through IBM branch offices.

A form for reader's comments appears at the back of this publication. Address any additional comments concerning the contents of this publication to IBM Corporation, Programming Publications, Department 636 Neighborhood Road, Kingston, New York 12401

This publication describes the internal logic and theory of operation of the Initial Program Loader, which brings in the programs which become the control program nucleus, and the Nucleus Initialization Program, which prepares those programs and the balance of main storage for operation of the control program.

The publication consists of eight major sections. The Introduction presents an overview of the purpose and functions of the programs and introduces the major topics presented in the manual. The section, Initial Program Loader presents a detailed discussion of that program's operation, including the use of tables, work areas, and control blocks. The address resolution procedure is described in detail as an adjunct to the listings of the program. The Nucleus Initialization Program section discusses the operation of that program and emphasizes the differences that exist for the different configurations of the control program. The Routine Lists section presents a summary of each of the major IPL and NIP routines. The section Tables and Work Areas contains illustrations of the tables, work areas, and control blocks used by the programs or initialized by the programs. The Flowcharts section contains charts that diagram the logic flow of the Initial Program Loader and the Nucleus Initialization Program. Appendix A: Initialization for MVT with Model 65 Multiprocessing describes the special initialization procedures for the Model 65 Multiprocessing System. Appendix B: NIP Interface Routine describes the special processing necessary to maintain correct base addresses when NIP uses subroutines.

Throughout this publication, references to control program configurations are simplified by the use of abbreviations. When used in the publication, MFT refers to systems capable of multiprogramming with a fixed number of tasks, and MVT refers to systems capable of multiprogramming with a variable number of tasks. MVT with Model 65 multiprocessing is not abbreviated in this publication. An appendix is devoted to a detailed discussion of MVT with Model 65 multiprocessing.

The reader of this publication should have a knowledge of the assembler language for System/360 and should be familiar with the following publications:

IBM System/360: Principles of Operation, GA22-6821

IBM System/360 Operating System:
- Introduction, GC28-6534
- Assembler Language, GC28-6524
- MFT Guide, GC27-6939
- MVT Guide, GC28-6720

In addition, information contained in other publications may prove helpful to understanding these programs. These publications are:

IBM System/360 Operating System:
- Data Management Macro Instructions, GC26-3794
- Data Management Services Guide, GC26-3746
- Messages and Codes, GC28-6631
- Operator's Reference, GC28-6691
- Programmer's Guide to Debugging, GC28-6670
- Service Aids, GC28-6719
- Storage Estimates, GC28-6551
- System Control Blocks, GC28-6628
- System Generation, GC28-6554
- MFT Supervisor PLM, GY27-7236
- MVT Supervisor PLM, GY28-6659

SUMMARY OF AMENDMENTS

Release 21

- Status Display Support
- Initialization for the Generalized Trace Facility
- Relocation of the IPL program
- Channel Check and Machine Check Handler support for the System/370 Model 135
- Operator Procedures for loading an alternative nucleus or limiting apparent storage size for System/370 machines

Release 20.1

- Time Sharing Option
- Channel Check and Machine Check Handler support for the System/370 Model 145

Release 20

- Extended Precision Floating Point Divide
- Initialization procedures for the Block Multiplexor channel
- Locating, formatting, and initializing the SYS1.DUMP data set

For the IBM System/360 Operating System to function in a computing system, the programs and their associated control blocks and work areas must be loaded into main storage and prepared for operation.

Loading the control program modules is the function of the Initial Program Loader, referred to as the IPL program.

After the IPL program completes the loading function, control is passed to the Nucleus Initialization Program, referred to as NIP, which performs the functions necessary to initiate operation of the control program. NIP also loads and initializes optional routines selected by the user.

This publication is a guide to the program listings of IPL and NIP. Where possible, the organization of the manual follows the logical flow of the programs discussed and the topics appear in the order of their appearance in the program listings. Unless otherwise indicated, IPL/NIP processing is common to both MFT and MVT. Only that processing that is unique to one or the other, or processing that is optional is noted as such. This section provides an overview and general discussion of the other sections in the manual. Detailed information on each routine can be found in the appropriate section.

THE INITIAL PROGRAM LOADER

The Initial Program Loader (IPL) clears main storage, determines main storage size, relocates its own instructions, reads in the selected control program nucleus and resolves address constants. When IPL has completed its operations, control is transferred to the Nucleus Initialization Program. IPL functions in the same manner for both control program configurations; the control program nucleus which IPL loads governs the option of the operating system.

The section "The Initial Program Loader" describes the IPL functions in detail and is divided into the following topics:

- Loading the IPL Program
- Determining the Nucleus
- Clearing Storage and Determining its Size
- Finding the Selected Nucleus
- Assigning Nucleus Control Section Addresses
- IPL Program Relocation
- Loading the Nucleus Control Section
- Replacing Nucleus Address Constants
- Giving Control to the Nucleus Initialization Program

THE NUCLEUS INITIALIZATION PROGRAM

The Nucleus Initialization Program (NIP) prepares the control program for operation by defining main storage areas and initializing certain tables, work areas, and control blocks. These NIP functions vary according to the control program option (MFT or MVT), so that NIP code for an MVT system will contain routines that do not appear in NIP code for an MFT system. These differences result from selective processing of the NIP macro instruction during system generation. According to the system being generated, program switches are set which cause sections of coding to be included in, or deleted from, the NIP load module produced by system generation. NIP also loads and initializes optional routines selected by operator command at system initialization.

The NIP section of this publication is organized in the order of appearance of routines in the NIP macro. Each routine is identified with the option or options of the control program in which it is included; for example, the NIP Relocation routine appears only in NIP for an MVT system.

LISTS OF ROUTINES

This section lists the IPL and NIP routines by their entry point names and gives a brief synopsis of the routine function, including the tables, work areas, and control blocks used by the routine. In cases where a routine is embedded within another routine and does not have a specific entry point name, the nearest entry point is given in parentheses.

TABLES AND WORK AREAS

This section presents illustrations of the tables, work areas, control blocks, and record formats referred to in this publication. Many of the control blocks used by IPL and NIP are standard system control blocks and can be found in other publications. For the convenience of the reader, however, they are included in this section.

FLOWCHARTS

This section contains the logic flowcharts for IPL and NIP. The charts are arranged to follow the same order as the presentation of the topic or function the chart represents. Charts for MVT with Model 65 multiprocessing are included.

APPENDIX A: INITIALIZATION FOR MVT WITH MODEL 65 MULTIPROCESSING

In addition to initialization for MVT, initialization for MVT with Model 65 multiprocessing requires special procedures. These procedures are performed by a special module (IEAMP650) and are described in Appendix A. The reader should understand the Nucleus Initialization Program for MVT before using the appendix.

APPENDIX B: NIP INTERFACE ROUTINE

NIP uses subroutines to perform repetitive functions during initialization processing. Because only one base register is available for executable code, NIP uses interface routines assembled in the constants area to maintain correct addressability in branching to these subroutines.

The Initial Program Loader (IPL) is a program which initializes main storage and loads the control program nucleus. The IPL program functions in exactly the same way regardless of the control program option to be loaded. As a result, IPL may perform some functions which are not necessary for a particular system initialization; the nucleus initialization program will not require some of the information. However, IPL can initialize the same generated system on varying machine configurations and use its work areas and tables to communicate with the nucleus initialization program. The nucleus initialization program is able to determine any restrictions, such as machine size, by using these tables and work areas.

The IPL records, located on the system residence volume, consist of three records. The first record is read into main storage by a hardware feature and causes the second record to be read into main storage above the area to be occupied by IPL. A transfer in channel (TIC) command by the first record causes the execution of the second, or "bootstrap", record. The bootstrap record causes the third record, the IPL program text, to be read into main storage, overlaying the first record. The IPL program:

- Determines the nucleus to be used (either the standard nucleus or an alternative user-selected nucleus can be used).
- Clears main storage and determines its size.
- Finds the selected nucleus on secondary storage.
- Assigns main storage addresses to the nucleus control sections.
- Relocates the unexecuted portion of its own instructions and work areas to prevent being overlaid when the nucleus control sections are read into main storage.

When these preparations are complete, the IPL program:

- Reads the nucleus initialization program into a predetermined area of main storage.
- Reads the nucleus control sections into main storage.
- Establishes addressability among the control sections by resolving address constants.

In performing these operations, IPL uses its own input/output routine (IEASTRIO), and interprets linkage editor output in order to resolve addresses.

IPL performs these operations for all of the control program options; the differences in control program initialization are handled by the nucleus initialization program (NIP).

When IPL successfully completes all processing, it transfers control to NIP.

If an error occurs during IPL, the system is placed in a wait state and an error code is stored in the program status word. These wait state codes are explained in Figure 1. Further discussion and user action required can be found in Messages and Codes.

LOADING THE IPL PROGRAM

To prepare for initial program loading, the operator mounts the system residence (SYSRES) volume on a direct access device and sets the load unit address switches to the unit address of that device. Initial program loading is then initiated by pressing the LOAD key.

Pressing the LOAD key causes a system reset, turns on the LOAD light, turns off the MANUAL light, and initiates a read operation from the selected input device. When the read operation is completed satisfactorily, the IPL PSW is obtained, the CPU starts operating, and the LOAD light turns off.

When the read operation is initiated, the selected input device starts transferring data. The first 24 bytes are read into storage locations 0-23. Storage protection, program controlled interruptions, and a possible incorrect-length indication are ignored. The doubleword read into location 8 is a channel command word (CCW) which causes the loading of the second IPL record, the "bootstrap" record.

The "bootstrap" record is loaded into storage at an address higher than the size of the IPL program text (the third record) to ensure that the record will not be overlaid by the IPL control section. The transfer in channel command at location 16 (in the first record) specifies the address of the bootstrap record. The bootstrap record is a chain of CCWs that cause the IPL control section to be read into main storage, beginning at location 0.

When the device provides channel end (the last CCW in the chain is executed), the unit address 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 CPU then fetches the doubleword in storage location 0 as a new PSW and proceeds as in a normal operation. The LOAD light then turns off.

Code	Bit Pattern	Meaning
01	0000 0001	I/O not operational
02	0000 0010	CSW stored on error
03	0000 0011	I/O not initiated
04	0000 0100	Error on TIO
05	0000 0101	Unit-check caused by other than TCC, EOT, EOCYL. Four Sense bytes are moved to location 84 for stand-alone dump.
06	0000 0110	Undefined error
07	0000 0111	No-console condition at IPL
17	0001 0111	Unit-check on Sense Command
18	0001 1000	Available storage exceeded for RLD records
19	0001 1001	Unexpected Program check

Figure 1. Wait State Error Codes

Clearing Registers

The first executable instruction in IPL loads zeros into registers 1 through 14 (register 15 is used as a base register). The address of IEAPCRET is then placed in register 10. IEAPCRET is the return point from the first expected program check. (Program check interruptions are handled by IPL. The program new program status word is constructed pointing to IEAINT. IEAINT is a branch on register 10; the contents of register 10 are modified by IPL routines to handle the next expected interruption.)

IPL then determines if an alternative nucleus has been selected or if main storage size has been limited by operator action.

DETERMINING THE NUCLEUS

The user has the option of selecting the nucleus to be loaded. The operator communicates this information to the IPL program by stopping at location 80 (hex), and storing the suffix for the alternative nucleus into location 8. On System/360 CPU models, the stopping is accomplished by setting the ADDRESS COMPARE switches to 80. On System/370 CPU models, the RATE switch is set to INSTRUCTION STEP before the operator presses the LOAD key. (The primary nucleus name is IEANUC01; user alternative nuclei are given unique names by appending 2 through 9 to the base name IEANUC0.) For further information about nucleus generation, see System Generation.

As its first operation after clearing registers, IPL tests location 8. If location 8 is not zero, the character found is appended to the nucleus name and the name thus formed is stored at IEANUCY for later use. If location 8 is zero, the primary nucleus (IEANUC01) is used.

At this time, the operator may also limit apparent storage size, by storing a hexadecimal character into location 9.

CLEARING STORAGE AND DETERMINING ITS SIZE

After the IPL program has determined the nucleus, its next operation is to determine if a limit has been set on main storage size. Location 9 (hex) is tested for zero. If not zero, the maximum storage size is set according to the character which has been loaded into location 9. The valid characters and their meanings are:

- X'C6' indicates maximum storage of 64K.
- X'C7' indicates maximum storage of 128K.
- X'A7' indicates maximum storage of 192K.
- X'C8' indicates maximum storage of 256K.
- X'A8' indicates maximum storage of 384K.
- X'C9' indicates maximum storage of 512K.
- X'D0' indicates maximum storage of 768K.
- X'D1' indicates maximum storage of 1024K.

IPL later clears storage, up to the indicated limit, by moving zeros. Even though more main storage might be available on the machine system, any addresses above the specified limit will be ignored by the control program, and any later attempts to address storage above the limit may result in a protection violation interruption.

Clearing Floating Point Registers

IPL next loads zeros into the floating-point registers, using a series of LDR instructions. If the system is not equipped with floating-point registers, a program check interruption occurs. The program new PSW (location 60 hex) points to IEAINT which is a branch instruction (BR 10). Since register 10 was previously loaded with the address of IEAPCRET, control is returned to that point. IEAPCRET is the instruction following the series of LDR instructions. Register 10 is then loaded with the address of IEAROUND, the routine which is to gain control after the next expected program interruption.

INDICATING SIZE OF MAIN STORAGE

IPL clears main storage using an MVC instruction in a loop, moving 256 zeros at a time. This continues until an addressing interruption occurs. The routine that clears storage is identified on the program listings by the name IEAZRLP3. The expected interruption is handled (via program new PSW and IEAINT) by IEAROUND. IEAROUND rounds main storage size (in register 9) to a doubleword boundary and stores this rounded value at IEAMAXC. The routine IEAKYLP then sets the storage key for each 2K block of storage to the supervisor key of 0.

FINDING THE SELECTED NUCLEUS

The IPL nucleus location routine (IEAPCKEY) searches for the chosen nucleus name on the primary system residence volume and determines the location of the nucleus data set. To locate the correct nucleus, the routine:

1. Passes the address of the system residence device, stored at location 2, to the I/O subroutine (IEASTRIO). (This device contains the system residence volume.)
2. Reads the label of the system residence volume to find the address of the VTOC, which contains the data set control block for the nucleus data set.
3. Reads the data set control block for the VTOC to determine the number of tracks per logical cylinder of the system residence device. (IPL obtains this value, since different types of direct access devices may be used.)
4. Reads the SYS1.NUCLEUS DSCB to determine the location of the partitioned data set directory.
5. Determines the location of the first scatter/translation record for the nucleus data set member from the partitioned data set directory record containing the nucleus name.
6. Reads the scatter/translation record into main storage above the text of the IPL program.

ASSIGNING NUCLEUS CONTROL SECTION ADDRESSES

For each nucleus control section, the IPL program assigns an address in main storage and calculates a relocation factor. The relocation factors are used to convert address constants to the actual main storage addresses.

Since IPL obtains the nucleus structure from the scatter/translation record, IPL can only calculate control section addresses after this record has been read into main storage. (The composition of the nucleus cannot be known in advance because several options that affect nucleus size are available at system generation.) More information about system generation is available in System Generation.

The scatter/translation record contains the scatter list and the translation table, which are provided by linkage editor to aid in the assignment of CSECT addresses. The scatter list contains suggested relative load addresses for the CSECTs; the translation table gives indexes for the displacement of each control section's scatter list entry from the start of the list.

Note: In assigning and calculating addresses for the nucleus control sections, IPL expects that the first two control sections in the nucleus member data set are the nucleus initialization program control section and the I/O interruption handler control section. The first CSECT (NIP) is loaded adjacent to the relocated IPL text in high-address storage, the second CSECT is loaded at absolute location 0, as the I/O interruption handler CSECT defines the permanent storage area for the control program. If these two CSECTS are not in this order, initialization and subsequent operation is unpredictable.

CALCULATING THE ADDRESSES

Using the relative origins obtained from the scatter list, IPL determines the main storage area required by each control section. IPL constructs a table of these sizes and uses it to assign CSECT loading addresses, which it places in an address table.

The IPL-constructed tables are of the same structure as the scatter table, and the entries associated with a given CSECT have the same relative position in each IPL table as in the scatter table.

Building the Size Table (SIZTABLE)

Using the scatter table information to calculate the size of every control section but the last, IPL:

1. Subtracts the control section origin from the next higher control section origin. This gives the area size required for the former CSECT.

2. Stores the difference in the size table in a position corresponding to that of the control section entry in the scatter table.

To calculate the size of the last CSECT, IPL subtracts that CSECT's relative origin from the size of the entire nucleus, which was obtained from the partitioned data set directory record.

Building the Address Table (ADRTABLE)

IPL makes address assignments first for the NIP and I/O interruption handler control sections, which have the translation table entries immediately following the initial dummy entry.

Since NIP remains in storage only temporarily, IPL loads it adjacent to the portion of IPL that has been relocated (which also is in storage temporarily). This address is obtained by subtracting the size of the area needed by the NIP control section from the relocation address of IPL. The difference is stored in NIP's entry in the address table.

IPL then assigns the I/O interruption handler an address of 0 since it contains the pre-assembled Program Status Words (PSWs), which are not relocatable.

Since other nucleus control sections may be placed in main storage in random order, their addresses in storage have the same numerical relation as their entries in the translation table. (The translation table is an alternative suggestion of loading order from linkage editor.) IPL calculates these addresses by:

1. Adding to the preceding entry in the address table the associated entry in the size table. For example, the second address placed in the table will be equal to the size of the I/O interruption handler, which has an address of zero.

2. Stores the sum in the address table in a location corresponding to that control section's entries in the scatter and size tables.

CALCULATING THE RELOCATION FACTORS

The IPL program uses the addresses it has assigned to the control sections to calculate the corresponding relocation factors. For each control section, IPL subtracts the suggested relative origin found in the scatter table from the assigned main storage address. The difference, which may be positive or negative, is stored in the relocation factor table (RLFTABLE). For example, consider the relocation factor for the I/O interruption handler CSECT. The assigned address is location 0, but the relative origin may be 2000. The relocation factor is then -2000.

The CSECT entries in the RLFTABLE are in the same order as in the scatter, size, and address tables. Therefore, RLFTABLE entries also are accessible through the translation table.

IPL PROGRAM RELOCATION

IPL, which was loaded beginning at location 0, must load the nucleus text into the area of storage it now occupies. To make room for the nucleus text, IPL relocates its tables and the unexecuted portion of code to upper main storage. The relocation address is determined by the size of storage and the size of the IPL tables and code to be relocated. The size of unexecuted IPL code and IPL tables is subtracted from an address determined by storage size. The result of the subtraction gives the beginning address of the relocated IPL. For storage sizes 512K and greater, IPL size is subtracted from 508K. For storage size 256, IPL size is subtracted from 252K. For other sizes less than 512K IPL size is subtracted from the highest available storage address. The relocated portion of IPL will never occupy storage above 508K. Relocation is accomplished using the IEAADDR routine. After IPL relocates its tables and unexecuted instructions, it moves zeros into the storage it occupied before relocation. Figures 2 and 3 illustrate main storage before and after IPL relocation.

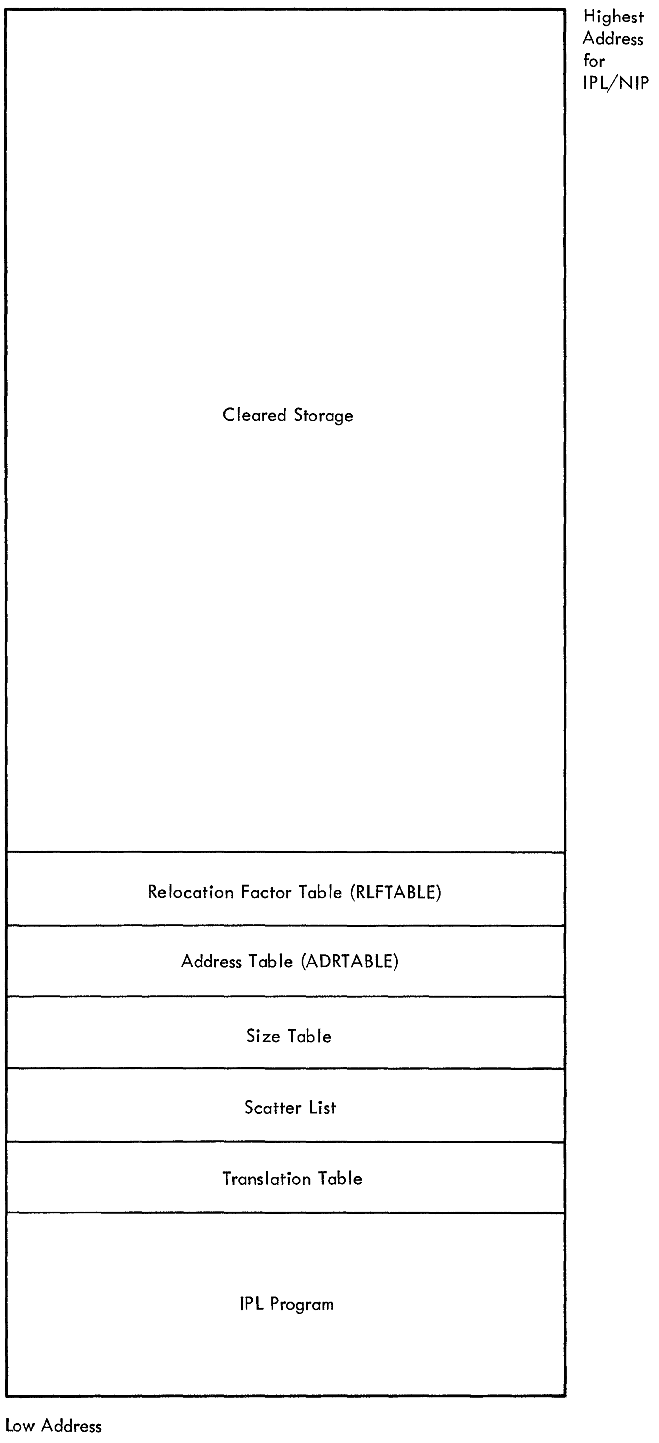


Figure 2. Main Storage Layout Before IPL Relocation

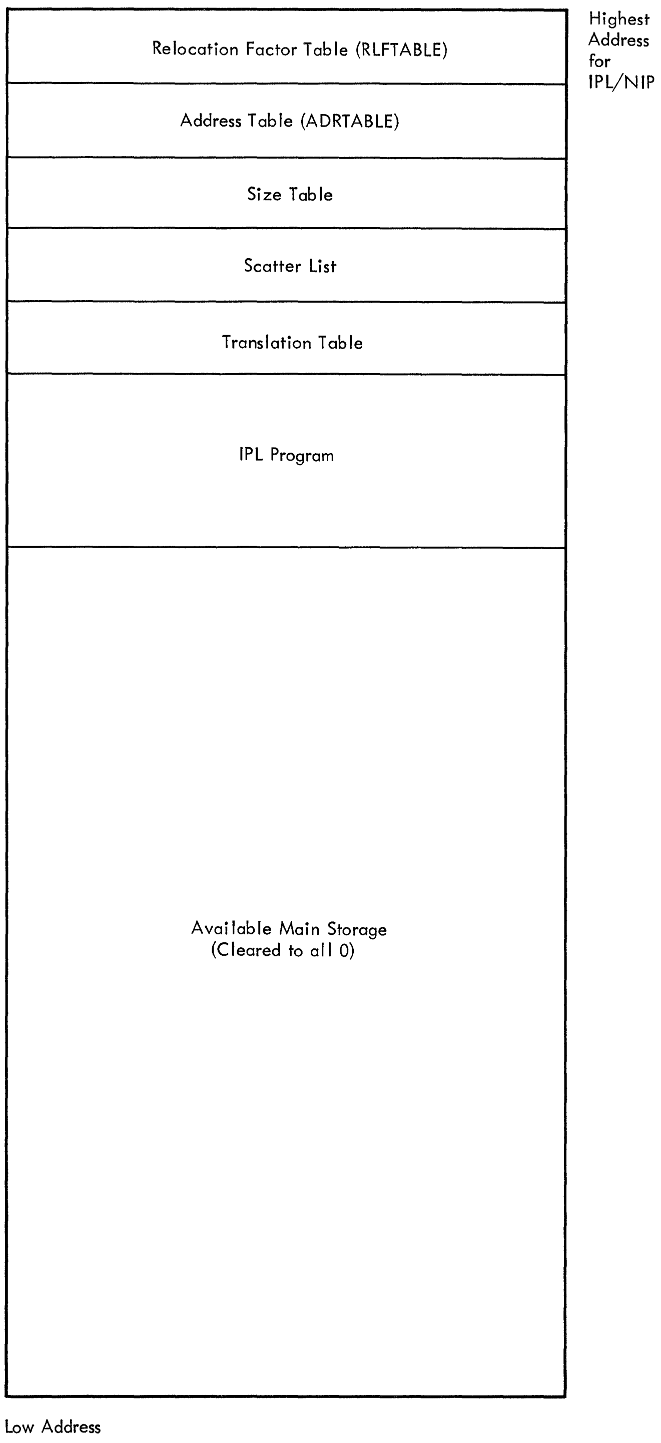

Highest Address for IPL/NIP
Relocation Factor Table (RLFTABLE)
Address Table (ADRTABLE)
Size Table
Scatter List
Translation Table
IPL Program
Available Main Storage (Cleared to all 0)
Low Address

Figure 3. Main Storage After IPL Relocation

LOADING THE NUCLEUS CONTROL SECTIONS

The IPL program reads in nucleus CSECTs in any sequence that it encounters them in the load module. First, the IPL program initializes a 260-byte buffer in main storage. It uses this buffer to read in control, RLD, and control/RLD records. IPL reads the first nucleus control record into the buffer. Then, to load each control section, the IPL program:

1. Determines, from the record in the buffer, the length of the following text record and the external symbol dictionary identification number (ESDID) of the control section containing the text record.
2. Finds the proper translation table entry, so that it can obtain the relocation factor. (The control section identification number is also the displacement of the corresponding entry from the start of the translation table. The IPL program uses this value as an index to the table to find the applicable entry.)
3. Finds the relocation factor for the control section by using the translation table entry (multiplied by four) as a displacement within the relocafactor table. (All text records in the same control section have the same relocation factor.)
4. Modifies a preassembled READ command in the control record by adding the relocation factor to the operand of the command, which has been set by linkage editor to the relative origin of the CSECT in the load module.

   Figure 4 shows the modification of the read commands for the text records of a single CSECT.

	Name	Operation	Operand
Preassembled Read Commands	RDRCD1	RD	100 (256 Bytes)
	RDRCD2	RD	356 (256 Bytes)
	RDRCD3	RD	612 (20 Bytes)
Read Commands With RLF = -20 added to Relative Origin	RDRCD1	RD	80 (256 Bytes)
	RDRCD2	RD	336 (256 Bytes)
	RDRCD3	RD	592 (20 Bytes)

Note: CSECT with assigned storage address of 80 and relative origin of 100 is assumed. This table shows modification of commands to read first three records of CSECT.

Figure 4. Read Command Modifications

5. Passes the preassembled READ command to the I/O subroutine (IEASTRIO), which then reads the text record into the loading address.
6. Reads into the buffer the control, control/RLD, or RLD record following the text record.
7. Moves any relocation information for the control section from the record in the buffer into the relocation dictionary (RLD) area above the high end of the nucleus. Figure 5 illustrates the placement of RLD information. (If the record is an RLD record, which contains only relocation information, IPL reads and transfers RLD information until it encounters a control or control/RLD record on the data set member).
8. Repeats the procedure from the first step until all of the nucleus text has been read into main storage.

Figure 5 shows the arrangement of storage after three control sections are loaded. IPL loads the CSECTs in the order in which it encounters them on disk and not in the order of their main storage addresses.

REPLACING NUCLEUS ADDRESS CONSTANTS

IPL establishes addressability among the control sections by converting nucleus address constants to their main storage equivalents. The IEARELOC routine uses the address table and the RLD information to make the conversions.

From the address table, IPL obtains the location of each control section: then from the RLD information, IPL finds the length and displacement from the control section origin of each address constant in the CSECT.

IPL adds to each of these address constants the relocation factor for the CSECT referred to by the constant. Finally, IPL replaces each address constant with the corresponding sum it has computed, which is the actual main storage address.

Figure 6 shows the layout of storage after all CSECTs have been loaded and address constants have been replaced.

GIVING CONTROL TO THE NUCLEUS INITIALIZATION PROGRAM

When it gives control to NIP, the IPL program loads, into general registers, the size of main storage, the address of the system residence device, the address of the sizetable, the address of the address table, the number of entries per table, and the address of the next doubleword above the end of the nucleus. The contents of general registers at IPL termination is shown in Figure 7. The IPL program then branches to absolute location 16C (hex). This location contains an instruction to load the program status word from location 170 (hex), which contains the starting address of the Nucleus Initialization Program.

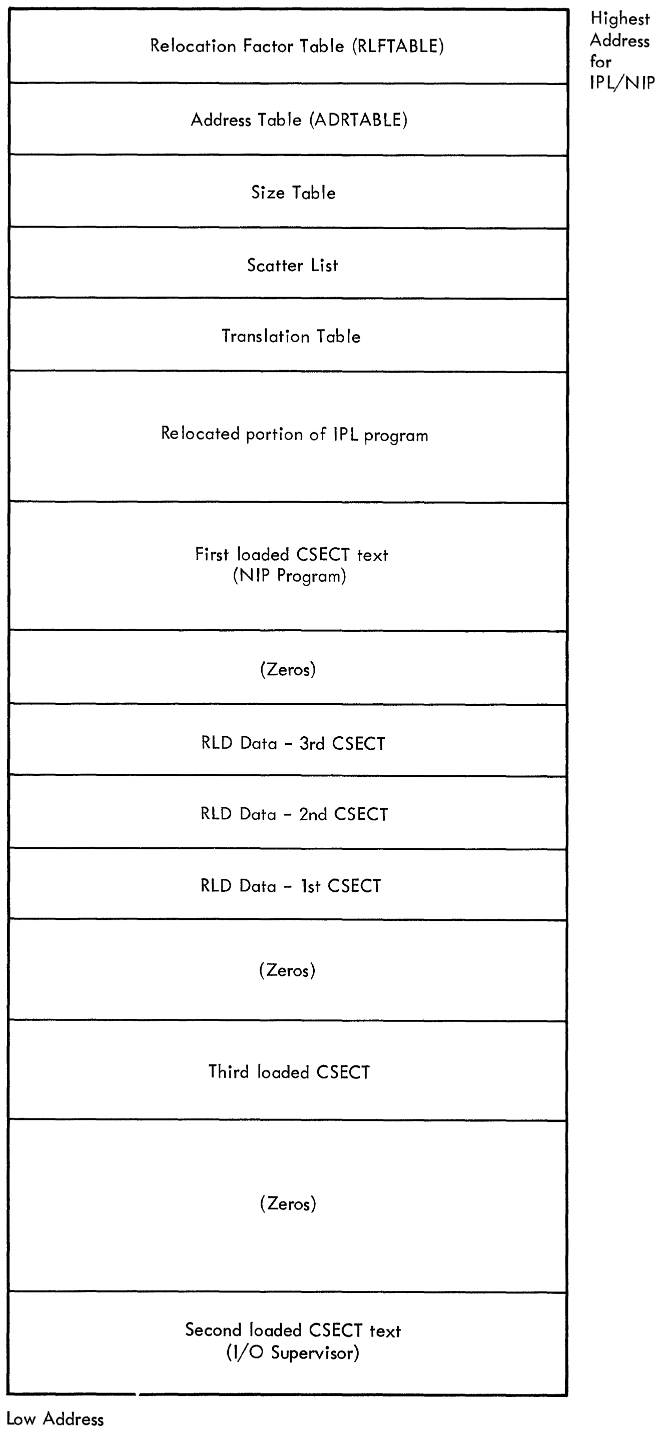


Figure 5. Main Storage After Loading First Three Sections

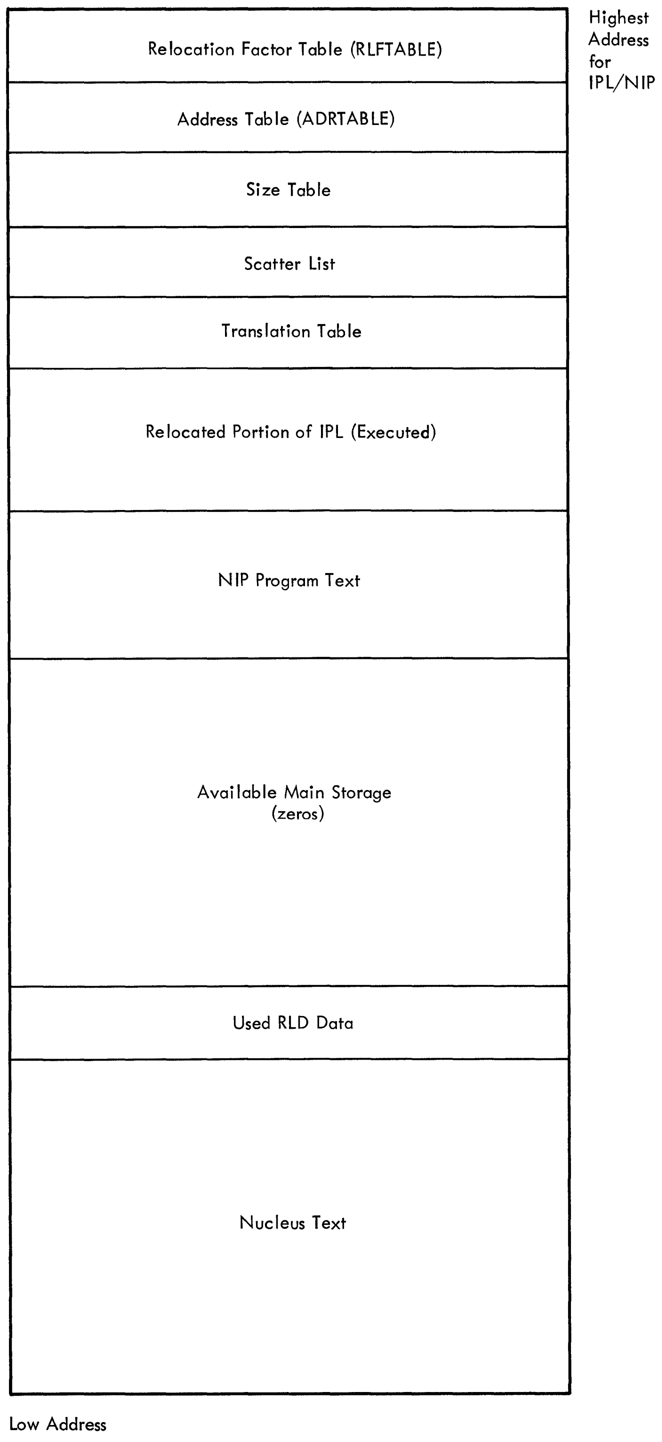

Relocation Factor Table (RLFTABLE)
Highest Address for IPL/NIP
Address Table (ADRTABLE)
Size Table
Scatter List
Translation Table
Relocated Portion of IPL (Executed)
NIP Program Text
Available Main Storage (zeros)
Used RLD Data
Nucleus Text
Low Address

Figure 6. Main Storage at IPL Termination

Reg	Content	Reg	Content
0	Varies	8	Address of ADRTABLE (excluding Dummy Entry)
1	Varies	9	Number of Entries per table (number of CSECTS)
2	Varies	10	SYSRES Device Address
3	Varies	11	Varies
4	Address of SIZTABLE (excluding Dummy Entry)	12	Varies
5	Varies	13	Varies
6	Storage Size	14	Varies
7	Address of End of Nucleus	15	Varies

Figure 7. Register Contents at IPL Termination

The Nucleus Initialization Program (NIP) is assembled at system generation from a system generation (SYSGEN) macro (SGIEA2NP) according to the options selected for the system being generated. Therefore, the order of the routines in NIP is the same for all levels of the control program, but NIP functions vary according to the control program selected -- some functions may be dropped from the assembly entirely, either because they are not supported by a control program level or not selected as a SYSGEN option.

This section describes the organization and functions of NIP in the order of the appearance of routines in the NIP macro instruction. Each function is identified by:

1. The control program level (MFT, MVT) to which it applies.

2. A routine or entry point name to assist the reader in relating the description to the program listing. A list of NIP routines and their entry points is included in the section "Lists of Routines."

If the control program is to be MVT with Model 65 multiprocessing, a special module (IEAMP650) must perform some of the initialization procedure. The additional initialization for MVT with Model 65 multiprocessing is discussed in Appendix A.

NIP creates some control blocks and tables which are used later by control program routines. NIP also uses control blocks and tables which are assembled as part of nucleus control sections. These control blocks and tables are discussed not only from the standpoint of NIP's use, but also the later use by control program routines. The control blocks and tables which are used or initialized by NIP are described in more detail in the section "Tables and Work Areas." Detailed descriptions of major system control blocks can be found in System Control Blocks.

Illustrations of main storage layout during various stages of NIP execution are included in this section. Where necessary, each control program level is illustrated separately.

NUCLEUS TABLE INITIALIZATION

The nucleus initialization program first initializes system tables which are used by NIP and then later used by control program routines. Some of these tables are optional, others vary slightly according to control program option. The initialization, therefore, depends on the control program option and the options selected at system generation.

INITIALIZING THE COMMUNICATIONS VECTOR TABLE POINTER

The first function of NIP, after establishing addressability, is to initialize the pointer to the communications vector table (CVT). The CVT is brought into storage as part of the I/O Supervisor control section, and NIP obtains the CVT address by means of an external reference (V-type address constant). NIP then stores the address of the CVT at location 10 (hex.), so that this address may later be used to place entries in the CVT or to retrieve information from the CVT.

CVT-Related Initialization

NIP stores, in its own constant area, the size of storage and the pointer to the end of the nucleus. IPL passes this information to NIP in registers 6 and 7, respectively. If, in a system with MVT, the rollout option has been selected, the main storage size is also placed in the rollout parameter list (see "Initializing the Rollout Data Set").

The address of the device which was used for IPL loading is saved for later use by NIP to determine the system residence (SYSRES) device.

Register 6, which contains the value found by IPL for main storage size, is decremented by one and the result (highest addressable byte in main storage) stored in the CVT at offset 164. In systems with MFT, this address is also stored in the supervisor validity-check routine which is resident in the nucleus.

The communications vector table contains addresses of control blocks and tables which are used by control program routines. Many of these addresses are resolved during the time that IPL reads in the nucleus, but NIP must adjust some of the addresses because NIP processing changes the locations of certain boundaries and control blocks. A detailed layout and description of the CVT can be found in System Control Blocks.

ESTABLISHING THE TRACE TABLE ADDRESS (OPTIONAL: MVT)

If system generation included the trace table option, NIP retrieves the three pointers (current entry pointer, start-of-table pointer, end-of-table pointer) and rounds them to eight-word boundaries. The address of the list of pointers is at location 84 (dec). NIP only adjusts the boundaries of the trace table; no clearing of trace table storage is performed.

The trace table is primarily a debugging tool; entries are made in this table for all I/O and SVC interruptions. For a detailed description of the trace table and an explanation of its use in debugging, see Programmer's Guide to Debugging.

BUILDING THE DUMMY TCB TABLE (MFT WITH SUBTASKING)

NIP constructs a dummy TCB table at location ATCHLOP2. This table holds a maximum of 15 entries, and is used by NIP until the system queue area initialization is completed. After the system queue area is initialized, the table is rebuilt in its permanent location (see "Building the TCB Address Table (MFT)").

TESTING FOR EXTENDED PRECISION FLOATING POINT SIMULATION

The Nucleus Initialization Program tests to determine if the Extended Precision Floating Point Divide feature is part of the system hardware. If the feature is present, a flag is set in the CVTOPTA field of the Communications Vector Table.

The Program Check New PSW address is replaced with the address of the routine EPFPRET. NIP then issues an extended precision instruction. If the operation is successful, bit 7 of the CVTOPTA field of the CVT is set and processing continues. Unsuccessful execution of the extended precision instruction causes a program check, and control goes to EPFPRET to determine the type of program check by examining the Program Check Old PSW. An operations interruption indicates that there are no floating point registers in the hardware, and the extended precision feature is not included.

The flag at bit 7 of CVTOPTA in the CVT is set for later use by the initial Extended Precision Simulator Routine (IEAEPSIM). A bit setting of 1 is used to indicate the presence of the Extended Precision Floating Point Divide feature; only the divide simulator is needed. A bit setting of 0 indicates that all extended precision instructions are to be simulated.

DETERMINING SIZE OF IBM 2361 CORE STORAGE (OPTIONAL: MFT, MVT)

Main storage may be expanded by including IBM 2361 Core Storage in the system. Main Storage Hierarchy Support for IBM 2361 Models 1 and 2 permits selective access to either processor storage (known as hierarchy 0) or the additional storage added by including 2361 Core Storage (the additional storage is known as hierarchy 1). NIP determines if 2361 Core Storage is in the system by comparing total storage size (determined by IPL) with 1024K. If the size is larger than 1024K, NIP divides the last storage address by 1024K. The remainder in the division (in register 0) is placed in IEAH0H1 as the size of hierarchy 0. If there is no remainder, processor size is assumed to be 1024K.

DETERMINING CONSOLE READINESS (MFT, MVT WITHOUT MCS)

Before NIP can communicate with the operator, it must determine whether the console is ready. To do this, the console initialization routine (IEACONS1) finds the primary console and checks its readiness. If the primary console is not ready, NIP finds the alternative console, checks it for readiness, and establishes it as the primary console. If neither the primary nor the alternative console is ready, NIP issues an LPSW instruction, placing the system in a wait state with an error code of 07 (hexadecimal) in the current Program Status Word (PSW). To recover, the console must be made ready and the IPL procedure repeated.

NIP finds the consoles by comparing names in the UCB table with the names provided for the primary and alternative consoles by the system generation program. Two names are given for each console: an input name and an output name. For a standard, noncomposite console, the operation of the console search routine is simplified since only input names need be compared, as both input and output names refer to the same console. For composite consoles, NIP first finds the input console and repeats its search to find the output console.

DETERMINING THE MASTER CONSOLE IN A SYSTEM WITH MULTIPLE CONSOLE SUPPORT (OPTIONAL: MFT, MVT)

If the system was generated with Multiple Console Support (MCS), NIP obtains the pointer to the master console entry in the Unit Control Module (UCM). The pointer is contained in the MCS Prefix to the UCM. NIP then tests the master console for availability. If the master console is available, the UCB for that unit is flagged to indicate that it is the master console, and the master console flag in the UCMDISP field of the unit's UCM entry is set to 1. A complete description and diagram of the UCM, including the MCS Prefix, can be found in the _MFT Supervisor_ and _MVT Supervisor Program Logic Manuals_.

NIP then determines whether the hard copy log is necessary. The hard copy log is required if the master console is a graphic device. If the master console is a graphic device, or if a second console is found available, the hard copy required flag is set to 1.

If the master console is found to be unavailable, NIP sets the "previously tested" flag (UCMXOR) in the UCM entry and obtains the entry for the alternate console of the master console. If the alternate console is not available, the UCM entry is flagged as tested, and the search for an available console proceeds to the next console on the alternate console chain. This search continues until either an available console is found or the end of the chain is reached.

If no available console is found during the search of the master console UCM entry chain, the console specified in the first UCM entry in the UCM base is tested for availability. Each device not flagged as "previously tested," and represented by a UCM entry is tested until an available console is found. If no console is available, the system is placed in a wait state with an error code of X'07'. When the console search is successfully completed, NIP resets the UCMXOR flag to 0 in all UCM entries that were tested and marked during the search. The address of the master console UCM entry is placed in the UCMMCENT field of the MCS Prefix to the UCM.

INITIALIZING TRANSIENT DISPLAY CONTROL MODULES (MULTIPLE CONSOLE SUPPORT WITH DISPLAY CONSOLES ONLY)

To initialize display (CRT) consoles, NIP first locates the transient display control module (TDCM) in main storage for each transient DCM group (in each transient DCM group, one TDCM is initially resident). NIP then uses a portion of the screen image buffer in the TDCM as a temporary DCM to write messages to the display console screen. This enables NIP to process each console assigned to a transient DCM group without bringing the TDCM for each console into main storage. Console initialization modules later remove the temporary DCM from the screen image buffer in the TDCM.

INITIALIZING READY DIRECT ACCESS UCBS

The unit control block initialization routine (IEUCB0) checks readiness of direct access devices. NIP recognizes direct access UCBs by testing the device type (DEVTYP) indicator field in each UCB. For each of these UCBs, NIP determines device readiness by issuing a TIO instruction. If the primary channel path for the device is not available, NIP attempts to check the device by using an alternate channel path (if any are available). The device is considered ready by NIP if any path (primary or alternate) receives a condition code other than not available and does not have a CSW stored with unit check status. When NIP finds a ready direct access UCB, the volume serial number of the mounted volume is obtained and placed in the UCB. The relative track address (TTR) of the volume table of contents (VTOC) of the mounted volume is also placed in the UCB. This information is later used by the allocation routines of the job scheduler, the access method routines, and I/O supervisor routines.

To avoid processing any UCB more than once, and for its later use, NIP builds a direct access device (DAD) table. This table contains device addresses assigned at system generation to each direct access device UCB. Each table entry consists of four half-word device addresses, the first of which is the primary address; the other three are secondary addresses.

NIP issues a RESERVE/RELEASE command sequence for each device that is designated as a shared direct access device at system generation. This is done to verify that the two channel switch feature is installed (the hardware reserve capability does exist). A direct access device that can be shared is indicated by a 1 in bit 2 of byte 2 in the UCBTYP field of the UCB. The command sequence is issued as many as ten times if the device is busy. Each retry is preceded by a 100-millisecond delay to avoid tying up the control unit. If after ten retries the device is still busy because it is being held reserved by another CPU, a message (IEA120A) is sent to the operator. The operator must decide whether the system should continue, leaving the device offline, or wait for the device to become available.

Note: The UCB for a direct access device contains a revised unit address for the device if the primary channel path for the device was not available and an alternate channel path was available.

INITIALIZING THE SYSTEM RESIDENCE UCB

The NIP routine CHKIPLDV ensures that the system residence (SYSRES) volume is mounted on a logically and physically connected device by verifying that there is a UCB for the device. The SYSRES unit address is passed from IPL to NIP, and NIP compares this address to the unit addresses in the UCBs until an equal compare is found. NIP then sets the status indicators in the UCB to indicate permanent residence and the system residence volume.

If no UCB specifying the device address is found, an operator message is issued indicating that the SYSRES volume must be remounted on a logically connected device, and the system is placed in a wait state with an error code of 3 in the current PSW. The nucleus must then be reloaded from the defined device by the IPL program.

CREATING AND INITIALIZING DATA EXTENT BLOCKS

To provide for a multi-extent SVC library and a multi-extent or multi-volume Linkage library, NIP builds the needed data extent blocks. These blocks are built at the high end of the nucleus and become a part of the nucleus. So that NIP can later adjust the nucleus boundaries, the end-of-nucleus address, which is stored in IEADMY, is also placed in IEANUCND. IEADMY is then incremented to include the area reserved for building the DEBs; IEANUCND, pointing to the actual end of the nucleus, is incremented each time a DEB is built and initialized. The addresses in IEADMY and IEANUCND are always rounded to a doubleword boundary.

Building and Initializing the SVCLIB and LOGREC Data Extent Blocks

The data extent block (DEB) for the SYS1.SVCLIB data set is built by NIP. The DEBs for SYS1.SVCLIB and SYS1.LOGREC are then initialized with information from the DSCB of the data set. (The DEB for SYS1. LOGREC is assembled and loaded as part of the nucleus, it is not built by NIP.) Since the SVC library may have multiple extents, its DEB is built, the DSCB checked for number of extents, and an appendage for each extent (up to 15 additional extents are possible) is added to the DEB. The DEB for SYS1.LOGREC (LOGREC is limited to one extent) is then initialized.

SYS1.SVCLIB and SYS1.LOGREC must reside on the system residence volume. The NIP subroutine IEACOMON obtains the information needed to initialize the DEB by first reading the volume label to find the volume table of contents (VTOC). NIP then:

1. Reads into a buffer the data portion of the data set control block (DSCB) for the data set DEB being initialized

2. Moves the absolute device address of the data set boundaries (start CCHH and end CCHH) from the buffer to the DEB.

3. Determines the number of tracks per cylinder from the device characteristics table and places this in the DEB.

4. Places the address of the UCB for the primary system residence device into each DEB. The address of an I/O appendage is placed in each DEB. (See Figure 8.)

GENERAL SYSTEM INITIALIZATION

NIP performs general system initialization functions. This initialization is primarily a housekeeping function; preparations either affect areas of main storage or obtain information from mounted volumes. The Nucleus Initialization Program:

- Checks and sets the timer (optional)
- Builds a temporary system queue area
- Initializes the SVC table
- Builds and initializes the Linkage Library DEB
- Locates, formats, and initializes the SYS1.DUMP data set
- Establishes communication with the operator (obtional)
- Locates and attempts to obtain a DSCB for the system parameter library (SYS1. PARMLIB) data set
- Establishes the linkage library list (LINKLIST) and builds and initializes the LINKLIB DEB for concatenated LINKLIB data sets
- Initializes the System Environment Recorder program (optional)

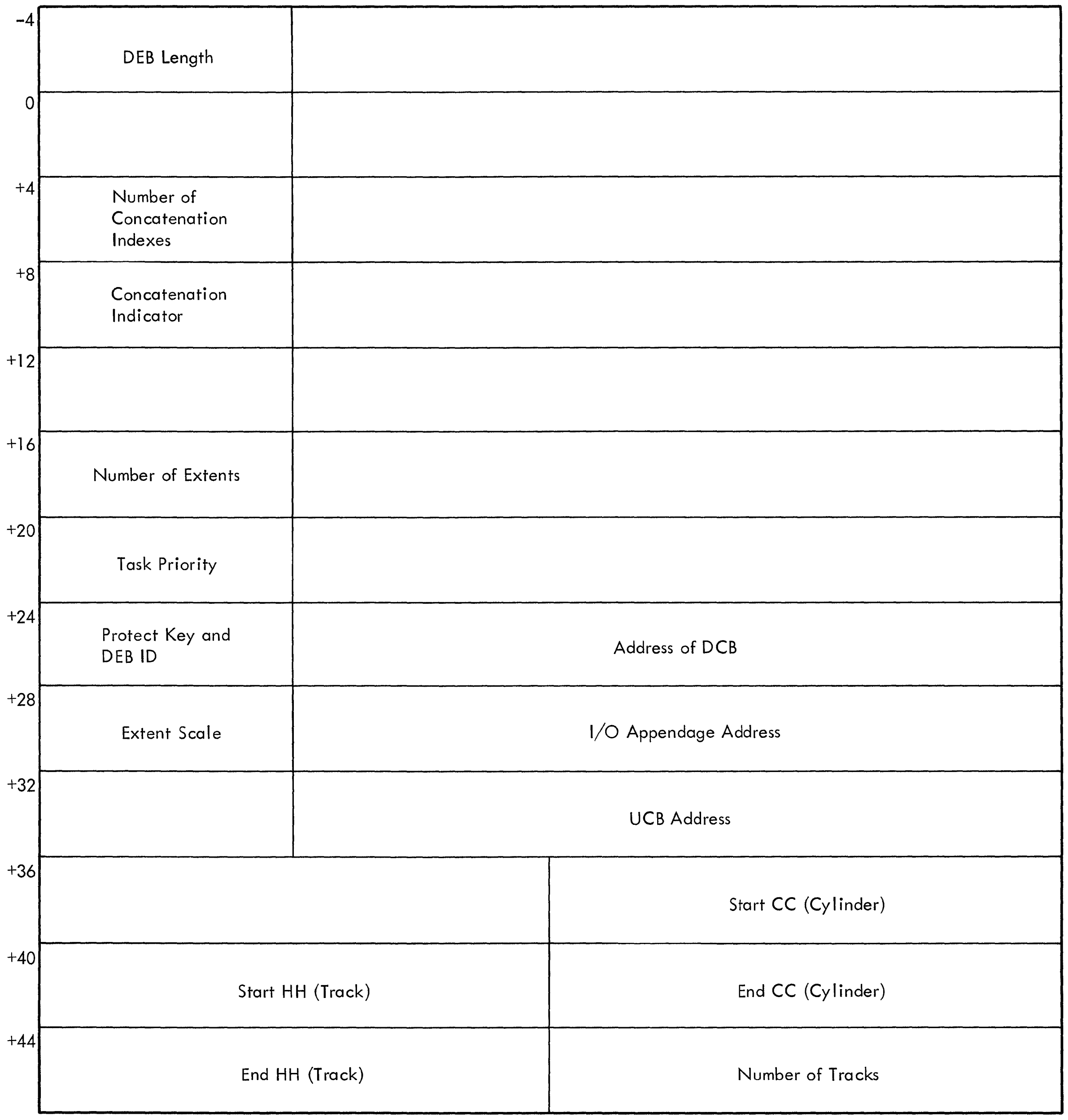


Figure 8. Data Extent Block Initialization

- Allocates, opens, and formats the Rollout data set (optional, MVT only)
- Initializes the SYS1.ASRLIB data set and the nucleus refresh table (NRT) for Machine Check Handler and Channel Check Handler (optional)
- Initializes time slice control elements (two for MVT with Model 65 multiprocessing) (optional)

General system initialization varies with the control program configuration selected. For example, the size, content, and use of the system queue area are different in MFT and MVT; different initialization routines are selected from the NIP macro at system generation for each of these systems. The Rollout option is supported only in systems with MVT; the routines performing this initialization function are included in NIP only if a system with MVT is generated.

CHECKING THE TIMER (OPTIONAL)

If the timer option was selected at system generation, NIP sets the time count at six hours and waits for the timer to decrement the count. The waiting is accomplished by loading a value into register 1 and executing a one-instruction BCT loop using register 1. The value used varies according to CPU Model as shown in Figure 9.

If the timer is inactive, NIP issues the message IEA100I TIMER IS NOT WORKING to notify the operator of the condition.

DEFINING CONTROL PROGRAM AREAS

After checking the timer, NIP begins definition of control program areas. Since NIP uses the control program to complete some of its functions (for example, NIP issues a BLDL macro instruction to initialize the SVC table), NIP defines main storage areas required for control program operation. A system with MFT requires one area, the system queue area; a system with MVT requires two areas, the system queue area and the master scheduler area.

The system queue area (SQA) is constructed in a temporary location to allow NIP to add required control blocks (DEBs) to the nucleus while using system functions requiring the SQA. The temporary location of the system queue area is dependent upon machine model, recovery management options, and control program options. Figure 10 gives the amount of main storage in bytes between the end of the nucleus and the temporary system queue area. The standard size will be increased by the amount indicated for each option included in the system.

System/360 or System/370 CPU Model	BCT Loop Value (Decimal)
System/360 Model 65, 75	21000
System/360 Model 85	56340
System/360 Model 91	111200
System/360 Model 195	94600
System/370 Model 155	25000
System/370 Model 165	50000
Others	17000

Figure 9. Timer Test BCT Loop Values

Machine Model	Options			
	Basic	SER	CCH	MFT with subtasking
195	6K	10K		
91	6K	7K		
others	6K	6K	4K	1K

Figure 10. Nucleus Dummy Buffer Size

The end-of-nucleus address is contained in two words: IEADMY, which contains the highest address of the nucleus loaded plus the dummy area; and IEANUCND, which originally contains the highest address of the nucleus and is updated as control blocks are added to the nucleus.

NIP uses the queue size specification set by system generation to build the system queue area.

In a system with MVT, the master scheduler area is temporarily defined as all of main storage from the end of the system queue area to the NIP control section area.

In a system with MVT, NIP constructs the following queue elements in the system queue area to describe the areas:

1. A dummy partition queue element (PQE) of 8 bytes, which points to the partition queue element (MSPQE1) for the master scheduler region.
2. A partition queue element (MSPQE1) which points to the master scheduler region. (If the Rollout option has been selected, a pointer to this PQE is stored in the Rollout TCB.)
3. A dummy PQE of 8 bytes which points to a partition queue element (HOPQE) for the free area.
4. A partition queue element (HOPQE) which is initialized later.
5. A descriptor queue element (DQE) that defines the size of the system queue area.
6. A free area queue element (FQE) which gives the size of unused system queue area.
7. A free block queue element (FBQE) that defines the size of the master scheduler region. (This control block is not located in the SQA, but rather in the temporary master scheduler region. It is built and chained at this time, however.)

8. A partition queue element (H1PQE) which is initialized to represent all hierarchy 1 storage.

9. A partition queue element (MSPQE2) for the master scheduler region in hierarchy 1. MSPQE2 is initialized only if a secondary link pack area (in hierarchy 1) is loaded.

If a system is generated with Main Storage Hierarchy Support for the IBM 2361 Core Storage, Models 1 and 2, the additional queue elements (items 8-9 in preceding list) are constructed. If the system is generated with a hierarchy structure, and 2361 Core Storage is in the system, a separate PQE (H1PQE) is constructed for 2361 Core Storage (hierarchy 1). H1PQE is temporarily chained to MSPQE1, and functions in place of the MSPQE for hierarchy 1 until the Link Pack Area modules are loaded. A corresponding FBQE is then established for 2361 Core Storage. The first address of 2361 Core Storage is one byte higher than the last address in processor storage (hierarchy 0). Space is reserved for a secondary master scheduler partition queue element (MSPQE2) which is initialized and chained to MSPQE1 only if a secondary link pack area exists in hierarchy 1 (2361 Core Storage) storage. If no link pack area is constructed in hierarchy 1, the space reserved for MSPQE2 is freed, and H1PQE is then the only PQE for hierarchy 1.

NIP also sets aside the area required for the initial supervisor request block (SVRB) at the end of the system queue area and stores the SVRB address in the transient area handler routine. Figure 11 shows the layout of main storage for an MVT system without 2361 Core Storage at this time. Figure 12 shows the layout of storage for an MVT system (excluding MVT with Model 65 multiprocessing) with 2361 Core Storage. If the Rollout option has been selected, the pointer to the high boundary of the system queue area is also stored in the Rollout boundary pointer.

Figure 13 shows the main storage layout for an MFT system at this time.

INITIALIZING THE SVC TABLE (MVT; OPTIONAL: MFT)

In a system with MVT, the routine which defines the system queue area also initializes the SVC table. In systems with MFT, this initialization is optional and can be located in the program listings by the instruction name SVXINIT.

NIP initializes all supervisor table entries for nonresident SVC routines. These type III and type IV routines reside in the SYS1.SVCLIB data set on the system residence volume. The routines may be either IBM- or user-supplied or both. The corresponding SVC table entries are flagged as representing nonresident routines.

When NIP recognizes an entry for a nonresident routine, the SVC number (nnnn) related to the entry is combined with the prefix IGC0 to obtain the routine name (IGC0nnnn). NIP then:

1. Issues a BLDL macro instruction to obtain module information from the SVC library directory.

2. Extracts from the directory record the low-order 18 bits of the TTR (track and record address) and the low-order 11 bits of the length of the first text record. NIP places these values into the SVC table entry.

If the BLDL routine fails to locate an SVC routine name, NIP issues an error message to the operator specifying the name of the routine. In MVT if an I/O error occurs or the BLDL for an SVC fails, NIP inserts the address of the nucleus resident error routine (IGCERROR).

CREATING AND INITIALIZING THE LINKAGE LIBRARY DEB

Creation and initialization of the linkage library DEB is similar to that for the SVC library, but because the data set (SYS1.LINKLIB) may reside on other than the system residence volume, NIP handles its initialization separately. If SYS1.LINKLIB is not on the system residence volume, it may be on another mounted volume or an unmounted volume. SYS1.LINKLIB _must_ be cataloged.

To determine the serial number of the volume containing the linkage library data set, NIP issues a LOCATE macro instruction before attempting DEB initialization. If the information obtained from the catalog data set indicates that the linkage library does reside on the system residence volume, NIP initializes the DEB. If the system catalog has no entry for SYS1.LINKLIB, NIP enters a wait state with an error code 03.

If LINKLIB does not reside on the system residence volume, NIP determines if the volume is mounted by searching the direct access device table (previously constructed by NIP) for the volume serial number. If the volume is mounted, NIP proceeds with initialization.

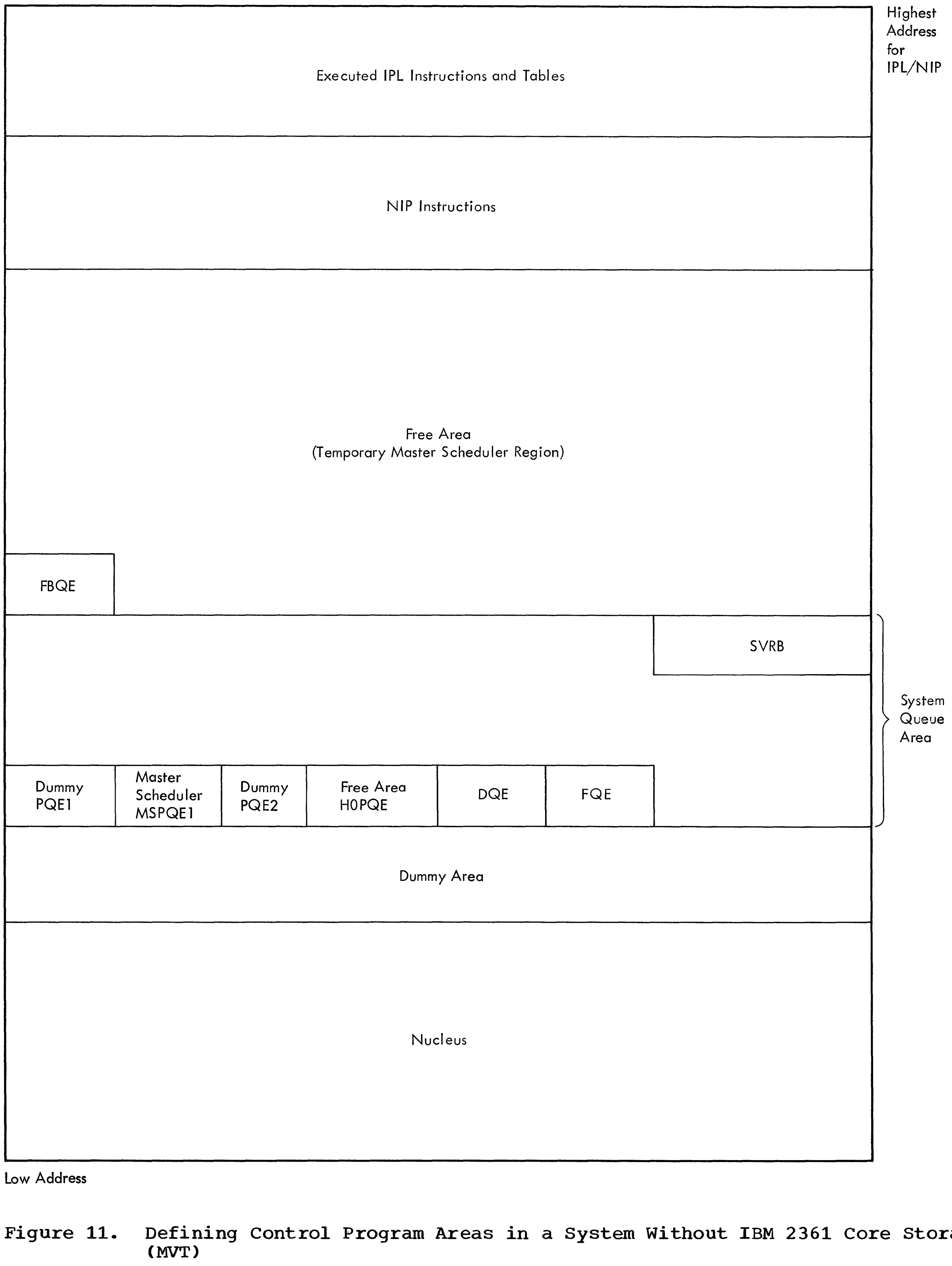


Figure 11. Defining Control Program Areas in a System Without IBM 2361 Core Storage (MVT)

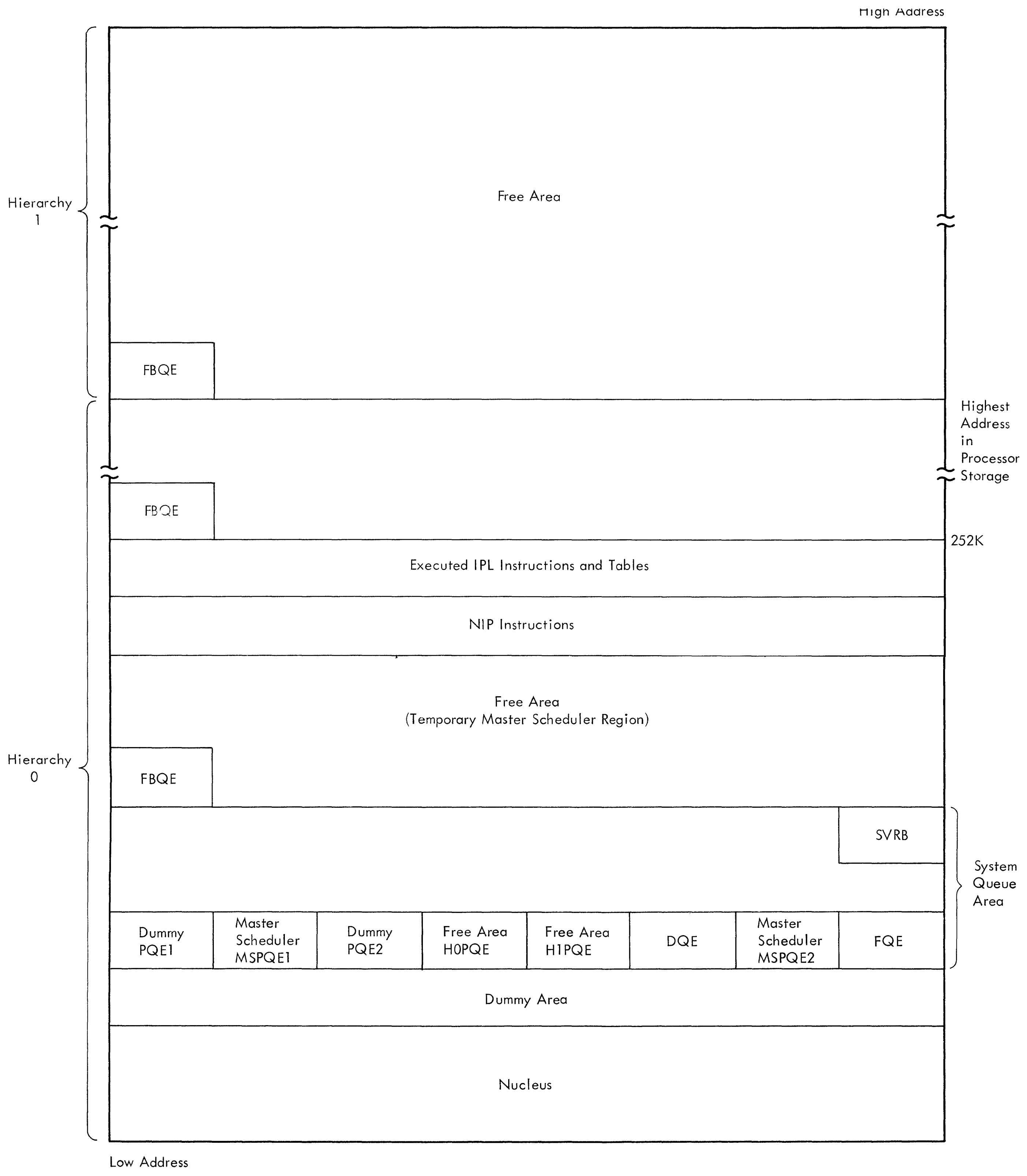


Figure 12. Defining Control Program Areas in a System With IBM 2361 Core Storage (MVT)

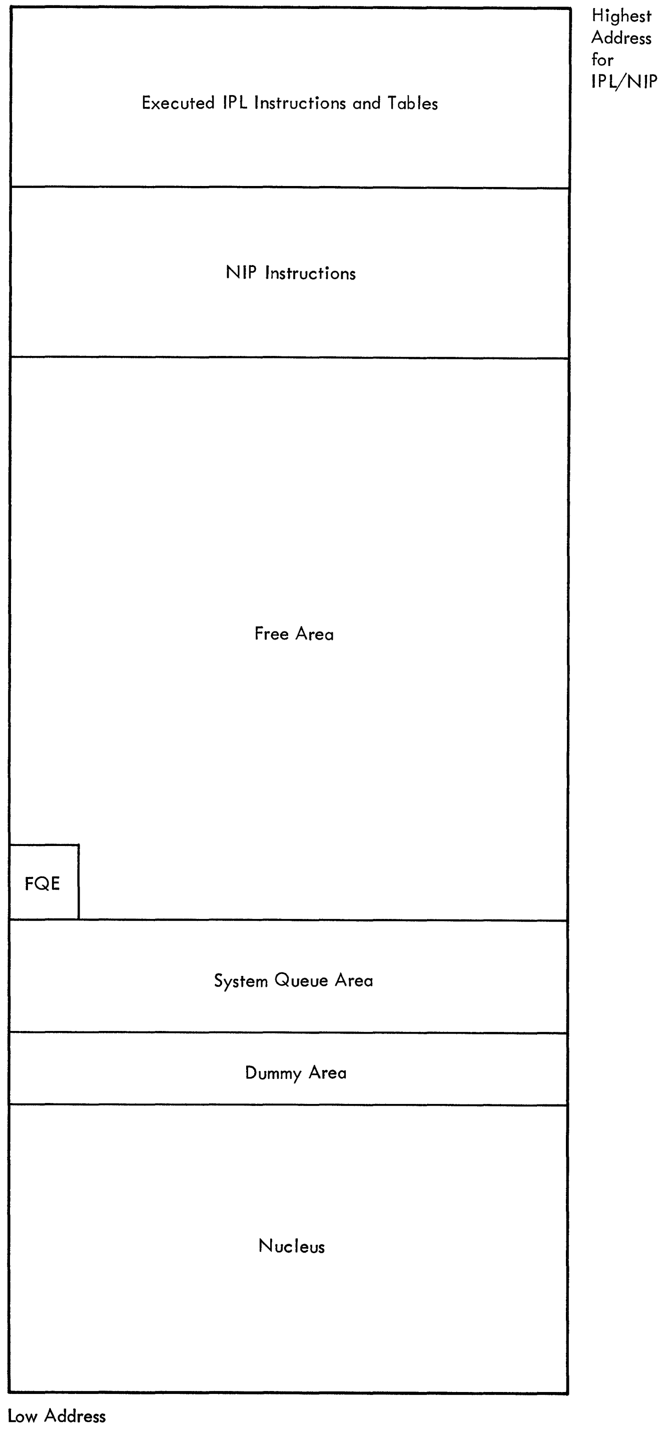


Figure 13. Defining Control Program Areas (MFT)

If the volume is not mounted, NIP issues a message to the operator requesting that the volume be mounted. NIP places the system in a simulated wait state until an I/O interruption indicates that the volume has been mounted. NIP then compares the requested serial number with the serial number of the volume mounted. If the serial numbers are not the same, NIP again requests the correct volume be mounted.

When the serial numbers are the same, NIP initializes the related UCB, marking it permanently resident so that the linkage library volume cannot be dismounted. The DEB is then initialized, using the subroutine IEACOMON.

LOCATING, FORMATTING, AND INITIALIZING THE SYS1.DUMP DATA SET

NIP determines whether the SYS1.DUMP data set exists on a direct access device or whether a tape device can be mounted. If SYS1.DUMP is on a direct access device, NIP must ensure that the data set is large enough and initialize the control blocks. The IEALOCAT routine is used by NIP to search the catalog to determine whether SYS1.DUMP resides on a direct access device.

Processing If No SYS1.DUMP Data Set Exists

If no catalog entry for SYS1.DUMP is found, a console message requests the operator to either enter the address of a tape device to be used for the data set or to state that there are no provisions for SYS1.DUMP. In an MFT system, if no provision is made for the data set, the CVTDAR pointer in the Communications Vector Table is cleared to zero, a message is issued to the operator to inform him that the SYS1. DUMP function is inoperative, and NIP processing continues. In an MVT system, the control blocks are initialized as much as possible, the CVTDAR pointer is set to the CVT and the same message is issued.

If the operator replies with the address of a tape unit, NIP verifies that the unit status of the tape is acceptable, that the tape is mounted, and that the tape is unlabeled. If any of these conditions is not met, a message is issued to the operator, and he is requested to repeat his actions. This will continue until NIP determines that all required conditions have been met. The control blocks (ECB, DCB, DEB, and IOB) required by the Write Dump routine are constructed at the high end of the nucleus and then initialized. NIP stores the addresses of the control blocks in the CVT. Regular NIP processing is resumed.

Processing a Cataloged SYS1.DUMP Data Set

If a catalog entry for the SYS1.DUMP data set is found, the NIP subroutine IEASERCH obtains the unit address and determines that the correct volume is mounted. Another subroutine, IEAUCBFN, searches for the UCB and obtains the UCB pointer. (If no UCB is found, processing continues as though no data set exists.)

Using specific device information from the UCB, NIP issues an OBTAIN macro instruction. If the OBTAIN fails, NIP determines the reason for failure and takes appropriate action. If failure was due to a permanent I/O error or an unsuccessful GETMAIN, processing continues as though no SYS1.DUMP data set exists. If space for the data set has not been allocated, NIP issues an ALLOCATE macro instruction for the required space and writes an EOF as the first record of the allocated data set. Processing then continues as described under "Final Control Block Initialization."

If the OBTAIN is successful, indicating the data set has been allocated, NIP determines if the data set consists of a single extent. If the data set does not consist of a single extent, NIP continues as described under "Scratch Data Set." For a single-extent data set, NIP determines whether or not it is large enough to contain a full storage dump. If the data set is too small, NIP reads the first record to test for an EOF that would indicate the data set is empty. If the first record is EOF, processing continues as described under "Scratch Data Set." If the data set is too small, but contains a dump, processing continues as if no data set exists. The message sent to the operator requests a tape to be mounted for the SYS1.DUMP data set. If the data set is large enough, NIP continues with final control block Initialization.

The possible conditions and the resultant actions taken are:

- Data set contains a dump, processing continues as if no data set exists. (Message is issued to operator.)
- Data set does not contain a dump, allocated space too small. Processing continues as described in the section "Scratch Data Set."
- Data set does not contain dump, allocated space is sufficient. Processing continues with final control block initialization.

Scratch Data Set

If a data set is not acceptable because it consists of multiple extents, or if an acceptable data set is too small, NIP attempts to scratch the data set. If the data set is successfully scratched, a message is issued to the operator to inform him of the action, and NIP issues an ALLOCATE macro instruction to reallocate the data set. NIP writes an EOF as the first record of the just-allocated data set. If either the scratch or allocate is unsuccessful, processing is resumed as though no SYS1.DUMP data set exists.

Final Control Block Initialization

When the data set has been initialized, NIP initializes the necessary control blocks (ECB, DCB, DEB, IOB) and stores the address of the blocks in the CVTDAR field of the Communications Vector Table.

DETERMINING USER OPTIONS (OPTIONAL)

If operator communication was selected at system generation, NIP issues the message IEA218I MOD=nnn [,ALTSYS=xxx] [mmm] ASSUMED. The model number (nnn) in the message is the model number specified at system generation. The ALTSYS= part of the message describes the alternate SYSRES device address and appears only if the Dynamic Device Reconfiguration option was selected during system generation. The machine designation (mmm) indicates that the model number (nnn) is part of the System/360 (mmm=360) or System/370 (mmm=370) series. This message is written so the operator can determine whether the correct model has been assumed and, if not, specify the correct model number in the MOD= parameter when he enters the system parameters. This message is also written to inform the operator of the alternate SYSRES device that was specified at system generation so that he may specifyanother device, if he desires, in the ALTSYS= parameter when he enters the system parameters. If an invalid address is specified, no message will be issued to indicate this condition. The system will use the alternate system residence device specified at system generation.

NIP then requests the operator to enter parameters for those options which may be modified by the user during NIP processing. NIP reads the reply, and when a parameter (consisting of a keyword and user-supplied values) is recognized, it places the parameter values in the buffer for the corresponding option. If options chosen at system generation are not specified by the operator, NIP uses the default specifications for the option. The operator may delete an option from the system by entering a comma in place of the parameter value (that is, RAM=,). In this case NIP sets a null switch in the buffer area for the option and processing will later be bypassed. If the operator enters a parameter in the wrong format, NIP issues a message indicating the error. The operator may respecify the parameter.

The user options which may be modified and the corresponding keywords are:

- Additional resident modules. In systems with MVT, these modules are loaded into the link pack area. In systems with MFT, these modules are loaded into the optional resident reenterable module area if included in the system. In systems with MFT that do not include the resident reenterable module area, only the resident access method modules can be loaded into the nucleus (RAM=).
- Additional or alternate resident error recovery modules (RERP=).
- A resident module list resulting from BLDL information (BLDL=).
- Additional resident SVC routines (RSVC=).
- User-specified request for hierarchy structure (HIARCHY=).
- User-specified model numbers for MCH, CCH, and block multiplexer initialization (MOD=).
- A larger system queue area (SQS=).
- An alternate SYSRES device different from that specified at system generation (ALTSYS=).
- User-specified device for hard copy log (HARDCPY=).
- User-specified minimum partition for job initiation (MIN=).

For systems with MVT only:

- Changes to time-slice groups and length of time slice (TMSL=).
- User-specified number of blocks for use as a buffer by the initiator (QBF=).
- User-specified number of 2K blocks for the master scheduler region size (MPS=).
- Additional resident modules to be loaded into the secondary Link Pack Area of hierarchy 1 (HRAM=).
- Additional resident SVC routines to be loaded into the secondary Link Pack Area of hierarchy 1 (HSVC=).

If the operator enters a parameter for an option not selected at system generation, NIP issues a message indicating the option is not supported.

INITIALIZING OPTIONAL CONTROL PROGRAM FUNCTIONS

Certain portions of main storage must be set aside for the later use of the control program. Some functions must be initialized for use by NIP, and other functions dependent on parameter lists established at system generation and stored in the SYS1.-PARMLIB data set must be initialized before NIP can complete its processing. Some of these functions may have been altered by the operator replies which have just been read from the console and stored in the appropriate parameter lists.

EMULATOR WARNING MESSAGE

If the emulation option was selected at system generation on the System/360 Model 85 or System/370 models (EMULATOR macro), NIP issues message IEA125I EMULATOR COMPATIBILITY FEATURE ASSUMED LOADED INTO WCS. This message is written to remind the operator that, if emulation is to be performed, the compatibility feature microcoded data set must have been loaded into Writable Control Storage (WCS) prior to IPL. NIP issues this message only if the current model ID is the same as the model ID specified at system generation. For information about loading WCS in the Model 85, see Emulating the IBM 7094 on IBM Models 85 and 165 using OS/360, GC27-6951.

INITIALIZING THE PARAMETER LIBRARY

The parameter library data set (SYS1.-PARMLIB) contains the parameter lists and module lists for system options. These lists are used by NIP to determine the modules to be loaded for the RAM, BLDL, and RSVC functions and the data sets to be concatenated with the system linkage library.

NIP issues a LOCATE macro instruction to find SYS1.PARMLIB. If no entry for SYS1. PARMLIB is found in the catalog, NIP assumes that the data set resides on the system residence volume. NIP then issues an OBTAIN macro instruction to determine the actual track address of the data set on the appropriate volume. This address is stored for later use by the initialization routines which must access SYS1.PARMLIB for parameter lists.

Note: If the OBTAIN macro instruction fails, none of the resident options can be initialized. The message IEA211I OBTAIN FAILED FOR SYS1.PARMLIB is written to the operator and followed by the message IEA208I, indicating the inoperative resident functions.

BUILDING THE LINKLIB LIST (PCP, MFT, MVT)

NIP builds the linkage library list (LINKLIST) and constructs and initializes the DEB as needed for multiple extent, multiple volume linkage library. NIP issues a GETMAIN macro instruction to obtain main storage in which to build the list. The linkage library list is then read from SYS1.PARMLIB into a buffer. Each data set name read into the buffer is then transferred to the dynamically acquired area as an entry in the LINKLIST. Each entry consists of a flag byte and a 44-byte name (left-adjusted, padded with blanks if necessary). The last name in the list is followed by 1 byte containing X'F0'.

After the last name is placed in the LINKLIST, NIP issues a LOCATE macro instruction for each data set name. The LOCATE is issued in the same order as the names in the list, and for each LOCATE, an addition is made to the LINKLIST. If the LOCATE is successful, the entry consists of a flag byte and a six-character volume label. The flag byte is set to X'00'. If the LOCATE is not successful, the entry consists of seven bytes set to X'FF'.

NIP then searches all the UCBS to find which devices have the requested volumes mounted. If a needed volume is not represented in the mounted volume identifier in the UCB, the flag byte is set to X'FF' and NIP issues the message IEA131A MOUNT LINKLIB VOL(s) for the volume indicated as not mounted. If the operator replies with an EOB signal from the console, the volume label field in all LINKLIST entries and the DSNAME flag byte specifying that volume are set to X'FF' (not used). When the correct volume is mounted, the UCB is marked permanently resident.

When all required volumes have been mounted (or their use canceled by operator action), NIP issues an OBTAIN macro instruction for each data set name. If the OBTAIN fails for any reason, the flag byte of the data set name and the volume label and its flag byte are overlaid with X'FF'. The appropriate message is issued to the operator.

If the OBTAIN is successful, the flag byte remains set to X'00' and a DEB is constructed and initialized in the same manner as the DEB for the SYS1.SVCLIB. No more than 15 data sets may be concatenated with SYS1.LINKLIB.

The pointer to the end of the nucleus, IEANUCND, is updated each time the DEB is expanded and initialized for a concatenated LINKLIB volume. Thus, IEANUCND always points to the highest address in the nucleus.

INITIALIZING FOR GENERALIZED TRACE FACILITY

NIP determines if the system being initialized is for a System/370. If it is, control register 8 is initialized to to enable MONITOR CALL interruptions for class 1 monitoring. The class 1 monitoring capability of System/370 is initiated by loading bits 16 through 31 of control register 8 with a hexadecimal 4000. In System/360, the MONITOR CALL instruction is simulated, and initialization is not required.

GTF is primarily a debugging tool that is invoked by the operator issuing a START command. When GTF is invoked, the optional trace table facility is disabled and the trace functions are performed by GTF. For additional information on the Generalized Trace Facility, refer to Service Aids.

ESTABLISHING BLOCK MULTIPLEXER CHANNEL CAPABILITY

NIP determines if the system being initialized supports the Block Multiplexer Channel and if the user selected the channel to be used in block multiplexer mode (this selection must be made at system generation). In systems with control registers, a control register flag is set to one, to indicate block multiplexer capability. The control register flag is checked by input/output supervisor routines to determine the correct I/O technique.

Block multiplexer support is determined by comparing the value at location MODELID with the entries in the list BLMPXCPU. MODELID reflects the system-generated model number or the model number supplied by the operator in reply to the message IEA101A SPECIFY SYSTEM PARAMETERS. BLMPXCPU is a subset of the list CPUTAB, which is a list of supported model numbers. The subset list contains those models that support the Block Multiplexer Channel.

BUILDING THE TCB ADDRESS TABLE (MFT)

NIP builds and initializes a table of TCB addresses for systems with MFT. The location of the table and the number of table elements are placed in the communications vector table. If MFT with subtasking is included in the system, the number of table elements is calculated to be the number of TCBs created during system generation (depending on the system options chosen), plus the number of possible attached subtasks that can execute concurrently. The latter value is based on the value specified by the user in the SQS= parameter (see "Expanding the System Queue Area").

RECOVERY MANAGEMENT INITIALIZATION

After the DEB for the SYS1.LINKLIB has been initialized, NIP initializes the System Environment Recorder (SER) program if it was selected at system generation. NIP determines if the operator specified a model in the MOD= parameter and uses this model number for initialization. If the operator did not specify a model in his reply, the model specified during system generation is used.

NIP checks the SER option table to determine the option to be used. The SER option table is built within NIP at system generation and contains the options for model/storage size variances selected by the user, since the level of SER support can be varied according to model and storage size combinations. If the table does not contain the correct model combined with either the actual storage size or a value less than the actual storage size, SER initialization is bypassed, the SEREP interface is left intact, and the message IEA217I SEREP INTERFACE ESTABLISHED is issued. If the option table contains an acceptable model/storage size entry, the specified SER option is initialized. If the table contains more than one acceptable entry, the SER option specified in the entry containing the storage size specification closest to the actual storage size is the option initialized.

After the appropriate SER option has been determined, the resident SER module, with its initialization appendage, is loaded by NIP from SYS1.LINKLIB. The module to be loaded is found by using a module name in the form IFBSRxmn, where x is the level of SER support (0 or 1), and mn is the model number.

If the SER0 option is selected, NIP establishes a logical connection between the resident portion of SER and the non-resident portion, module IFBSER00, which resides on the linkage library. NIP finds the address of IFBSER00 and stores the address in the resident portion of SER.

To find the address of IFBSER00 in the linkage library, NIP reads the partitioned data set directory record for the module. From the relative track address in this record, NIP calculates the absolute address (CCHHR) of the first text record and places this address, along with the unit address of the device, in the SER0 control section in the nucleus. In the final NIP routine, a read CCW, which points to a location 20 bytes above the end of the nucleus, is placed in the resident SER location IFBADDR.

REBUILDING THE SYSTEM QUEUE AREA

After all additions to the high end of the nucleus have been completed, the system queue area can be relocated to its permanent location. The queue area is moved to a location adjacent to the end of the nucleus (indicated by IEANUCND).

In a system with MFT, the system queue area is moved and the FQE for the dynamic area is adjusted to reflect the address of the first byte of storage above the SQA. This address is obtained by adding the system generation chosen size of the SQA to the address in IEANUCND, the end-of-nucleus pointer. Free area then is defined as all storage from the end of the SQA to the beginning of NIP code (see Figure 14).

In a system with MVT, the SQA is moved in the same manner and the control blocks (HOPQE, FBQE) are adjusted to indicate the change in storage layout (see Figure 15).

In both systems, the size of the SQA may be changed by operator response to the message IEA101A SPECIFY SYSTEM PARAMETERS (see "Expanding the System Queue Area").

Message Buffer for MCS (Optional)

After the system queue area has been relocated to its permanent position, a 2K portion is obtained for use as a message buffer in a system with MCS. This area is used to store copies of all system messages and operator replies that are issued prior to completion of console initialization. The area is freed by either the communications task console initialization routine (if the hard copy log is a console), or by the log initialization routine (if the hard copy log is the system log).

Note: A temporary buffer is used for NIP messages before the SQA is relocated. The message buffer does not contain any messages generated as a result of the "list" parameter in response to the IEA101A SPECIFY SYSTEM PARAMETERS message.

INITIALIZATION FOR TIME-SLICING (OPTIONAL)

If the time-slicing option was selected at system generation, NIP converts the millisecond value in the time-slice control elements (TSCEs) to timer units. Because this option may be canceled by the operator, NIP first checks the time-slice value in the first TSCE for zero. This value is set to zero if time slicing has been canceled, and NIP then branches around the routine. If time slicing is not canceled, NIP compares the millisecond value for the time-slice interval with the minimum value of 20 milliseconds. If the value is less than the minimum, NIP increases it to 20 milliseconds, issues a message to the operator that the interval has been increased, and proceeds with millisecond-to-timer unit conversion.

NIP multiplies the millisecond value in the TSCE by 1000, and divides the result by 26, obtaining the number of timer units. This value is placed in the TSCE in the field TSCEMSEC. The process is repeated for each TSCE on the chain of time-slice control elements.

ALLOCATING, OPENING, AND FORMATTING THE ROLLOUT DATA SET (OPTIONAL: MVT)

Allocation

The rollout data set (SYS1.ROLLOUT) must be cataloged. The data set may reside on any system-supported DASD, but must occupy no more than a single volume.

NIP locates the rollout data set by using the catalog to obtain its volume serial number. If the data set has not been catalogued, NIP notifies the operator and bypasses further rollout processing. When the volume serial number has been found, NIP compares it with the numbers in the direct access device table to determine whether the desired volume has been mounted on the correct device type, as indicated in the catalog entry. If the volume has not been mounted, NIP requests that the operator mount it.

NIP then calculates the number of records per track, using the device type of the rollout data set (obtained from the device table) and a 1024-byte record size. This number is converted to the corresponding number of cylinders, and is used later by the Rollout program.

NIP issues an OBTAIN macro instruction for the DSCB(s) of the rollout data set. This is to ensure that sufficient space is available for the data set. If a DSCB is not found, space has not been allocated; NIP issues an ALLOCATE macro instruction for the required space. If the space has been allocated but is not sufficient, NIP issues a SCRATCH macro instruction to eliminate it, and an ALLOCATE macro instruction to provide new space of sufficient size. If sufficient space is not available, NIP sets a program switch to prohibit rollouts and bypasses further rollout processing.

Whenever NIP issues an ALLOCATE macro instruction it follows with an OBTAIN macro instruction to verify that the new DSCB has been created.

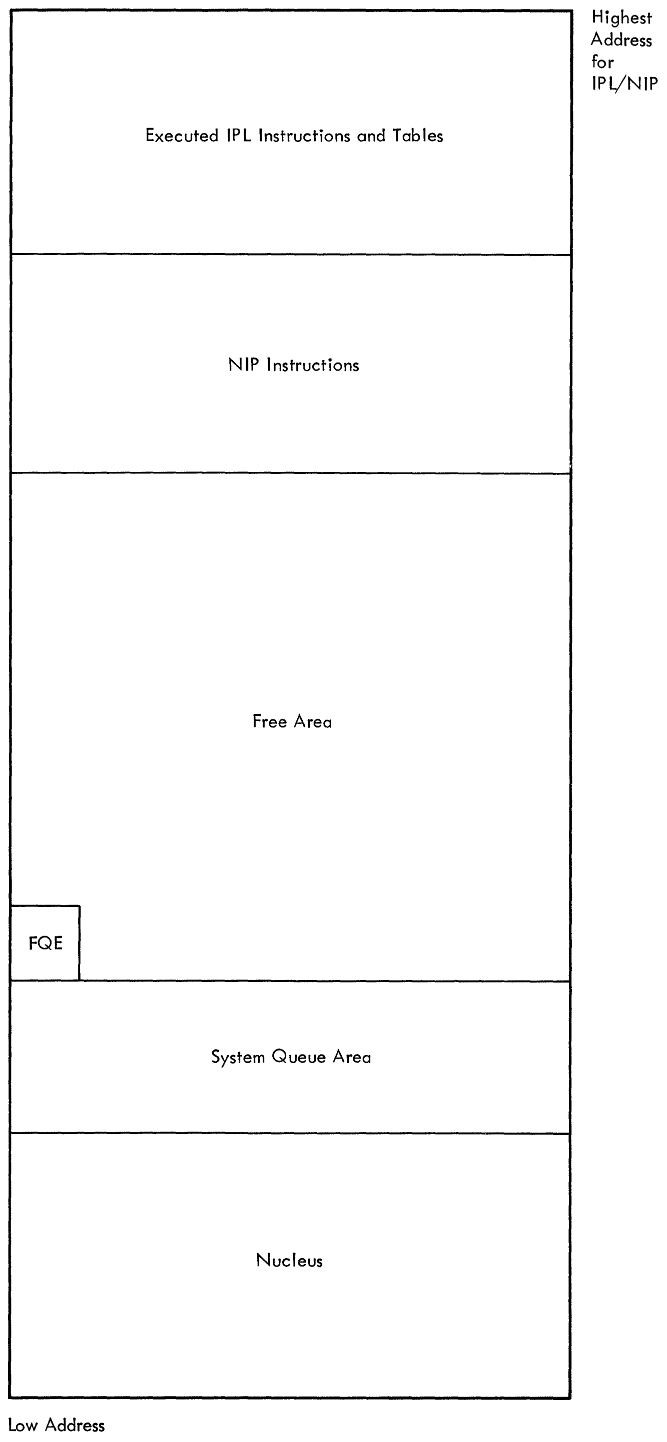

Highest Address for IPL/NIP
Executed IPL Instructions and Tables
NIP Instructions
Free Area
FQE
System Queue Area
Nucleus
Low Address

Figure 14. Main Storage in an MFT System After Rebuilding the System Queue Area

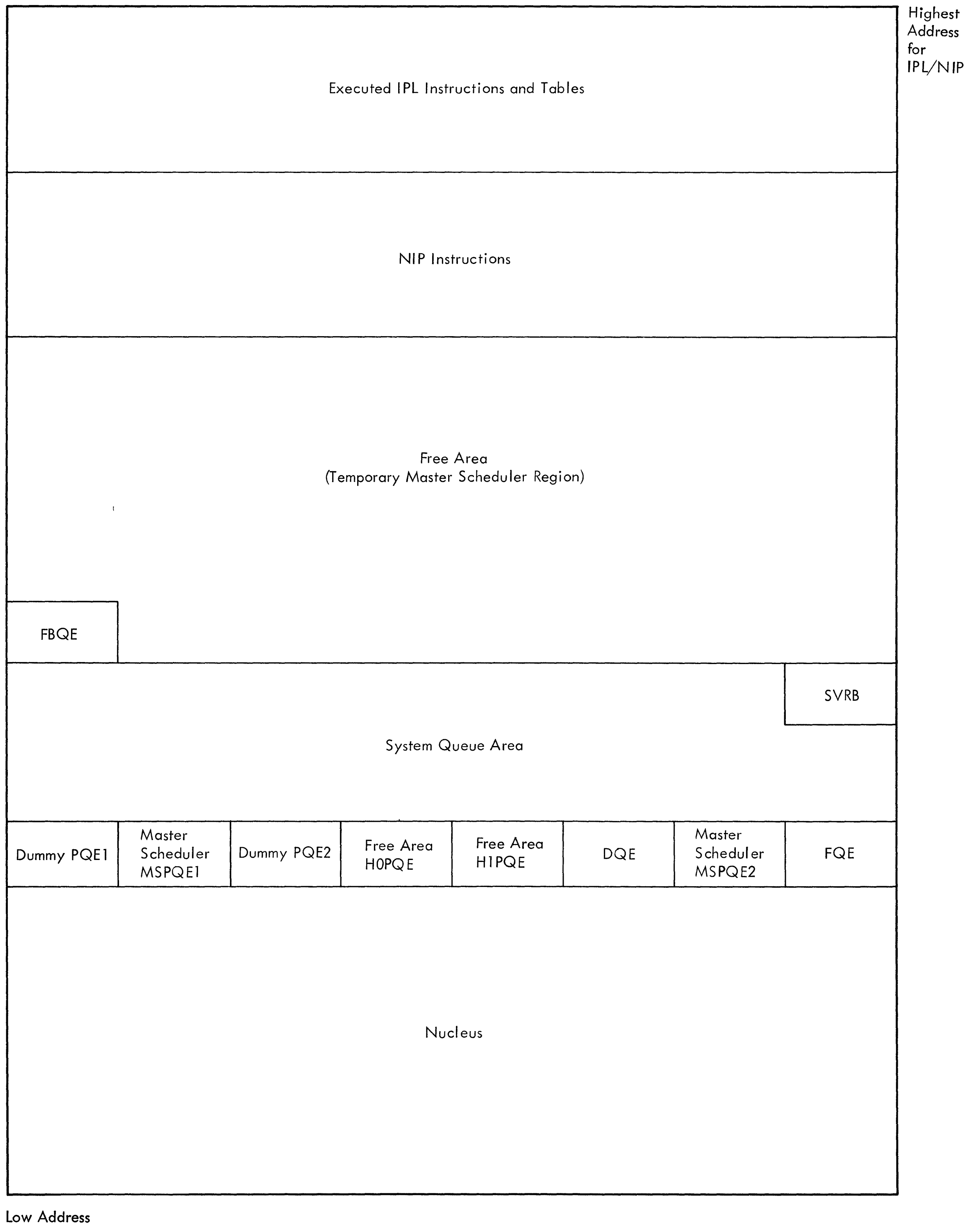


Figure 15. Main Storage in an MVT System After Rebuilding the System Queue Area

Opening and Formatting

NIP opens and formats the rollout data set. Additional formatting may be required from one IPL to the next (for example, if the system link pack area is modified); if the data set has been partially formatted, NIP begins additional formatting at the point where previous formatting ended.

NIP opens the rollout data set by building the necessary control blocks and issuing an OPENJ macro instruction.

NIP uses the "last TTR written" field. It completes rollout initialization by attaching the DEB to the rollout DCB, and the rollout appendage vector table to the DEB.

INITIALIZATION FOR MACHINE-CHECK HANDLER, MODEL 65

When the MCH option has been selected at system generation, NIP initializes pointers for MCH and completes the SYS1.ASRLIB data set by writing on it copies of all refreshable nucleus modules. NIP uses the MOD= parameter of the "specify system parameters" message to set the IGFMOD field to the IBM System/360 model number. The MCH programs use this field to determine whether recovery processing is to take place.

NIP locates the SYS1.ASRLIB data set through the catalog. If an entry for the data set cannot be located in the catalog, or if it resides on a volume for which there is no UCB, NIP notifies the operator and cancels the MCH function. If the data set is located on an unmounted volume, NIP requests that the operator mount the volume, and waits until SYS1.ASRLIB has been successfully located and mounted.

NIP then obtains the address on SYS1. SVCLIB of the MCH module which will be the first to be loaded into the transient area. The address is placed in the IGFENVCK field of the MCH common area. If the module cannot be found in SYS1.SVCLIB, or if an I/O error occurs, NIP indicates that MCH initialization has failed.

After the IGFENVCK field has been initialized, NIP processes the 12-byte entries in the nucleus refresh table (IGFNUC01), which was prepared at system generation. Each entry identifies a refreshable nucleus module. Using the module name from the nucleus refresh table and the size and address from IPL SIZTABLE and IPL ADRTABLE, NIP writes the modules on SYS1.ASRLIB. NIP then places the disk address and length of each record in the nucleus refresh table. Finally, the table itself is written on SYS1.ASRLIB as the last record, and the address of the table is stored in IGFNUC01, an 8-byte field with this format:

Bytes	Contents
0-4	CCHHR (the address)
5	X'00'
6-7	length of the table

NIP indicates to the operator that initialization of SYS1.ASRLIB is incomplete if: (a) NIP is unable to find a matching address in IPL ADRTABLE for an entry in the nucleus refresh table; (b) NIP is unable to continue writing modules in SYS1.ASRLIB because of insufficient space. If an I/O error occurs while NIP is writing on SYS1. ASRLIB, NIP notifies the operator and cancels the MCH function.

If the resident BLDL option has been selected, NIP places the address of the resident LINK area into the IGFBLDL field of the MCH common area, and places the address of the resident SVC BLDL area into the IGFSVBLD field of the MCH common area.

If resident Type 3 and 4 SVC routines have been specified, NIP stores the address of the resident SVC area in the IGFSVCQ field of the MCH common area.

At the successful completion of MCH initialization processing, NIP stores X'FF' in the high-order byte of the IGFBLDL field of the MCH common area.

INITIALIZATION FOR MACHINE-CHECK HANDLER, SYSTEM/360 MODEL 85 AND SYSTEM/370

The Machine-Check Handler (MCH) for System/360 Model 85 and System/370 is initialized by the module IGFMCHF0. This module is part of MCH, but it is executed during NIP processing. Before attempting to load the MCH module, NIP compares the model number of the system being initialized to the model number specified at system generation. For the Model 85, if the model numbers are not equal, MCH initialization is not performed.

However, for System/370 machines that have MCH, the secondary model support (SMS) option is provided. That is, MCH initialization is provided for all System/370 machines (except the Model 195) which are specified at system generation as primary or secondary models, regardless of the System/370 machine on which the system is IPLed. If the model numbers are equal or if SMS is specified at system generation, NIP loads the MCH resident nucleus (IGFMCH10 for the Model 85, IGFMCHE0 for other models) into the dynamic area and passes control to the loaded module. The

MCH resident nucleus module performs its own initialization and relocates itself (see Figure 16) so that it becomes contiguous with the control program nucleus. The pointer to the end of the nucleus is updated to point to the end of the MCH resident nucleus and control is returned to NIP. NIP deletes the copy of the load module from the dynamic area and then loads the MCH initialization module IGFMCHF0 and passes control to it.

<u>IGFMCHF0 Processing -- Stage 1</u>

During Stage 1 of its processing, IGFMCHF0 performs the following functions:

1. Allocates space for the MCH Transient Area. Allocation is accomplished by adding to the end-of-nucleus pointer the number of bytes needed for the transient area. For the Model 85, 3K bytes are allocated; for System/370, only 1K bytes are needed. IGFMCHF0 then loads into the transient area the module which will initially reside there (for System/360 Model 85, the initiator/terminator; for System/370, the Soft Machine-Check Handler).

2. Allocates space for, and initializes the Model Dependent Common Area.

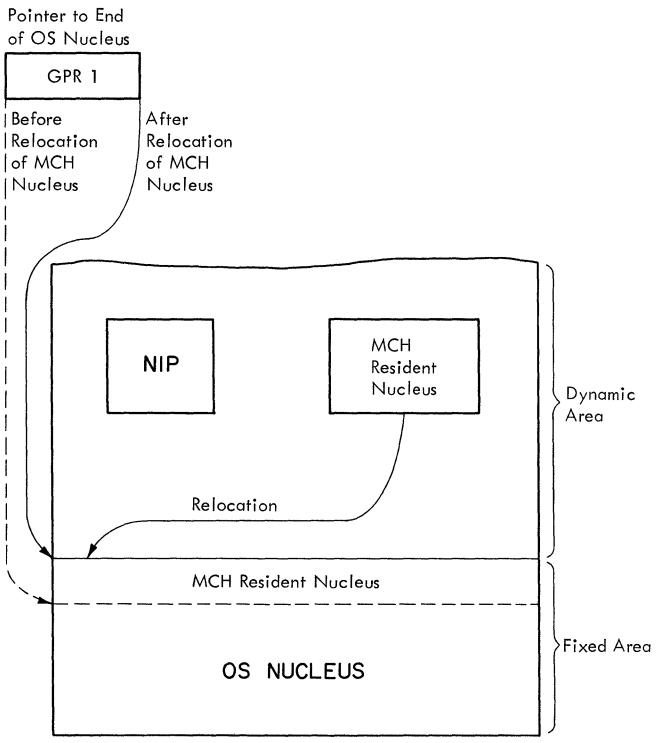

Pointer to End of OS Nucleus
GPR 1
Before Relocation of MCH Nucleus
After Relocation of MCH Nucleus
NIP
MCH Resident Nucleus
Dynamic Area
Relocation
MCH Resident Nucleus
Fixed Area
OS NUCLEUS

Figure 16. Loading the MCH Resident Nucleus

3. Initializes IGFMSB00, the machine status block in the control program nucleus, with information for machine status control and multiple console support (MCS) control information for the nucleus.

4. Allocates the Model Independent Common Area. This area serves as the MCH communications area.

5. Allocates the Fixed Logout Save Area. This step is performed only for System/370 initialization. The fixed logout save area is 280 bytes.

6. Allocates the Extended Logout (System/370 models only). A pointer to the extended logout is placed in control register 15.

7. Initializes control register 14 with the machine check mask (System/370 only).

8. Initializes the Model Independent Common Area.

9. Initializes pointers in the MCH nucleus.

10. Initializes the Dispatcher. A pointer to the MCH nucleus Post ECB routine is placed in the Dispatcher.

11. Initializes the machine-check new PSW.

12. Initializes the Module Scheduler. The IDs and TTRs of the MCH transient modules on SYS1.SVCLIB are placed in the MCHTTRS field in the MCH Model Independent Common Area, and the successor IDs for those modules having successors are placed in the MCHNXIDS field.

13. Allocates Subsystem Common Area (Optional).

Figure 17 illustrates the structure of the MCH Area at this stage of NIP processing.

After initializing the Module Scheduler, IGFMCHF0 returns control to NIP. NIP saves the end-of-nucleus pointer and deletes the copy of IGFMCHF0 that was loaded in the dynamic area.

INITIALIZATION FOR THE CHANNEL-CHECK HANDLER

The Channel-Check Handler supports the 2860, 2870, 2880 stand-alone channels, and the Model 135, 145 and 155 integrated channels.

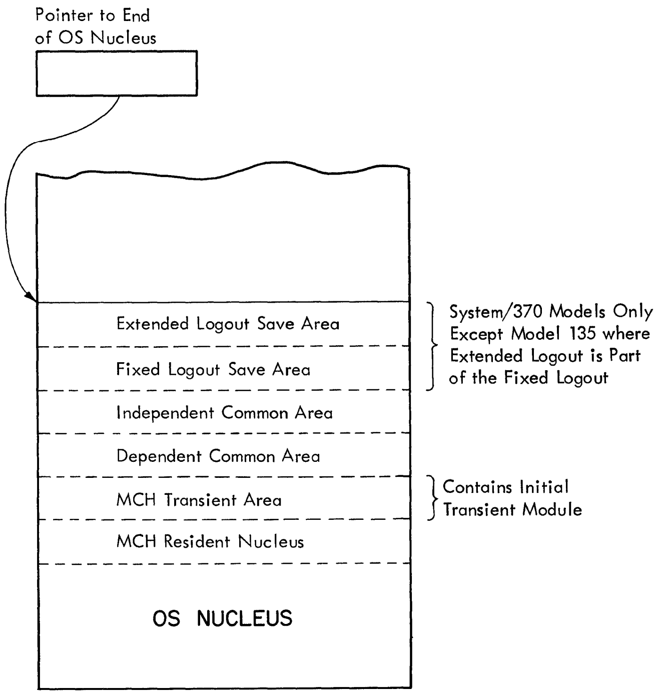

Pointer to End of OS Nucleus
Extended Logout Save Area
Fixed Logout Save Area
Independent Common Area
Dependent Common Area
MCH Transient Area
MCH Resident Nucleus
OS NUCLEUS
System/370 Models Only Except Model 135 where Extended Logout is Part of the Fixed Logout
Contains Initial Transient Module

Figure 17. MCH Nucleus at Conclusion of IGFMCHF0 Stage 1 Processing

The Channel-Check Handler (CCH) consists of one central channel- and model-independent module which is permanently resident, and six channel-dependent channel error analysis modules, one for each of the supported channels. In addition, there is a CCH Initialization routine which is used for initialization purposes only, and a move routine within each of the analysis modules.

At system generation, the channels that are to be part of the system are specified; they are indicated in the channel configuration word. For a system which includes CCH, the central CCH module is loaded as part of the nucleus. The analysis modules for the specified channels reside on SYS1. LINKLIB, and the needed modules are loaded by NIP.

During nucleus initialization, the CCH initialization routine indicates to NIP which analysis modules are to be made resident. NIP dynamically loads those modules; only those channel-dependent analysis modules needed to support a specific channel configuration occupy space in the nucleus.

If the CCH option was selected at system generation and the MOD= parameter value is 65 or greater, NIP loads the CCH initialization routine IGFCCHIN. NIP gives control to IGFCCHIN, and passes three parameters: the address of the channel configuration word, the address of a dummy buffer (indicating the unused area of storage immediately after the resident nucleus), and the address of a parameter list. The CCH Initialization routine places indicators in the parameter list to indicate to NIP which of the analysis modules are to be made resident.

The CCH Initialization routine first determines what channels are both in the system and online, and uses this information to fill the parameter list. The parameter list consists of a two-byte hexadecimal code for each module needed (the last two digits of the module name). Additionally, the channel type for each channel is stored in the first byte of the appropriate entry in the channel pointer table and a hexadecimal code is stored in the fourth byte of the same entry, to be used by the move routine in establishing linkages.

The CCH Initialization routine then sets up a record entry area three times the size of the maximum record length needed for the channels in the system. This area is established beginning at the address of the dummy buffer and is initialized to zero. The CCH Initialization routine places the address of this area in the parameter table and in the central CCH module. The size of the individual record is saved in the halfword before the record entry area and in the CCH module. The master byte (see "CCH Communications Scheme," below) and the release number are also stored ahead of the record area. If a 145 channel or a 2880 channel is attached to a Model 165, the address of the channel logout area is placed in location 172 (dec). For the 2880 channel attached to a Model 165, the address of the channel logout area is also placed in word 5 of the parameter table. The CCH Initialization routine returns control to NIP, passing the address of the unused buffer space and the address of the parameter list.

NIP then deletes the CCH Initialization module. Using the parameter list, NIP loads each analysis module needed into the high address portion of main storage and passes control to the move routine (in the module), passing the address of the remaining dummy buffer.

The move routine obtains the address of the channel pointer table and examines the fourth byte of each address entry in the table for the hexadecimal code that identifies the module to be moved. When the move routines locates the code, it stores the address of the module in the three right-hand bytes of that entry, if it is unused, overlaying the hexadecimal code. The process is continued for each address entry in the table.

The move routine then moves the channel analysis module (except for the move routine itself) into the dummy buffer and updates the buffer address.

When the move routine completes the required move, it returns control to NIP, passing the address of the unused buffer. NIP deletes the move routine and loads the next analysis module, passing control to its move routine. This procedure continues until all modules identified in the parameter list have been loaded.

CCH Communications Scheme

The CCH Communications Scheme is illustrated in Figure 18.

The address of the first word of the LOGREC DCB is placed in the CVTDCB field of the Communications Vector Table (CVT) by system generation. The first word of the LOGREC DCB contains the address of a five-word parameter table which is the CCH portion of the I/O RMS Communications Area. The address of the parameter table is also contained in the word located at displacement -4 from the beginning of the CCH pointer table.

The parameter table contains the following information:

First word:
: contains the address of the beginning of a table of five ERPIBs which reside within the CCH central module.

Second word:
: first byte: contains a flag field which indicates, when other than zero, that there are record entries to be written. The field is initially set to zero.

: second, third, and fourth bytes: contain the address of the beginning of the record entry area, which contains three record entries of the maximum length needed for the channels in the system.

Third word:
: CCH base register value.

Fourth word:
: contains the address of the channel pointer table in the central CCH module. The channel pointer table contains pointers for each channel to the appropriate analysis module. A zero in the address entry for a channel indicates that the channel is not supported. The first byte of each address entry contains the channel ID information placed there by the CCH Initialization routine. The channel pointer table can be expanded to provide for more than seven channels.

Fifth word:
: 2880 Logout pointer.

All these tables reside in the nucleus before CCH is initialized. During initialization, the record entry area and the analysis modules are appended to the nucleus by the CCH Initialization routine and the move routines within each module. The CCH Initialization routine also stores a word preceding the record entry area. This word contains:

First byte:
: this is the master byte for CCH. Each bit indicates the presence of a type of channel.

Second byte:
: this byte contains the release level of the system in binary.

Third and fourth bytes:
: these bytes contain the maximum size for a record entry. This information is also in displacement -8 from the beginning of the channel pointer table.

LOADING OPTIONAL ERROR RECOVERY PROCEDURE MODULES

Error Recovery Procedure (ERP) modules from the SVC library are loaded according to the lists specified in the RERP= parameter. (If the RERP parameter is not used in reply to the SPECIFY SYSTEM PARAMETERS message, the default list, SYS1.PARMLIB member IEAIGE00 is used.) NIP uses the same procedures to locate and load the ERP modules as for the RAM and RSVC modules. However, NIP creates a separate chain of ERP-related CDEs and saves a pointer to this chain at location IEAAERP. The pointer in the TCB is zeroed as for RAM and RSVC modules. NIP restricts ERP loading to those modules which have module names beginning with IGE0.

INITIALIZATION FOR DYNAMIC DEVICE RECONFIGURATION SYSRES SUPPORT

If Dynamic Device Reconfiguration SYSRES support was selected at system generation, the operator has the option to change the alternate system residence device specified at system generation by specifying ALTSYS= as a system parameter. If this is done, NIP places the new alternate system residence device address in the I/O RMS Communications Area for use by Dynamic Device Reconfiguration SYSRES.

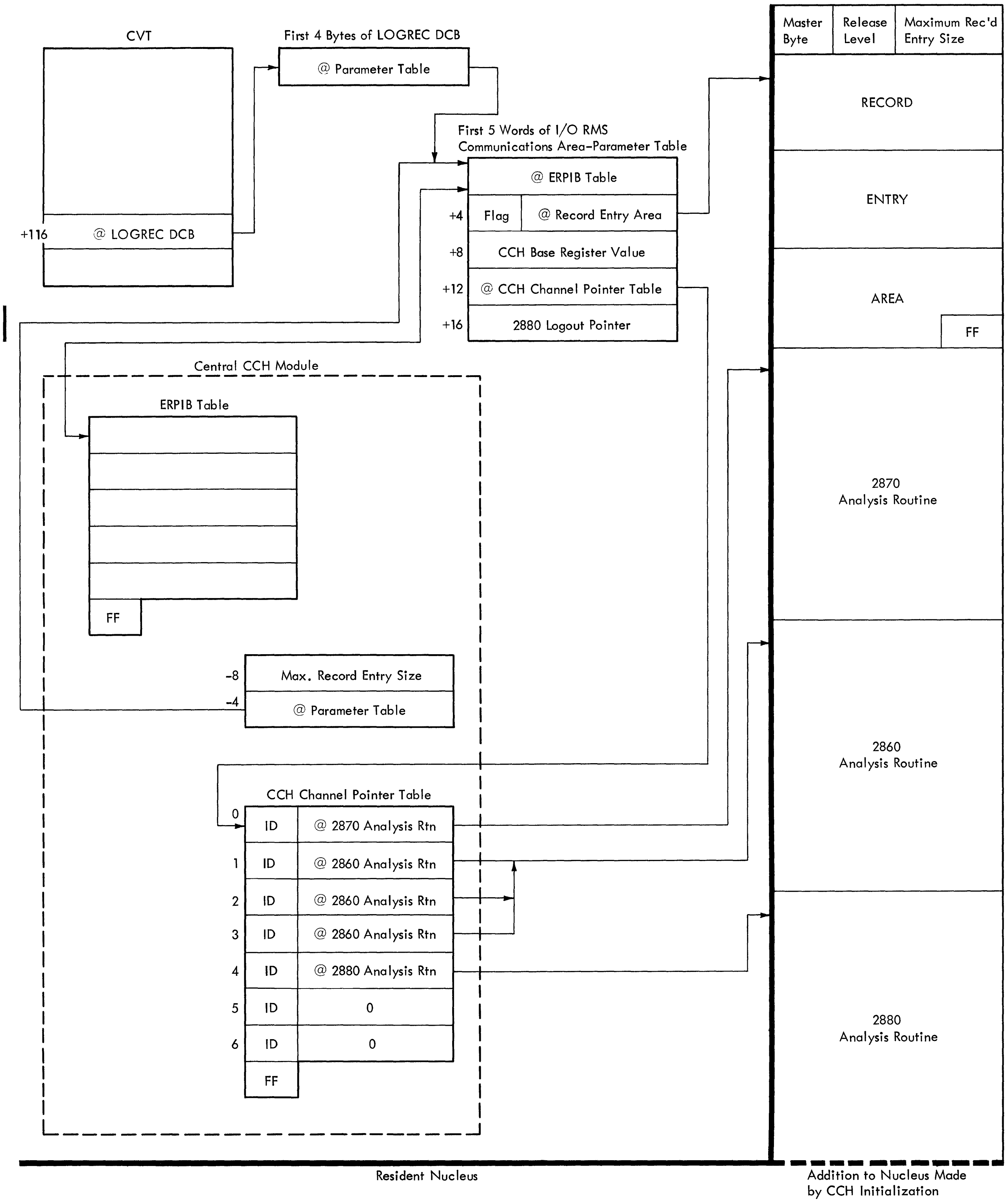


Figure 18. Communications Scheme After Initialization

RESETTING MAIN STORAGE DIVISIONS

So that it may complete its operations, NIP resets main storage divisions. The program expands the system queue area if the SQS= parameter was specified. In a system with MVT, NIP frees upper main storage by relocating its unexecuted instructions.

EXPANDING THE SYSTEM QUEUE AREA (OPTIONAL)

If the operator specified a larger system queue area in response to the SPECIFY SYSTEM PARAMETERS message, NIP resets the area's upper boundary to include the requested supplemental area.

In a system with MFT _without subtasking_, the operator enters, in the SQS= parameter, the total size of the system queue area desired. NIP checks this value to determine that it is at least 1600 bytes (the minimum size of the system queue area), and then adjusts the upper boundary. The upper boundary is rounded to a doubleword boundary, and the FQE for the dynamic area is adjusted to the resultant boundary.

In MFT _with subtasking_ the user must calculate, in addition to the minimum requirements of an MFT system without subtasking, an increase in system queue area size as follows:

Minimum requirements = N(A+B+C+4) + 4D + E

where:

- N is the number of simultaneously active tasks possible.
- A is the length of one floating-point register save area in one TCB (if necessary).
- B is the length of one task control block (TCB).
- C is the length of one timer queue element (TQE); this is zero if the interval timer is not included in the system.
- D is the number of system tasks generated during system generation.
- E is the amount of storage space required by system options selected by the user during system generation.

Note: The user should be aware that the system writer issues an ATTACH macro instruction. Thus, the user should add the number of writers active in the system to the factor N (the number of simultaneously active tasks possible). Additional information about storage requirements can be found in _Storage Estimates_.

In a system with MVT, the operator enters the supplemental size as the number of additional 2K blocks of storage to be added. NIP converts this number to a number of bytes and adds this supplemental size to the system queue area end address to determine the new upper boundary.

NIP updates affected control blocks and pointers to show the new, expanded size. The new free area size is placed in the FQE. A pointer to the FQE and the area size are placed in the DQE. NIP also updates the system environment recorder and main storage supervisor pointers to the new system queue area. NIP then places a new SVRB pointer in the transient area handler routine of the second level interruption handler.

RELOCATING NIP (MVT)

Before continuing main storage initialization, NIP relocates its unexecuted portion, thus freeing upper storage. NIP moves its instructions to the 2K block following the system queue area.

To maintain addressability, NIP computes the displacement factor for the move and uses the factor to recalculate its internal address constants. The program also resets address constants in channel command words (CCWs) and resets base registers.

When the move is complete, NIP branches to the relocated instructions to continue storage preparation.

CONSTRUCTING THE LINK PACK AREA (MVT)

After the final size of the system queue area is determined and upper storage is freed, NIP constructs the link pack area. The area consists of the optional resident BLDL list, the standard modules which are always resident, and any modules indicated by the user. These modules may be from the linkage library or the SVC library.

CONSTRUCTING THE RESIDENT REENTERABLE MODULE AREA (OPTIONAL: MFT)

The modules to be loaded into the resident reenterable module area are selected from lists contained in the SYS1.PARMLIB data set, as described by the RAM= parameter. These modules may come from the linkage or SVC libraries, and should not include SVC modules which may be loaded into the resident SVC area.

CONSTRUCTING THE RESIDENT BLDL LIST (OPTIONAL)

When the resident BLDL option is chosen, NIP constructs lists of BLDL entries for modules in SYS1.SVCLIB or SYS1.LINKLIB. In MFT systems, the lists reside as part of the nucleus. In MVT systems, the lists are part of the Link Pack Area. These modules are listed in member IEABLDxx of SYS1. PARMLIB (xx is the parameter value supplied by the operator, or is 00 by default).

NIP first builds lists of entries, containing only the names of the specified modules. In a system with MFT, the list is built adjacent to the system queue area (see Figure 19). In MFT systems, the BLDL list is built in the part of main storage where it will remain, and becomes part of the nucleus. In a system with MVT, the lists are built immediately following the relocated NIP instructions and NIP then issues a GETMAIN macro instruction to obtain permanent storage at the upper end of storage for the BLDL list. NIP then moves the constructed table to its permanent location (see Figure 20).

NIP issues the BLDL macro instruction to fill in the list entries with the linkage library directory information. Should a permanent I/O error occur during BLDL execution, a message is sent to the operator and NIP operation continues. Should any specified module directory entries not be found in LINKLIB or SVCLIB, a message is sent to the operator giving the corresponding module names.

If the user requested a listing of modules with resident BLDL list entries, NIP writes, in operator messages, the module names as it obtains them from SYS1.PARMLIB.

LOADING THE LINK PACK AREA MODULES (MVT)

In a system with MVT, NIP loads into the link pack area any SVC library and linkage library modules required by the control program or specified by the user as resident. The standard modules are listed internally in NIP, but the user lists are contained in the SYS1.PARMLIB data set. A message is issued to the operator to inform him that the optional routines (BLDL, RAM, RSVC, RERP) are not operative if NIP was unable to locate and open the SYS1.PARMLIB data set. NIP first loads the standard modules, then the optional resident access modules, and finally the resident SVCs. If the system includes IBM 2361 Core Storage and Main Storage Hierarchy Support, a secondary Link Pack Area may exist in hierarchy 1 storage. Only those modules that are specified by the user options HRAM= and HSVC= are loaded into this secondary LPA.

Standard Modules

For each standard module to be loaded, NIP issues a BLDL macro instruction. When NIP obtains from BLDL the partitioned data set directory information for the module, it sets the attribute field to show that the associated routine is reentrant regardless of Linkage Editor assigned attributes. NIP then issues a LOAD macro instruction specifying the constructed BLDL list for that module.

Optional Linkage Library Modules

When the standard modules are loaded, NIP obtains the module lists specified by the RAM= parameter values. (The default module list is IEAIGG00.) NIP issues the BLDL macro instruction for each module. After NIP sets the attribute field of the found module BLDL list, the program issues a LOAD macro instruction.

When loading of standard and of any optional modules is complete, NIP obtains a pointer to the chain of contents directory entries (CDEs) for the loaded modules. NIP flags each CDE to show that NIP loaded the module for permanent residence. NIP sets the pointer to the chain in the nucleus-resident LINK, LOAD, XCTL service routine and zeros the temporary chain pointer in the master task control block (TCB).

Optional Type 3 and Type 4 SVC Modules

Lists indicated by the RSVC= parameter values are used to load modules from the SVC library. (The default list is SYS1. PARMLIB member IEARSV00.) NIP uses the same procedure to locate and load the SVC modules as for the RAM modules. However, NIP forms a separate chain of SVC-related CDEs. NIP saves a pointer to this chain at location IEAQSVCQ. The pointer in the TCB is zeroed as for the linkage library modules.

Also, when the loaded module is the first (or only) module of an SVC routine, NIP updates the corresponding SVC table entry to show that the module is resident and to give the module's main storage address. (These modules are recognized by their names which are of the form IGC00xxx.)

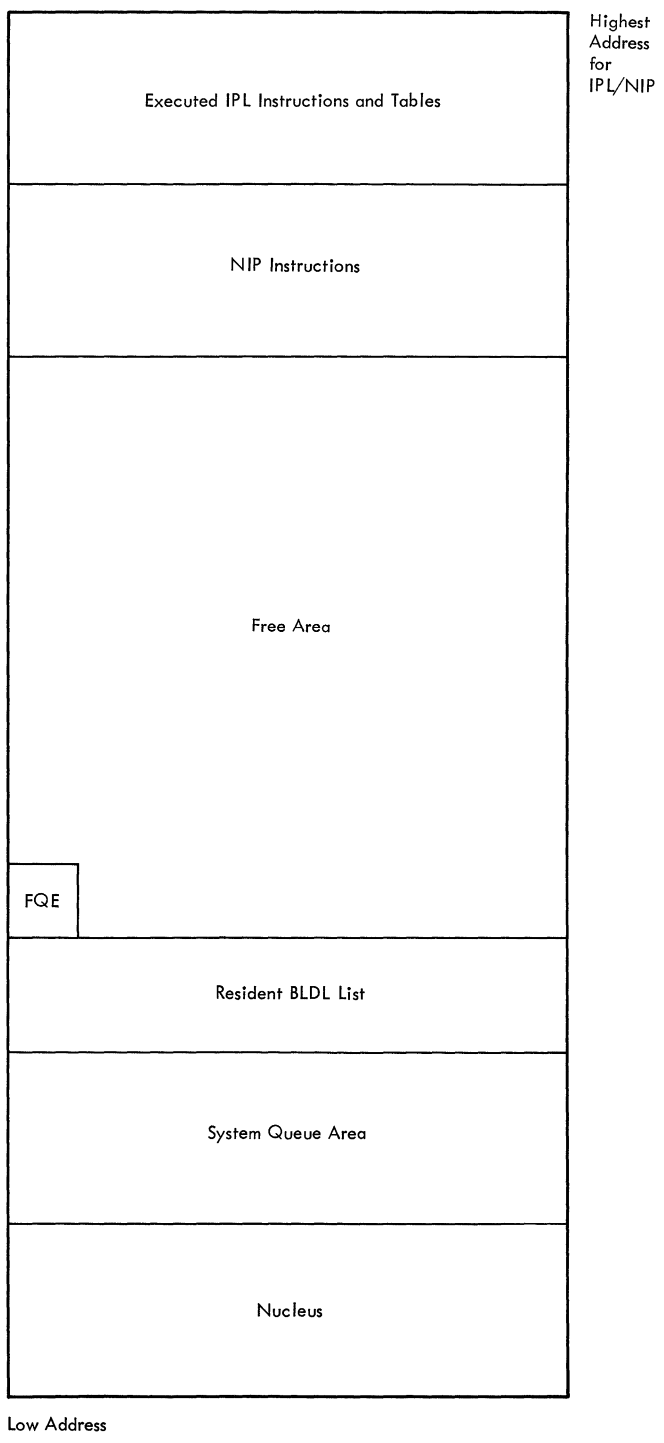


Figure 19. MFT Main Storage After Resident BLDL List construction

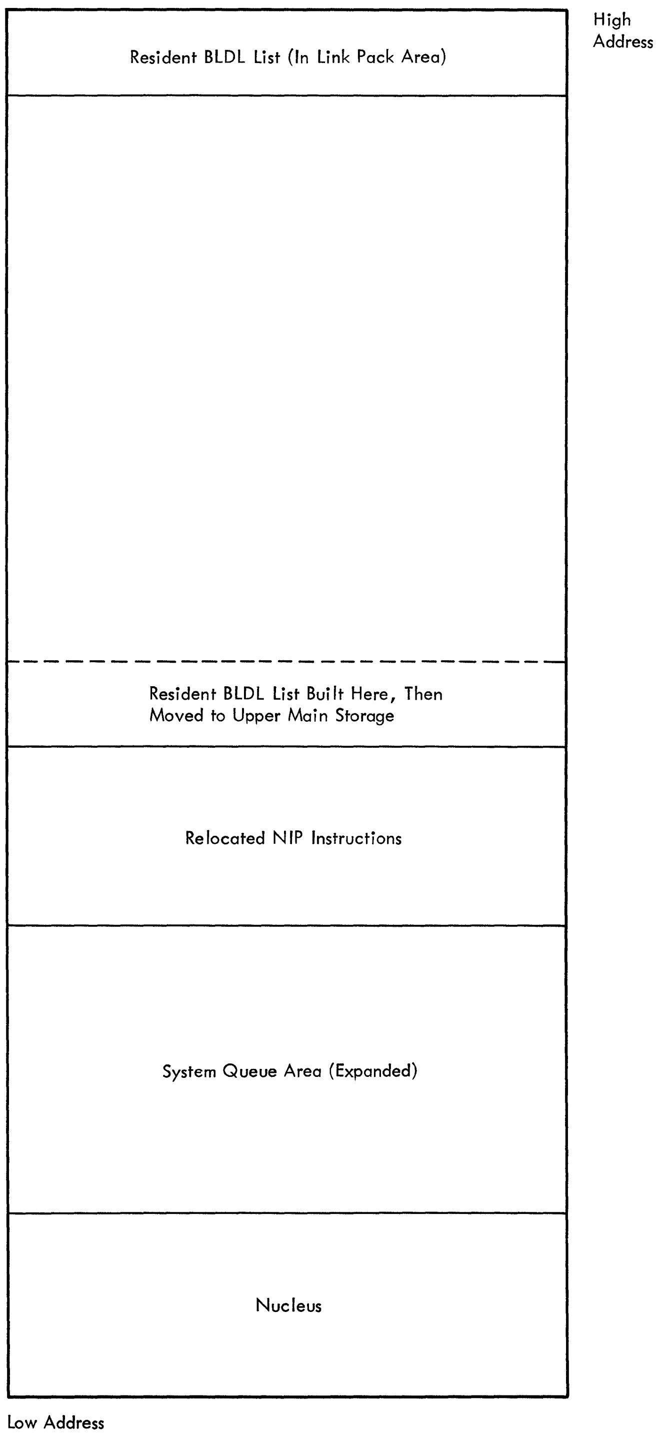

High Address
Resident BLDL List (In Link Pack Area)
Resident BLDL List Built Here, Then Moved to Upper Main Storage
Relocated NIP Instructions
System Queue Area (Expanded)
Nucleus
Low Address

Figure 20. MVT Main Storage After Resident BLDL Construction

Optional Error Recovery Procedure Modules

Lists indicated by the RERP= parameter values are used to load error recovery procedure (ERP) modules from the SVC library. (The default list in SYS1.PARMLIB member IEAIGE00). NIP uses the same procedure to locate and load the ERP module as for RAM and RSVC modules. However, NIP forms a separate chain of ERP-related CDEs, and saves a pointer to this chain at location IEAAERP. The pointer in the TCB is zeroed as for the RAM and RSVC modules. NIP restricts ERP loading to those modules which have names beginning with the four characters IGE0.

LOADING THE RESIDENT REENTERABLE MODULE AREA MODULES (OPTIONAL: MFT)

In a system with MFT which includes the optional resident reenterable module area, NIP loads into this area any SVC or linkage library modules specified by the user in the RAM= option. The user's module lists are contained in the SYS1.PARMLIB data set. If NIP is unable to locate a DSCB for the SYS1.PARMLIB data set, a message is issued to the operator to inform him that the optional routines are inoperative.

Optional Linkage Library Modules

NIP obtains the module lists specified by the RAM= parameter values. (The default module list is IEAIGG00.) NIP issues the BLDL macro instruction for each module. After NIP sets the attribute field of the found module BLDL list, it issues a LOAD macro instruction.

When loading of the modules is completed, NIP obtains a pointer to the chain of loaded program request blocks (LPRBs) for the loaded modules. NIP places the pointer in the communications vector table (CVT) and then zeros out the temporary chain pointer in the communication task TCB.

IGFMCHF0 PROCESSING -- STAGE 2

After the link pack area has been initialized, NIP again loads IGFMCHF0 for its stage 2 processing.

For System/360 Model 85 and System/370 machines (excluding the Model 195), IGFMCHF0 initializes pointers to the resident SVCLIB BLDL Table and LINKLIB BLDL Table in the MCH Independent common area. Then, if the IPLed model number is 135 or 145, IGFMCHF0 returns to NIP, which deletes the copy of IGFMCHF0 that was just loaded. For Model 85 and System/370 Models 155 and 165, IGFMCHF0 then computes checksums for all refreshable data in the control program nucleus and the link pack area. The computed checksums are written out on SYS1. ASRLIB, where they will be available if the need arises to refresh the related data. For an explanation of checksumming as a method of refreshing, see IBM System/360 Operating System: Machine-Check Handler for IBM System/360 Model 85, Program Logic Manual, GY27-7184, or Machine-Check Handler for IBM System/370 Models 155 and 165, GY27-7198.

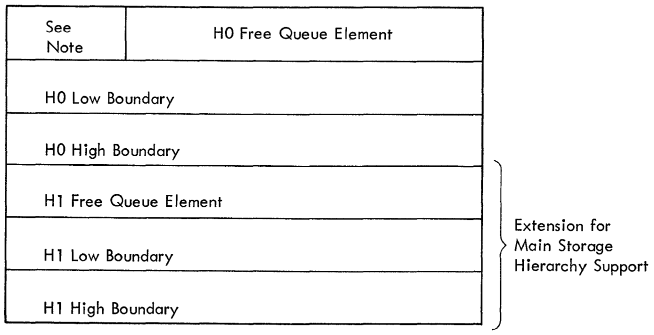
MCH Initialization for System/370 is then complete, and control is returned to NIP. NIP deletes the copy of IGFMCHF0 which was just loaded. One more step occurs for System/360 Model 85 MCH initialization. NIP checks to determine if the alternate multiply algorithm should be loaded. If the function is required, NIP loads the Alternate Multiply Control module and passes control to that module so that the necessary initialization functions can be performed. The Alternate Multiply Control module returns control to NIP, and NIP then deletes the module before continuing system initialization.

LOADING OPTIONAL RESIDENT ROUTINES (MFT)

Optional resident routines are loaded in the same manner in an MFT system as in an MVT system, except that the routines are loaded into different areas of storage. In an MFT system, the RAM, BLDL, RERP, and RSVC resident modules are loaded into the area of storage adjacent to the system queue area. The boundary boxes and end-of-nucleus pointers are updated to reflect the additional size of the control program area (nucleus).

PREPARING MAIN STORAGE WHEN THE SYSTEM CONTAINS MAIN STORAGE HIERARCHY SUPPORT FOR IBM 2361 CORE STORAGE (MFT)

If the computing system being initialized contains IBM 2361 Core Storage units and the control program has been generated with Main Storage Hierarchy Support, a boundary box extension is initialized to indicate the space available. The boundary box extension is created during system generation. NIP determines if 2361 Core Storage is present on the system and places the addresses of the upper and lower limits in the boundary box extension (see Figure 21). The FQE for the area is built and the FQE pointer in the boundary box is initialized. In a system with MFT, a six-word boundary box for each partition defined in system generation exists in the master scheduler resident data area. However, NIP initializes only the master scheduler boundary box; the initialization of partitions is the function of the master scheduler initialization routine. The boundary box for the master scheduler describes all



Note: Bit 7 of Byte 0 in the H0 Boundary Box is set to 1 to indicate the presence of the extension for Main Storage Hierarchy Support.

Figure 21. MFT Boundary Box and Extension for Main Storage Hierarchy Support

of processor (hierarchy 0) storage, 2361 Core Storage (hierarchy 1) is described as free area.

PREPARING MAIN STORAGE (MFT)

In a system with MFT, main storage which is to be used for problem programs is divided into two or more logically discrete areas called partitions. Creating these partitions is a function of the MFT master scheduler. NIP rounds the nucleus boundary and sets protection keys for the control program area and problem program (dynamic) area. Control is then passed to the master scheduler, which completes storage preparation. Passing control to the master scheduler is explained later under "NIP Termination."

PREPARING MAIN STORAGE (MVT)

After building the link pack area in a system with MVT, NIP determines the final main storage divisions required for control program operation. At this time, NIP divides free main storage into the master scheduler region of predetermined size and the dynamic area, which is all the unused storage. (See "Defining Control Program Areas" for control block use and storage designation.)

ESTABLISHING THE FINAL MASTER SCHEDULER REGION (MVT)

NIP sets the master scheduler region size to six 2K blocks. If MPS=nn was specified, NIP overrides the pre-set value with the size indicated by nn, provided that the requested amount of storage is available for allocation. NIP designates storage adjacent to the link pack area as the final master scheduler region (see Figure 22). NIP sets the master scheduler partition queue element (MSPQE1) in the system queue area to the address of the new FBQE constructed at the beginning of the master scheduler region. The FBQE points to MSPQE1 and gives the size of the master scheduler region.

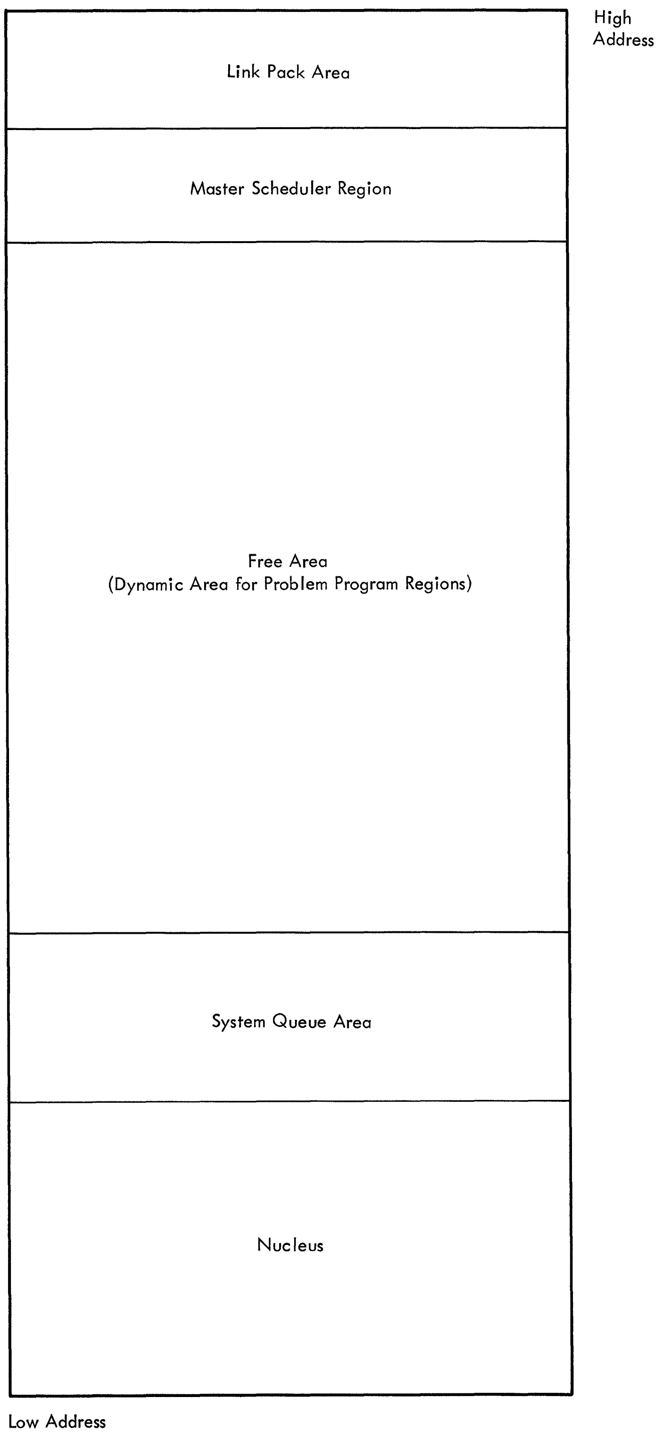


Figure 22. Final MVT Main Storage

NIP builds two SPQEs in subpool 255 of the system queue area to allow the master scheduler and communications tasks to share subpool zero. NIP issues a GETMAIN macro instruction to obtain 16 bytes of storage from subpool 255. The format of the two SPQEs is shown in Figure 23.

The master scheduler SPQE is flagged as owned, while the communications task SPQE is flagged as shared. Each SPQE is chained to the SPQE chain for the task with which it is associated. If either of the SPQEs for subpool zero is the only SPQE on the chain, it is also flagged as the last SPQE on the chain.

DEFINING THE DYNAMIC AREA (MVT)

NIP defines the storage space between the system queue area and the master scheduler region as the dynamic area for problem programs. This space includes the area occupied by NIP. (The NIP instructions are transparent to the control program when NIP passes control to it.) NIP constructs an FBQE at the start of dynamic area, thus overlaying some NIP instructions. The FBQE points to the dynamic area partition queue element (HOPQE) and contains the size of the dynamic area. HOPQE, which was previously unused, points to the FBQE. Figure 22 shows main storage after NIP processing.

VERIFYING HARD COPY REQUIREMENTS FOR MCS (OPTIONAL)

If the HARDCPY parameter was specified, either at system generation or as an operator response when entering system parameters, NIP ensures that the hard copy device is available.

Either a device address or the SYSLOG data set may be specified for the hard copy log. (SYSLOG is a valid specification only if LOG support was included in system generation.) If SYSLOG is specified, NIP attempts to locate the SYS1.SYSLOGa (where a is either X or Y) data set in the system catalog. NIP searches first for the SYS1. SYSLOGX data set; if that is not found, the search is repeated for the SYS1.SYSLOGY data set. If neither data set can be

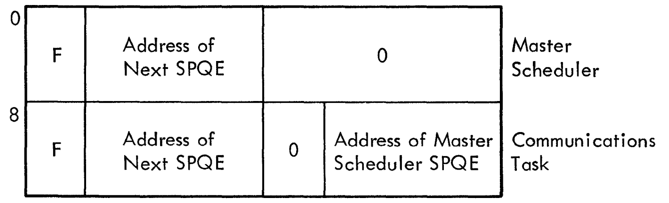


Figure 23. Shared Subpool 0 SPQEs (MVT)

found, NIP terminates the hard copy search after issuing the appropriate message. If the device on which SYSLOG is mounted was not on-line at IPL, the search is terminated. Normal NIP processing continues.

If the hard copy specification indicates a device address, NIP searches for the corresponding UCM entry to ensure that the address represents a valid console. If no UCM entry is found, the HARDCPY SPECIFICATION INVALID message is issued to the master console and the search is terminated. Normal NIP processing continues.

When an acceptable and available device is found, NIP sets the system flag in the MCS prefix to the UCM base, as appropriate, to indicate:

- that hard copy log is required.
- that hard copy is to be included in the SYSLOG.
- that commands are to be included in the system log.
- that the timer is operative.

If the hard copy specification was a device address, the address of the UCM entry for that device is placed in the hard copy pointer field of the MCS prefix to the UCM base. If the hard copy device is the SYSLOG device, zeros are placed in the hard copy pointer field.

ESTABLISHING HARD COPY OUTPUT CAPABILITY (OPTIONAL)

If the system includes MCS, a hard copy output device may be required. NIP tests the hard copy required flag in the MCS prefix to the UCM base. If hard copy output is required, NIP determines whether an acceptable device has been previously specified. If not, the SPECIFY HARDCPY message is issued to the master console. An acceptable operator response is required before processing can continue. NIP tests for the device (or SYSLOG data set) availability, and if the device was not on-line at IPL or for any other reason is not available, repeats the SPECIFY HARDCPY message. This procedure is repeated until an available device is obtained. If no hard copy device is available, re-IPL is required.

NIP TERMINATION (MFT)

In a system with MFT, NIP passes control to the master scheduler which performs initialization of the dynamic area. NIP sets the storage protection keys of the dynamic area to 0 and adjusts the master scheduler boundary box to describe a partition that includes all main storage above the end of the nucleus. (In a system which includes IBM 2361 Core Storage and Main Storage Hierarchy Support, the master scheduler partition includes all of hierarchy 0, and the boundary box for hierarchy 1 is initialized to indicate all of hierarchy 1 as free storage.) A pointer to the original contents of the master scheduler boundary box is passed to the master scheduler initialization routine when NIP relinquishes control.

NIP then constructs an RB at the low address of the temporary master scheduler partition and establishes XCTL code to the master scheduler initialization module IEESD569. The master scheduler task control block (TCB) is made dispatchable, and the NEW pointer in the NEW/OLD doubleword (used for task switching by the dispatcher) is set to zero. NIP then branches to the dispatcher. Figure 24 shows main storage at NIP termination in an MFT system.

NIP TERMINATION (MVT)

When NIP has completed storage initialization, the program passes control to the first module (IEEVIPL) of the master scheduler. NIP issues a LINK macro instruction specifying the module name.

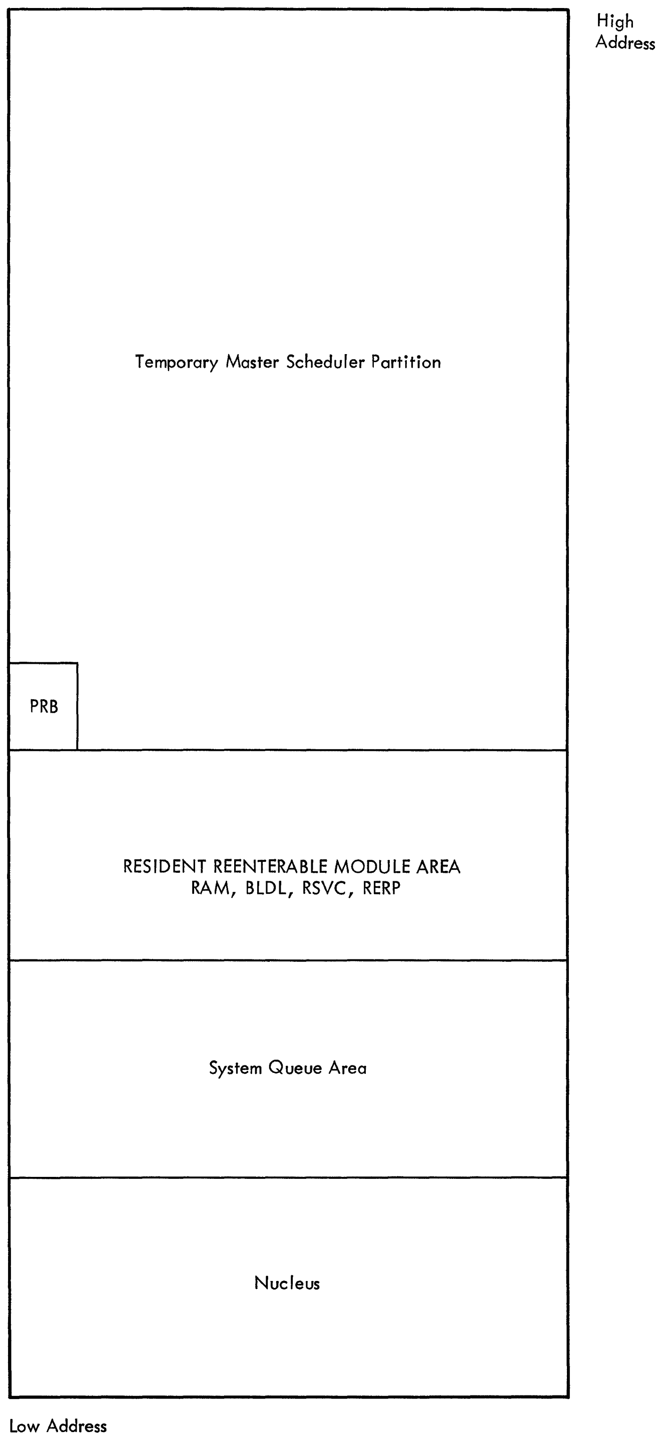


Figure 24. Final MFT Main Storage

This section contains routine lists for the Initial Program Loader and the Nucleus Initialization Program.

Routine names are given for convenience in relating routine descriptions to the location of the instructions in the program listing. Each routine name given for NIP is the address or the entry point name in the listing nearest to the start of the routine. The names given for the parts of the IPL program are actual routine names.

INITIAL PROGRAM LOADER PROGRAM CONTROL SECTION

Load Module Name: IEAIPL00
Control Section Name: IEAIPL
Routine Names:

IEAADDR	IPL Relocation: Moves unexecuted part of IPL to upper end of main storage to make room for the nucleus at low end.
IEACOMLP	Nucleus Location: Locates the nucleus on the primary system residence device.
IEACOMPR	Nucleus Selection: Selects the nucleus to be loaded.
IEAHOOP	Control Section Data Organization: Computes and arranges loading data before the nucleus is loaded.
IEALOAD	Nucleus Load: Loads the nucleus into main storage.
IEAMAIN	Hardware Initialization: Clears main storage and machine registers, and sets storage protection keys.
IEARELOC	RLD Relocation: Calculates absolute values for address constants in the nucleus text.
IEASTRIO	IPL Common I/O Subroutine: Issues START I/O instructions and tests for successful completion of I/O operations.

NUCLEUS INITIALIZATION PROGRAM CONTROL SECTION

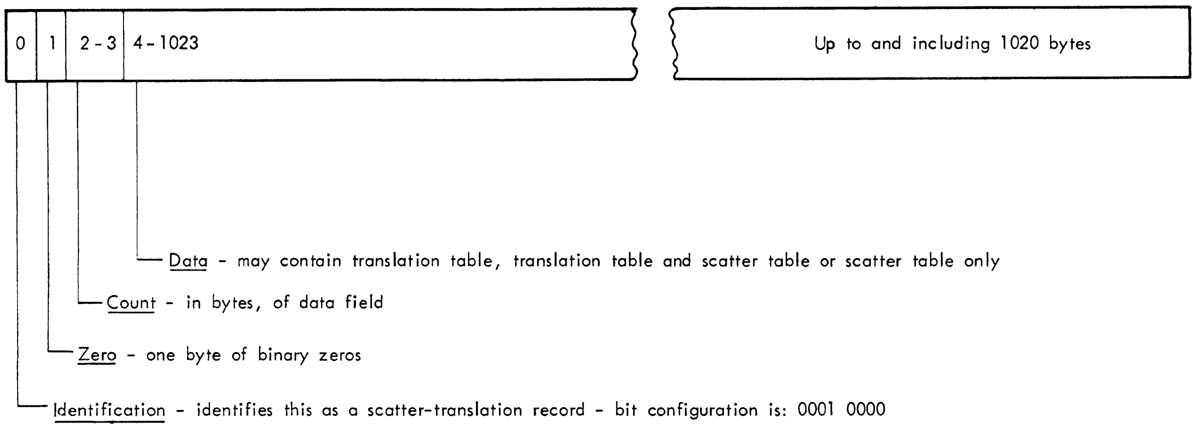
Load Module Names: IEANUC01 through IEANUC0Z

Control Section Name: IEANIP0
Routine Names:

IEACOMON	Data Extent Block Initialization Common Subroutine: Inserts data set boundary data into data extent blocks.
IEACONS1	Console Initialization: Saves the unit address of the operator's console for the use of NIP and the control program.
IEACOREX	Partition Adjustment: Decreases the size of the master scheduler region by the size of each module loaded into the link pack area.
IEAENDRM	Link Pack Area Load: Loads modules into the link pack area in conjunction with the IEARMLDR routine.
IEAGETXT	List Finder: Finds lists for link pack area loading and calculates a pointer to the address of the first module.
IEAFNDTX	Module Locator: Obtains the relative track address (TTR) of a module from the partitioned data set (PDS) directory record for the module.
CHKIPLDV	Data Extent Block Initialization: Identifies the unit control block for the system residence device and initializes data extent blocks for the supervisor call library and the system log.
IEALISTK	Names of lists of required modules.
IEALOCAT	Volume Serial Number Subroutine: Issues a LOCATE (SVC 26) macro instruction to find the serial number of a volume containing a particular data set.
IEAMOUNT	Mount Message Subroutine: Issues requests to the operator to mount a desired volume.
IEAMS1ER	MS/1 Pointer Routine: Sets pointers for main storage initialization procedures.
IEANIP4	Address Conversion: Sets the base register and converts control block addresses to absolute values in main storage.

IEANOUCB	System Residence Error: Indicates to the operator that no unit control block corresponds to the system residence device. The nucleus must be reloaded under this condition.
IEAOPMSG	Operator Communication: Provides operator communication.
IEARDVOL	Volume Contents Reader: Reads the volume table of contents (VTOC) of a volume mounted on a specified device.
IEARMLDR	Link Pack Area Load: Loads modules into the link pack area in conjunction with the IEAENDRM routine.
IEASCAN1	Linkage Library Initialization: Finds the Linkage library and initializes its data extent block.
IEASETK	Storage Protection Key Initialization: Sets storage protection keys of all 2048-byte blocks of main storage.
IEASTRIO	NIP Common I/O Subroutine: Issues START I/O instructions and tests for successful I/O completion.
IEATIMER	Timer Initialization: Checks and sets the system timer.
IEAUCBFN	Unit Control Block Finder: Finds the address of a unit control block corresponding to a given unit address.
IEAUCB0	Unit Control Block Initialization: Initializes direct access device unit control blocks and the direct access device table.
SENSESS	Unit Check Subroutine: Handles unit check interruptions to clear possible contingent connection conditions from a 2841 Control Unit.
SVXINIT	SVC Initialization: Finds the length and relative track address of each externally stored supervisor call routine and places the value into the supervisor call table.
(None)	Main Storage First Initialization: Sets initial values for the system queue area and master scheduler region.
(None)	Main Storage Second Initialization: Sets the final value for the system queue area, sets an intermediate value for the master scheduler region, and relocates the unexecuted portion of NIP.
(None)	Main Storage Final Initialization: Calculates and assigns final values for main storage areas.
(None)	System Environment Recorder Initialization: Gives to the portion of the SER program in main storage the address of the externally stored portion.

SCATTER/TRANSLATION RECORD



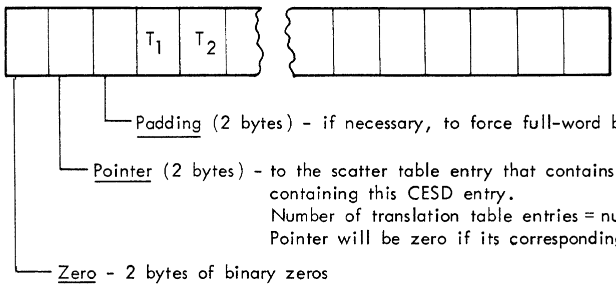
Data - may contain translation table, translation table and scatter table or scatter table only

Count - in bytes, of data field

Zero - one byte of binary zeros

Identification - identifies this as a scatter-translation record - bit configuration is: 0001 0000

Translation Table



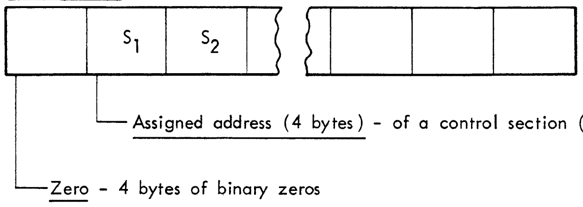
Padding (2 bytes) - if necessary, to force full-word boundary alignment of scatter table.

Pointer (2 bytes) - to the scatter table entry that contains the address of the control section containing this CESD entry.
Number of translation table entries = number of CESD entries + 1.
Pointer will be zero if its corresponding CESD entry is not SD, PC, CM or LR.

Zero - 2 bytes of binary zeros

NOTE: (One 2-byte entry for each external symbol)

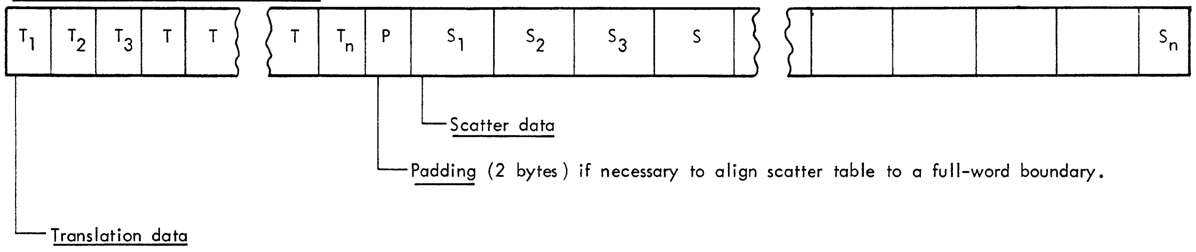
Scatter Table



Assigned address (4 bytes) - of a control section (SD, PC or CM) (one entry for each CSECT)

Zero - 4 bytes of binary zeros

Translation Table and Scatter Table



Scatter data

Padding (2 bytes) if necessary to align scatter table to a full-word boundary.

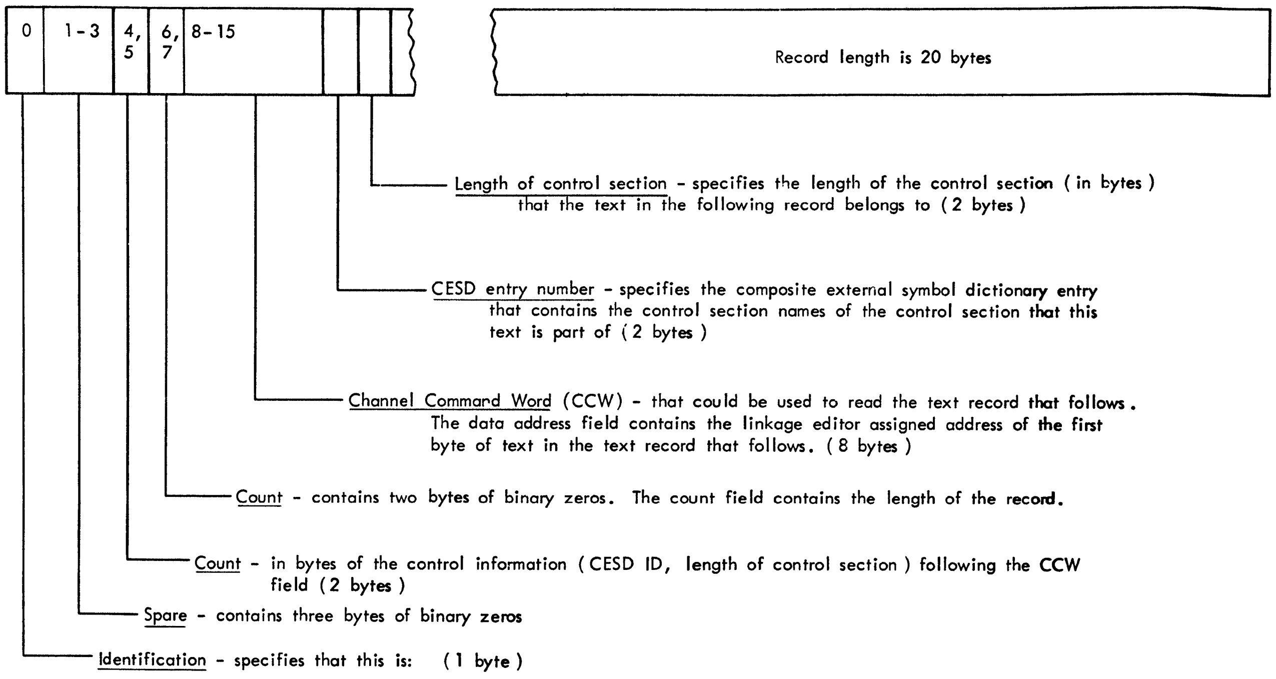
Translation data

NOTE: Translation table follows extent list in main storage.
Translation table entries are two bytes in length, scatter table entries four bytes in length.

Legend for Types of Entries in Composite External Symbol Dictionary (CESD)

SD = section definition
LR = label reference
PC = private code
CM = common

CONTROL RECORD



Length of control section - specifies the length of the control section (in bytes) that the text in the following record belongs to (2 bytes)

CESD entry number - specifies the composite external symbol dictionary entry that contains the control section names of the control section that this text is part of (2 bytes)

Channel Command Word (CCW) - that could be used to read the text record that follows. The data address field contains the linkage editor assigned address of the first byte of text in the text record that follows. (8 bytes)

Count - contains two bytes of binary zeros. The count field contains the length of the record.

Count - in bytes of the control information (CESD ID, length of control section) following the CCW field (2 bytes)

Spare - contains three bytes of binary zeros

Identification - specifies that this is: (1 byte)

- A control record - 0000 0001
- The control record that precedes the last text record of this overlay segment - 0000 0101
- The control record that precedes the last text record of the module - 0000 1101

RELOCATION DICTIONARY (RLD) RECORD

0	1 - 3	4, 5	6, 7	8-15	16-255

Record length can be between 24 and 256 bytes

RLD data -- see below

Spare - contains 8 bytes of binary zeros

Count - in bytes of the relocation dictionary information following the spare 8 byte field (2 bytes)

Count - contains two bytes of binary zeros

Spare - contains three bytes of binary zeros

Identification - specifies that this is: (1 byte)
A relocation dictionary record - 0000 0010
The last record of the segment - 0000 0110
The last record of the module - 0000 1110

RLD Data

R	P	F	A	F	A

F	A	R	P	F	A	R	P	F	A

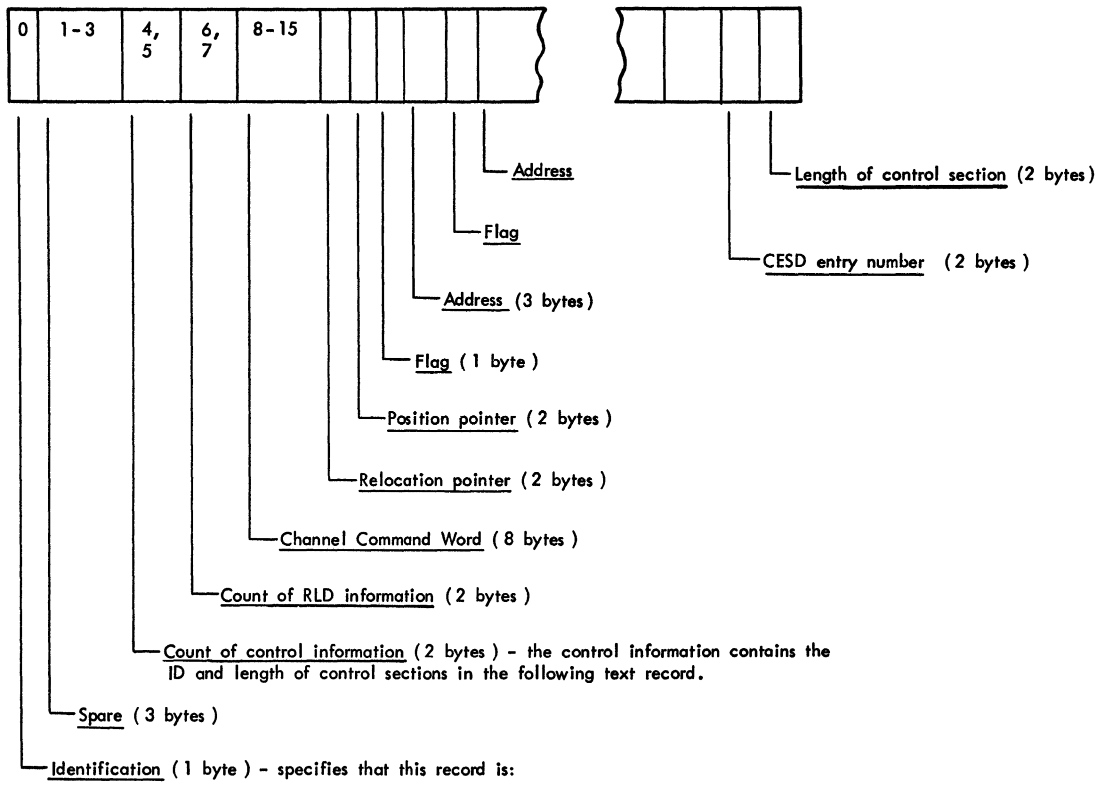
Address - Linkage editor assigned address of the address constant (3 bytes)

Flag - specifies miscellaneous information as follows: (1 byte) when byte format is xxxxLLST:
xxxx specifies the type of this RLD item (address constant)
0000 -- non-branch type in assembler language, a DC A (name)
0001 -- branch type (in assembler language, a DC V (name)
0010 -- pseudo register displacement value
0011 -- pseudo register cumulative displacement value
1000 and 1001 -- this address constant is not to be relocated, because it refers to an unresolved symbol.
LL specifies the length of the address constant
01 -- two byte
10 -- three byte
11 -- four byte
S specifies the direction of relocation
0 -- positive
1 -- negative
T specifies the type of RLD item following this one
0 -- the following RLD item has a different relocation and/or position pointer
1 -- the following RLD item has the same relocation and position pointers as this one, and therefore is omitted

Position pointer - contains the entry number of the CESD entry (or translation table entry) that indicates which control section the address constant is in (2 bytes)

Relocation pointer - contains the entry number of the CESD entry (or translation table entry) that indicates which symbol's value is to be used in the computation of the address constant's value (2 bytes)

CONTROL AND RELOCATION DICTIONARY RECORD


0
1-3
4, 5
6, 7
8-15
Address
Length of control section (2 bytes)
Flag
CESD entry number (2 bytes)
Address (3 bytes)
Flag (1 byte)
Position pointer (2 bytes)
Relocation pointer (2 bytes)
Channel Command Word (8 bytes)
Count of RLD information (2 bytes)
Count of control information (2 bytes) - the control information contains the ID and length of control sections in the following text record.
Spare (3 bytes)
Identification (1 byte) - specifies that this record is:

- A control and RLD record - 0000 0011
- A control and RLD record that is followed by the last text record of a segment - 0000 0111
- A control and RLD record that is followed by the last text record of a module - 0000 1111

Note: For detailed descriptions of the data fields see:

Relocation Dictionary Record
Control Record

The record length will vary from 20 to 260 bytes.

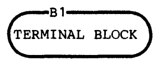
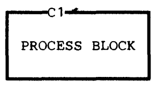
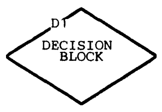
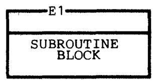
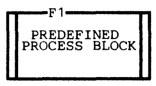
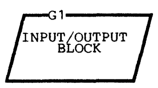
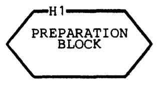
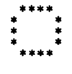
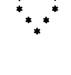
SCATTER EXTENT LIST

Bytes		
0	EXLLNTH (Total size of extent list)	
4	Number of relocation factors	
8	Length of first non-contiguous block	
12	Length of second non-contiguous block	
16	Length of third non-contiguous block	
	1 byte	3 bytes
	Hex. 80*	Length of last non-contiguous block
	0	Address of first non-contiguous block
	0	Address of second non-contiguous block
	0	Address of third non-contiguous block
	• • • •	• • • •
	0	Address of last non-contiguous block

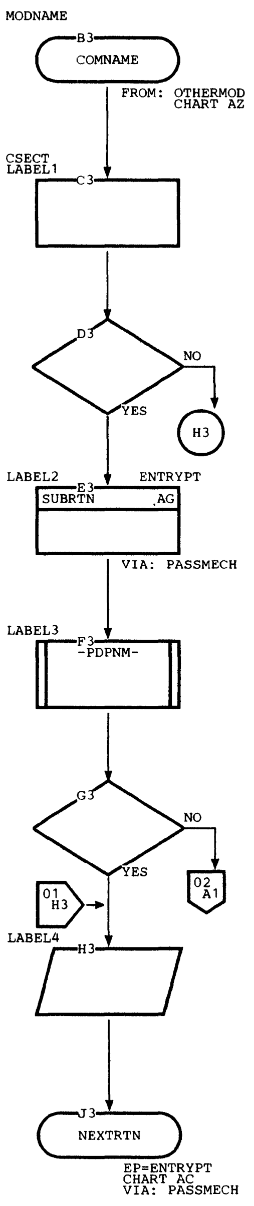
* Indicates the end of the immediately preceding length-of-block list. Used by the GETMAIN routine.

FLOWCHARTS

The flowcharts in this manual have been produced by an IBM program, using ANSI symbols. The symbols are defined in the left column below, and examples of their use are shown at the right.

SYMBOL	DEFINITION
	INDICATES AN ENTRY OR TERMINAL POINT IN A FLOW-CHART; SHOWS START, STOP, HALT, DELAY, OR INTERRUP-TION. MAY ALSO INDICATE RETURN TO THE CALLING PROGRAM.
	INDICATES A PROCESSING FUNCTION OR A DEFINED OP-ERATION CAUSING CHANGE IN VALUE, FORM OR LOCATION OF INFORMATION.
	INDICATES A DECISION OR SWITCHING-TYPE OPERATION THAT DETERMINES WHICH OF A NUMBER OF ALTERNATE PATHS SHOULD BE FOLLOWED.
	INDICATES A SUBROUTINE OR MODULE THAT IS DESCRIBED IN THIS MANUAL
	INDICATES A SUBROUTINE OR MODULE THAT IS INCLUDED IN THE FLOWCHARTS OF AN-OTHER MANUAL.
	INDICATES GENERAL I/O FUNCTIONS, SUCH AS GET, PUT, READ, WRITE, SIO, AND DEVICE-CONTROL MACRO INSTRUCTIONS.
	INDICATES A PROCESS THAT CHANGES SYSTEM OPERATION, FOR EXAMPLE, SETS A SWITCH, MODIFIES AN INDEX REGISTER, OR INITIALIZES A ROUTINE.
ONPAGE CONNECTOR 	INDICATES ENTRY TO OR EXIT FROM ANOTHER BLOCK ON THE SAME FLOWCHART PAGE.
OFFPAGE CONNECTOR 	INDICATES ENTRY TO OR EXIT FROM A BLOCK ON ANOTHER PAGE OF THE SAME SET OF FLOWCHARTS.

EXAMPLE



COMMENTS

B3: MODNAME IS THE LOAD MODULE OR LIBRARY NAME OF THE ROUTINE DESCRIBED BY THIS FLOWCHART.

COMNAME IS THE COMMON NAME OF THE ROUTINE.

OTHERMOD INDICATES THE MODULES PASSING CONTROL TO THIS MODULE AND THEIR FLOW-CHARTS.

C3: CSECT IS THE CSECT NAME OR OTHER ENTRY POINT AT WHICH PROCESSING BEGINS.

LABEL1 IS THE LABEL OF THE FIRST INSTRUCTION.

D3: PROGRAM EXECUTION CONTINUES WITH BLOCK H3 WHEN THE DECISION IS NO, OR BLOCK E3 WHEN THE DECISON IS YES.

E3: LABEL2 IS THE LABEL OF THE SECTION OF CODE IN THIS ROUTINE FROM WHICH CONTROL IS PASSED TO THE SUBROUTINE. CONTROL RETURNS TO THE NEXT INSTRUCTION FOLLOW-ING THE SUBROUTINE CALL.

ENTRYPT IS THE ENTRY POINT.

SUBRTN IS THE COMMON NAME OF THE SUB-ROUTINE IN FLOWCHART AG.

VIA: PASSMECH INDICATES HOW CONTROL PASSES FROM COMNAME TO SUBRTN.

F3: LABEL3 IS THE LABEL OF THE SECTION OF CODE FROM WHICH CONTROL IS PASSED TO THE PREDEFINED PROCESS PDPNM, WHICH IS DOCUMENTED IN ANOTHER PUBLICATION (-PDPNM- MAY ALSO BE USED IN A PROCESS-ING BLOCK).

G3: EXECUTION CONTINUES WITH BLOCK H3 WHEN THE DECISION IS YES, OR WITH BLOCK A1 ON PAGE 2 OF THIS SET OF FLOWCHARTS WHEN THE DECISION IS NO.

THE OFFPAGE CONNECTOR MARKED 01H3 INDI-CATES THAT EXECUTION CONTINUES WITH BLOCK H3 FROM ANOTHER PAGE OF THIS SET OF FLOW-CHARTS. THIS CONNECTOR IS ALSO PAIRED WITH THE ONPAGE CONNECTOR FROM BLOCK D3.

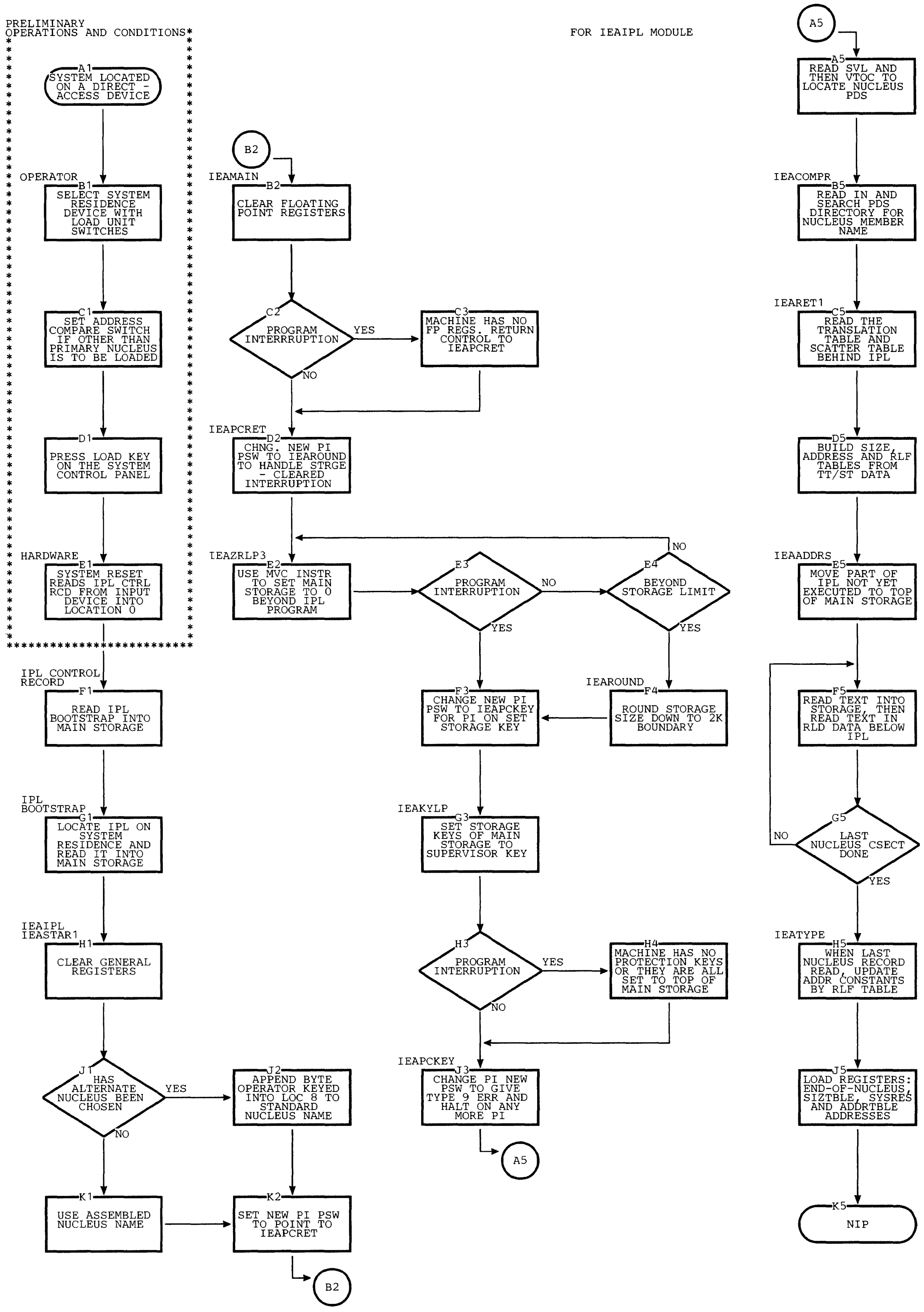
H3: LABEL4 IS THE LABEL OF A SECTION OF CODE OF THIS ROUTINE THAT INITIATES I/O.

J3: NEXTRTN IS THE COMMON NAME OF THE ROUT-INE THAT EXECUTES AFTER THIS ROUTINE.

ENTRYPT IS THE ENTRY POINT OF NEXTRTN, WHICH IS DESCRIBED IN CHART AC.

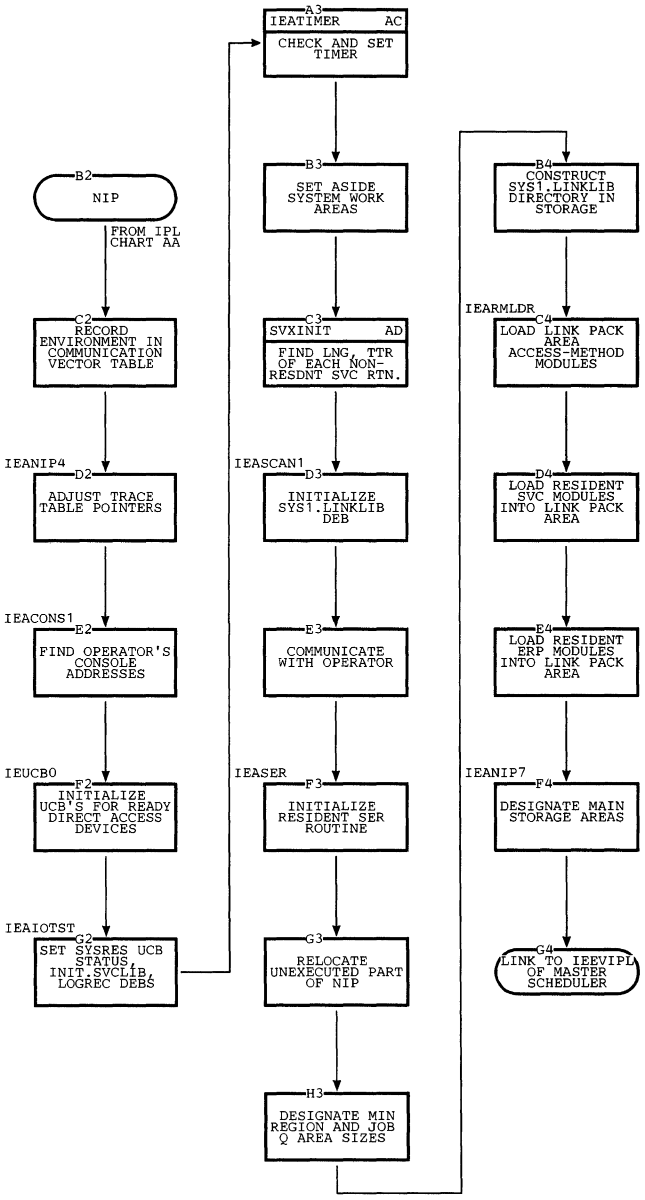
VIA: PASSMECH INDICATES HOW CONTROL PASSES FROM COMNAME TO NEXTRTN.

Chart AA. Initial Program Loader Control Flow



PRELIMINARY OPERATIONS AND CONDITIONS
FOR IEAIPL MODULE
A1 SYSTEM LOCATED ON A DIRECT - ACCESS DEVICE
OPERATOR
B1 SELECT SYSTEM RESIDENCE DEVICE WITH LOAD UNIT SWITCHES
C1 SET ADDRESS COMPARE SWITCH IF OTHER THAN PRIMARY NUCLEUS IS TO BE LOADED
D1 PRESS LOAD KEY ON THE SYSTEM CONTROL PANEL
HARDWARE
E1 SYSTEM RESET READS IPL CTRL RCD FROM INPUT DEVICE INTO LOCATION 0
IPL CONTROL RECORD
F1 READ IPL BOOTSTRAP INTO MAIN STORAGE
IPL BOOTSTRAP
G1 LOCATE IPL ON SYSTEM RESIDENCE AND READ IT INTO MAIN STORAGE
IEAIPL IEASTAR1
H1 CLEAR GENERAL REGISTERS
J1 HAS ALTERNATE NUCLEUS BEEN CHOSEN
YES
NO
K1 USE ASSEMBLED NUCLEUS NAME
B2
IEAMAIN
B2 CLEAR FLOATING POINT REGISTERS
C2 PROGRAM INTERRRUPTION
YES
NO
C3 MACHINE HAS NO FP REGS. RETURN CONTROL TO IEAPCRET
IEAPCRET
D2 CHNG. NEW PI PSW TO IEAROUND TO HANDLE STRGE - CLEARED INTERRUPTION
IEAZRLP3
E2 USE MVC INSTR TO SET MAIN STORAGE TO 0 BEYOND IPL PROGRAM
E3 PROGRAM INTERRUPTION
NO
YES
E4 BEYOND STORAGE LIMIT
NO
YES
IEAROUND
F4 ROUND STORAGE SIZE DOWN TO 2K BOUNDARY
F3 CHANGE NEW PI PSW TO IEAPCKEY FOR PI ON SET STORAGE KEY
IEAKYLP
G3 SET STORAGE KEYS OF MAIN STORAGE TO SUPERVISOR KEY
H3 PROGRAM INTERRUPTION
YES
NO
H4 MACHINE HAS NO PROTECTION KEYS OR THEY ARE ALL SET TO TOP OF MAIN STORAGE
IEAPCKEY
J3 CHANGE PI NEW PSW TO GIVE TYPE 9 ERR AND HALT ON ANY MORE PI
A5
J2 APPEND BYTE OPERATOR KEYED INTO LOC 8 TO STANDARD NUCLEUS NAME
K2 SET NEW PI PSW TO POINT TO IEAPCRET
B2
A5
A5 READ SVL AND THEN VTOC TO LOCATE NUCLEUS PDS
IEACOMPR
B5 READ IN AND SEARCH PDS DIRECTORY FOR NUCLEUS MEMBER NAME
IEARET1
C5 READ THE TRANSLATION TABLE AND SCATTER TABLE BEHIND IPL
D5 BUILD SIZE, ADDRESS AND RLF TABLES FROM TT/ST DATA
IEAADDRS
E5 MOVE PART OF IPL NOT YET EXECUTED TO TOP OF MAIN STORAGE
F5 READ TEXT INTO STORAGE, THEN READ TEXT IN RLD DATA BELOW IPL
G5 LAST NUCLEUS CSECT DONE
NO
YES
IEATYPE
H5 WHEN LAST NUCLEUS RECORD READ, UPDATE ADDR CONSTANTS BY RLF TABLE
J5 LOAD REGISTERS: END-OF-NUCLEUS, SIZTBLE, SYSRES AND ADDRTBLE ADDRESSES
K5 NIP

Chart AB. Nucleus Initialization Program Control Flow



B2 NIP
FROM IPL CHART AA
C2 RECORD ENVIRONMENT IN COMMUNICATION VECTOR TABLE
IEANIP4 D2 ADJUST TRACE TABLE POINTERS
IEACONS1 E2 FIND OPERATOR'S CONSOLE ADDRESSES
IEUCBO F2 INITIALIZE UCB'S FOR READY DIRECT ACCESS DEVICES
IEAIOTST G2 SET SYSRES UCB STATUS, INIT.SVCLIB, LOGREC DEBS
A3 IEATIMER AC CHECK AND SET TIMER
B3 SET ASIDE SYSTEM WORK AREAS
C3 SVXINIT AD FIND LNG, TTR OF EACH NON-RESDNT SVC RTN.
IEASCAN1 D3 INITIALIZE SYS1.LINKLIB DEB
E3 COMMUNICATE WITH OPERATOR
IEASER F3 INITIALIZE RESIDENT SER ROUTINE
G3 RELOCATE UNEXECUTED PART OF NIP
H3 DESIGNATE MIN REGION AND JOB Q AREA SIZES
B4 CONSTRUCT SYS1.LINKLIB DIRECTORY IN STORAGE
IEARMLDR C4 LOAD LINK PACK AREA ACCESS-METHOD MODULES
D4 LOAD RESIDENT SVC MODULES INTO LINK PACK AREA
E4 LOAD RESIDENT ERP MODULES INTO LINK PACK AREA
IEANIP7 F4 DESIGNATE MAIN STORAGE AREAS
G4 LINK TO IEEVIPL OF MASTER SCHEDULER

Chart AC. Timer Initialization

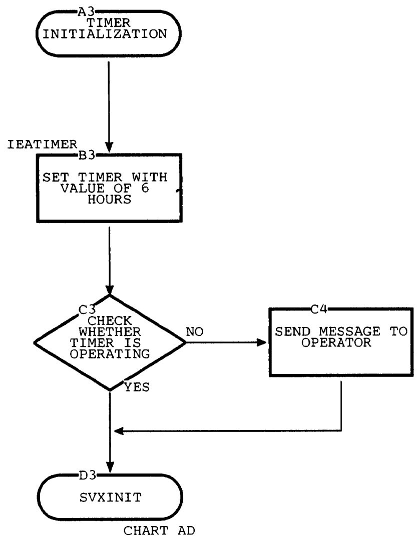


Chart AD. Supervisor Call (SVC) Table Initialization

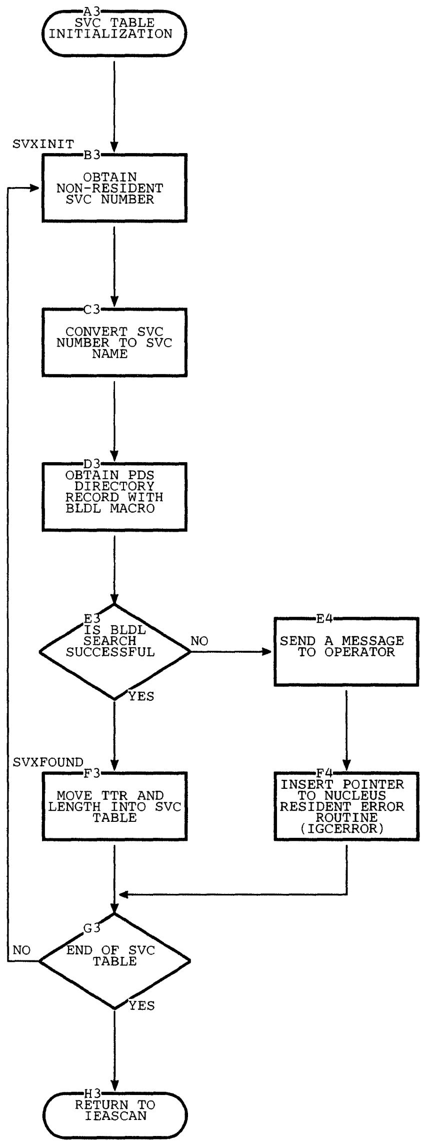
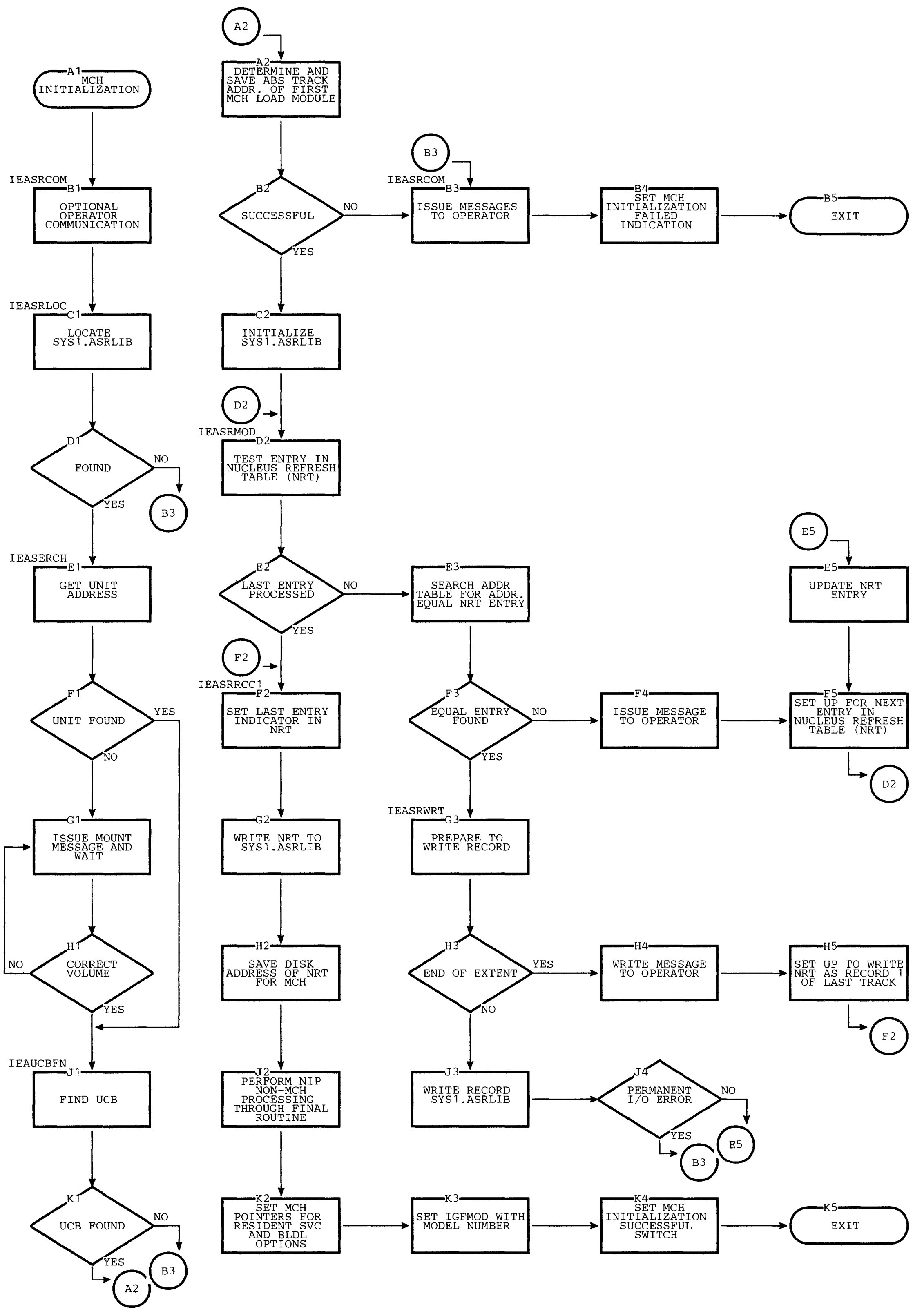

A3
SVC TABLE INITIALIZATION
SVXINIT
B3
OBTAIN NON-RESIDENT SVC NUMBER
C3
CONVERT SVC NUMBER TO SVC NAME
D3
OBTAIN PDS DIRECTORY RECORD WITH BLDL MACRO
E3
IS BLDL SEARCH SUCCESSFUL
NO
YES
E4
SEND A MESSAGE TO OPERATOR
SVXFOUND
F3
MOVE TTR AND LENGTH INTO SVC TABLE
F4
INSERT POINTER TO NUCLEUS RESIDENT ERROR ROUTINE (IGCERROR)
G3
END OF SVC TABLE
NO
YES
H3
RETURN TO IEASCAN

Chart AE. Machine-Check Handler Initialization (Model 65)



A1 MCH INITIALIZATION
IEASRCOM B1 OPTIONAL OPERATOR COMMUNICATION
IEASRLOC C1 LOCATE SYS1.ASRLIB
D1 FOUND
NO
B3
YES
IEASERCH E1 GET UNIT ADDRESS
F1 UNIT FOUND
YES
NO
G1 ISSUE MOUNT MESSAGE AND WAIT
H1 CORRECT VOLUME
NO
YES
IEAUCBFN J1 FIND UCB
K1 UCB FOUND
NO
B3
YES
A2
A2
A2 DETERMINE AND SAVE ABS TRACK ADDR. OF FIRST MCH LOAD MODULE
B2 SUCCESSFUL
NO
YES
C2 INITIALIZE SYS1.ASRLIB
D2
IEASRMOD D2 TEST ENTRY IN NUCLEUS REFRESH TABLE (NRT)
E2 LAST ENTRY PROCESSED
NO
YES
F2
IEASRRCC1 F2 SET LAST ENTRY INDICATOR IN NRT
G2 WRITE NRT TO SYS1.ASRLIB
H2 SAVE DISK ADDRESS OF NRT FOR MCH
J2 PERFORM NIP NON-MCH PROCESSING THROUGH FINAL ROUTINE
K2 SET MCH POINTERS FOR RESIDENT SVC AND BLDL OPTIONS
B3
IEASRCOM B3 ISSUE MESSAGES TO OPERATOR
E3 SEARCH ADDR TABLE FOR ADDR. EQUAL NRT ENTRY
F3 EQUAL ENTRY FOUND
NO
YES
IEASRWRT G3 PREPARE TO WRITE RECORD
H3 END OF EXTENT
YES
NO
J3 WRITE RECORD SYS1.ASRLIB
K3 SET IGFMOD WITH MODEL NUMBER
B4 SET MCH INITIALIZATION FAILED INDICATION
F4 ISSUE MESSAGE TO OPERATOR
H4 WRITE MESSAGE TO OPERATOR
J4 PERMANENT I/O ERROR
NO
E5
YES
B3
K4 SET MCH INITIALIZATION SUCCESSFUL SWITCH
B5 EXIT
E5
E5 UPDATE NRT ENTRY
F5 SET UP FOR NEXT ENTRY IN NUCLEUS REFRESH TABLE (NRT)
D2
H5 SET UP TO WRITE NRT AS RECORD 1 OF LAST TRACK
F2
K5 EXIT

Chart AF. CCH Initialization Routine

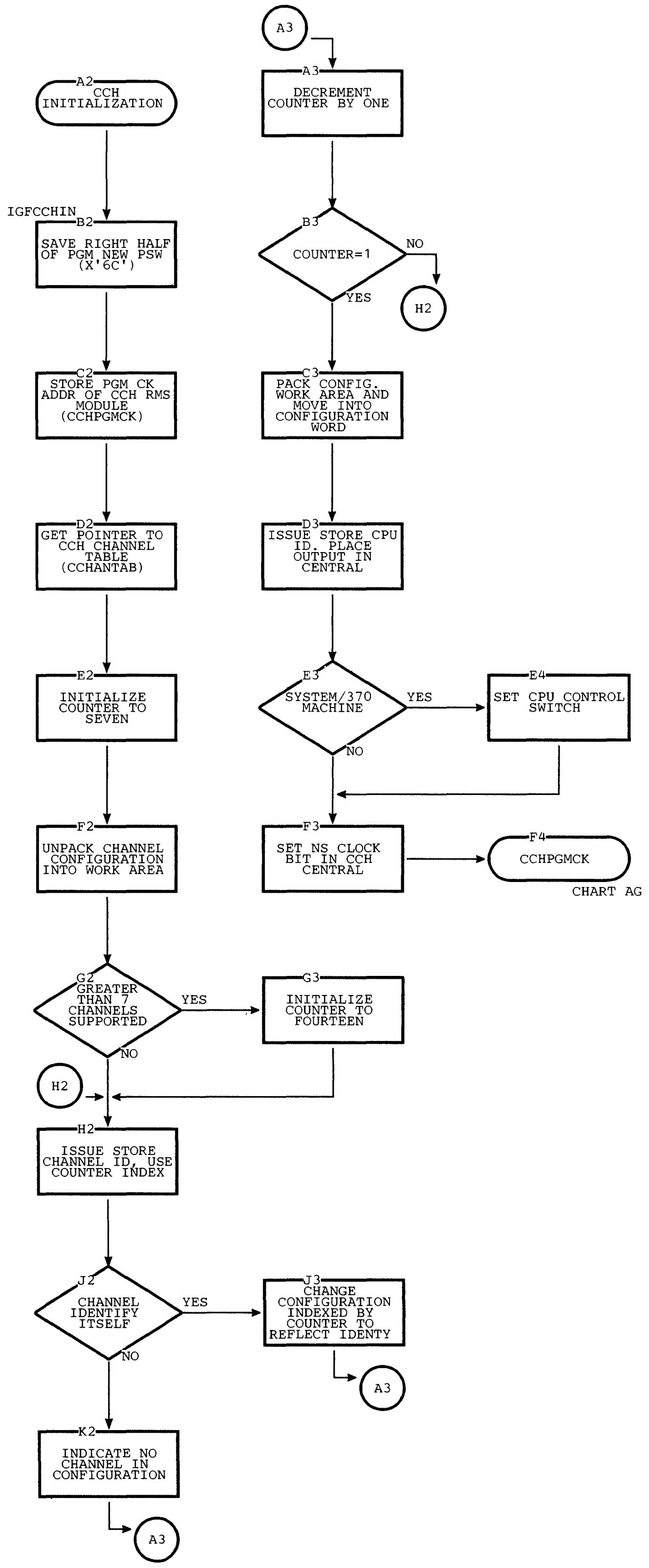

A2 CCH INITIALIZATION
IGFCCHIN
B2 SAVE RIGHT HALF OF PGM NEW PSW (X'6C')
C2 STORE PGM CK ADDR OF CCH RMS MODULE (CCHPGMCK)
D2 GET POINTER TO CCH CHANNEL TABLE (CCHANTAB)
E2 INITIALIZE COUNTER TO SEVEN
F2 UNPACK CHANNEL CONFIGURATION INTO WORK AREA
G2 GREATER THAN 7 CHANNELS SUPPORTED
YES
NO
G3 INITIALIZE COUNTER TO FOURTEEN
H2
H2 ISSUE STORE CHANNEL ID, USE COUNTER INDEX
J2 CHANNEL IDENTIFY ITSELF
YES
NO
J3 CHANGE CONFIGURATION INDEXED BY COUNTER TO REFLECT IDENTY
A3
K2 INDICATE NO CHANNEL IN CONFIGURATION
A3
A3
A3 DECREMENT COUNTER BY ONE
B3 COUNTER=1
NO
H2
YES
C3 PACK CONFIG. WORK AREA AND MOVE INTO CONFIGURATION WORD
D3 ISSUE STORE CPU ID. PLACE OUTPUT IN CENTRAL
E3 SYSTEM/370 MACHINE
YES
E4 SET CPU CONTROL SWITCH
NO
F3 SET NS CLOCK BIT IN CCH CENTRAL
F4 CCHPGMCK
CHART AG

Chart AG. CCHPGMCK

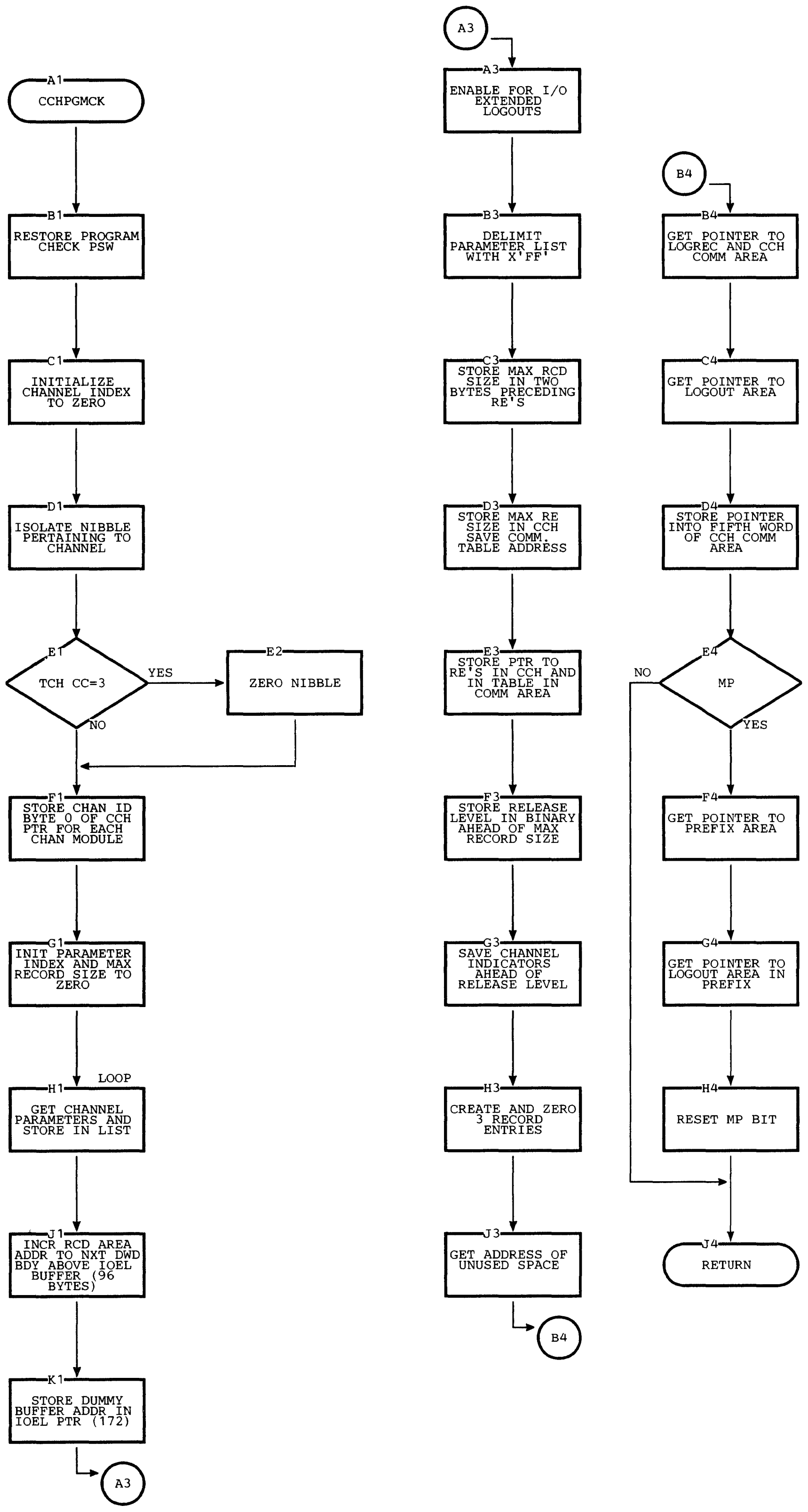

A1
CCHPGMCK
B1
RESTORE PROGRAM CHECK PSW
C1
INITIALIZE CHANNEL INDEX TO ZERO
D1
ISOLATE NIBBLE PERTAINING TO CHANNEL
E1
TCH CC=3
YES
NO
E2
ZERO NIBBLE
F1
STORE CHAN ID BYTE 0 OF CCH PTR FOR EACH CHAN MODULE
G1
INIT PARAMETER INDEX AND MAX RECORD SIZE TO ZERO
LOOP
H1
GET CHANNEL PARAMETERS AND STORE IN LIST
J1
INCR RCD AREA ADDR TO NXT DWD BDY ABOVE IOEL BUFFER (96 BYTES)
K1
STORE DUMMY BUFFER ADDR IN IOEL PTR (172)
A3
A3
A3
ENABLE FOR I/O EXTENDED LOGOUTS
B3
DELIMIT PARAMETER LIST WITH X'FF'
C3
STORE MAX RCD SIZE IN TWO BYTES PRECEDING RE'S
D3
STORE MAX RE SIZE IN CCH SAVE COMM. TABLE ADDRESS
E3
STORE PTR TO RE'S IN CCH AND IN TABLE IN COMM AREA
F3
STORE RELEASE LEVEL IN BINARY AHEAD OF MAX RECORD SIZE
G3
SAVE CHANNEL INDICATORS AHEAD OF RELEASE LEVEL
H3
CREATE AND ZERO 3 RECORD ENTRIES
J3
GET ADDRESS OF UNUSED SPACE
B4
B4
B4
GET POINTER TO LOGREC AND CCH COMM AREA
C4
GET POINTER TO LOGOUT AREA
D4
STORE POINTER INTO FIFTH WORD OF CCH COMM AREA
E4
MP
NO
YES
F4
GET POINTER TO PREFIX AREA
G4
GET POINTER TO LOGOUT AREA IN PREFIX
H4
RESET MP BIT
J4
RETURN

Chart AH. CCH Move Module

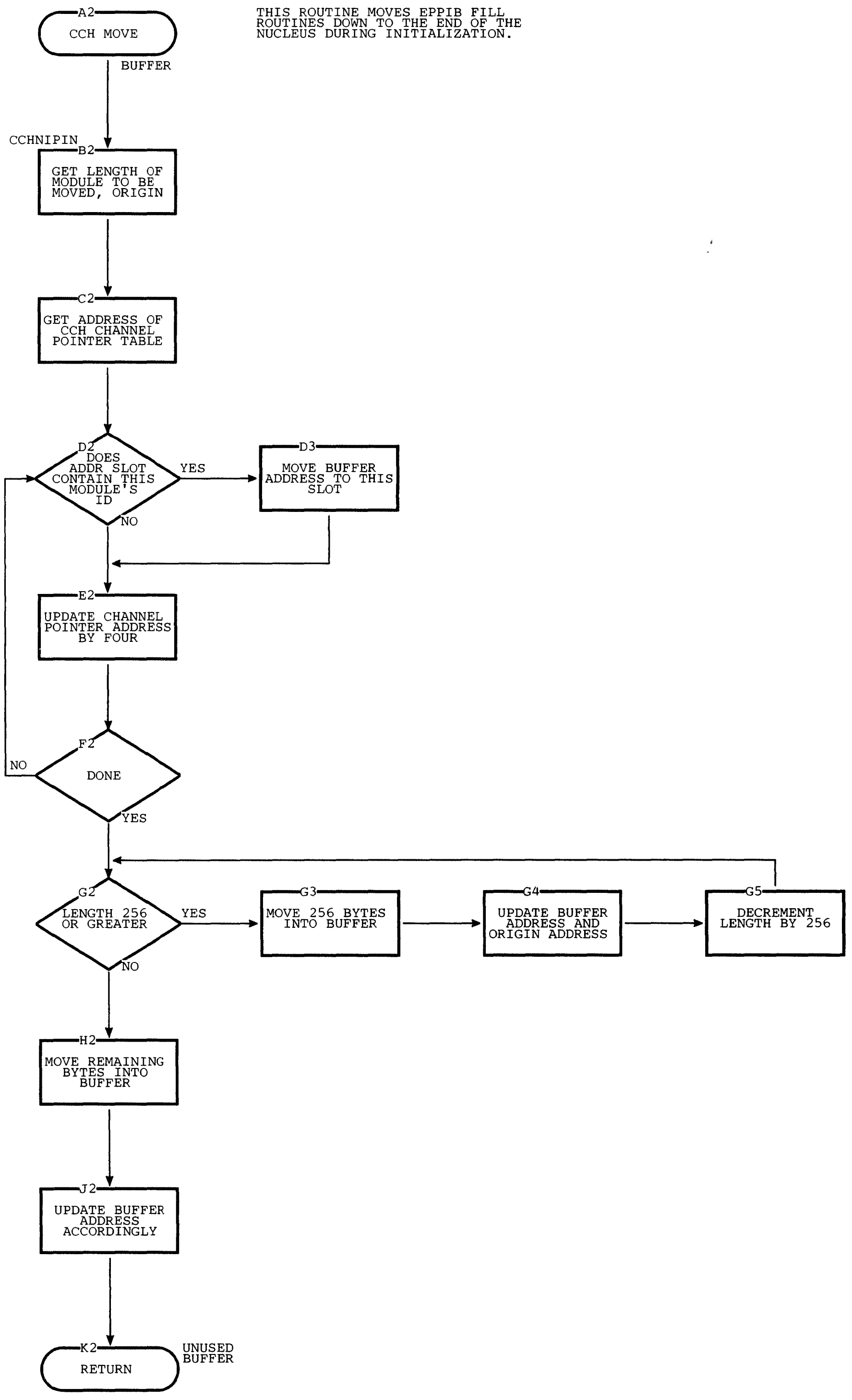

A2
CCH MOVE
BUFFER
THIS ROUTINE MOVES EPPIB FILL ROUTINES DOWN TO THE END OF THE NUCLEUS DURING INITIALIZATION.
CCHNIPIN
B2
GET LENGTH OF MODULE TO BE MOVED, ORIGIN
C2
GET ADDRESS OF CCH CHANNEL POINTER TABLE
D2
DOES ADDR SLOT CONTAIN THIS MODULE'S ID
YES
NO
D3
MOVE BUFFER ADDRESS TO THIS SLOT
E2
UPDATE CHANNEL POINTER ADDRESS BY FOUR
F2
DONE
NO
YES
G2
LENGTH 256 OR GREATER
YES
NO
G3
MOVE 256 BYTES INTO BUFFER
G4
UPDATE BUFFER ADDRESS AND ORIGIN ADDRESS
G5
DECREMENT LENGTH BY 256
H2
MOVE REMAINING BYTES INTO BUFFER
J2
UPDATE BUFFER ADDRESS ACCORDINGLY
K2
RETURN
UNUSED BUFFER

Chart AI. Initialization for Rollout/Rollin

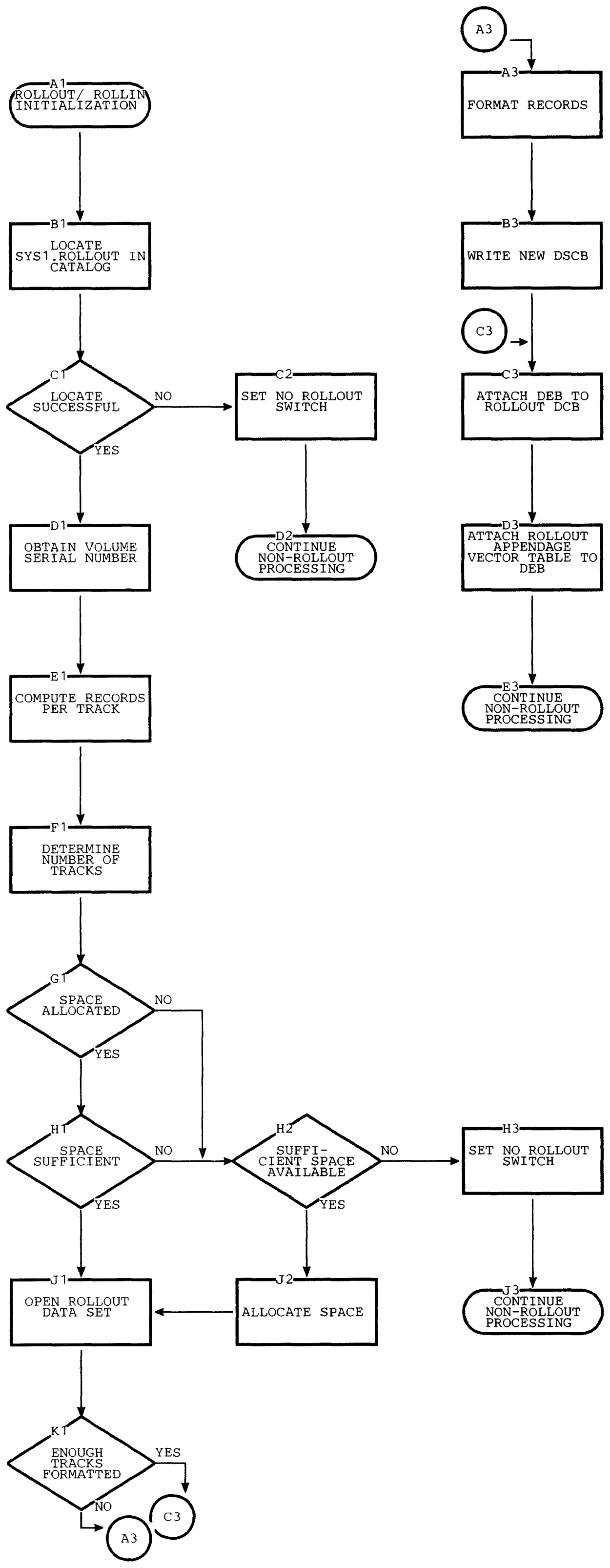
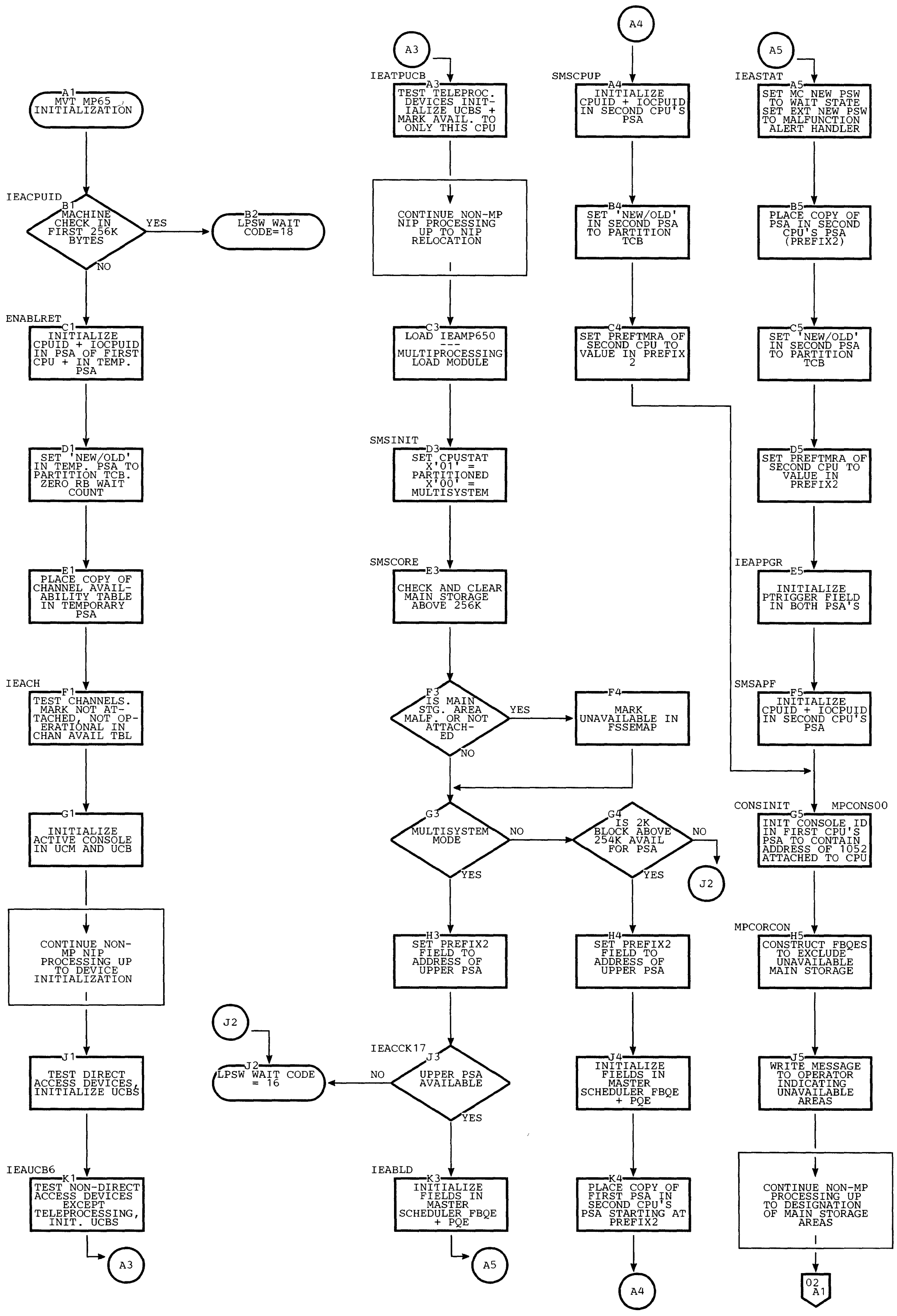
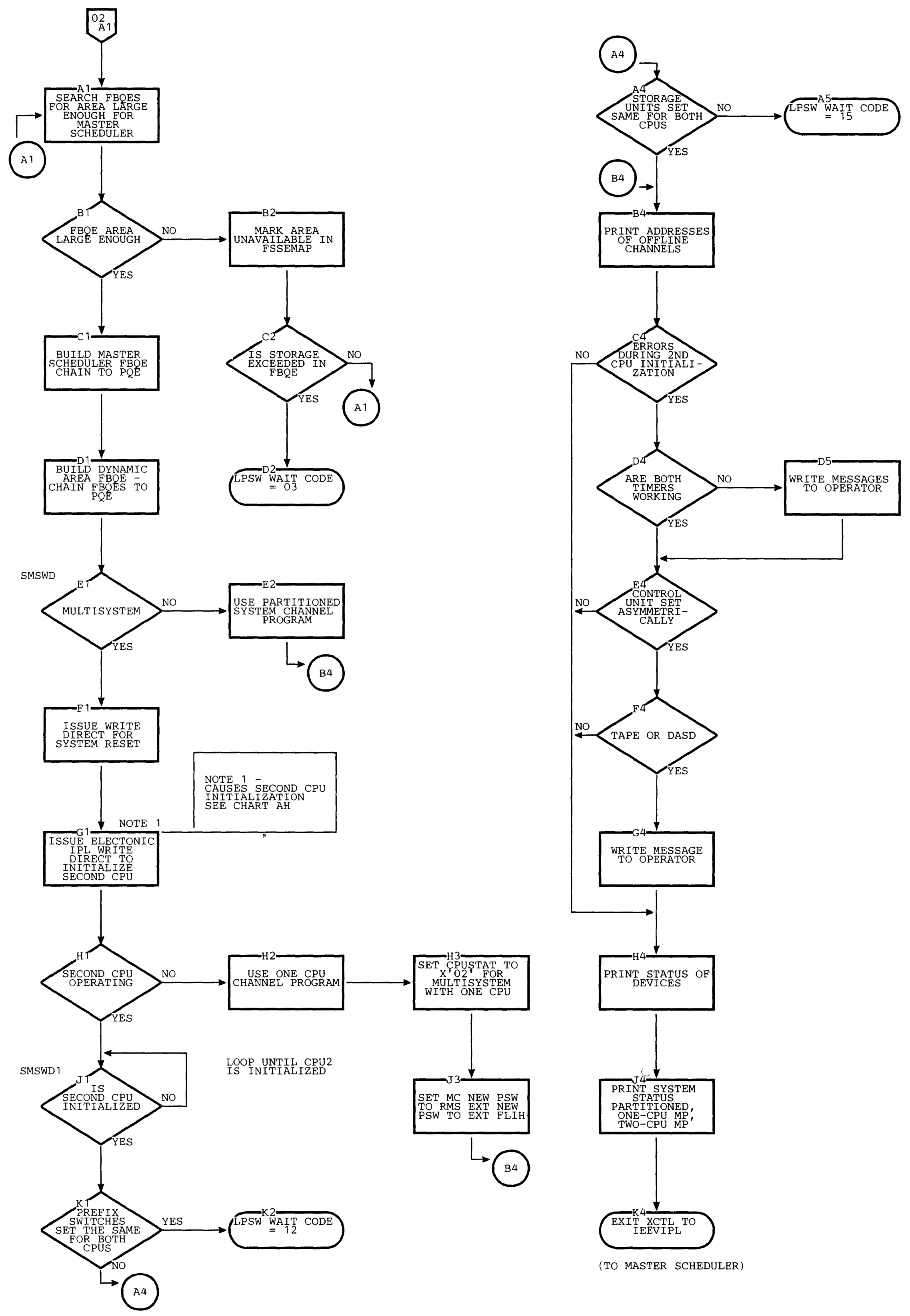

A1 ROLLOUT/ ROLLIN INITIALIZATION
B1 LOCATE SYS1.ROLLOUT IN CATALOG
C1 LOCATE SUCCESSFUL
NO
YES
C2 SET NO ROLLOUT SWITCH
D2 CONTINUE NON-ROLLOUT PROCESSING
D1 OBTAIN VOLUME SERIAL NUMBER
E1 COMPUTE RECORDS PER TRACK
F1 DETERMINE NUMBER OF TRACKS
G1 SPACE ALLOCATED
NO
YES
H1 SPACE SUFFICIENT
NO
YES
H2 SUFFI-CIENT SPACE AVAILABLE
NO
YES
H3 SET NO ROLLOUT SWITCH
J3 CONTINUE NON-ROLLOUT PROCESSING
J2 ALLOCATE SPACE
J1 OPEN ROLLOUT DATA SET
K1 ENOUGH TRACKS FORMATTED
YES
NO
C3
A3
A3
A3 FORMAT RECORDS
B3 WRITE NEW DSCB
C3
C3 ATTACH DEB TO ROLLOUT DCB
D3 ATTACH ROLLOUT APPENDAGE VECTOR TABLE TO DEB
E3 CONTINUE NON-ROLLOUT PROCESSING

Chart AJ. MVT with Model 65 Multiprocessing Initialization (Chart 1 of 2)



A1 MVT MP65 INITIALIZATION
IEACPUID B1 MACHINE CHECK IN FIRST 256K BYTES
YES
NO
B2 LPSW WAIT CODE=18
ENABLRET C1 INITIALIZE CPUID + IOCPUID IN PSA OF FIRST CPU + IN TEMP. PSA
D1 SET 'NEW/OLD' IN TEMP. PSA TO PARTITION TCB. ZERO RB WAIT COUNT
E1 PLACE COPY OF CHANNEL AVAILABILITY TABLE IN TEMPORARY PSA
IEACH F1 TEST CHANNELS. MARK NOT ATTACHED, NOT OPERATIONAL IN CHAN AVAIL TBL
G1 INITIALIZE ACTIVE CONSOLE IN UCM AND UCB
CONTINUE NON-MP NIP PROCESSING UP TO DEVICE INITIALIZATION
J1 TEST DIRECT ACCESS DEVICES, INITIALIZE UCBS
IEAUCB6 K1 TEST NON-DIRECT ACCESS DEVICES EXCEPT TELEPROCESSING, INIT. UCBS
A3
A3
IEATPUCB A3 TEST TELEPROC. DEVICES INITIALIZE UCBS + MARK AVAIL. TO ONLY THIS CPU
CONTINUE NON-MP NIP PROCESSING UP TO NIP RELOCATION
C3 LOAD IEAMP650 MULTIPROCESSING LOAD MODULE
SMSINIT D3 SET CPUSTAT X'01' = PARTITIONED X'00' = MULTISYSTEM
SMSCORE E3 CHECK AND CLEAR MAIN STORAGE ABOVE 256K
F3 IS MAIN STG. AREA MALF. OR NOT ATTACHED
YES
NO
F4 MARK UNAVAILABLE IN FSSEMAP
G3 MULTISYSTEM MODE
NO
YES
G4 IS 2K BLOCK ABOVE 254K AVAIL FOR PSA
NO
YES
J2
H3 SET PREFIX2 FIELD TO ADDRESS OF UPPER PSA
H4 SET PREFIX2 FIELD TO ADDRESS OF UPPER PSA
J2
J2 LPSW WAIT CODE = 16
IEACCK17 J3 UPPER PSA AVAILABLE
NO
YES
IEABLD K3 INITIALIZE FIELDS IN MASTER SCHEDULER FBQE + PQE
A5
A4
SMSCPUP A4 INITIALIZE CPUID + IOCPUID IN SECOND CPU'S PSA
B4 SET 'NEW/OLD' IN SECOND PSA TO PARTITION TCB
C4 SET PREFTMRA OF SECOND CPU TO VALUE IN PREFIX 2
J4 INITIALIZE FIELDS IN MASTER SCHEDULER FBQE + PQE
K4 PLACE COPY OF FIRST PSA IN SECOND CPU'S PSA STARTING AT PREFIX2
A4
A5
IEASTAT A5 SET MC NEW PSW TO WAIT STATE SET EXT NEW PSW TO MALFUNCTION ALERT HANDLER
B5 PLACE COPY OF PSA IN SECOND CPU'S PSA (PREFIX2)
C5 SET 'NEW/OLD' IN SECOND PSA TO PARTITION TCB
D5 SET PREFTMRA OF SECOND CPU TO VALUE IN PREFIX2
IEAPPGR E5 INITIALIZE PTRIGGER FIELD IN BOTH PSA'S
SMSAPF F5 INITIALIZE CPUID + IOCPUID IN SECOND CPU'S PSA
CONSINIT G5 MPCONS00 INIT CONSOLE ID IN FIRST CPU'S PSA TO CONTAIN ADDRESS OF 1052 ATTACHED TO CPU
MPCORCON H5 CONSTRUCT FBQES TO EXCLUDE UNAVAILABLE MAIN STORAGE
J5 WRITE MESSAGE TO OPERATOR INDICATING UNAVAILABLE AREAS
CONTINUE NON-MP PROCESSING UP TO DESIGNATION OF MAIN STORAGE AREAS
02 A1

Chart AJ. MVT with Model 65 Multiprocessing Initialization (Chart 2 of 2)



02 A1
A1
A1 SEARCH FBQES FOR AREA LARGE ENOUGH FOR MASTER SCHEDULER
B1 FBQE AREA LARGE ENOUGH
NO
YES
B2 MARK AREA UNAVAILABLE IN FSSEMAP
C2 IS STORAGE EXCEEDED IN FBQE
NO
A1
YES
D2 LPSW WAIT CODE = 03
C1 BUILD MASTER SCHEDULER FBQE CHAIN TO PQE
D1 BUILD DYNAMIC AREA FBQE - CHAIN FBQES TO PQE
SMSWD
E1 MULTISYSTEM
NO
E2 USE PARTITIONED SYSTEM CHANNEL PROGRAM
B4
YES
F1 ISSUE WRITE DIRECT FOR SYSTEM RESET
NOTE 1 - CAUSES SECOND CPU INITIALIZATION SEE CHART AH
NOTE 1
G1 ISSUE ELECTONIC IPL WRITE DIRECT TO INITIALIZE SECOND CPU
H1 SECOND CPU OPERATING
NO
H2 USE ONE CPU CHANNEL PROGRAM
H3 SET CPUSTAT TO X'02' FOR MULTISYSTEM WITH ONE CPU
J3 SET MC NEW PSW TO RMS EXT NEW PSW TO EXT FLIH
B4
YES
SMSWD1
J1 IS SECOND CPU INITIALIZED
NO
LOOP UNTIL CPU2 IS INITIALIZED
YES
K1 PREFIX SWITCHES SET THE SAME FOR BOTH CPUS
YES
K2 LPSW WAIT CODE = 12
NO
A4
A4
A4 STORAGE UNITS SET SAME FOR BOTH CPUS
NO
A5 LPSW WAIT CODE = 15
YES
B4
B4 PRINT ADDRESSES OF OFFLINE CHANNELS
C4 ERRORS DURING 2ND CPU INITIALI-ZATION
NO
YES
D4 ARE BOTH TIMERS WORKING
NO
D5 WRITE MESSAGES TO OPERATOR
YES
E4 CONTROL UNIT SET ASYMMETRI-CALLY
NO
YES
F4 TAPE OR DASD
NO
YES
G4 WRITE MESSAGE TO OPERATOR
H4 PRINT STATUS OF DEVICES
J4 PRINT SYSTEM STATUS PARTITIONED, ONE-CPU MP, TWO-CPU MP
K4 EXIT XCTL TO IEEVIPL
(TO MASTER SCHEDULER)

Chart AK. MVT with Model 65 Multiprocessing Second CPU Initialization

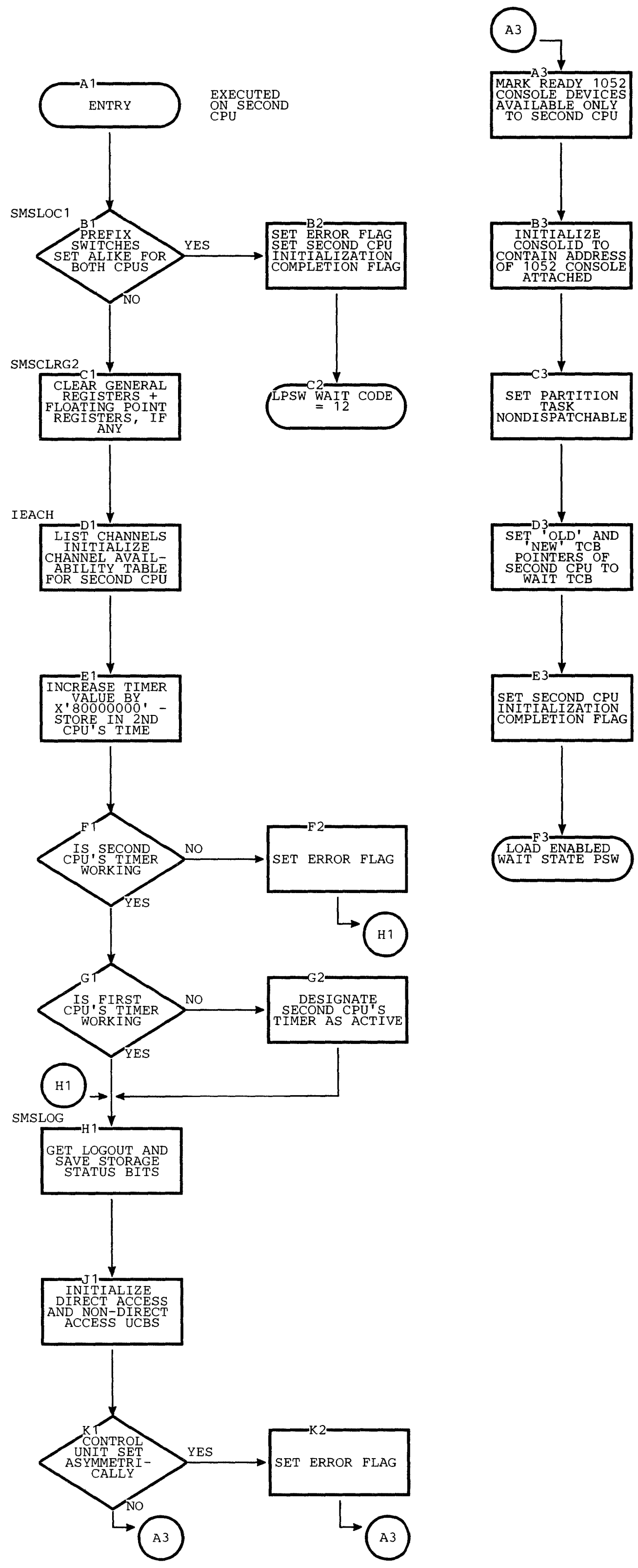

A1 ENTRY
EXECUTED ON SECOND CPU
SMSLOC1
B1 PREFIX SWITCHES SET ALIKE FOR BOTH CPUS
YES
NO
B2 SET ERROR FLAG SET SECOND CPU INITIALIZATION COMPLETION FLAG
C2 LPSW WAIT CODE = 12
SMSCLRG2
C1 CLEAR GENERAL REGISTERS + FLOATING POINT REGISTERS, IF ANY
IEACH
D1 LIST CHANNELS INITIALIZE CHANNEL AVAIL-ABILITY TABLE FOR SECOND CPU
E1 INCREASE TIMER VALUE BY X'80000000' - STORE IN 2ND CPU'S TIME
F1 IS SECOND CPU'S TIMER WORKING
NO
YES
F2 SET ERROR FLAG
H1
G1 IS FIRST CPU'S TIMER WORKING
NO
YES
G2 DESIGNATE SECOND CPU'S TIMER AS ACTIVE
H1
SMSLOG
H1 GET LOGOUT AND SAVE STORAGE STATUS BITS
J1 INITIALIZE DIRECT ACCESS AND NON-DIRECT ACCESS UCBS
K1 CONTROL UNIT SET ASYMMETRI-CALLY
YES
NO
K2 SET ERROR FLAG
A3
A3
A3
A3 MARK READY 1052 CONSOLE DEVICES AVAILABLE ONLY TO SECOND CPU
B3 INITIALIZE CONSOLID TO CONTAIN ADDRESS OF 1052 CONSOLE ATTACHED
C3 SET PARTITION TASK NONDISPATCHABLE
D3 SET 'OLD' AND 'NEW' TCB POINTERS OF SECOND CPU TO WAIT TCB
E3 SET SECOND CPU INITIALIZATION COMPLETION FLAG
F3 LOAD ENABLED WAIT STATE PSW

Chart AL. Initialization for SYS1.DUMP Data Set (Chart 1 of 3)

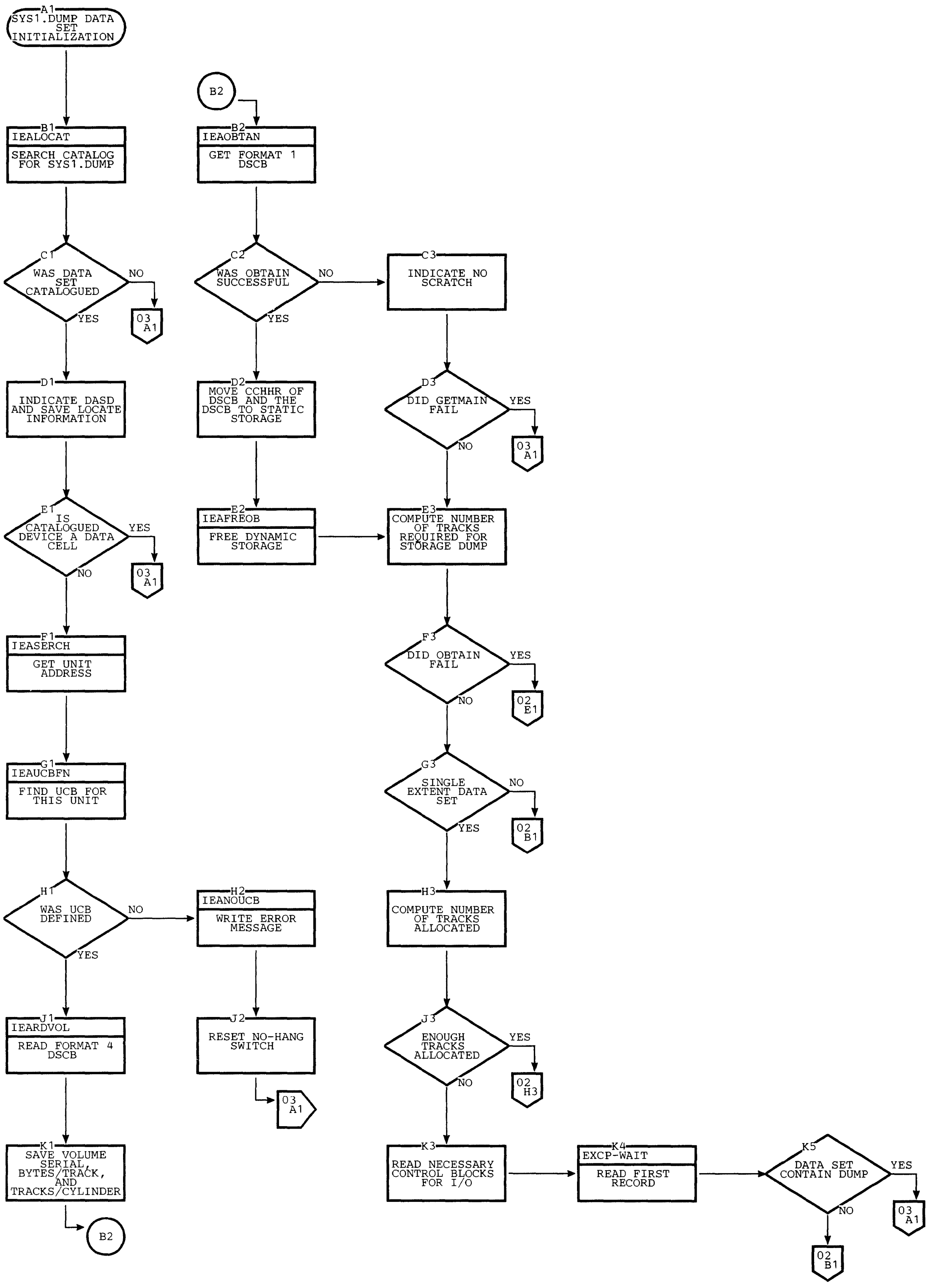

A1 SYS1.DUMP DATA SET INITIALIZATION
B1 IEALOCAT SEARCH CATALOG FOR SYS1.DUMP
C1 WAS DATA SET CATALOGUED
NO
03 A1
YES
D1 INDICATE DASD AND SAVE LOCATE INFORMATION
E1 IS CATALOGUED DEVICE A DATA CELL
YES
03 A1
NO
F1 IEASERCH GET UNIT ADDRESS
G1 IEAUCBFN FIND UCB FOR THIS UNIT
H1 WAS UCB DEFINED
NO
YES
J1 IEARDVOL READ FORMAT 4 DSCB
K1 SAVE VOLUME SERIAL, BYTES/TRACK, AND TRACKS/CYLINDER
B2
B2
B2 IEAOBTAN GET FORMAT 1 DSCB
C2 WAS OBTAIN SUCCESSFUL
NO
YES
D2 MOVE CCHHR OF DSCB AND THE DSCB TO STATIC STORAGE
E2 IEAFREOB FREE DYNAMIC STORAGE
H2 IEANOUCB WRITE ERROR MESSAGE
J2 RESET NO-HANG SWITCH
03 A1
C3 INDICATE NO SCRATCH
D3 DID GETMAIN FAIL
YES
03 A1
NO
E3 COMPUTE NUMBER OF TRACKS REQUIRED FOR STORAGE DUMP
F3 DID OBTAIN FAIL
YES
02 E1
NO
G3 SINGLE EXTENT DATA SET
NO
02 B1
YES
H3 COMPUTE NUMBER OF TRACKS ALLOCATED
J3 ENOUGH TRACKS ALLOCATED
YES
02 H3
NO
K3 READ NECESSARY CONTROL BLOCKS FOR I/O
K4 EXCP-WAIT READ FIRST RECORD
K5 DATA SET CONTAIN DUMP
YES
03 A1
NO
02 B1

Chart AL. Initialization for SYS1.DUMP Data Set (Chart 2 of 3)

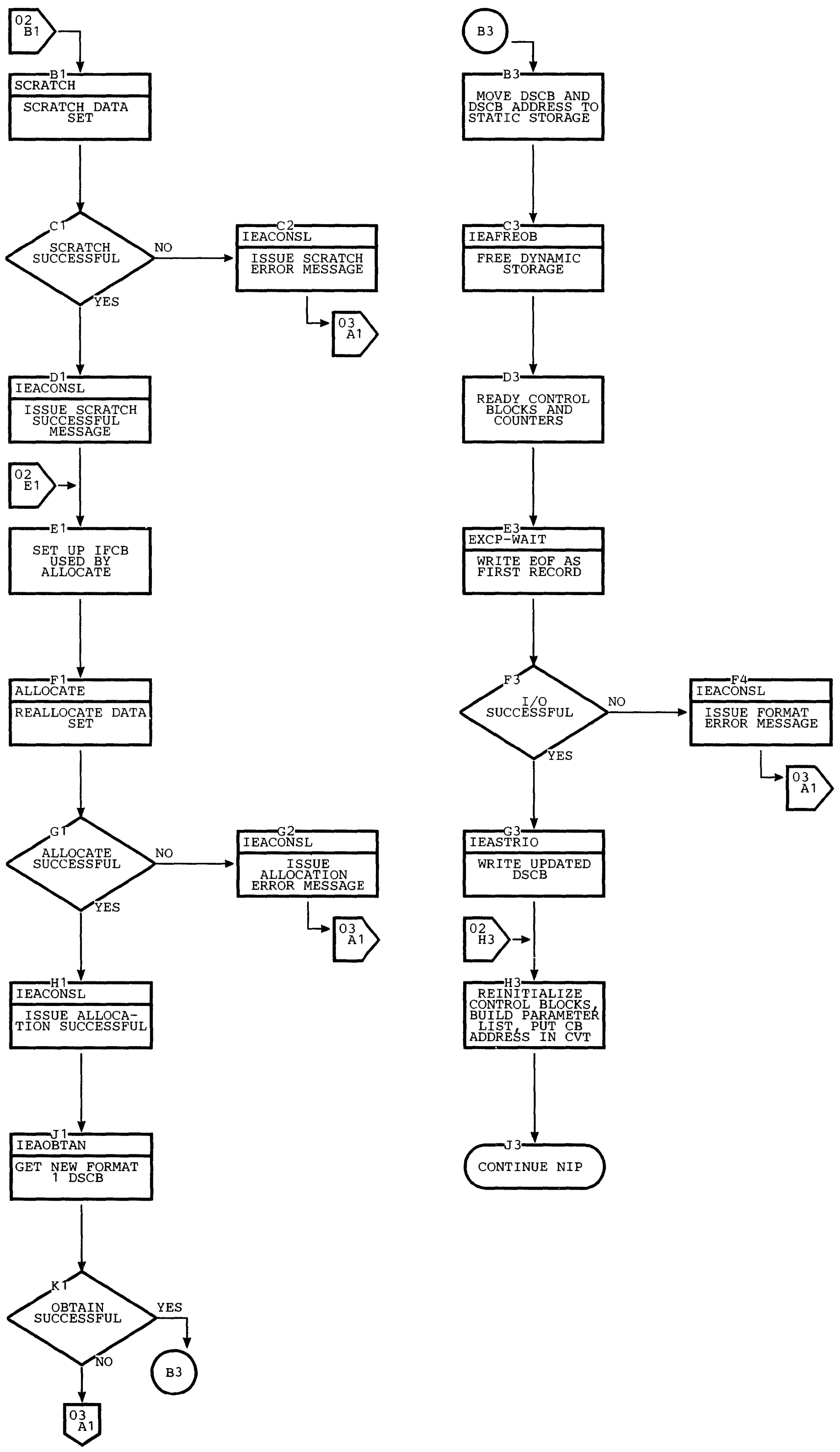
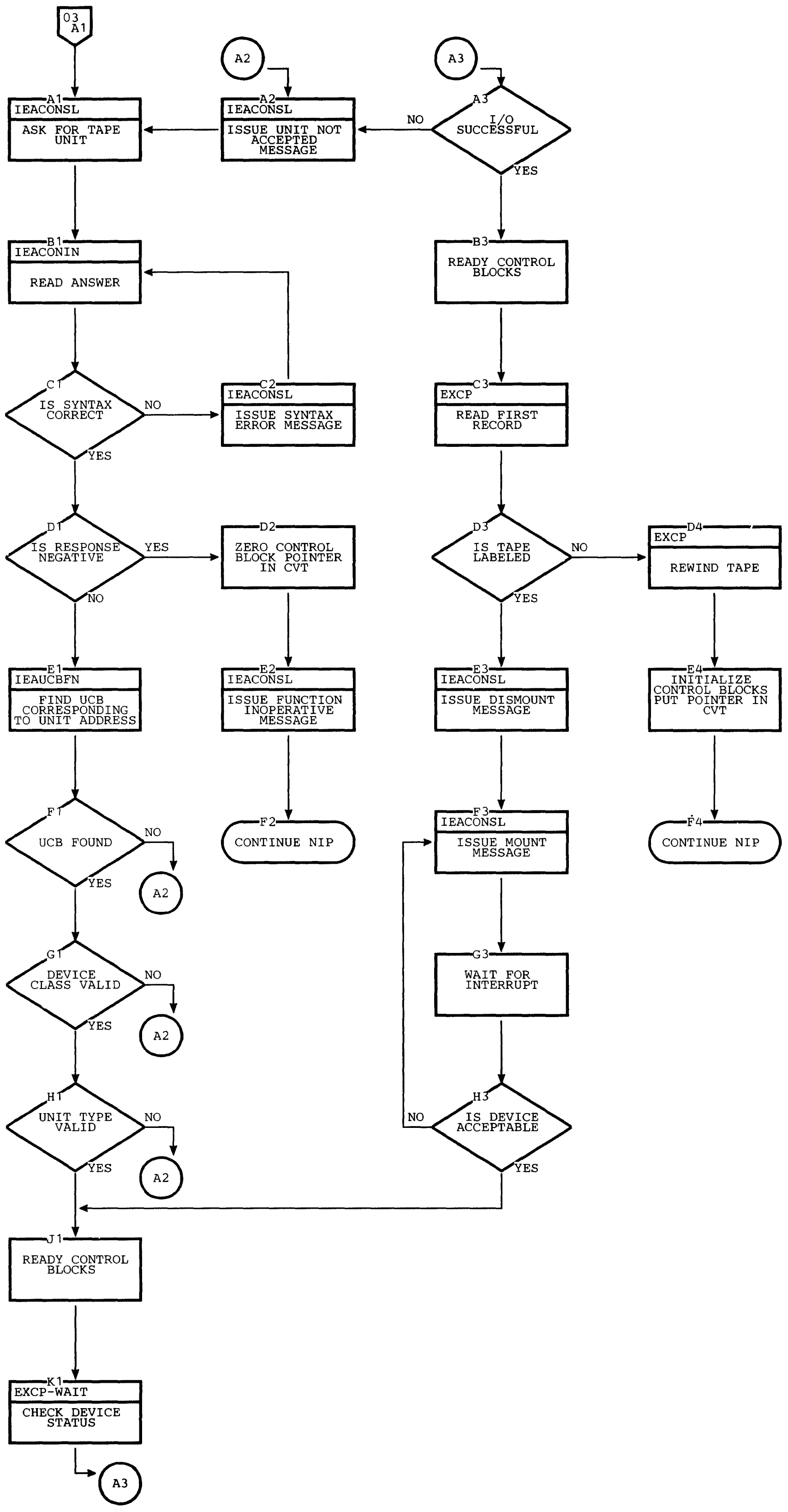


Chart AL. Initialization for SYS1.DUMP Data Set (Chart 3 of 3)



APPENDIX A: INITIALIZATION FOR MVT WITH MODEL 65 MULTIPROCESSING

In MVT with Model 65 multiprocessing, NIP performs the additional functions of initializing entries in the Prefixed Storage Area (PSA), defining main storage to allow blocks of storage to be logically omitted from the system, and initializing the second CPU.

The NIP routines which initialize the PSA, define main storage, and perform initialization for the second CPU are contained in a separate load module (IEAMP650), brought into main storage by NIP after the system queue area has been rebuilt in its permanent location. The layout of main storage, after the multiprocessing NIP load module has been brought in, is shown in Figure 25.

Until the multiprocessing NIP load module (IEAMP650) is loaded, the higher main storage units contained in the configuration are not represented by any FBQE. IEAMP650 determines the size of main storage to enable FBQE representation for all main storage and to place the second PSA in the highest 4K block of main storage. Ultimately, NIP relocates itself above IEAMP650.

PRELIMINARY INITIALIZATION AT ENTRY TO NIP

NIP ensures that the lower 256K bytes of main storage are not malfunctioning, and establishes a temporary Prefixed Storage Area (PSA) for the second CPU.

CHECKING THE FIRST MAIN STORAGE UNIT

If a machine check occurred when the IPL program cleared main storage (due to a malfunctioning storage area), a uniprocessing system enters the WAIT state at entry to NIP, because the PSW containing the NIP routine entry point address is enabled for machine checks. However, in MVT with Model 65 multiprocessing, NIP can logically omit from the system malfunctioning areas of main storage above 256K. Therefore, the PSW for the multiprocessing NIP module is disabled for machine checks, and NIP determines if a machine check occurred in the first 256K bytes of main storage by issuing a Diagnose Log instruction and checking the logout in the diagnostic scan out area of the PSA. If a machine check did occur in the first 256K bytes, a WAIT state PSW is loaded with an error code of 18. Otherwise, a PSW enabled for machine check is loaded, and the NIP routine continues. (Pending machine checks are cleared when a Diagnose Log instruction is executed.)

TEMPORARY PREFIXED STORAGE AREA INITIALIZATION

Certain entries in the Prefixed Storage Area (PSA) are used by multiprocessing routines prior to final PSA initialization. Therefore, NIP creates a 4K byte temporary PSA for the second CPU at the top of the first 256K bytes of main storage, and initializes the following PSA fields:

- The CPUID and IOCPUID bytes are initialized. NIP examines the CPU1 ID bit (bit 1 of byte X'C2') in the logout, and places a X'C1' in the CPU identification byte (CPUID) if the bit is on; if off, NIP places a X'C2' in CPUID. The opposite value is placed in CPUID in the temporary PSA. In messages printed at the console, CPU 1 is called CPU A; CPU 2 is called CPU B.

NIP also initializes IOCPUID, a full word in the PSA, which is used by IOS to indicate the CPU that started the last I/O operation on a particular device. This entry is set to X'00000000' for CPU A and to X'00000008' for CPU B.

- The doubleword TCB pointers (IEATCBP) in the temporary PSA are set to the partition TCB, and the RB wait count is set to zero. (See "Initializing the TCB Pointers (IEATCBP)" for detail.)
- A copy of the Channel Availability Table is placed in the temporary PSA (see "Initializing the Channel Availability Table").

UNIT CONTROL BLOCK INITIALIZATION

Unit Control Blocks are initialized to reflect the status of each device and the attachment of a device to a particular CPU. The availability of each CPU's channels is determined to allow one CPU to access a device through the other CPU's channel if its own channel is unavailable. Thus, in a multiprocessing system, a device is not marked offline if only one CPU's channel to the device is unavailable.

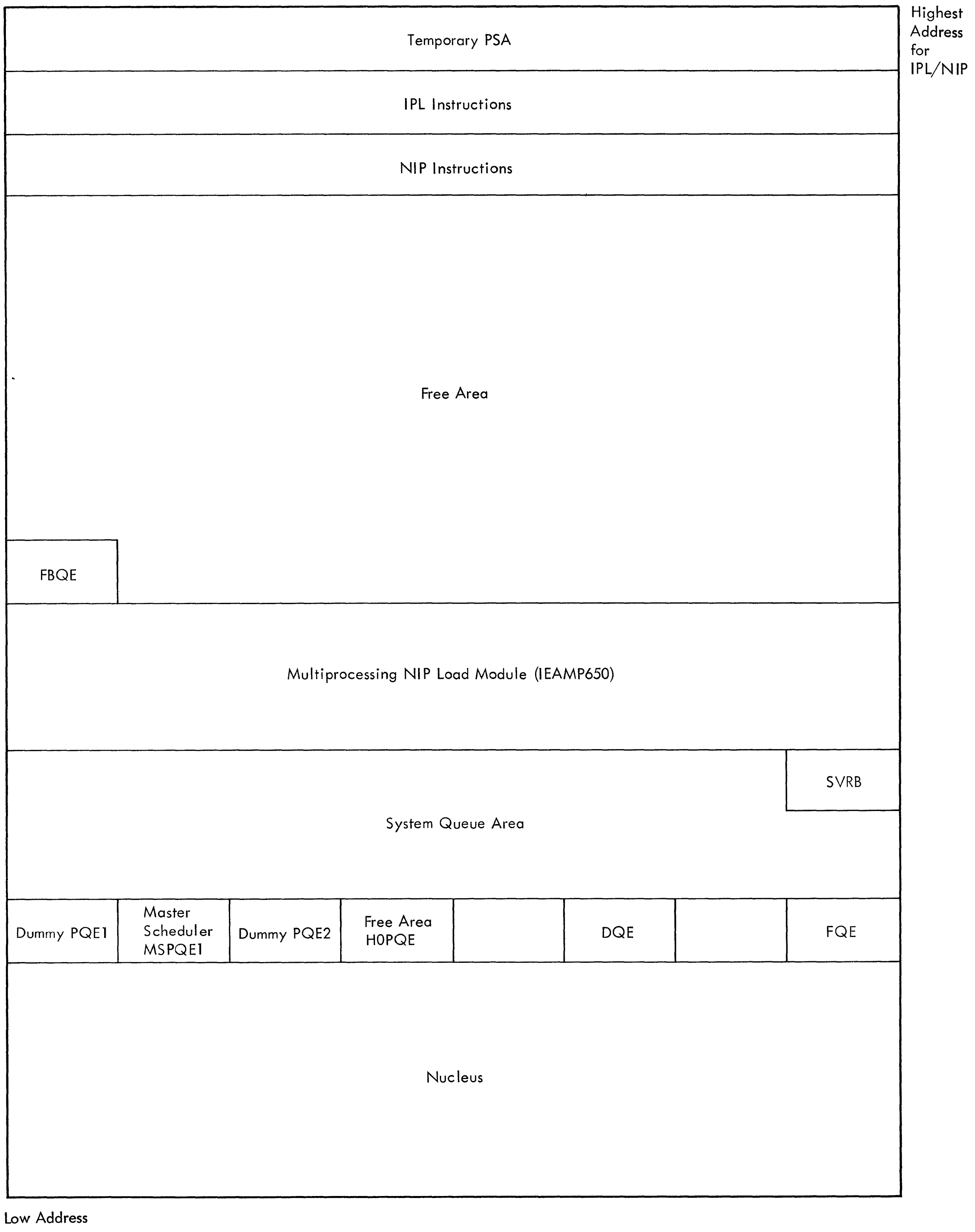


Figure 25. Main Storage After Multiprocessing NIP Module Has Been Loaded

Channel	Byte 0	Byte 1
6	0001 0110	0000 0000
5	0001 0101	0000 0000
4	0001 0100	0000 0000
3	0001 0011	0000 0000
2	0001 0010	0000 0000
1	0001 0001	0000 0000
0	0001 0000	0000 0000

Byte 0 indicates channel status and number; Byte 1 indicates control unit/device address. There is an entry in the Channel Availability Table for each device.

Byte 0	Setting	Meaning
Bit 0	0	Channel not busy
	1	Channel is busy
Bit 1	0	Channel is operational
	1	Channel is not operational
Bit 2	0	Channel is attached to system
	1	Channel is not attached to system
Bit 3	0	Channel is initialized
	1	Channel is not initialized

Figure 26. Channel Availability Table At System Generation

INITIALIZING THE CHANNEL AVAILABILITY TABLE

NIP determines if any channel path is not available to the CPU because the channel is in TEST MODE, is powered down, or not physically attached to the system. An unavailable channel is marked offline in the Channel Availability Table which is located in the PSA and shown in Figure 26. The status of each channel is described by a two-byte entry which represents a 16-bit Channel-Control Unit-Device address.

Each channel is tested by issuing a TCH instruction which determines the setting of bit 1 in byte 0 (operational/not operational). NIP then determines from the IOS constant IECHICHA (established at system generation) the highest numbered channel attached to the system, and sets bit 2 of byte 0 to 1 for all channels above the highest numbered channel. Bit 2 indicates that the channel is not attached to the system.

The Channel Availability Table is used by the VARY channel routine to logically remove or attach a channel to the CPU. If any of the four high-order bits are set, a condition code of not operational is received when the channel is tested.

<u>Determining Console Readiness</u>

NIP locates the first available console from the list of console devices specified by the SYSGEN program and marks it active in the Unit Control Module. A console is available if (1) the channel is available, and (2) the device is operational.

If the console device can be accessed by only one CPU, as in the case of a 1052 with a 1052 adapter, byte 0 of the multiprocessing addition to the UCB is initialized as shown in Figure 27.

Byte 0 Setting		Meaning
Bit 6	Bit 7	
0	0	device accessible to both CPUs or neither CPU
1	0	device accessible only to CPU A
0	1	device accessible only to CPU B

Figure 27. MVT with Model 65 Multiprocessing Addition to UCB

INITIALIZING DIRECT ACCESS DEVICE UCBS

In a multiprocessing system, before a device is tested, the channel to that device is checked for availability. If the channel is marked unavailable in the Channel Availability Table, the device is marked offline and not ready in its UCB. Otherwise, the device is tested as in the uniprocessing system.

INITIALIZING NON-DIRECT ACCESS DEVICE UCBS

In a multiprocessing system, before testing each non-direct access device, the channel is checked for availability. If unavailable, the UCB for the device is marked offline. If the channel is available, a TIO is issued to the device. If a condition code of not operational results, the UCB is marked offline. If a CSW is stored, a SENSE command is issued to clear the contingent connection and the UCB remains marked online. If the device is a tape drive, the device is tested to determine if it physically exists; if not, the UCB is marked offline and not ready. If a CSW is not stored, the UCB remains marked ready and online.

Teleprocessing Devices

A teleprocessing device is attached to the first CPU that issues a TIO and finds the device available. Therefore, teleprocessing devices can be attached to either one of the two CPUs. The multiprocessing addition to the UCB, located at a displacement of -4, is initialized as shown in Table 6.

PREFIXED STORAGE AREA (PSA) INITIALIZATION

Since each multiprocessing CPU works on tasks independently, and because program status words must be stored separately, a multiprocessing system requires two PSAs; one located in lower main storage and one in upper main storage. In the multisystem mode, NIP creates an additional 4K byte PSA for the second CPU; in the partitioned mode, a 2K byte PSA. The 2K byte PSA is located at upper main storage, and its entries are initialized to allow multiprocessing routines to operate as if two CPUs were functioning.

In the multisystem mode, the CPU with its prefix switch disabled has the lower PSA, and the CPU with its prefix switch enabled has the upper PSA. The prefix switch is a hardware mechanism which allows a routine to access its own CPU's PSA by referring to lower 4K byte locations and to access the other CPU's PSA by referring to upper 4K byte locations. If the switch is disabled (CPU has lower PSA), addresses generated by the CPU remain unchanged. If the switch is enabled (CPU has upper PSA), addresses generated by the CPU are checked to determine whether they are less than 4K bytes or greater than the prefix value. (The prefix value is 4K bytes less than the highest main storage address indicated on the configuration control panel.) If less than 4K bytes, the prefix value is added to the address; if greater than the prefix value, the prefix value is subtracted from the address. Figure 28 shows the layout of main storage after the second PSA is established.

NIP initializes the following multiprocessing fields in the PSAs:

- CPU status byte
- Storage Element Status Map
- Prefix field
- External and machine-new PSWs
- PSA of second CPU
- TCB pointers
- Timer Prefix field
- PTRIGGER field
- CPU identification bytes
- Console Identification bytes
- Channel Availability Table

INITIALIZING THE CPU STATUS BYTE (CPUSTAT)

NIP determines the status of the multiprocessing system and records it by setting the CPUSTAT byte to one of the following values:

Setting	Indication
00000000	Multisystem with two CPUs
00000001	Partitioned with one CPU
00000010	Multisystem with one CPU

To determine the system status, NIP tests the multisystem log bit, bit 7 of byte X'C1' in the logout. If on, the system is operating in multisystem mode, and CPUSTAT is set to X'00'; if off, the system is operating in partitioned mode, and CPUSTAT is set to X'01'. In the multisystem mode, after completing initialization for the first CPU, NIP determines whether the second CPU is in operation; if not, CPUSTAT is then set to X'02'.

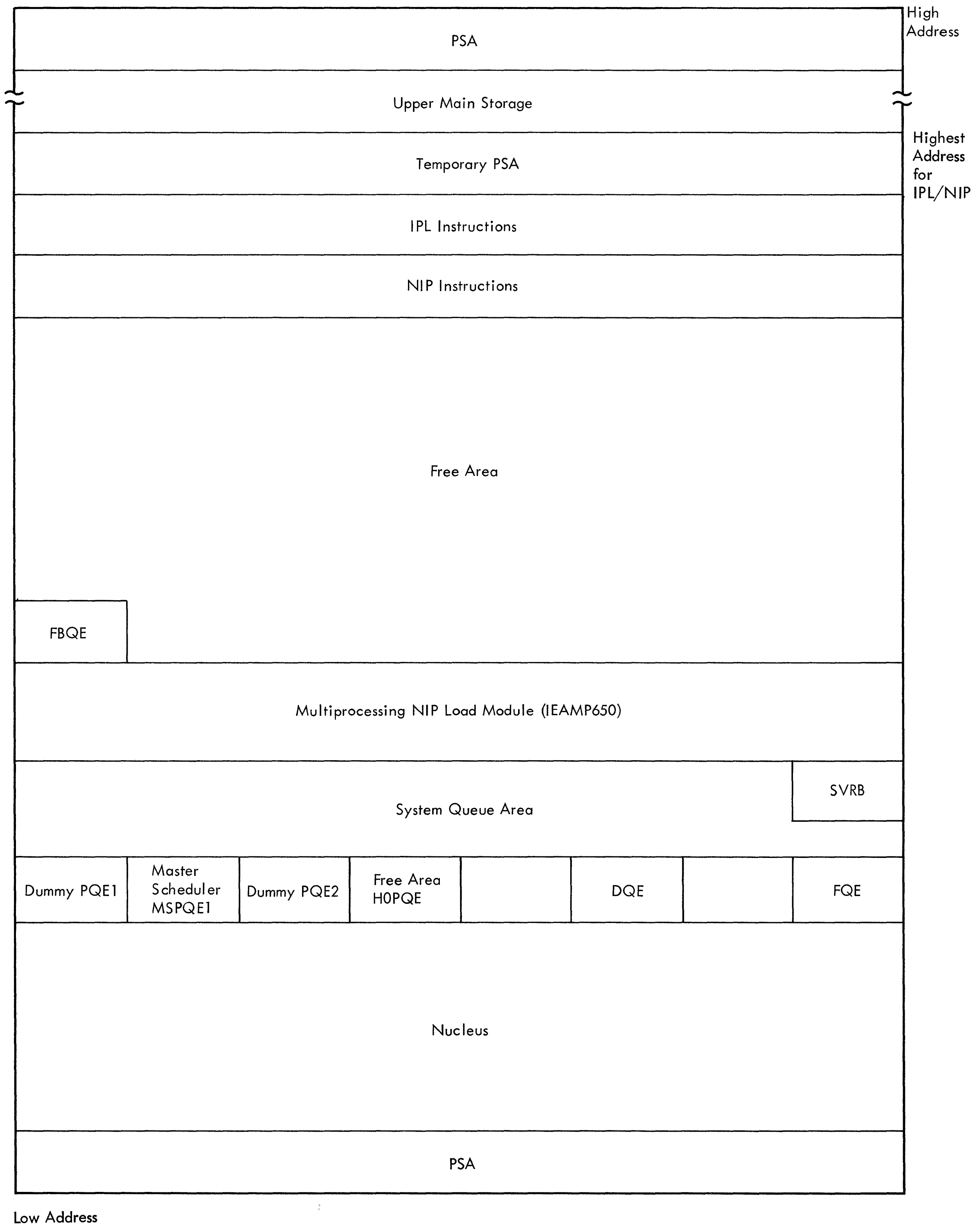


Figure 28. Main Storage After the Second PSA Has Been Established

INITIALIZING THE STORAGE ELEMENT STATUS MAP (FSSEMAP)

NIP clears main storage above 256K and detects the unavailability of main storage units that are physically not in the system or blocks that are malfunctioning. NIP initializes FSSEMAP to reflect the status of main storage, by setting a pair of bits for each 2K storage block to one of the following values:

Setting	Indication
00	available 2K storage block
11	unavailable 2K storage block (not chained into system queue)

NIP first examines the storage status bits in the logout to determine whether any 256K storage units are physically omitted from the system and are therefore unavailable. If any are, the corresponding bit setting in FSSEMAP is B'11'.

The available main storage above 256K is then checked to ensure that it is functioning properly. A machine check interruption in any 2K block of main storage causes that block to be represented in FSSEMAP as unavailable.

The main storage check routine also determines whether:

1. Both CPUs can access each available 256K storage unit
2. The upper PSA area is available (see "Defining Main Storage")

INITIALIZING THE PREFIX (PREFIX2)

NIP places the address of the upper storage PSA in PREFIX2, a word in the PSA which enables a routine to access the other CPU's PSA. This value, less 1, is also stored in the CVT as the highest addressable main storage byte.

INITIALIZING THE EXTERNAL AND MACHINE NEW PSWS

In the multisystem mode, NIP sets the external interruption new PSW to point to the malfunction alert handler, a routine which determines if a malfunction alert signal (issued by the other CPU when it experiences a machine check interruption) has been received. If not, the External First-Level Interruption Handler is entered. In the partitioned mode, NIP leaves the external interruption new PSW pointing to the External First-Level Interruption Handler.

In the multisystem mode, NIP sets the machine check new PSW to contain a Wait State error code of A21. In the partitioned mode, NIP leaves the machine check New PSW pointing to the Recovery Management Support (RMS) routines. If, in the multisystem mode, NIP later finds that only one CPU is operating, the machine check new PSW is reset to point to RMS.

CONSTRUCTING THE SECOND PSA

The second PSA is constructed by duplicating common fields from the first PSA. NIP places a copy of the lower PSA in upper main storage, starting at the address in PREFIX2. NIP then initializes the fields in the PSAs which differ for each CPU.

INITIALIZING THE TCB POINTER (IEATCBP)

In a multiprocessing system, each CPU performs different tasks; therefore, a doubleword TCB pointer is located in the PSA of each CPU. NIP sets the "new" and "old" pointers in the IEATCBP field of the second CPU to point to the partition TCB, a dummy high priority task. When only one CPU is operating, the "new" and "old" pointers in the second PSA point to this dummy task which is always dispatchable. Thus, the Dispatcher routine will never dispatch another task on the nonexistent CPU.

During second CPU initialization, the partition TCB is set non-dispatchable, and the IEATCBP pointer of the second CPU is set to point to the WAIT task, a special task which has no associated programs and which the Dispatcher always dispatches in the Wait State.

INITIALIZING THE PTRIGGER FIELD

In the multisystem mode, NIP examines the Prefix trigger log bit, bit 2 of byte X'88'. If on, the prefix switch is enabled for this CPU, and PTRIGGER byte of the PSA is set to C'P'; if off, the prefix switch is disabled, and PTRIGGER is set to C' '. The second CPU's PTRIGGER field is set to the opposite value. The ABDUMP routine uses this field to determine which PSA belongs to each CPU.

INITIALIZING THE TIMER PREFIX FIELD (PREFTMRA)

Although each CPU has a timer, only the timer designated as active is used by timer routines. Initially, the timer of the CPU that performed the IPL routine is designated as active, the timer of the other CPU as inactive. To enable a routine to access the active timer, PREFTMRA contains zeros if the active timer is located in the same PSA, or contains the PREFIX2 value if the active timer is located in the other PSA. Thus, the active timer can be referenced by adding the PREFTMRA value to the timer's PSA displacement value.

If, during second CPU initialization, the second CPU's timer is found to be working, while the first CPU's timer is not, the second CPU's timer is designated as active.

INITIALIZING THE CPU IDENTIFICATION BYTES

The CPUID and IOCPUID bytes in the first CPU's PSA were initialized at entry to NIP (see "Temporary PSA Initialization"). Now, NIP initializes these bytes in the second CPU's PSA. CPUID and IOCPUID are set to X'C1' and X'00000000' for CPU A, or to X'C2' and X'00000008' for CPU B.

INITIALIZING THE CONSOLE IDENTIFICATION BYTES (CONSOLID)

CONSOLID is initialized to contain the address of an available 1052 console. In a system with Multiple Console Support (MCS), CONSOLID is initialized to contain the address of the highest available non-composite console on the master console's alternate chain. If none are available, it is initialized to contain the address of any non-composite console.

INITIALIZING THE CHANNEL AVAILABILITY TABLE

The multiprocessing Channel Availability Table, contained in the PSA, is initialized before the Unit Control Blocks are initialized (see "Unit Control Block Initialization").

DEFINING MAIN STORAGE

In a multiprocessing system, 256K storage units may not be physically present in the system, or may be malfunctioning. NIP logically removes these sections from the system and allocates main storage regions within the available area.

DETERMINING UNAVAILABLE MAIN STORAGE

Unavailable main storage is determined when the PSA is initialized. (See "Initializing the Storage Element Status Map (FSSEMAP)".) NIP checks the storage status log bits in the logout area to determine if any main storage units are not attached to the first CPU. If so, the corresponding address range is marked offline in FSSEMAP. After the second CPU has been initialized, NIP compares the storage status log bits of both CPUs. If they differ, a message is issued to the operator indicating that the 256K storage units are set asymmetrically, and a Wait State PSW with an error code of 15 is loaded.

NIP then checks and clears main storage above 256K. Any 2K storage blocks in which a machine check occurs is marked unavailable in FSSEMAP. If a block of main storage is malfunctioning, but a machine check does not occur, a Wait State PSW with an error code of 14 is loaded.

After main storage has been cleared, NIP checks FSSEMAP to determine whether the upper PSA (i.e., the upper 4K in the multisystem mode or a 2K block in the partitioned mode if the IPL CPU does not have prefixing enabled) is marked unavailable. If it is, a Wait State PSW with an error code of 16 is loaded.

DEFINING THE FREE AREA

The size of the upper PSA (2K if in partitioned mode, 4K if in multisystem mode) is subtracted from the highest available main storage byte, and this value is stored in the CVT as the highest addressable main storage byte. The size of the free area of main storage, from the system queue area to the upper PSA, is stored in the Master Scheduler FBQE and MSPQE1.

If any 2K blocks of main storage are marked unavailable in FSSEMAP, NIP constructs the free area FBQEs so that the unavailable blocks are logically excluded from the system. NIP then issues a message to the operator identifying the unavailable areas. (A VARY storage online command may be issued later to make the main storage blocks logically available.)

Figure 29 shows the layout of main storage if storage areas are logically omitted from the system.

FINAL MAIN STORAGE PREPARATION

After building the link pack area, NIP divides free main storage into the master scheduler region and the dynamic area.

Establishing the Final Master Scheduler Region

NIP searches the free area FBQEs, starting with the last FBQE, to find a free block which is large enough (at least 18K) for the master scheduler region. Any free block not large enough is marked unavailable in FSSEMAP.

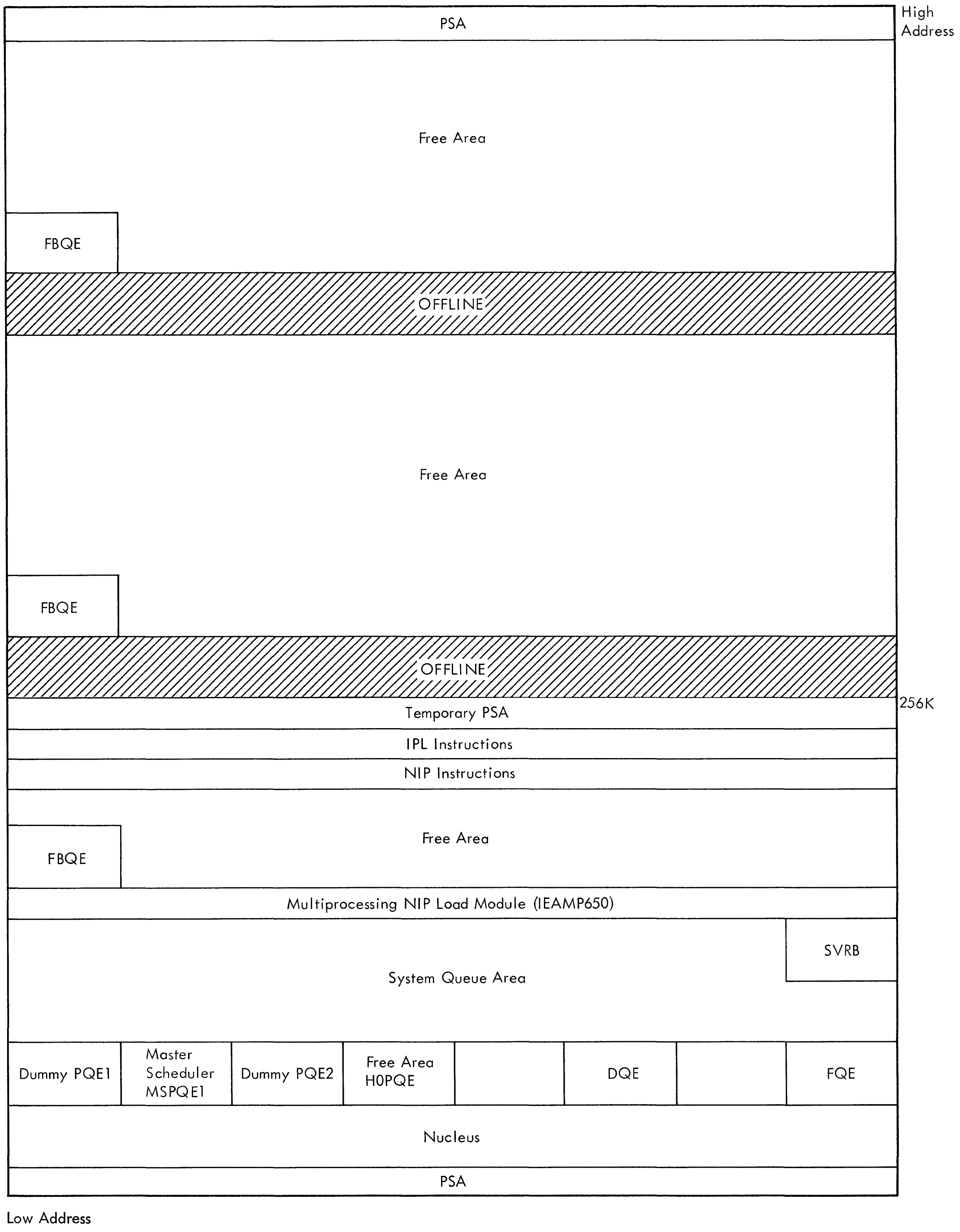


Figure 29. Main Storage After Free Area Has Been Established

When a free block is found, the size of the master scheduler region is subtracted from the size of the free block, and a new master scheduler FBQE is constructed at the beginning of the master scheduler region. The master scheduler partition queue element (PQE1) in the system queue area is set to the address of the master scheduler FBQE, and the master scheduler FBQE set to point PQE1.

Defining the Dynamic Area

NIP constructs an FBQE at the start of the dynamic area, overlaying some NIP instructions. The dynamic area FBQEs are then chained so that the first and last FBQEs point to the dynamic area PQE (PQE2), and PQE2 points to the first and last FBQEs. PQE2 contains the size of the dynamic area, from the end of the system queue area to the beginning of the master scheduler region.

Figure 30 shows final main storage layout.

SECOND CPU INITIALIZATION

In the multisystem mode, the multiprocessing NIP load module determines whether the second CPU is in operation, and, if it is, does the following:

- Clears the registers
- Initializes the Channel Table
- Initializes the Timer
- Checks devices for availability to the second CPU
- Initializes the Console Identification bytes
- Initializes the TCB pointer
- Checks for error conditions

If the second CPU is not in operation, NIP:

- Sets CPUSTAT to X'02' to indicate multisystem mode with one CPU.
- Sets the machine check new PSW to the Recovery Management Support routines.

NIP issues a WRITE DIRECT instruction with a X'40' in the I_2 field which causes the second CPU to load the PSW (LPSW) from its location 0. This location was set by NIP (first CPU) to point to location X'80' (second CPU). Location X'80' has also been set by NIP (first CPU) to contain a branch to the second CPU Initialization Routine. In addition, prior to issuing the WRITE DIRECT instruction, NIP sets a byte in main storage which is reset by NIP on the second CPU. If this byte is not reset, the second CPU is not in operation.

While initialization of the second CPU is in process, the first CPU's NIP routine tests the completion byte in main storage. When initialization of the second CPU is complete, this byte is reset to X'00', the second CPU enters an enabled Wait State, and the first CPU's NIP routine checks for any errors found during second CPU initialization.

INITIALIZING THE CHANNEL AVAILABILITY TABLE

NIP tests each channel by issuing a TCH instruction and initializes the Channel Availability Table in the second CPU's PSA (see "Unit Control Block Initialization").

INITIALIZING THE TIMER

The value of the active timer on the first CPU is increased by X'80000000' and the result stored in the second CPU's timer. NIP waits for the timer to decrement. If the timer does decrement, NIP determines whether the first CPU's timer is also working. If not, NIP designates the timer on the second CPU as the active timer by switching the values in the PREFTMRA field for the two CPUs and switching the TIMER values for the two CPUs.

DETERMINING DEVICE AVAILABILITY

A TIO instruction is issued to each device whose channel is available to the second CPU. The UCB is marked as follows:

Result of TIO	UCB Setting
Not operational	offline/not ready
CSW stored	online/ready
Available (CSW not stored)	online/ready

If the results for both CPUs do not agree (that is, a device is neither online nor offline to both CPUs), an error flag is set. However, if the results for both CPUs do not agree because a channel is unavailable to one CPU, the error flag is not set. Thus, NIP sets the error flag only when the control units are set asymmetrically.

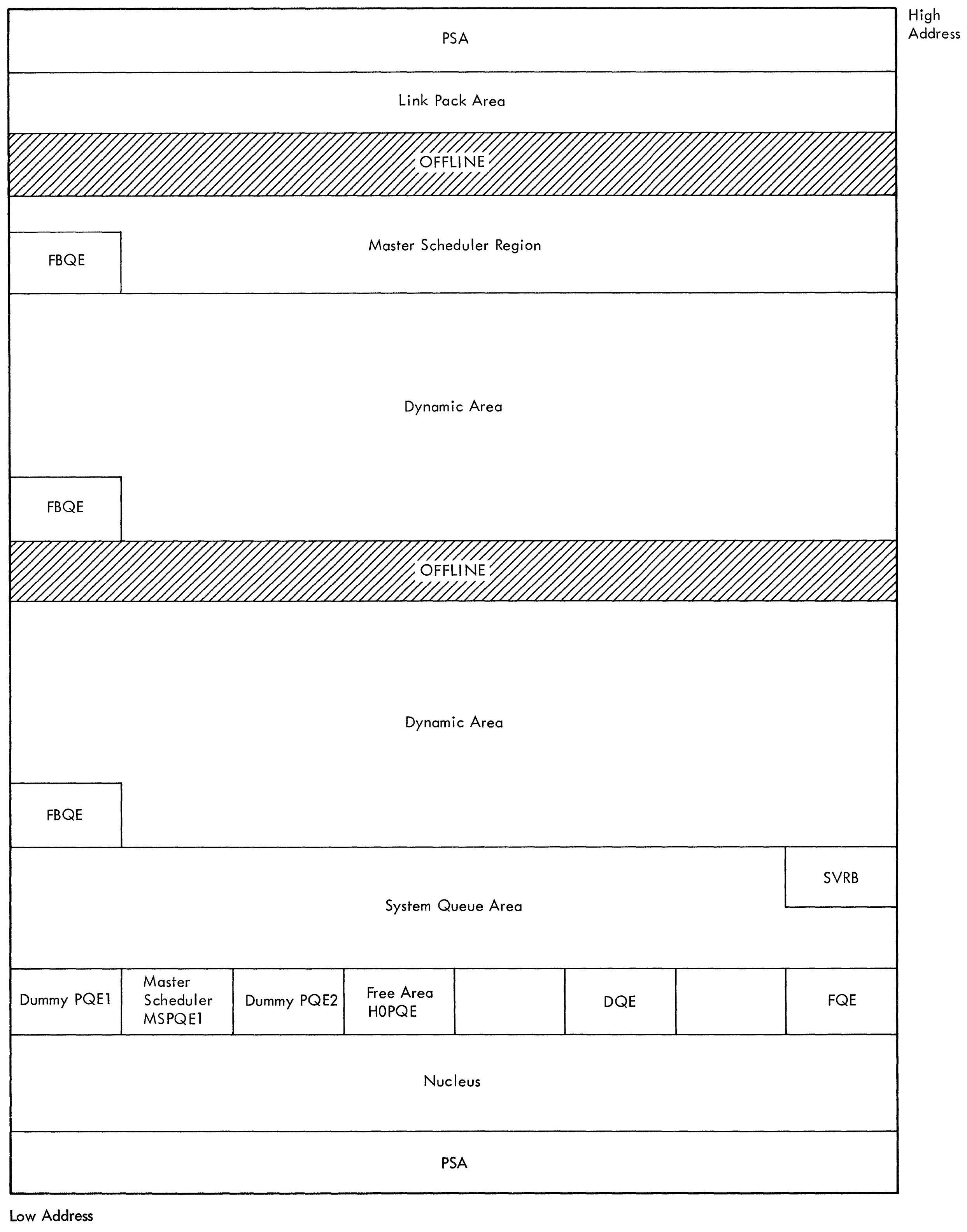


Figure 30. Final Main Storage Layout (MVT with Model 65 Multiprocessing)

If a console device that can be accessed by only one CPU (such as the 1052) is found available to the second CPU, the UCB word at a displacement of -4 is initialized as follows:

	Setting	Indication
Byte 0, Bit	6 7	
	1 0	device accessible to CPU A
	0 1	device accessible to CPU B

INITIALIZING THE TCB POINTER (IEATCBP)

NIP sets the partition task TCB non-dispatchable and sets the 'new' and 'old' TCB pointers to the WAIT task TCB.

CHECKING FOR ERROR CONDITIONS

The first CPU's NIP routine now checks for the following possible error conditions found during second CPU initialization:

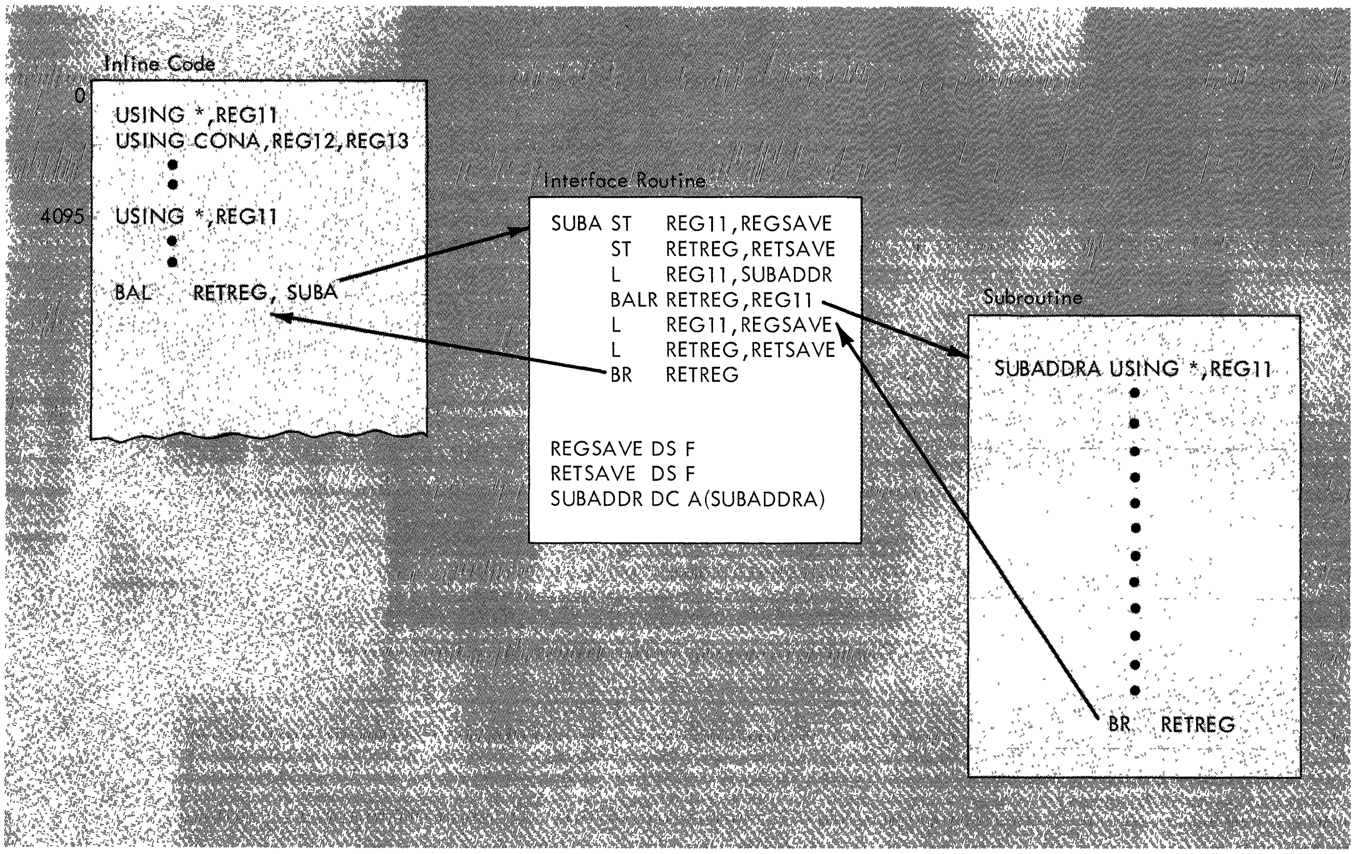
- If the prefix switches of both CPUs are set alike (both enabled or disabled), a message is written to the operator, and a Wait State PSW with an error code of 12 is loaded.
- If the storage status log bits of both CPUs are not set alike (that is, if both CPUs cannot access the same storage units using the same addresses), a message is written to the operator, and a Wait State PSW with an error code of 15 is loaded.
- If the second CPU's timer is not working, a message is issued to the operator.
- If any control units for tape or direct-access devices are set asymmetrically, a message is issued to the operator.

NIP TERMINATION

NIP issues messages to the operator indicating the channels (if any) that are marked offline in the Channel Availability Table, the status of each device, and the system initiated (partitioned, multiprocessor, or one-CPU multiprocessor). Control is then passed to the first module (IEEVIPL) of the master scheduler.

Unlike IPL, the assembly of the Nucleus Initialization Program depends on the type of control program being generated (MFT, MVT) and the options selected at system generation for that control program. Only those routines needed to support the control program and the selected options are included in the assembly. To save main storage, NIP uses subroutines to perform repetetive functions. The processing of "inline" code is sequential, and addressability is established for each 4096 bytes of code without regard to the previous base address. However, because only one base register (register 11) is available for all executable code (registers 12 and 13 are base registers for the NIP constants area), an interface routine for each subroutine is necessary to maintain proper addressability.

For each subroutine used by the in-line code, an interface routine similar to that shown in Figure 31 is included in the NIP constants area. When a BAL instruction is encountered in the in-line code, the interface routine receives control and saves the base address and the return address of the in-line code before passing control to the subroutine. The subroutine also uses register 11 to establish its addressability. Upon completion of processing, the subroutine returns control to the interface routine which reestablishes the return address and base address of the in-line code, and then branches to the in-line code. Notice that the symbolic address in the BAL instruction is not that of the subroutine, but that of the interface routine.



```
USING *,REG11
USING CONA,REG12,REG13
   •
   •
USING *,REG11
   •
   •
BAL    RETREG, SUBA
```

```
SUBA ST   REG11,REGSAVE
     ST   RETREG,RETSAVE
     L    REG11,SUBADDR
     BALR RETREG,REG11
     L    REG11,REGSAVE
     L    RETREG,RETSAVE
     BR   RETREG

REGSAVE DS F
RETSAVE  DS F
SUBADDR DC A(SUBADDRA)
```

```
SUBADDRA USING *,REG11
            •
            •
            •
         BR   RETREG
```

Figure 31. NIP Interface Routine

INDEX

Indexes to program logic manuals are consolidated in the publication IBM System/360 Operating System: Program Logic Manual Master Index, GY28-6717. For additional information about any subject listed below, refer to other publications listed for the same subject in the Master Index.

Where more than one page reference is given, the major reference is first.

IBM System/360 Operating System; Initial Program Loader and Nucleus Initialization Program; Program Number 360-CI-535

Order No. GY28-6661-5

READER'S COMMENT FORM

Your views about this publication may help improve its usefulness; this form will be sent to the author's department for appropriate action. Using this form to request system assistance or additional publications will delay response, however. *For more direct handling of such requests, please contact your IBM representative or the IBM Branch Office serving your locality.*

How did you use this publication?

☐ As an introduction	☐ As a text (student)
☐ As a reference manual	☐ As a text (instructor)

☐ For another purpose (explain)____

____

Please comment on the general usefulness of the book; suggest additions, deletions, and clarifications; list specific errors and omissions (give page numbers):

Cut or Fold Along Line

What is your occupation?____

Number of latest Technical Newsletter (if any) concerning this publication:______________________

Please include your name and address in the space below if you wish a reply.

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments.)

Your comments, please . . .

This manual is part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. Your comments on the other side of this form will be carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

Cut or Fold Along Line

Fold Fold

First Class
Permit 40
Armonk
New York

Business Reply Mail
No postage stamp necessary if mailed in the U.S.A.

Postage will be paid by:

International Business Machines Corporation
Department 636
Neighborhood Road
Kingston, New York 12401

Fold Fold

IBM®

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

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

GY28-6661-5

IPL/NIP Printed in U.S.A. GY28-6661-5

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]