IBM VSE/Enterprise Systems Architecture
VSE Central Functions

Initial Program Load and Job Control Diagnosis Reference

Version 6 Release 6

SC33-6325-02

IBM VSE/Enterprise Systems Architecture
VSE Central Functions

Initial Program Load and Job Control Diagnosis Reference

Version 6 Release 6

SC33-6325-02

Note!

Before using this information and the product it supports, be sure to read the general information under “Notices” on page ix.

Third Edition (March 2002)

This edition applies to Version 2 Release 6 of IBM Virtual Storage Extended/Enterprise Systems Architecture (VSE/ESA), Program Number 5690-VSE, and to all subsequent releases until otherwise indicated in new editions or Technical Newsletters.

This manual is not available in print. It has been updated for softcopy May 1998, September 2000 and March 2002.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the addresses given below.

A form for readers' comments is provided at the back of this publication. If the form has been removed, address your comments to:

IBM Deutschland Entwicklung GmbH
Department 3248
Schoenaicher Strasse 220
D-71032 Boeblingen
Federal Republic of Germany

You may also send your comments by FAX or via the Internet:

Internet: s390id@de.ibm.com
FAX (Germany): 07031-16-3456
FAX (other countries): (+49)+7031-16-3456

When you send information to IBM, you grant IBM a non-exclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

Figures

Notices

References in this publication to IBM* products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any of the intellectual property rights of IBM may be used instead of the IBM product, program, or service. The evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, are the responsibility of the user.

This publication is intended primarily for use by IBM personnel responsible for program service. The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. It is not intended as a description of a programming interface. The use of this information is a customer reponsibility. Service for errors, ommissions, accuracy, or completeness will not be provided.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785, USA.

Programming Interface Information

This book documents information that is NOT intended to be used as a Programming Interface of *VSE/ESA*.

Trademarks and Service Marks

The following terms, denoted by an asterisk (*) in this publication, are trademarks of the IBM Corporation in certain countries:

ECKD
Enterprise Systems Architecture/370
Enterprise Systems Architecture/390
ESA/370
ESA/390
IBM
SQL/DS
VM/ESA
VSE/ESA

Preface

This manual is intended for use by IBM personal responsible for servicing the VSE/ESA* Initial Program Load and Job Control. The publication supplements the listings of two VSE/ESA components: Initial Program Load and Job Control.

Publication Organization: Each component is discussed in five chapters.

Introduction: It summarizes functions and program characteristics of the two components and those of the service programs and phases related to them.

Program Design: It shows the function-to-phase relationship, the storage map of the program, and a description of each phase comprising essential information in a form similar to the prologue in the listings and a sequence of operation with important labels, showing the logic flow. For very small phases, this sequence of operation may be given already under the heading “Function.”

Program Organization: It shows the calling structure of phases.

Data Areas: It describes the use and data flow of those data areas which have more than just a phase-internal significance.

Diagnostics: It contains statement (or command)-to-phase, message-to-phase, and phase-to-module cross-references and interface conventions.

Bibliography:

Prerequisite reading for this manual is:

VSE/ESA Guide to System Functions

Related manuals referenced in the text are:

VSE/ESA System Control Statements
VSE/ESA Planning
VSE/ESA Installation
VSE/ESA Diagnosis Tools
VSE/ESA System Macros User's Guide
VSE/ESA System Macros Reference
VSE/ESA Messages and Codes
VSE/ESA Guide to System Functions

Also referenced are the hardware manuals:

IBM Enterprise Systems Architecture/390 - Principles of Operation
IBM Virtual Machine/Enterprise Systems Architecture: CP Programming Services

and the following Diagnosis Reference manuals:

Supervisor
Logical Transients
LIOCS, Vol.1 - General Information and Imperative Macros
LIOCS, Vol.2 - SAM
Error Recovery and Transients

Summary of Changes

VSE/ESA Version 1 Release 1

- Enterprise Systems Architecture*
- More Partitions
- Changed Storage Layout
- ECKD*

VSE/ESA Version 1 Release 3

- 31-bit Addressing
- Data Spaces
- Virtual Disk

VSE/ESA Version 2 Release 1

- Improved Console Support
- Stand-alone Integration
- Single Supervisor
- IBM 3494 Tape Library Dataserver

VSE/ESA Version 2 Release 2

- Year 2000

VSE/ESA Version 2 Release 3

- Daylight Saving

VSE/ESA Version 2 Release 5

- Improved Installation Support

Chapter 1. Introduction -- Initial Program Load

The IPL Procedure in the Context of the VSE System: The IPL procedure must be performed each time

- the system is powered up;
- the system I/O configuration changes, that is, you want to add or delete or display device assignments;
- you want to reallocate your page data set;
- you want to reallocate your label area;
- the lock file is or has to be reallocated;
- you want to reassign
  - your VSAM master catalog (SYSCAT),
  - your recorder file (SYSREC),
  - your system console (SYSLOG) (in some cases this is possible by the job control statement ASSGN),
- you want to change the group of system programs residing in the SVA;
- you want to reallocate the system GETVIS space and/or the SVA;
- you want to load a new supervisor;
- you want to respecify VSIZE, VIO, VPOOL, or NOPDS option;
- you want to respecify ATL, BUFLD, BUFSIZE, CHANQ, DASDFP, ESM, NPARTS, PASIZE, RSIZE, SDSIZE, SEC, SERVPART, SPSIZE, SUBLIB, UNATT, VMCF;
- you want to specify APPC/VM network information for SQL/DS* in a VM Guest Sharing environment.

IPL is divided into two successive processing steps:

- the bootstrap program
- command processing

The bootstrap phases reside initially in the system sublibrary SYSLIB and are written to predefined locations at the beginning of the disk when the system library is defined. From there, they are loaded by hardware IPL into main storage, see Figure 4 on page 15. $$A$IPLR and $$A$IPLE are then loaded from the system sublibrary SYSLIB by $$A$PLBK or $$A$PLBF, respectively.

The function of the *bootstrap program* is to load a supervisor, a dispatcher, and the console support, and to build a minimal (3-device) system, the basis for IPL command processing. The 3-device system consists of SYSRES, SYSLOG, and SYSUSE; IPL commands are entered via SYSUSE, and since SYSUSE may be assigned to the same device as SYSRES (IPL ASI procedure) or as SYSLOG, the 3-device system may physically consist of two devices only.

The bootstrap program operates in supervisor mode and prepares the system for IPL command processing by setting up I/O tables for a minimal system configuration in low storage.

The function of *command processing* is to bring up the complete system by processing the IPL commands specified by the user. The commands ADD and DEL prepare the LUB and PUB tables in IOFLD at the high end of storage. When one of the commands DLA, DPD, DLF, DEF, or SVA is read (and the ADD and DEL process is thereby shown to be finished) these tables are substituted for the "3-device system" tables in the supervisor and the rest of the system is adjusted according to the ADD, DEL, and, if present, SYS commands.

IPL can be performed either in an automated way via the ASI (automated system initialization) procedure, or interactively. If the first option is taken, the same functions are performed, but without operator intervention.

IPLing the Stand-Alone VSE System: For *stand-alone processing* all phases needed by IPL reside on tape in the sequence of their operation. The first two bootstrap phases $$A$IPL2 and $$A$PLBT are for tape only. All others phases are common for disk and tape IPL. No IPL commands are required for stand-alone system initialization.

The Load-Print-Control-Buffers Function in the VSE System: For most printers, a print control buffer has to be loaded. The buffers of a printer can be loaded as follows:

1. Automatically during IPL. Phase $$BUFLDR together with $$BUFLD1 and $$BUFLD2 is executed as part of the IPL procedure. If PRT1, 3800, 4248 or 1403U printers are attached to the system, $$BUFLDR together with the corresponding standard buffer image phases must be available in the system library.
2. Dynamically by issuing the LFCB or LUCB attention commands.
3. Dynamically by issuing the LFCB macro in a problem program. (This macro can only be used to load the FCB of a printer).
4. As a separate job step by executing the SYSBUFLD program.

Case 1 is documented in "The $$BUFLDR Function" on page 92 and case 4 is documented in "The SYSBUFLD Program" on page 97.

Cases 2 and 3 are documented in *VSE/ESA Logical Transients Diagnosis Reference*, LY33-9165.

Chapter 2. Design Information -- Initial Program Load

Function Overview

The two parts of IPL, bootstrap program load and command processing, are shown in overview charts. The bootstrap program load is shown in Figure 1; command processing is shown in Figure 2 on page 9.

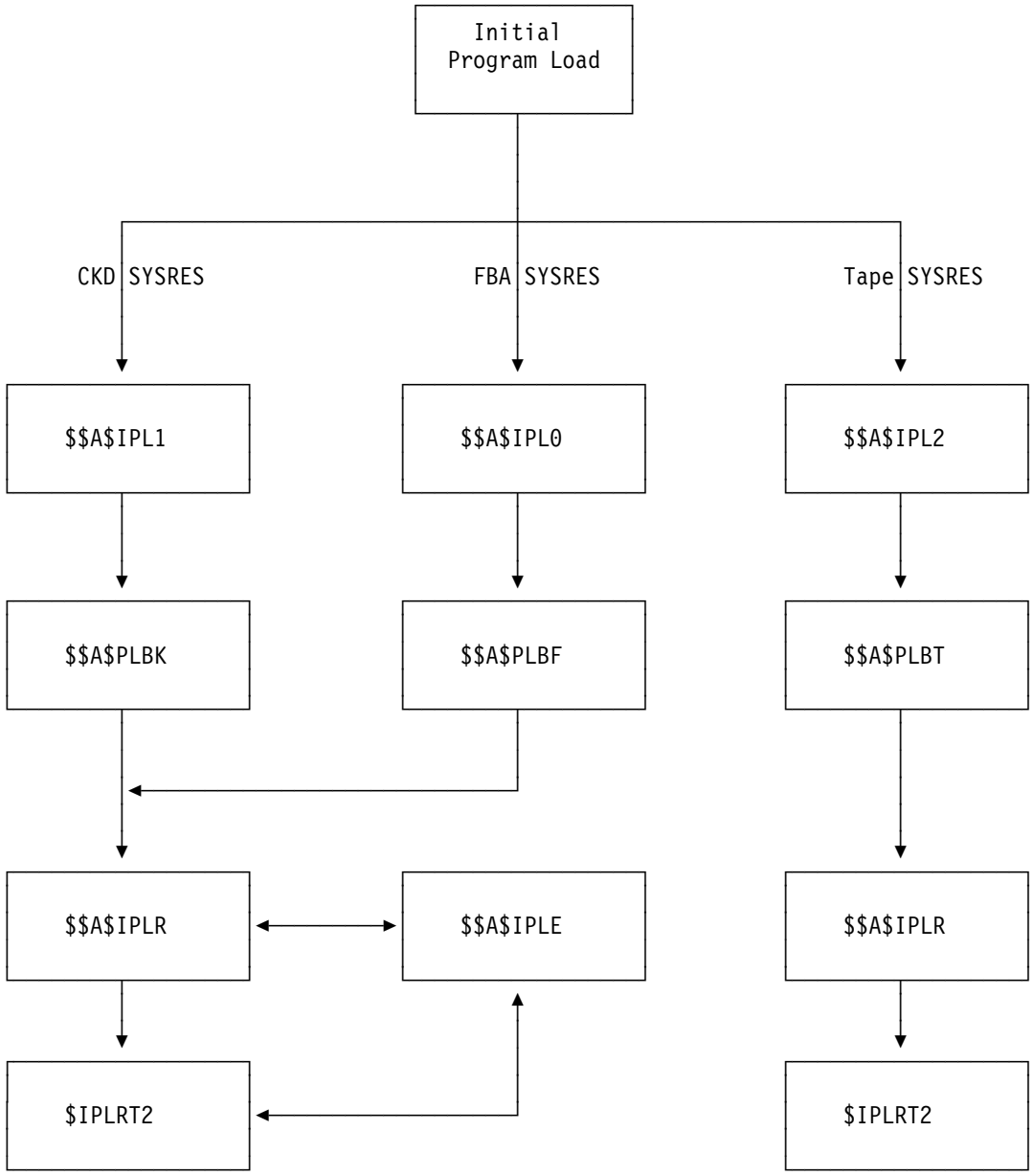


Figure 1 (Part 1 of 9). IPL Bootstrap Overview

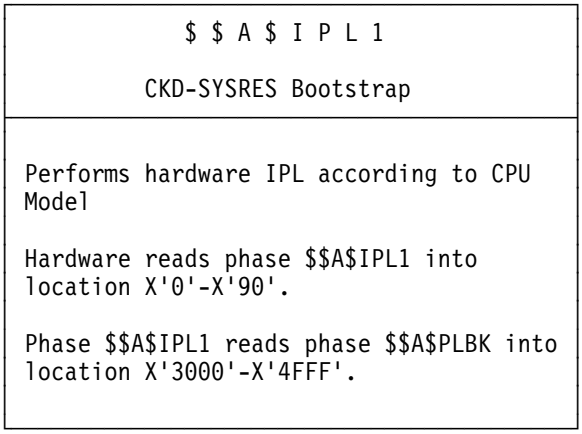


Figure 1 (Part 2 of 9). IPL Bootstrap Overview

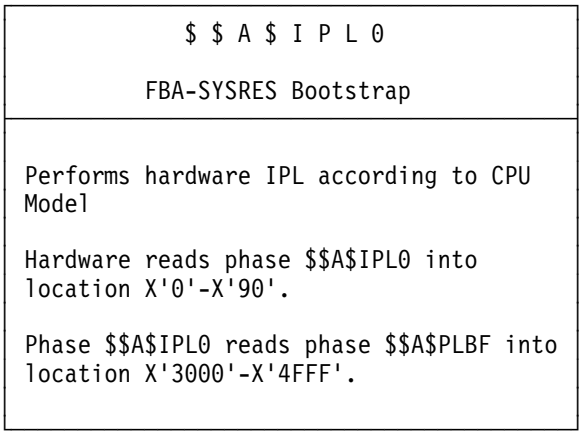


Figure 1 (Part 3 of 9). IPL Bootstrap Overview

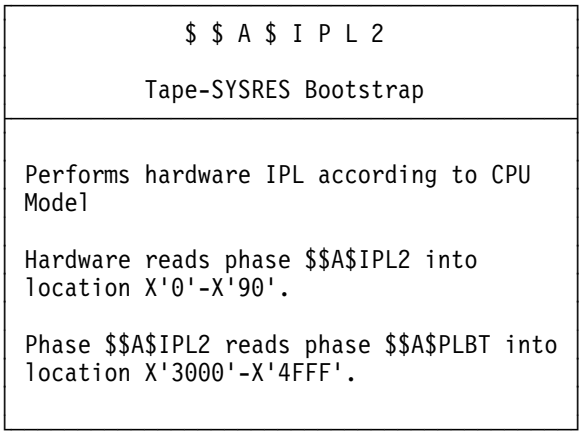


Figure 1 (Part 4 of 9). IPL Bootstrap Overview

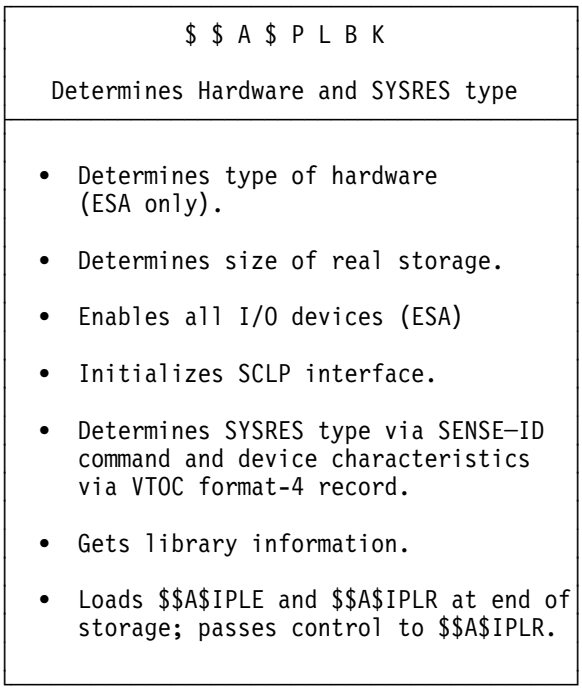


Figure 1 (Part 5 of 9). IPL Bootstrap Overview

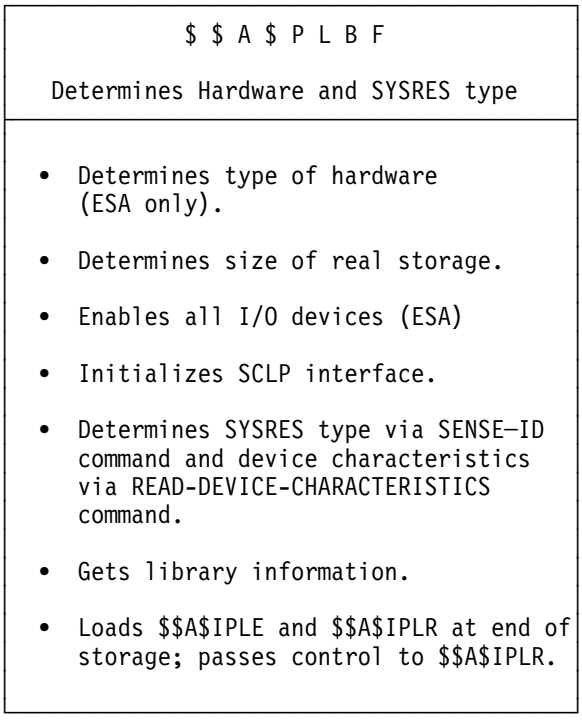


Figure 1 (Part 6 of 9). IPL Bootstrap Overview

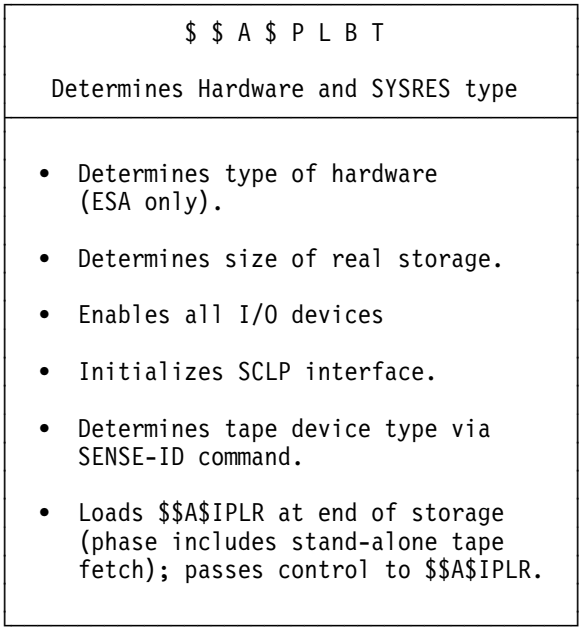


Figure 1 (Part 7 of 9). IPL Bootstrap Overview

$ $ A $ I P L R
Loads Supervisor,builds 3-device-system
• Prepares for operation with/without $$A$CDL0. • Prepares for operation with Integrated Console if requested. • Locates system console (SYSLOG). • Requests a SYSLOG device, if not located. • Requests a supervisor name and options (e.g. VSIZE, VIO). • Loads supervisor and dispatcher. • Initiates system. • Builds communication area for CMD-processing (IOFLD). • Locates IPL communication device (SYSUSE). • Builds 3-device system I/O tables. • Initializes program management. • Loads and initializes console support. • Loads and passes control to $IPLRT2.

Example of a communication device list (CDL) as to be specified in phase $$A$CDL0:

Communication Devices	
0009	= 1050A
000C	= 2540R
041F	= 3540
0420	= 3277

Figure 1 (Part 8 of 9). IPL Bootstrap Overview

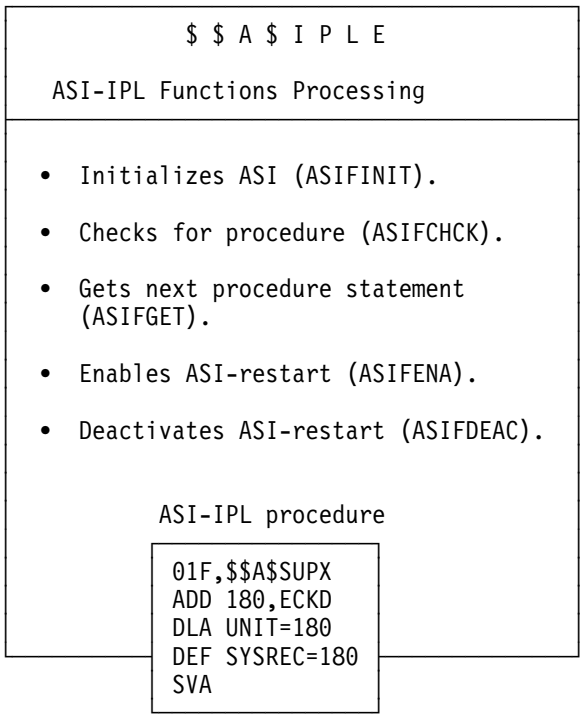


Figure 1 (Part 9 of 9). IPL Bootstrap Overview

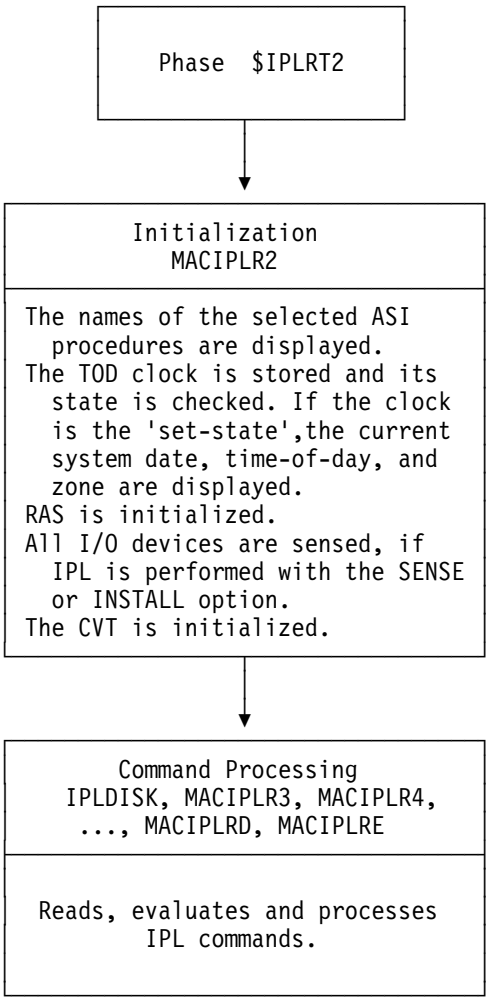


Figure 2 (Part 1 of 2). IPL Command-Processing Overview

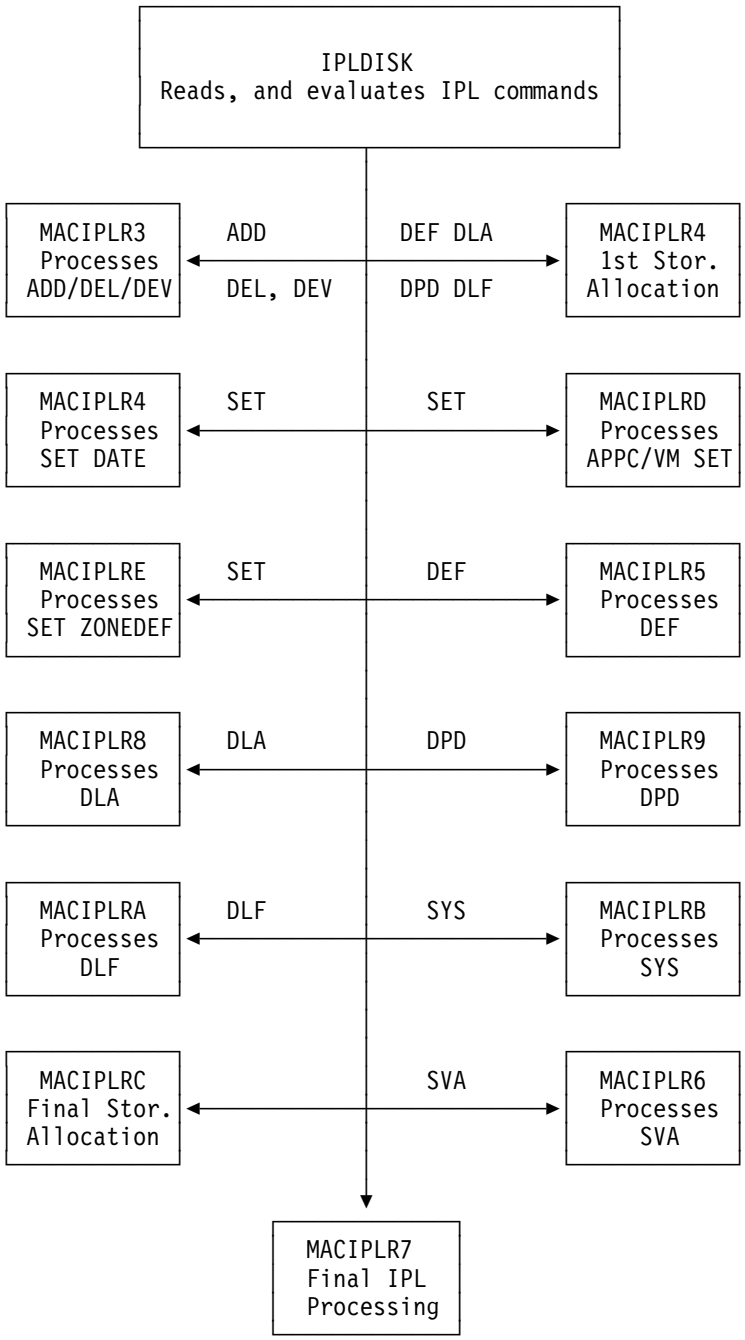


Figure 2 (Part 2 of 2). IPL Command-Processing Overview

Function-to-Phase List

The following list shows all IPL phases with their functions.

Subcontinent	Phase/Macro	Function
Bootstrap Processing	$$A$IPL0	FBA-Bootstrap Loader. Block 0
	$$A$IPL1	CKD-Bootstrap Loader. Head 0, Record 1 - 2
	$$A$IPL2	Tape-Bootstrap Loader - Stand Alone System
	$$A$PLBF	FBA-Bootstrap Prog. Block 2 - 17
	$$A$PLBK	CKD-Bootstrap Prog. Head 1, Record 1
	$$A$PLBT	Tape-Bootstrap Prog. - Stand Alone System
	$$A$IPLR	IPL Retrieval Program
	$$A$CDL0	Communic. Device List. User Specified
	$$A$IPLE	IPL Processing of ASI
	IPLBMAC	I/O Routines Included by $$A$PLBF/K/T, $$A$IPLR/E
Command Processing	$IPLRT2	IPL Command Processor Phase - includes CSECTS below
	IPLDISK	CSECT IJBIPLR2 - Common Routines and Tables
	MACIPLR2	CSECT IJBIPL2 - Initialization
	MACIPLR3	CSECT IJBIPL3 - Processes ADD/DEL/DEV
	MACIPLR4	CSECT IJBIPL4 - Processes SET Command, Allocates I/O Areas
	MACIPLR5	CSECT IJBIPL7 - Processes DEF Command
	MACIPLR6	CSECT IJBIPL6 - Processes SVA Command
	MACIPLR7	CSECT IJBIPL7 - End Processing
	IPLUNATT	CSECT IJBIPUNS - Unattended Node Support
	MACIPLR8	CSECT IJBIPL8 - Processes DLA Command
	MACIPLR9	CSECT IJBIPL9 - Processes DPD Command
	MACIPLRA	CSECT IJBIPLA - Processes DLF Command
	MACIPLRB	CSECT IJBIPLC - Processes SYS Command

Figure 3 (Part 1 of 2). Function-to-Phase

Subcontinent	Phase/Macro	Function
	MACIPLRC	CSECT IJBIPLC - Allocates more Supervisor Areas
	MACIPLRD	CSECT IJBIPLD - Processes APPC/VM SET Command
	MACIPLRE	CSECT IJBIPLE - Processes SET ZONEBDY/ZONEDEF Command
	IPLCGEN	Included by MACIPLR8/9/A
	$$BUFLDR $$BUFLD1 $$BUFLD2	Loads or Modifies Print Buffer

Figure 3 (Part 2 of 2). Function-to-Phase

ASI-IPL Function

Usually the system is initialized automaticly. This process is called automated system initialization (ASI). In case the ASI procedure is defective, the operator can choose interactive processing.

Bootstrap Loading

The IPL load parameter is checked for correctness and the appropriate initialization options are set.

Phase $$A$IPLE, which controls the ASI process, will be loaded at the very end of processor storage and before $$A$IPLE the Decompress Routine will be loaded. Bit ASILOAD in the IPL communication region IOFLD is turned on to signal that $$A$IPLE is loaded. This interface area, IOFLD, is now allocated in front of the Decompress Routine. The temporary communication area of phase $$A$IPLR, IPLRCOM, is copied into IOFLD.

Phase $$A$IPLR is loaded in front of IOFLD. The load points of $$A$IPLE and $$A$IPLR are saved into IOFLD.

If the Integrated Console is requested as system console by the appropriate load parameter, phase $$A$CISS (stand alone IC support) is loaded in front of $$A$IPLR.

Bootstrap of 3-Device System

When $$A$IPLR requests a SYSLOG cuu address, it checks bit ASI#LOAD if $$A$IPLE has sucessfully been loaded. If yes, it tries to start an ASI process:

$$A$IPLE is called to perform the ASIFINIT function. It tries to obtain the SYSLOG cuu address from the first record of the IPL procedure. This is possible only if either $ASIPROC the IPL default procedure $IPLESA exists in the system.

For the specification of $ASIPROC and of IPL procedures see *VSE/ESA Guide to System Functions* or “Description of Master Procedure $ASIPROC” on page 88.

If the Integrated Console is to be system console, the communication to the IC is initialized. If done sucessfully, the integrated console is established as the SYSLOG device.

In case the SYSLOG device could not be located or could not be addressed, the system waits for an attention interrupt to recognize the system console.

Once $$A$IPLE successfully read the first record of the IPL procedure, bit ASI#PROC is turned on together with return code zero after return from ASIFINIT.

Before extracting more information from the first ASI command, $$A$IPLR checks, if IPL was performed with the IPL prompting load parameter. If it was, the operator is prompted for another IPL procedure name.

ASI extracts the SUPVR-name from the IPL procedure specified. If the supervisor name is invalid or cannot be found in the system sublibrary SYSLIB, the operator is prompted for another name.

Bootstrap Supervisor Retrieval

The supervisor is loaded from the system sublibrary by IPL stand-alone retrieval routines. In the same way the dispatcher is loaded behind the supervisor, and then anchored into the supervisor.

If no ASI procedure was started (bit ASI#PROC is zero), then $$A$IPLR requests the SYSUSE cuu address from the operator. When ASI is in process, it copies the SYSRES PUB into the SYSUSE PUB, builds a dummy diskette label record, and prepares the IPL command processing and JCL switches as if the IPL commands would come from a diskette.

$$A$IPLR proceeds to complete the 3-device system and initializes the supervisor sub-components. Then bit ASI#SUP is turned on to indicate that the supervisor is now prepared for use. This flag is tested in $$A$IPLE IOS. If the bit is 0, I/O has to be performed by subroutine IORTN in macro IPLBMAC. If it is 1, I/O is done from supervisor IOS (EXCP). Normally, the directory lookup and the reading of the first library block of the IPL procedure are done using the private I/O subroutine, while the IPL commands on subsequent library blocks will be read via supervisor EXCP.

$$A$IPLR loads the console support phases (via LOAD SVC) at a fixed location behind supervisor and dispatcher. Then it passes control to the console router for initialization of the support.

$$A$IPLR then loads the IPL command processing phase $IPLRT2 (via LOAD SVC) and passes control to this phase.

Command Processing

The diskette-read subroutine checks bit ASI#PROC if ASI is in process. If yes, the next IPL command to be processed is obtained from the IPL procedure via an ASIFGET call to $$A$IPLE. The record is moved into the input buffer (CDBUFF) and processed. If IPL command processing detects an invalid command (coming from a card reader or diskette), it switches the input device to SYSLOG, thus offering an update or skip facility of the command in error.

Phase $IPLRT2 (MACIPLR4) calls $$A$IPLE at the entry point ASIFDEAC for ASI restart deactivation and displays message 0J10I IPL RESTART POINT BYPASSED. Once this message has been displayed, it is no longer possible to restart IPL.

Stand-Alone IPL

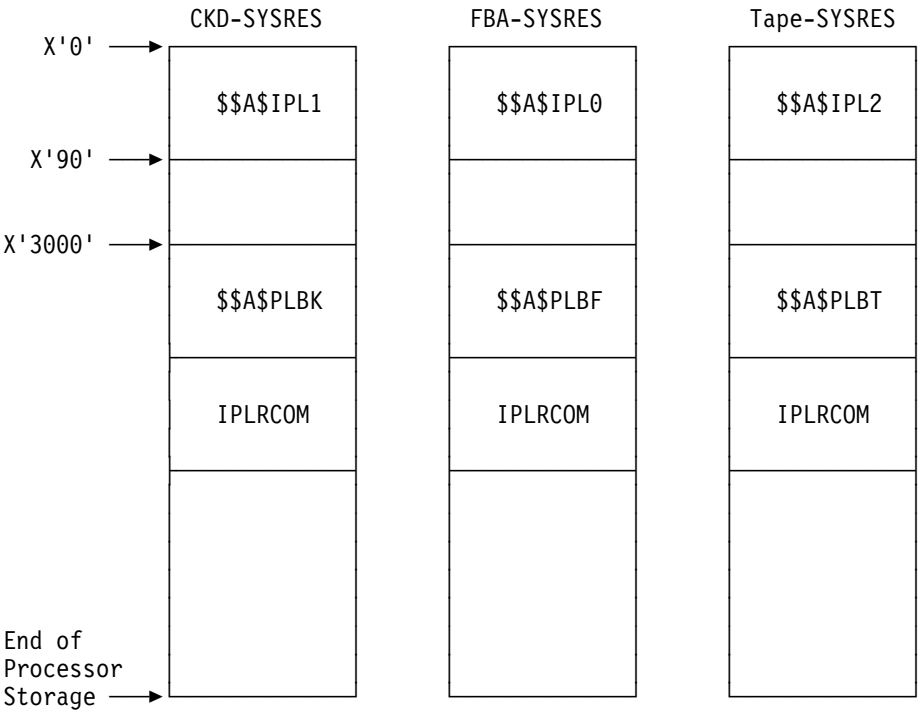
If a stand-alone system is IPL'ed from tape, IPL basicly proceeds as described above. The main difference: there is no ASI procedure, but the system is automatically initialized with predefined values.

Normally the system comes up without any messages unless exceptional conditions occur, e.g the clock is in non-set state, or too many I/O devices are operational. In that case the operator is prompted for SET or DEL commands at the system console.

Dynamic Storage Map

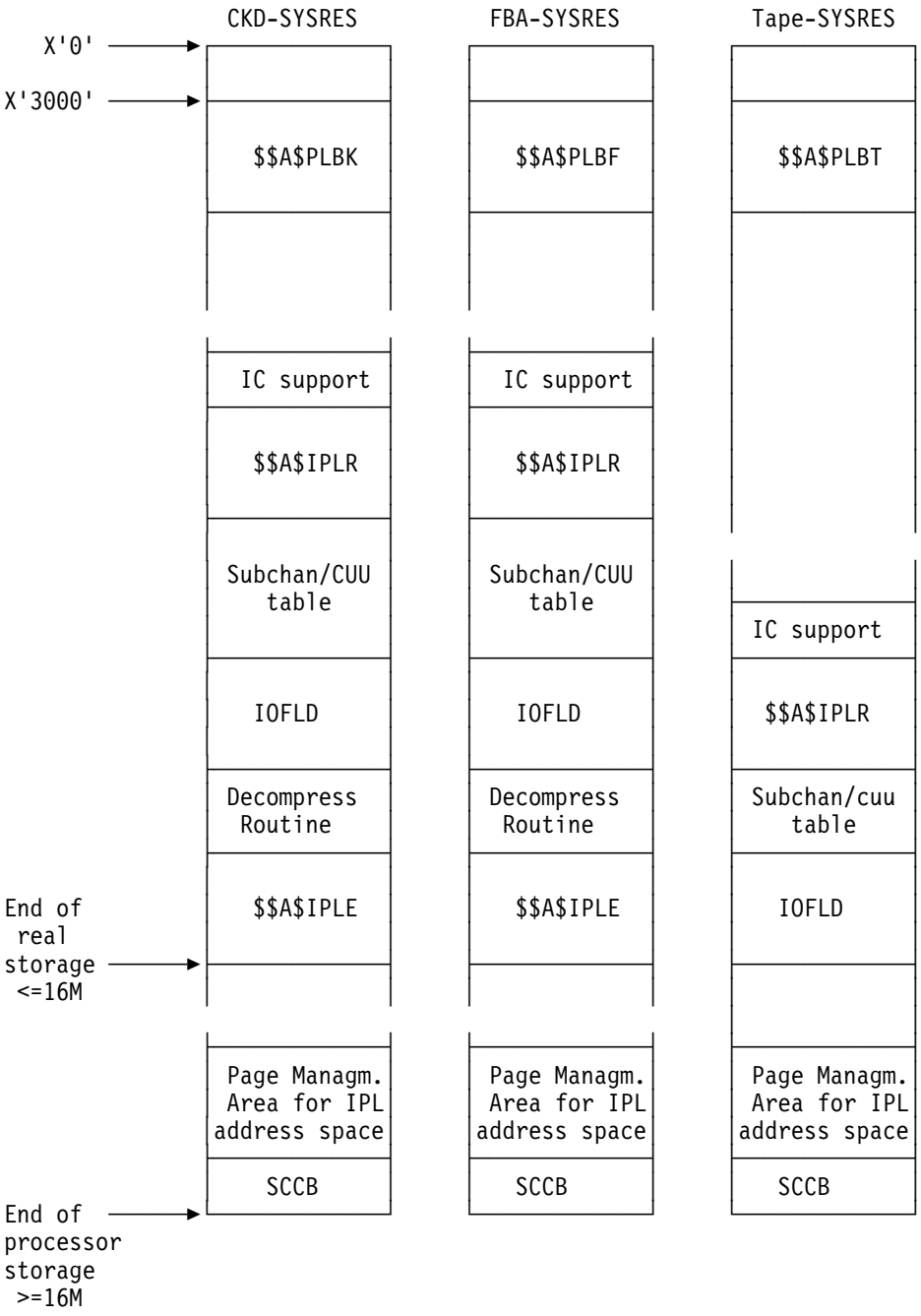
Figure 4 shows how the IPL program is laid out in storage at six different points of the process on a real machine.

```
Step 1: Processor storage when hardware IPL is activated
Step 2: When $$A$PLBF K or T pass control to $$A$IPLR
Step 3: During IPL bootstrap process
Step 4: Before dynamic table allocation in MACIPLR4
Step 5: After dynamic table allocation in MACIPLR4
Step 6: After final supervisor table allocation and first IPL code shift
Step 7: After allocation of virtual storage and IPL code shift to virtual
Step 8: After allocation of 24-bit shared areas
```



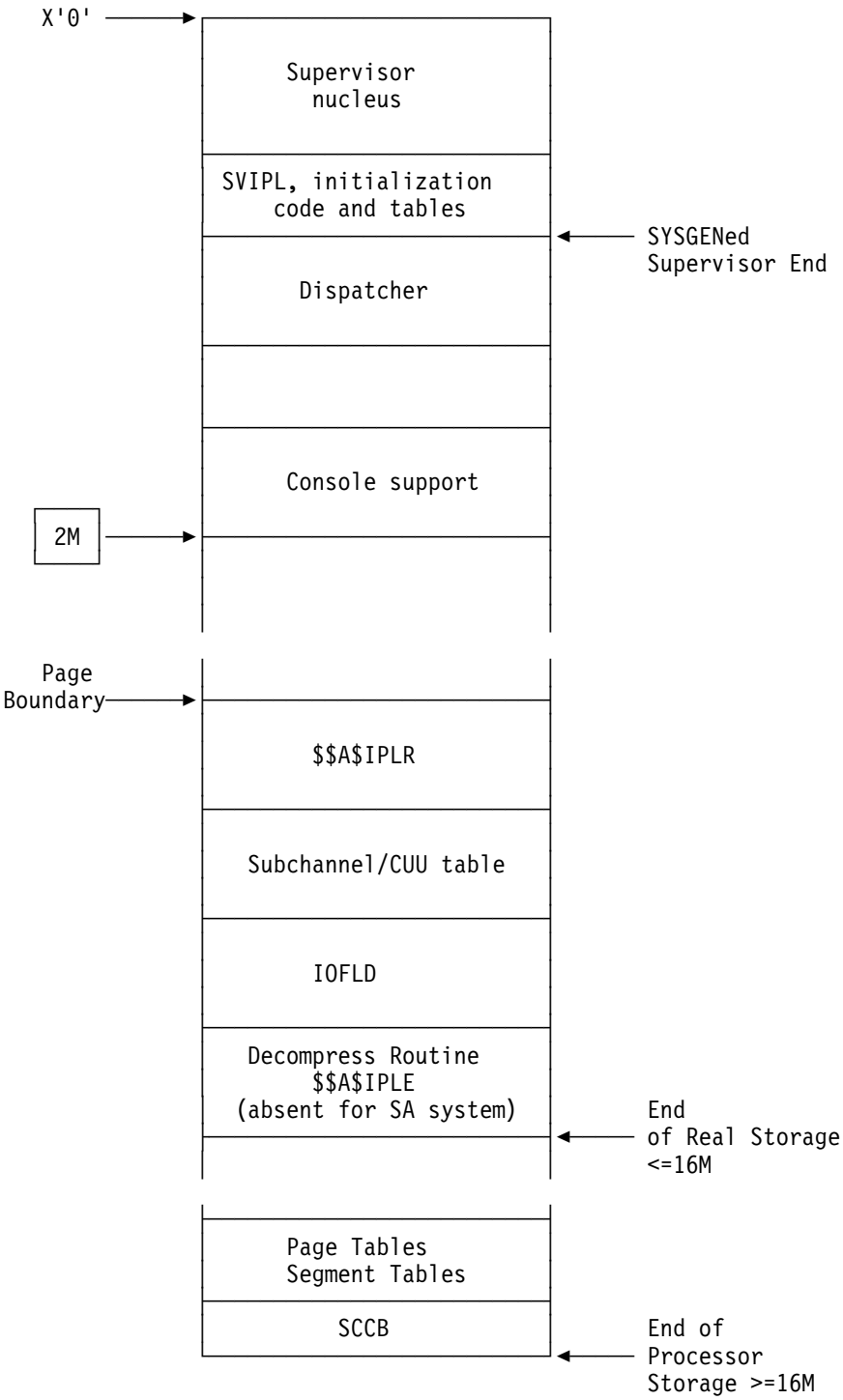

```
Step 1 - Real Storage after Hardware-IPL is activated
```

Figure 4 (Part 1 of 8). IPL Storage Map



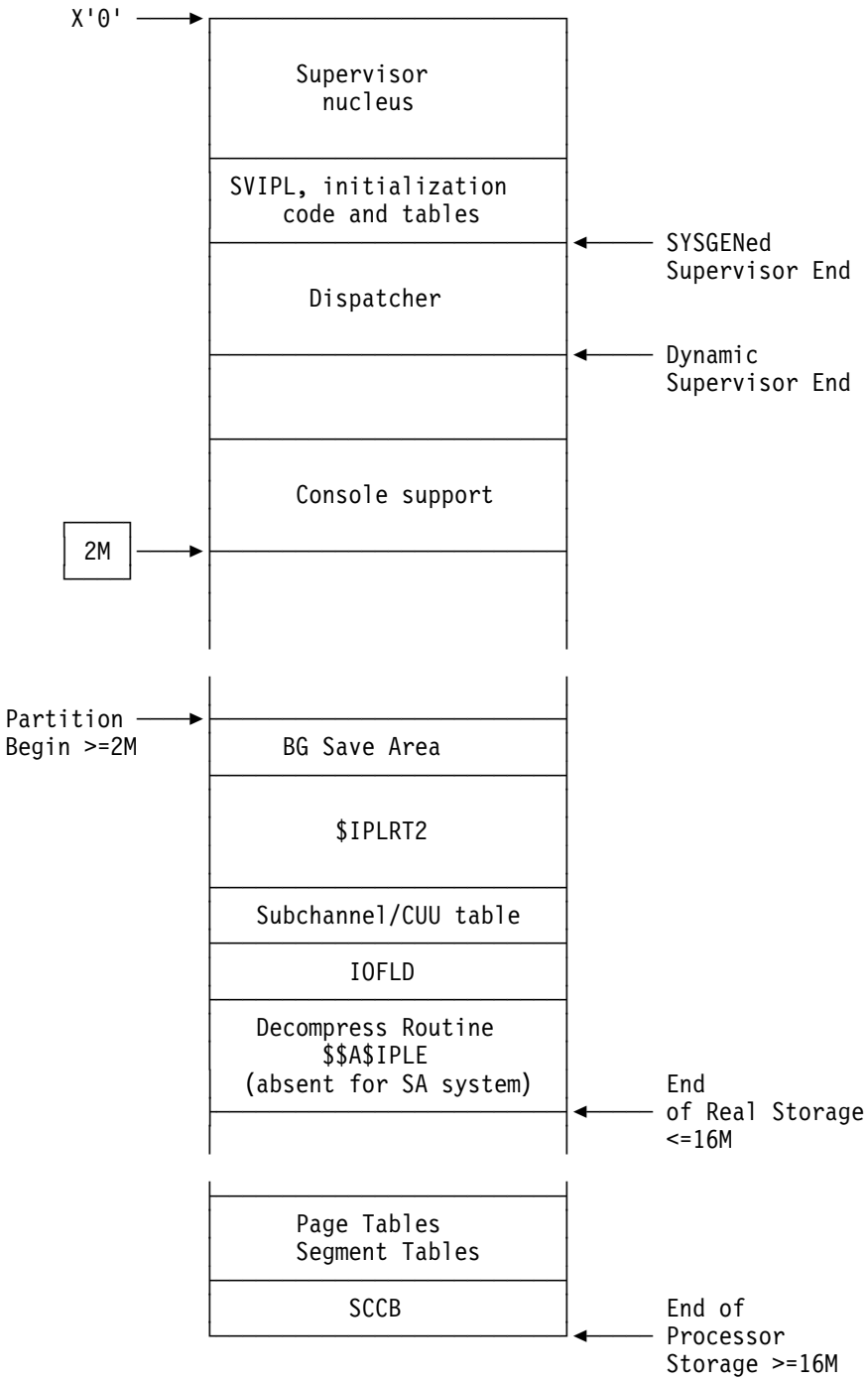
Step 2 - Tranfer of Control to $$A$IPLR (bootstrap processing)

Figure 4 (Part 2 of 8). IPL Storage Map



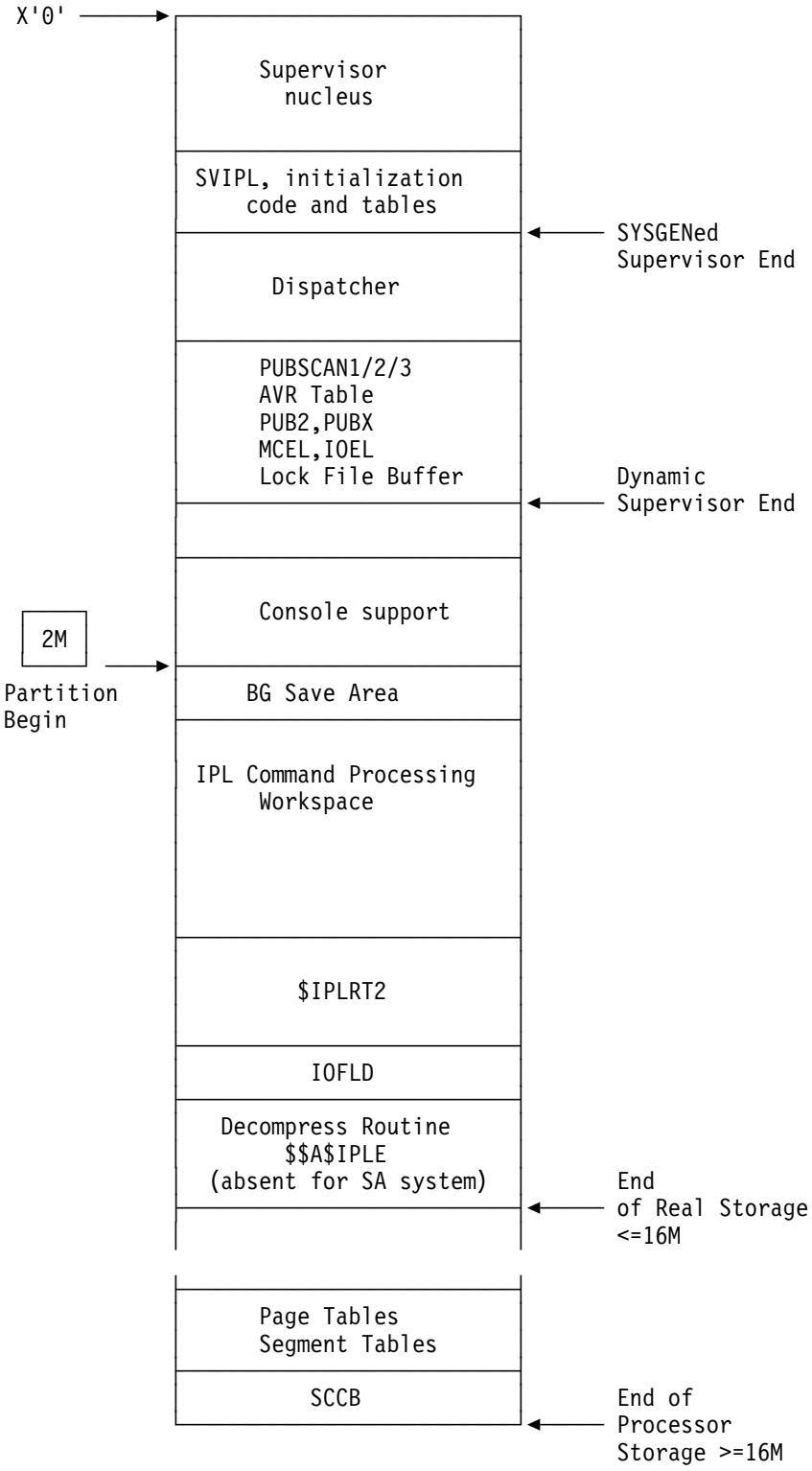
Step 3 - Real Storage at End of IPL-Bootstrap Processing

Figure 4 (Part 3 of 8). IPL Storage Map



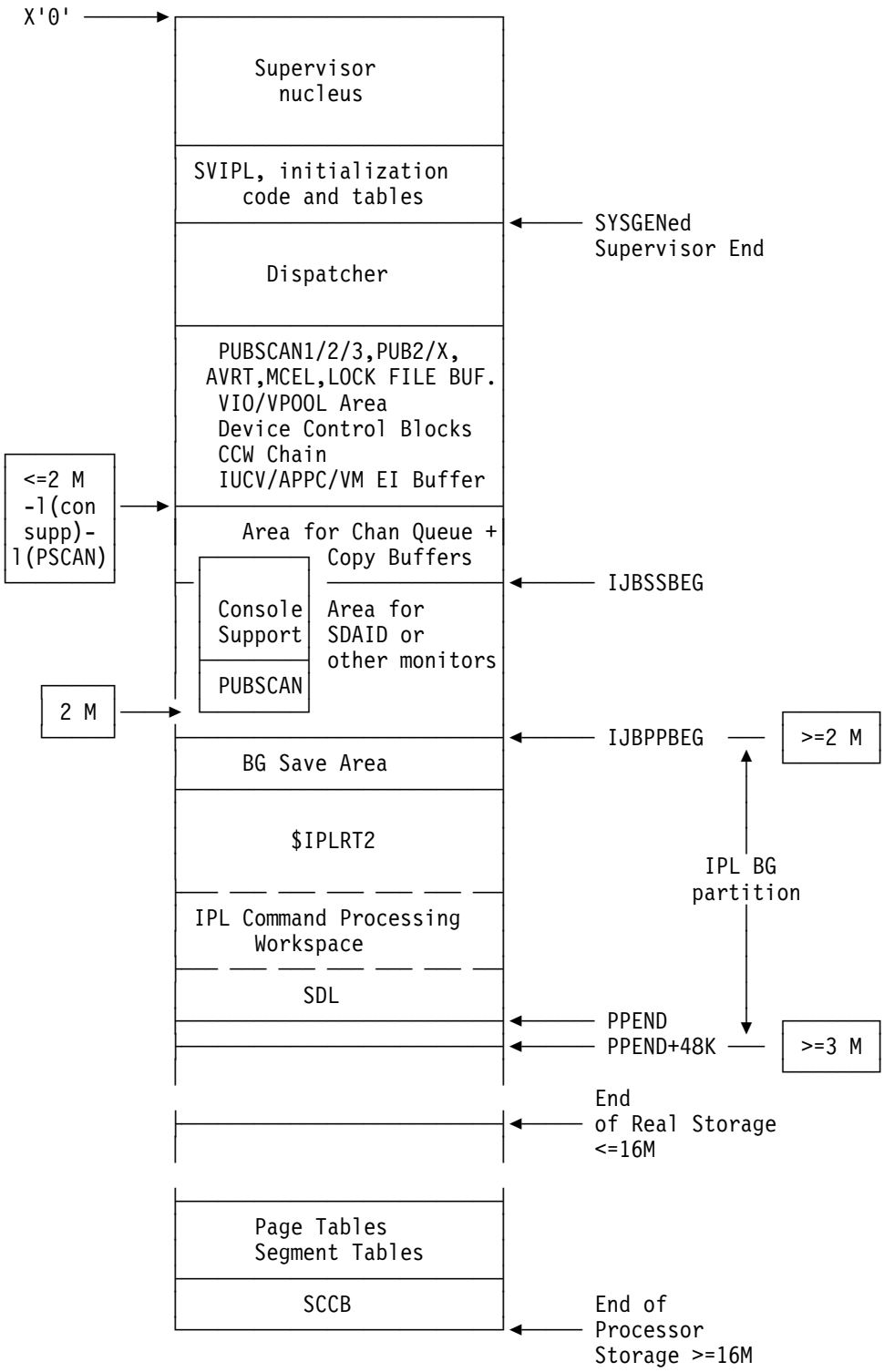
Step 4 - Before Dynamic Table Allocation in MACIPLR4

Figure 4 (Part 4 of 8). IPL Storage Map



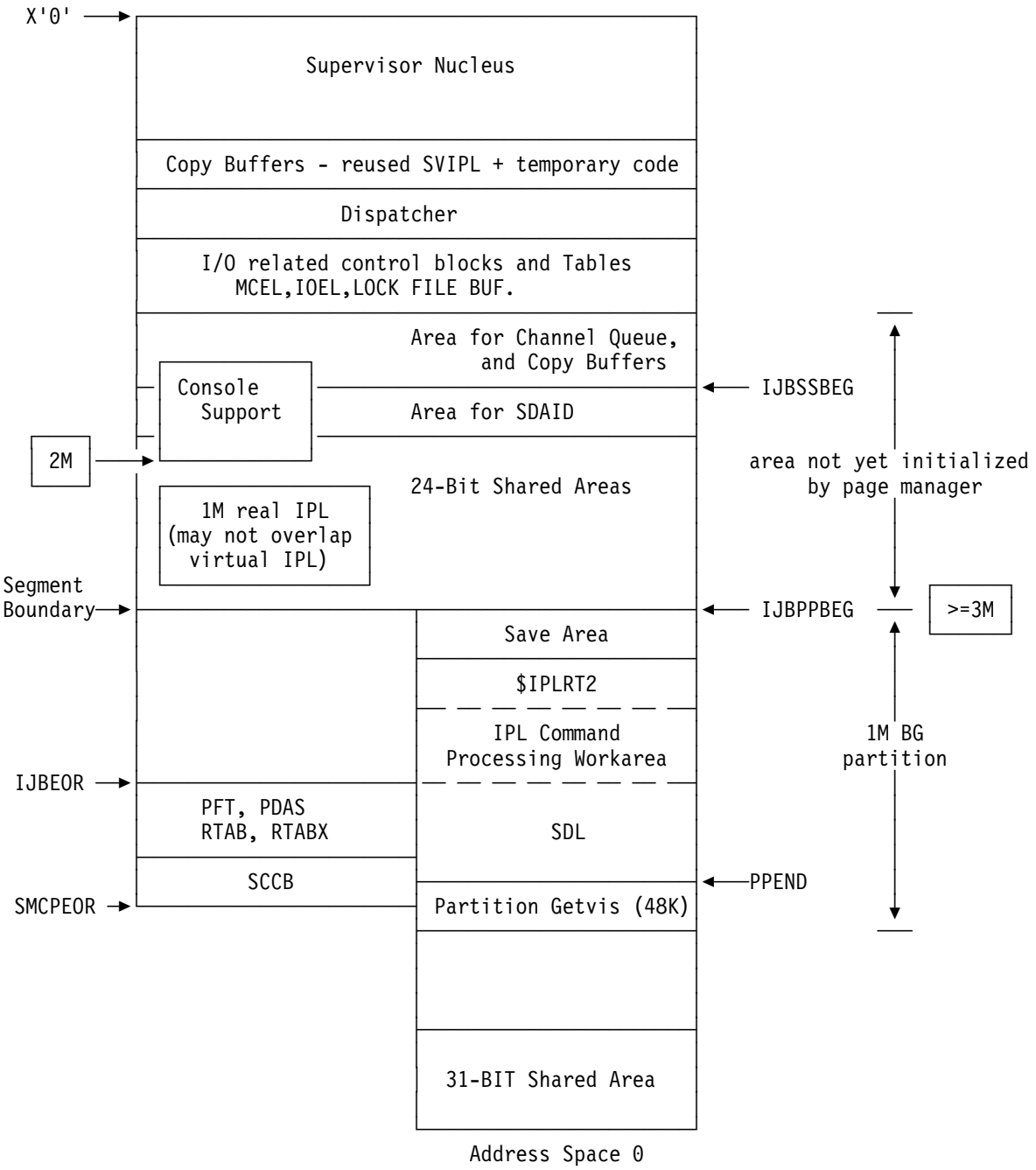
Step 5 - After First Dynamic Table Allocation in MACIPLR4

Figure 4 (Part 5 of 8). IPL Storage Map



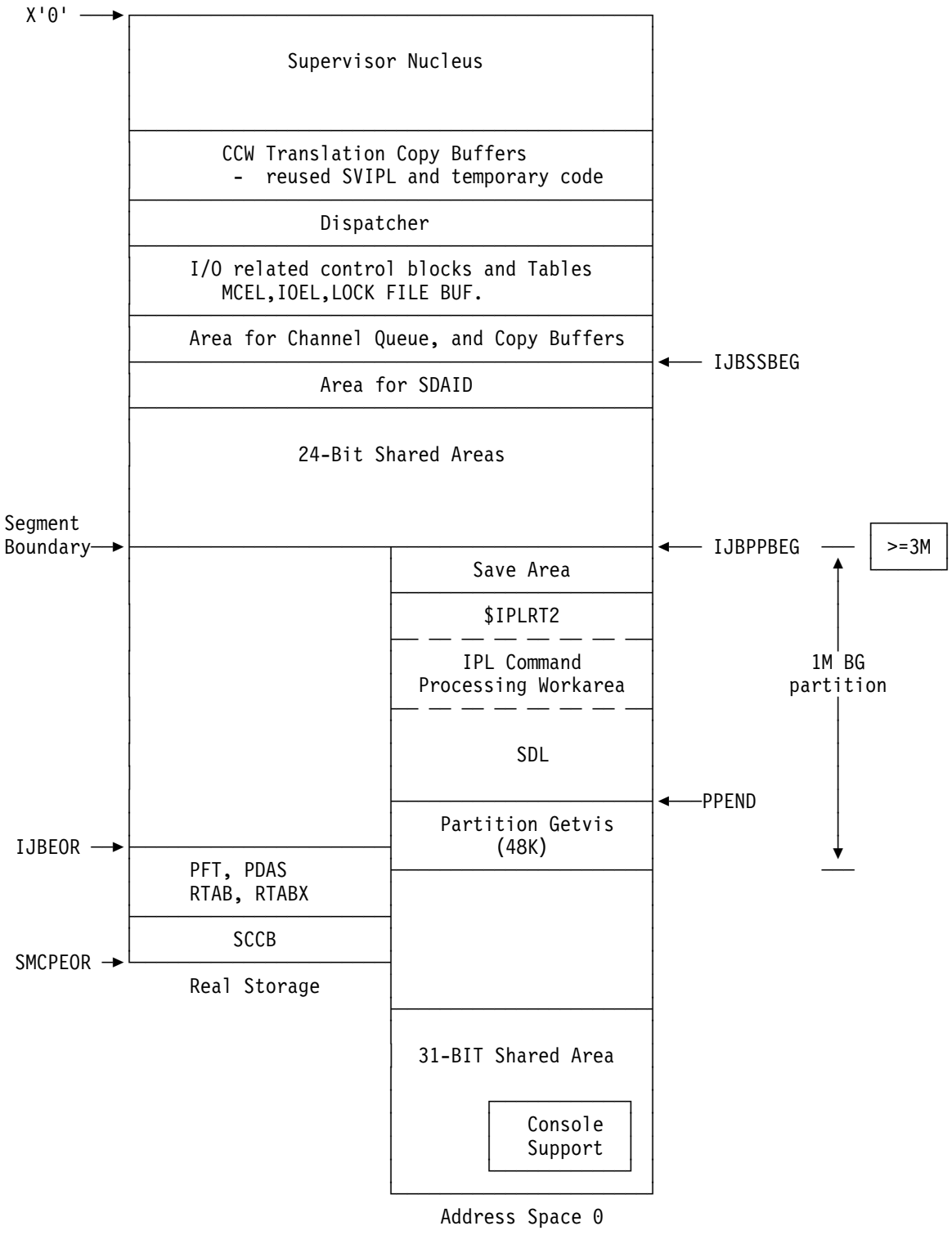
Step 6 - After SDL Preparation in MACIPLR6

Figure 4 (Part 6 of 8). IPL Storage Map



Step 7 - After First Call of $INTVIRT in MACIPLR7

Figure 4 (Part 7 of 8). IPL Storage Map



Step 8 - After Second Call of $INTVIRT in MACIPLR7

Figure 4 (Part 8 of 8). IPL Storage Map

Description of IPL Phases

Every phase is described in a summary, first by listing its important characteristics in the same way as the code does. The last item, however, “Sequence of Operation,” gives an overview of the control flow with the most important labels and indicates where the main exits occur, by means of an arrow (------->). The exits indicate either entries in the same phase, or a branch to another phase.

Phase $$A$IPL0

Module Name: $$A$PLBF

Entry Point: $$A$IPL0

Function: IPL basic bootstrap for an FBA SYSRES by loading a PSW and four CCWs.

Called By: Hardware

Phases Called: $$A$PLBF

Data Areas Used: None

Messages: None

Input: SYSRES device number in the load frame on the screen display.

Output: Phase $$A$PLBF loaded into location X'3000'.

Exit Normal: to $$A$PLBF

Exit Error: Hard wait

Register Use: None

Sequence of Operation: The following processes take place for a FBA SYSRES:

Routine	Description
$$A$IPL0:	When load control is activated, the microprogram reads IPLPSW, IPLCCW1, and IPLCCW2 into locations 0 to 23 of main storage from block 0 on SYSRES.
IPLPSW:	PSW with $$A$PLBF load address.
IPLCCW1:	IPL block 0 (512 bytes) is read into locations 0 to 511, overlaying IPLPSW and both CCWs.
IPLCCW2:	IPLCCW2 is executed via command chaining but serves as a NOP only.
IPLCCW3:	IPLCCW3 prepares the system to read data with a block count of 16, starting at block 2 of SYSRES, thus bypassing the volume label.
IPLCCW4:	CCW4 reads phase $$A$PLBF.
	IPLPSW is made the current PSW and control is transferred to $$A$PLBF.

Phase $$A$IPL1

Module Name: $$A$PLBK

Entry Point: $$A$IPL1

Function: IPL basic bootstrap for a CKD SYSRES by loading a PSW and five CCWs.

Called By: Hardware

Phases Called: $$A$PLBK

Data Areas Used: None

Messages: None

Input: SYSRES device number in the load frame on the screen display.

Output: Phase $$A$PLBK loaded into location X'3000'.

Exit Normal: to $$A$PLBK

Exit Error: Hard wait

Register Use: None

Sequence of Operation: The following processes take place for a CKD SYSRES:

Routine	Description
$$A$IPL1:	When load control is activated, the microprogram reads IPLPSW, IPLCCW1, and IPLCCW2 from disk address 00.00.1 (CC.HH.R) on SYSRES into locations 0 to 23 of the main storage.
IPLPSW:	PSW with $$A$PLBK load address.
IPLCCW1:	IPLCCW1 reads record 00.00.2 (CC.HH.R) containing IPLCCW3, IPLCCW4, IPLCCW5 and the SEEK address of $$A$PLBK (SKADR1) into page 0, locations 24 to 63.
IPLCCW2:	SEEK is performed for cylinder 00 track 01.
IPLCCW3:	IPLCCW3, IPLCCW4, IPLCCW5 read phase $$A$PLBK from disk address 00.01.1 on SYSRES into location X'3000' and following. IPLPSW is made the current PSW and control is transferred to phase $$A$PLBK.

Phase $$A$IPL2

Module Name: $$A$PLBT

Entry Point: $$A$IPL2

Function: IPL basic bootstrap for a tape SYSRES by loading a PSW and a CCW.

Called By: Hardware

Phases Called: $$A$PLBT

Data Areas Used: None

Messages: None

Input: SYSRES device number in the load frame on the screen display.

Output: Phase $$A$PLBT loaded into location X'3000'.

Exit Normal: to $$A$PLBT

Exit Error: Hard wait

Register Use: None

Sequence of Operation: The following processes take place for a tape SYSRES:

Routine	Description
$$A$IPL2:	When load control is activated, the microprogram reads IPLPSW and IPLCCWT into locations 0 to 15 of main storage from the first file on tape.
IPLPSW:	PSW with $$A$PLBT load address.
IPLCCWT:	IPLCCWT reads phase $$A$PLBT.
	IPLPSW is made the current PSW and control is transferred to $$A$PLBT.

Phase $$A$PLBF

Module Name: $$A$PLBF

Entry Point: $$A$PLBF

Function: Determines configuration.

(For details see “Sequence of Operation” below.)

Called By: $$A$IPL0

Phases Called: $$A$IPLR

Data Areas Used:

```
IOFLD       -- IPL communication area in high storage
IPLRCOM     -- IPL communication area in the phase
Low storage for hard wait codes
```

Messages: None

Input:

- SYSRES device address cuu stored in fixed location (low storage).
- IPL load parameter
- VOL1 label record
- System library

Output:

- IPLRCOM, a communication area built for $$A$IPLR, mapped into IOFLD
- Phase $$A$IPLE loaded at storage end if ASI is installed. (See section “ASI-IPL Function” above.)
- Decompress routine loaded before $$A$IPLE if ASI is installed.
- IOFLD allocated before compress routine or at storage end if no ASI is installed.
- Subchannel/cuu table as retrieved from the IOCDS.
- $$A$IPLE entry point saved into IOFLD.
- Phase $$A$IPLR loaded before IOFLD.

Exit Normal: to $$A$IPLR

Exit Error: Hard wait (code stored into storage starting at location 0); see also the chapter “Wait Codes for IPL in Low Storage” on page 122.

The codes are:

```
X'C1E2'           Machine check on clear storage
X'07E6'           IPL I/O error
X'F0C9F0F0C1'     Real storage too small = message 0I00A
X'F0C9F0F1C1'     Incorrect SYSRES format = message 0I01A
X'F0C9F0F6C1'     Unknown SYSRES device type = message 0I06A
X'F0C9F0F7C1'     IPL phase $$A$IPLR not found = message 0I07A
X'F0C9F1F4C1'     Service call exceptional condition = message 0I14A
X'F0C9F6F8C1F0F1' Unsupported hardware = message 0I68A, RC=01
```

Register Use:

```
R 8: Link register
R10: Base register
R14: Pointer to IPLRCOM
```

Sequence of Operation Summary: (For a detailed operation description with labels, see phase $$A$PLBK.)

Routine	Description
GETRESC:	Determines SYSRES device type by executing the SENSE-ID CCW and reads the device constants by executing the READ-DEVICE-TYPE CHARACTERISTICS CCW.
CONNECT:	Get library information. Loads phase $$A$IPLE if itself and a member of type PROC and the compress routine exists. Completes communication area with $$A$IPLR (IPLRCOM) ------->$$A$IPLR

Phase $$A$PLBK

Module Name: $$A$PLBK

Entry Point: $$A$PLBK

Function: Determines configuration.

Called By: $$A$IPL1

Phases Called: $$A$IPLR

Data Areas Used:

```
Control blocks
SIDTABLE  -- Table of all CKD-DASDs permitted as SYSRES
IOFLD     -- IPL communication area in high storage
IPLRCOM   -- Communication area in the phase
Low storage for hard wait codes
```

Messages: None

Input:

- SYSRES device address cuu stored in fixed location.
- IPL load parameter
- VOL1 label record
- VTOC format-4 record
- System library

Output:

- IPLRCOM, communication area for $$A$IPLR, mapped into IOFLD.
- $$A$IPLE loaded at storage end if ASI is installed. (See section “ASI-IPL Function” above.)
- Compress routine loaded before $$A$IPLE if ASI is installed.
- IOFLD allocated before compress routine or at storage end if ASI is not installed.
- Subchannel/cuu table as retrieved from the IOCDS.
- $$A$IPLE entry point saved into IOFLD.
- $$A$IPLR loaded before IOFLD.

Exit Normal: to $$A$IPLR

Exit Error: Hard wait (code stored into storage starting at location 0); see also the chapter “Wait Codes for IPL in Low Storage” on page 122.

The codes are:

```
X'C1E2'           Machine check on clear storage
X'07E6'           IPL I/O error
X'F0C9F0F0C1'     Real storage too small = message 0I00A
X'F0C9F0F1C1'     Incorrect SYSRES format = message 0I01A
X'F0C9F0F6C1'     Unknown SYSRES device type = message 0I06A
X'F0C9F0F7C1'     IPL phase $$A$IPLR not found = message 0I07A
X'F0C9F1F4C1'     Service call exceptional condition = message 0I14A
X'F0C9F6F8C1F0F1' Unsupported hardware = message 0I68A, RC=01
```

Register Use:

```
R 8: Link register
R10: Base register
R14: Pointer to IPLRCOM
```

Sequence of Operation for $$A$PLBK, $$A$PLBF and $$A$PLBT: After relocating all address constants in PSWs and CCWs, the program must:

- Determine type of hardware (see description of routine DETHW in chapter "Macro IPLBMAC" on page 42).
- Determine size of real storage (see description of routine CLEARCOR in chapter "Macro IPLBMAC" on page 42).
- Locate and enable all I/O devices (see description of routine CONTINXA in chapter "Macro IPLBMAC" on page 42).
- Determine the SYSRES device characteristics (GETRESC).
- Get library information, load phases $$A$IPLR and $$A$IPLE (CONNECT).

The program saves all necessary information about hardware and SYSRES device characteristics in an area described by IPLRCOM. The start address of this area and the load address of $$A$IPLR are passed to this phase via special registers.

Routine	Description
GETRESC:	Extracts the SYSRES volume serial number from the volume label record or block. Determines the SYSRES device-type via a SENSE-ID instruction. To get the device capacity, performs a READ DEVICE CHARACTERISTICS instruction only if SYSRES is an FBA device. If it is a CKD- device, reads the format-4 record (first physical record) of the VTOC whose start address is extracted from the volume label record. The capacity information from the VTOC format-4 record is found in a device characteristics table to identify the CKD-SYSRES device type only if the SYSRES device does not support the SENSE-ID instruction.
CONNECT:	Reads the system library descriptor. Searches for the SYSLIB index entry. Searches for a member of type PROC in SYSLIB. Searches SYSLIB for phase $$A$IPLE and compress routine if type PROC exists. Loads $$A$IPLE and compress routine. Allocates IOFLD. Loads phase $$A$IPLR. Loads phase $$A$PLBK from SYSLIB to check code level. Rewrites $$A$PLBK to SYSRES fixed address and restarts IPL with current $$A$PLBK version, if executing bootstrap and version from SYSLIB not identical. Maps IPLRCOM into IOFLD. ------->$$A$IPLR

Phase $$A$PLBT

Module Name: $$A$PLBT

Entry Point: $$A$PLBT

Function: Determines configuration.

Called By: $$A$IPL2

Phases Called: $$A$IPLR

Data Areas Used:

```
SIDTABLE    -- Table of all supported tapes
IOFLD       -- IPL communication area in high storage
IPLRCOM     -- IPL communication area in the phase
Low storage for hard wait codes
```

Messages: None

Input:

- SYSRES device number cuu stored in fixed location (low storage).
- IPL load parameter.
- Subsequent phases on tape in their order of execution.

Output:

- IPLRCOM, a communication area built for $$A$IPLR, mapped into IOFLD
- IOFLD allocated at the end of storage.
- Subchannel/cuu table as retrieved from the IOCDS.
- Phase $$A$IPLR loaded before IOFLD.

Exit Normal: to $$A$IPLR

Exit Error: Hard wait (code stored into storage starting at location 0); see also the chapter “Wait Codes for IPL in Low Storage” on page 122.

The codes are:

```
X'C1E2'           Machine check on clear storage
X'07E6'           IPL I/O error
X'F0C9F0F0C1'     Real storage too small = message 0I00A
X'F0C9F0F7C1'     IPL phase $$A$IPLR not found = message 0I07A
X'F0C9F1F4C1'     Service call exceptional condition = message 0I14A
X'F0C9F6F8C1F0F1' Unsupported hardware = message 0I68A, RC=01
```

Register Use:

```
R 8: Link register
R10: Base register
R14: Pointer to IPLRCOM
```

Sequence of Operation Summary: (For a detailed operation description with labels, see phase $$A$PLBK.)

Routine	Description
GETRESC:	Determines SYSRES device type and mode by executing the SENSE-ID CCW.
CONNECT:	Calls tape FETCH to load phase $$A$IPLR.
	Completes communication area with $$A$IPLR (IPLRCOM) ------->$$A$IPLR
SATFCH:	Stand-alone fetch: loads phase directory entries and phases from stand-alone tape ------->code generated by the supervisor macro SATFCH.

Phase $$A$IPLR

Module Name: $$A$IPLR

Entry Point: $$A$IPLR

Function: Common IPL retrieval program.

Called By: $$A$PLBF, $$A$PLBK or $$A$PLBT

Phases Called:

```
$IPLRT2
$$A$CDL0 (Communication device list)
```

If ASI:

```
$$A$IPLE, functions ASIFINIT and ASIFENA
```

Data Areas Used:

```
AVRADR   -- Automatic volume recognition DSECT (macro AVRLIST)
CDLEDS   -- Communication device list entry DSECT ($$A$CDL0)
CHNTBL   -- Channel control table (supervisor)
COMREG   -- Partition communication region DSECT
            (macro MAPCOMR)
DIRECTRY -- Layout of directory entry
DKETLABL -- Diskette label record DSECT
DTSID    -- Layout of SENSE-ID data
IOFLD    -- IPL communication area
IRB      -- Interruption request block
MAPCIRPL -- Parameter list for WRITE/READ request to Integrated Console
MAPCORIP -- Parameter list for initialization call to Console Router
MAPSALPL -- Parameter list for load request to tape fetch (stand alone)
ORB      -- Operation request block
PUB      -- PUB DSECT
PUBX     -- PUB extension
PTE      -- Page table
PMCOM    -- Page manager communication area
SCBADR   -- Space control block
SCHIB    -- Subchannel information block
SGLOWC   -- Supervisor low core
SGSPDT   -- Control block for Service Processor support
SMCB     -- Storage management control block (macro SMCB)
SMCOM    -- Storage management communication area
STE      -- Segment table
SUPAVT   -- Supervisor vector table
SUPAVTEX -- Supervisor vector table extension
SVASVDL  -- SVA directory of system phases
SYSCOM   -- System communication region DSECT
VCTEADR  -- Volume recognition table
VIOCM    -- VIO communication table
XACUUD   -- XA subchannel - CUU table
```

Messages Caused and Issued:

0I00	0J03
0I02	0J04
0I03	0J05
0I04	0J07

0I05	0J08
0I07	0J09
0I08	0J32
0I24	0J34
0I54	0J50
0I68	0J65
0I96	0J77
0J02	

Input: $$A$IPLR communication area IPLRCOM

Output: Modified supervisor areas

Exit Normal: to BEGIN in $IPLRT2

Exit Error: Hard wait (code stored into storage starting at location 0); see also the chapter “Wait Codes for IPL in Low Storage” on page 122.

The codes are:

```
X'07E6'.'cuu'    IPL I/O error on device 'cuu'
X'07E6'.'IC ree'  Integrated console error, return code 'r', error code 'ee'
X'07E6'.'CSrree'  Console router error, return code 'rr', error code 'ee'
X'F0C9F0F0C1'    Real storage is too small = message 0I00A
X'F0C9F0F7C1'    IPL phase $IPLRT2 not found = message 0I07A
X'F0C9F2F4C1'    Unrecoverable I/O error = message 0I24A
X'F0C9F5F4C1'.'phase_name'  Phase not found = message 0I54A
X'F0D1F5F0C1'    Unsupported SYSLOG device = message 0J50A
```

Register Use:

```
R 8: Link register
R10: 1st base register
R11: 2nd base register
R12: 3rd base register
R14: Pointer to IOFLD
```

Sequence of Operation: Since the function of this phase is rather varied and of multiple character, it is shown first in a summary list and then in a more detailed presentation.

Function Summary:

- Analyses IPL load parameter (TESTLP).
- Loads IC support phase $$ACISS and the optional device list $$A$CDL0 (LOADIC).
- Initializes ASI and retrieves device number of system console (REQLOG).
- Initializes Integrated Console if requested (REQIC).
- Initializes CRT or line mode console if not IC requested (REQCRT).
- Requests a supervisor name (REQSUP).
- Extracts supervisor name and options from supervisor parameters command (PSUP, called by routine REQSUP).
- Loads supervisor and dispatcher.
  From tape in stand-alone environment:
  - Gets phase directory (TGETDIR).
  - Calculates phase load point (GETPLDP).
  - Loads phase into storage (TGETEXT).

  From disk:
  - Gets phase desriptor (GETPDESC).
  - Gets phase length to get load point (GETPLEN).

  - Calculates phase load point (GETPLDP).
  - Loads phase into storage (GETPHASE).
- Checks the specified VSIZE, VPOOL, VIO values (OKSUP).
- Initializes supervisor flags and fields (label INITSYS).
- Initializes the channel control table (INICCTAB).
- Calls transient supervisor routine (INITCTLR) to initialize control registers.
- Updates background start and end addresses, and save area address.
- Initializes the paging tables for 16M IPL address space (INITDAT).
- Sets storage key for supervisor and background (label SETKEY).
- Builds IOFLD for command processing (label BLDIOFLD).
- Locates IPL communication device (label BUILD3DS).
- Builds supervisor I/O tables for 3 device system (label B3DSASLU).
- Initializes program management (label FETCHIN).
  - Completes DASD control blocks for SYSRES, calls FETCH initialization.
  - Stand-alone environment: loads tape fetch $IJBTFCH.
- Loads and initializes IPL console support (INITCS).
- Loads IPL command processor and passes control to it (label SVC4).

Routine	Description
$$A$IPLR:	Control is passed to this phase from $$A$PLBF or $$A$PLBK. The addresses in PSWs and CCWs are relocated. Calls subroutine TESTLP to check the IPL load parameter. Message 0I04I is built and default supervisor name is initialized.
SMRT2:	The system sublibrary SYSLIB is searched for the $IPLRT2 phase descriptor and the extracted phase length is saved for load point calculation.
SMRTMAX:	SYSLIB is searched for the phase descriptor of the common VTOC handler (maximum size phase loaded by IPL). The extracted phase length is used for calculation of the program space needed by IPL.
CDL0:	Calls subroutine LOADIC to load phases $$A$CISS and $$A$CDL0.
GETCON:	Calls subroutine REQLOG to extract the device number of the system console from the ASI procedure.
GETCONIC:	Calls subroutine REQIC, if the integrated console is requested as system console (by load parameter).
GETCONL:	Calls subroutine REQCRT to determine the system console, if the Integrated Console is not requested, or could not be initialized.
SUPREQ:	Calls subroutine REQSUP to extract or request the supervisor name and parameters.
FINDSUP:	In stand-alone environment the supervisor and dipatcher are loaded by calling subroutines • TGETDIR to retrieve the phase and get the phase length, • GETPLDP to calculate the load point, and • TGETEXT to finally load the phase. IPL terminates if the standard supervisor or dispatcher are not found.
LOADSUP:	From disk the supervisor and dipatcher are loaded by calling subroutines • GETPDESC to retrieve the phase descriptor, • GETPLEN to extraxt the phase length from the descriptor, • GETPLDP to calculate the load point, and • GETPHASE to finally load the phase.

If the supervisor is not found, routine REQSUP is reentered to issue message 0I03D requesting the name of a valid supervisor.

If the specified dispatcher is not found, the standard dispatcher is loaded. IPL terminates in case of an unsuccesful load.

NOASIW0: Issues message if Turbo Dispatcher is activated.

Calls subroutine OKSUP to process supervisor parameter values.

INITSYS: The SCCB address is placed into the supervisor SGSPDT table.

Indicators are set, signalling that IPL is in progress, running under VM, or LPAR mode.

INITSYSX: Calls subroutine INICCTAB to have the channel control table built.

The supervisor routine INITCTLR is called to initialize the control registers. It resides in the supervisor copy-buffer area and is overlaid with table and buffer space during later processing.

The generated background beginning address and system save area address are updated leaving as much space as possible to the dynamic supervisor area allocation.

The (location X'50') timer is initialized.

Calls routine INITDAT to set up paging tables.

The attention routine is disabled, and write operations on the SYSRES device are enabled.

SETKEY: The storage protect key of the supervisor area is set to 0, while the storage protect key of the background area (the remaining real storage) is set to 1.

BLDIOFLD: Formats an area near the high address end of processor storage for building I/O related tables.

The PUBs, FOCL, NICL, FICL, diskette label record (if a diskette is an IPL communication device), and device specific information about SYSLOG and SYSUSE are stored in this area (described by DSECT IOFLD). The tables will be used by phase $IPLRT2 to complete the system I/O-configuration process (ADD/DEL - MACIPLR3, device sensing - MACIPLR2 and MACIPLR4).

BUILD3DS: The 3-device system is built. If ASI is not in process the system is put into the wait state and the operator has the option of selecting the communication device for IPL.

In stand alone-environment, or for the integrated system console, IPL does not wait, but assumes the system console as communication device.

B3DSASLU: The PUBs built for SYSRES, SYSLOG, and SYSUSE (if SYSUSE is not equal to SYSLOG) are copied into the supervisor PUB table and the background LUBs for SYSRES, SYSLOG, and SYSUSE are assigned to the corresponding PUB table entries. The FOCL (first-on-channel list) and the PUB scan tables are built for this 3-device system. Figure 9 on page 111 in Chapter 4, “Data Areas -- Initial Program Load” on page 103 shows an example of the I/O tables for a 3-device system built by $$A$IPLR.

FETCHIN: In a stand-alone environment tape FETCH $IJBTFCH is loaded into the supervisor area.

FETCHINA: The logical transient area (pointed at from SYSCOM via IJBLTA) contains initialization code for the FETCH system task. A temporary AVR (automatic volume recognition) table is allocated and the volume-characteristics of the SYSRES device placed into this table. This information together with library related information (disk address of library, SYSLIB descriptor, library block size, etc.) are passed to the FETCH initialization routine.

FETCHINE: If IPL restart is indicated in ASI#REST, than $$A$IPLR is reloaded and passed control from the beginning again.

XCTL: FETCH, initialized before, will now be used for program loading via SVC 4. Therefore IPL is enabled for interrupts: A supervisor state PSW with the DAT bit off and I/O and external interrupts enabled is loaded.

Routine INITCS is called to load and initialize the IPL console support. At the end of IPL the same phases will be loaded into the SVA and the support will be reinitialized.

For loading the command processing phase $IPLRT2 and transferring control to it, the SVC 4 and branch register 1 instruction sequence is moved into IOFLD, from where it is executed. Thus it is possible to overlay phase $$A$IPLR with $IPLRT2. The address of IOFLD is passed via general register 7 to $IPLRT2.

Subroutine **Description**

INICCTAB: The channel control table is built. The length of the I/O extended logout area and the machine check extended logout area for this CPU model is calculated.

REQLOG: Normally the cuu of the SYSLOG device is specified on the first command of the IPL ASI procedure. Thus ASI processing is initialized, the default IPL ASI procedure and its first command are retrieved.

REQSUP: In case IPL was performed with the appropriate IPL load parameter on the load panel, message 0I03D is issued to ask the operator for IPL parameters. If any of the parameters is specified incorrectly, or a procedure is not found, message 0I03D requests valid IPL parameters once more.

If one of the IPL parameters was STOP=SUP, then the supervisor parameters command is displayed, together with message 0J05D, and the operator has the opportunity to override the supervisor parameters.

Calls subroutine PSUP, to check the supervisor parameters command.

If the supervisor is not found (later after label LOADSUP), this routine is reentered to issue message 0I03D requesting the name of a valid supervisor.

TESTLP: Calls subroutine TESTLP to analyse the IPL load parameter, and to save the requested options.

LOADIC: Loads phase $$A$CISS which contains the IPL bootstrap support for the integrated console.

If phase $$A$CDL0 is found in the system library, the text of the phase is read into storage; if not, the normal process of system preparation continues.

REQIC: MSG 0I04I is transmitted to the integrated console service $$ACISS. If the request was successful, IPL determines a device number for the Integrated Console and builds a dummy SYSLOG PUB. The internal device number must not be defined in the IOCDS to avoid conflicts with a real device. It is kept until an 'ADD cuu,CONS' command is given.

REQCRT: First MSG 0I04I is transmitted to the SYSLOG device specified in the ASI procedure, and the device type (1050 or CRT) is determined. If the I/O operation was not successful, VSE enters the wait state and waits for an attention interrupt.

The operator has to press REQUEST or ENTER to cause the attention interrupt, which then transmits the SYSLOG device number. If the device is not supported by VSE IPL stores 0J50A in bytes 0-4 and enters the wait state.

$$A$CDL0 contains the communication device list (CDL). The list specifies the system console (SYSLOG) and the I/O device (SYSLOG and/or SYSUSE) that may present inter-

rupts to IPL and thus establish themselves as communication device(s) for transmitting IPL commands.

If $$A$CDL0 is in the system library, the program checks if the SYSLOG device number is in the CDL. If yes or if there is no $$A$CDL0, the SYSLOG device is accepted. Otherwise IPL waits for another SYSLOG device number to be transmitted.

Note: IPL may loop indefinitely if the CDL has been specified incorrectly. Therefore it may be useful, each time that IPL enters the wait state repeatedly, to enter manually the device number for SYSLOG and SYSUSE into locations X'10' to X'17' in the hexadecimal format X'00000cuu00000cuu'. (For device number 000 this interface does not work.) To resume processing press REQUEST/ENTER at SYSLOG.

On acceptance of the SYSLOG device address, a PUB is built and the phase writes message 0I04I.

After having built the PUB, the phase makes SYSLOG available for operator/system communication for the remaining system preparation process; all error messages are written to SYSLOG.

If Phase $$A$CDL0 is found in the system library, the text of the phase is read into storage; if not, the normal process of system preparation continues.

INITCS: Loads the console support from a predefined address (2M) in storage downwards.

Loads phases

- $IJBSPDT, the Integrated Console support, if the Integrated Console is selected as system console,
- $IJBCRT, the CRT console support, if a CRT or line mode console is the system console,
- $IJBxDEF, the appropriate NLS support, which is either $IJBEDEF for English, $IJBGDEF for German, $IJBSDEF for Spanish, or $IJBJSDEF for Japanese,
- $IJBPHCF containing Hard Copy support,
- $IJBCSIW containing the IPL console buffers, and
- $IJBCSIO containing the console router support.

The console router initialization module gets control, to initialize the complete console support, which works with preallocated buffers during IPL.

IPL terminates, if one of the phases could not be loaded, or the initialization is not sucessful.

GETPLDP: The dispatcher load point behind the supervisor is calculated, and stored into the SVASVDL. It is checked whether the phase will fit into processor storage without overlaying IPL.

GETPDESC: The system sublibrary index is scanned for the phase name.

Supervisor name as specified in the (entered on SYSLOG or default) IPL ASI procedure, or as entered on SYSLOG directly.

Dispatcher name as specified in the IPL load parameter or default.

GETPLEN: The length of the phase is extracted from the phase descriptor.

GETPHASE: The CCW chain for loading the phase is constructed in an area in front of module $$A$CISS. The phase is finally loaded.

OKSUP: The specified VSIZE, VPOOL, and VIO values are checked in this sequence for consistency. Messages 0J08I, 0J09I, 0J32I, 0J34I, 0J40I, 0J41A are issued in case of inconsistencies. The specification of VPOOL, and VIO are rounded to 64k. The specification of VSIZE is checked for minimum/maximum, and is finally placed into the supervisor.

PSUP: The listing option LOG/NOLOG is evaluated with LOG as default; the NOPDS and NOMSG options are saved. The VSIZE, VPOOL, and VIO specifications are syntax-checked and saved for later evaluation.

If a specification is invalid, message 0I96I is displayed. Then routine REQSUP is reentered to issue message 0I03D requesting valid supervisor parameters.

INITDAT: The following tables are initialized for the IPL address space of 16M:

- page table
- segment table

SATFCH: Tape FETCH for stand-alone environment. Called by IPL subroutines TGETDIR AND TGETEXT (see macro IPLBMAC).

Phase $$A$IPLE

Module Name: $$A$IPLE

Entry Point: $$A$IPLE

Function: Reads IPL information from procedure library.

Called By: For Functions:

```
$$A$IPLR                    ASIFINIT and ASIFENA
$IPLRT2  (IPLDISK)          ASIFGET
$IPLRT2  (MACIPLR4)         ASIFDEAC
```

Phases Called: None

Data Areas Used:

```
COMREG   -- Partition communication region DSECT (macro MAPCOMR)
IOFLD    -- IPL communication area in high storage
Low storage for hard wait codes and interrupt information
SYSCOM   -- System communication region DSECT
```

Messages: None

Input:

```
ASIFLAG   = flag byte
ASIFUNC   = AL1(function-code)
ASIENTRY  = A($$A$IPLE)
ASIIPLR   = A($$A$IPLR)
ASIIPLN   = name of IPL procedure
ASIJCLN   = name of JCL procedure
ASIXNPSW  = supervisor External New PSW
```

Output:

```
ASIFLAG   = flag byte
ASIIPLN   = name of IPL procedure
ASIJCLN   = name of JCL procedure
ASIRC     = AL1 (return code)
ASIWORK   = IPL command
IPLRFLG1  = TYPE specification
```

Exit Normal: Return to caller

Exit Error:

```
Hard wait (code X'07E6' -- IPL I/O error)
Return to caller
```

Register Use:

```
R 6: Return register
R 8: Link register
R10: 1st base register
R11: 2nd base register
R14: Pointer to IOFLD
```

Sequence of Operation:

When the phase is called for the first time, it executes the necessary relocations and sets the IPL and JCL default procedure names.

The function performed by the phase is determined by the function code passed to this phase by the calling phase in ASIFUNC. The codes signify:

```
ASIFINIT   =   0 -- Initialize ASI-IPL

ASIFCHCK   =   4 -- Check whether the procedure named
                    in ASIIPLN exists

ASIFGET    =   8 -- Get next procedure record

ASIFENA    =  12 -- Enable external interrupts for
                    an ASI restart

ASIFDEAC   =  16 -- Disable external interrupts for
                    an ASI restart
```

These functions are provided at the following labels:

Routine	Description
INITASI:	Initially the compress routine is loaded. Then the system sublibrary SYSLIB is scanned for the ASI master procedure $ASIPROC. This procedure defines what procedures should be used for IPL and job control depending • on the processor (CPU-ID) installed and/or the SYSRES DASD type (format 1 of $ASIPROC) or • on the SYSRES DASD type (format 2 of $ASIPROC). See “Description of Master Procedure $ASIPROC” on page 88 for details. The procedure is scanned record by record to find a matching entry. The TYPE specification is evaluated and the information is stored in IPLRFLG1. The IPL and job control procedure names are extracted. If the IPL procedure is found, the library block containing its first record is read. If $ASIPROC or the specified IPL procedure does not exist or no matching processor ID is found, the default IPL procedure ($IPLESA) is looked for and the library block containing its first record is read. ASI is indicated to be in process. The first record is placed into IOFLD to be inspected by the caller. The job control default procedure name ($$JCL) is used if none has been found in $ASIPROC.
CHCKASI:	Checks if there is an entry in the SYSLIB member directory for the specified procedure name and reads it.
GETASI:	Gets next procedure record (ASIFGET) and places it into IOFLD. If there are no more records available, an end-of-data condition is indicated.
ENABLE:	Enables the ASI restart facility: The current PSW is enabled for external interrupts. The external new PSW is saved and replaced by an ASI PSW. Pressing the INTERRUPT key at the console will result in an external interrupt which is

handled by this phase. Only a flag (ASI#REST) is set to indicate that ASI has to be restarted. ASI restart results in reloading $$A$IPLR and passing control back to the beginning of phase $$A$IPLR which will display message 0I03D.

The operator may proceed with ASI by specifying another procedure name for IPL and for job control or terminate ASI and proceed with interactive IPL by specifying a supervisor name.

DEACASI: Deactivates the ASI restart facility (ASIFDEAC) by restoring the external new PSW to its original contents.

JCLASI: In each partition COMREG, an indication (ASIPL) is set into JCSW5 that job control has to proceed with ASI and the job control procedure name is placed into the field COMUSCR. This is a subfunction of routine ASIFDEAC.

ACCPRMR: Subroutine reading one by one the procedure member library blocks and decompressing one by one its records into IOFLD.

DECOMPR: Subroutine to decompress one record out of the library block containing PROC-type records.

IOS: I/O subroutine, either calling IORTN or issuing an EXCP macro, dependent on the supervisor being loaded and initialized successfully.

RESTART: The restart function is activated by pressing the interrupt key on the machine-panel. This transfers control to subroutine RESTART, which turns on the ASI#REST flag and resumes processing.

Note: ASI restart is a means to interrupt the automatic IPL process and use another IPL procedure or perform an interactive IPL.

HAREST: Each time before returning to the caller, $$A$IPLE checks for the ASI#REST bit being set. If it is set, IPL processing is stopped immediately, the current PSW is disabled for external interrupts, $$A$IPLR is re-loaded into its previous locations and control passed back to $$A$IPLR.

Note that the ASI-restart facility is enabled as a subfunction of ASIFINIT and explicitly by calling ASIFENA.

$$A$IPLR will disable the ASI restart facility when reading the supervisor into storage since for that period of time the new PSW area in low storage is overlaid.

Before final allocation of the paging tables and subsequent passing control to IPL command processing, $$A$IPLR checks the ASI#REST bit in IOFLD and if it is set, restarts from the beginning.

Macro IPLBMAC

Macro Name: IPLBMAC

Function: Macro to generate common code and data for IPL phases depending on parameters passed. The macro may be called three times:

- For DSECT generation (GEN=DSECT)
- For code generation (GEN=CODE)
- For generation of ESA I/O routines (GEN=IORTN)

Called By:

```
$$A$PLBK
$$A$PLBF
$$A$IPLE
$$A$IPLR
$IPLRT2
```

Macros Used:

AVRLIST
INLCDENT
INLCDESC
INLCLBCF
INLCLDES
INLCMBRX
INLCSLXE
IRB
MAPCOMR
MAPDNTRY
MAPDTS
MAPLOWC
MAPPTE
MAPPIB
MAPSCB
MAPSSID
MAPSTE
MAPSVIPL
MAPVCTE
ORB
PMCOM
SCHID
SGLOWC
SGSPDT
SMCB
SNSDVIP
SMCOM
SYSCOM
SUPAVT
SUPAVTEX
XACUUD

Input:

```
Name of the calling phase (MOD=$$A$IPLR is default.)
GEN=DSECT or GEN=CODE or GEN=IORTN
```

Output: Generated code, subroutines and DSECTs.

Routine	Description
DETHW:	ESA-hardware is required. If the program does not successfully execute a PALB instruction, IPL terminates with error 0I68A. In addition, the program tries the CUSE and CMPSC instructions and remembers whether these hardware features are installed. For information on the PALB, CUSE and CPMSC assembler instructions, see *IBM System/ESA Principles of Operation.*
TESTVM:	On VM systems the virtual machine definition is cheched: V=V, or V=R, or V=F. If the virtual machine is V=V, DIAGNOSE instructions are used to determine the size of real storage and to clear storage. Otherwise it is treated like native processor storage. See *IBM Virtual Machine/Enterprise Systems Architecture: CP Programming Services* for DIAGNOSE instructions.
CLEARCOR:	On native processors, IPL gets the real storage size by an 'endless-loop' which refers to real storage from the end of phase $$A$PLBK up to the end of real storage. The loop is interrupted by an addressing exception when real storage is exhausted, or it ends at the highest address defined for the architecture (2GB-1). In this loop a TB instruction is used to clear and recover storage. The virtual storage size cannot be known from the hardware. It is specified by the VSIZE operand of the supervisor parameters command. The last page of real storage is reserved for SCLP services.
PMTAB:	Real storage is allocated for the page management tables.
CONTINXA:	All I/O devices are located and enabled. Only one path per I/O device is enabled. A temporary subchannel-cuu table is built (below IOFLD), which is valid, until the final I/O configuration is determined in MACIPLR4.
DOSCLP:	The processor is questioned for hardware features and for the IPL load parameter.
RELOCATE:	Relocates CCWs and PSWs.
CONVERT:	Converts a hexadecimal value to a decimal value.
PROSTOP:	Processes STOP= parameter.
PROTYPE:	Processes TYPE= parameter.
STOPCHK:	Checks whether ASI stop has to be performed for the current command.
KEYSCAN:	Keyword-scan routine for ASI.
CMDESC:	Reads the system library descriptor. Searches for the SYSLIB index entry. Searches for the member descriptor via the member index.
SMDESC:	Searches the member index from higher levels to lower levels for the level one member index entry, i.e. the member descriptor.
SSLXE:	Searches the sublibrary index for the SYSLIB entry.
ACCSLXE:	Searches one library block for the SYSLIB index entry.
ACCLMX:	Scans through the level n index library block and returns either the physical relative byte address (PRBA) of the level n-1 index library block for the searched member (n 1) or the member descriptor (n = 1)
ACCPHMR:	Reads a member of type PHASE into storage.

NXTREC: The current record pointer is set to the address of the next record within the library block in storage.

GETLB: Reads one or more library blocks into storage, truncating the library block control field of the previous library block for multiple library block read.

XPRBA: Converts the physical relative byte address (PRBA) to the physical address CC.HH.R for CKD or block number for FBA SYSRES device.

EXTRVIF: Extracts the pointer to the phase descriptor from the member descriptor.

TGETDIR: Builds the request parameter list and calls tape FETCH to load a phase directory entry from stand-alone tape.

TGETEXT: Builds the request parameter list and calls tape FETCH to load a phase from stand-alone tape.

IORTN: Performs ESA I/O and I/O error checking.
The devices handled are DASDs and tapes supported as SYSRES, and the SYSLOG devices (except the Integrated Console, which communicates by a different protocol).

Exit Error: MNOTE if module name is invalid

Sequence of Operations: The following Figure 5 shows the control flow between the library access sub-routines. A line downwards and from left to right means: 'passes control'. A line from right to left and upwards means: 'returns control'.

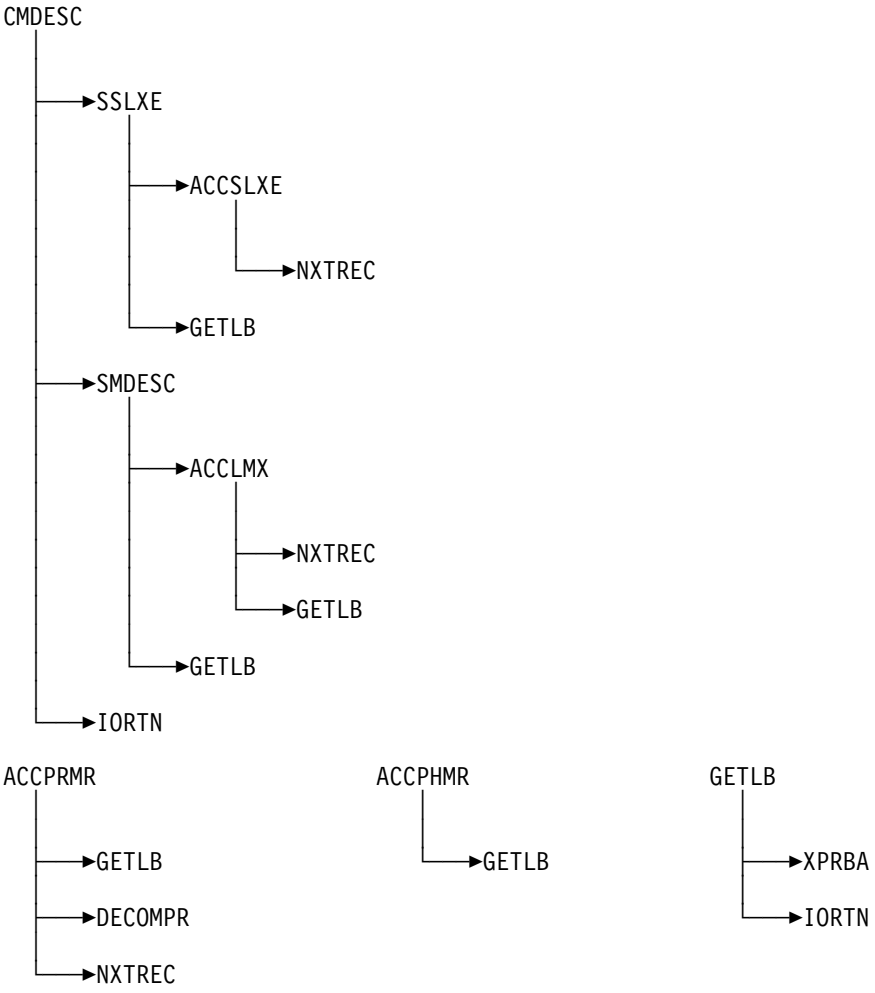


Figure 5. Control Flow: Library Access Routine

Module IJBIPL

IJBIPL serves to generate a link module by calling the following macros, which in turn generate the corresponding command processing phase:

Macro	Phase Generated
IPLDISK	$IPLRT2
MACIPLR2	
MACIPLR3	
MACIPLR4	
MACIPLR5	
MACIPLR6	
MACIPLR7	
IPLUNATT	
MACIPLR8, IPLCGEN	
MACIPLR9, IPLCGEN	
MACIPLRA, IPLCGEN	
MACIPLRB	
MACIPLRC	
MACIPLRD	
MACIPLRE	

Macro IPLDISK

Phase Name: $IPLRT2

Module Name: IJBIPL

CSECT Name: IJBIPLR2

Entry Points:

```
ABNCHK   -- Error message exit
CHANQLIM -- Calculates minimum and default CHANQ value
CHECKSHR -- Checks sharing capability by issueing RESERVE command
DECRTN   -- Converts a field to binary
DOWTO    -- Writes suppressed messages to HC file only
DUPKEYW  -- Error exit for a duplicate keyword specification
FDSRTN   -- Finds next operand in IPL command
            GETSET,GETADD,GETDEL,GETDEV,GETDEF,GETDLA,GETDLF,GETDPD,
            GETSYS AND GETSVA -- Entry points where processing
            for the corresponding commands begins
FETCH4   -- Transfers control to MACIPLR4, allocation of I/O tables
FETCH7   -- Transfers control to MACIPLR7, final IPL processing
ILLCD0   -- Error message exit with previous command buffering
ILLCD1   -- Error message exit
INPCONS  -- Reads from console
INPDKET  -- Reads from diskette
INVCMD   -- Invalid command error exit
INVKEYW  -- Error exit for an invalid keyword specification
INVSEQ   -- Error exit for an invalid command sequence
INVSPEC  -- Error exit for an invalid specification
IOHLD    -- Reads from card reader
IPLTERM  -- Error exit for termination of IPL by hard wait
ISSMSG94 -- Issues message 0I94A and terminates IPL
LOADCHCK -- Checks return codes of LOAD macro and builds
            messages 0I54A or 0I94A if applicable.
LOGSTR   -- Gets command from system console
MISKEYW  -- Error exit for missing keyword
MSGRTN   -- Message writing subroutine
MONITOR  -- Branches to subroutines to read and evaluate
            the command
MONITRSA -- Branches to appropriate command processing routines
            in stand-alone environment
OPRTN    -- Translates commands into internal code
            Branches to one of the following, using the table
            OPTAB:
            GETSET, GETADD, GETDEL, GETDEV, GETDEF,
            GETDLA, GETDLF, GETDPD, GETSYS, GETSVA,
            OR INVCMD.
OPRTN30  -- Secondary entry point in routine OPRTN
OUTP     -- Writes to the console
QSCEIO   -- Quiesce ongoing I/O
READRT   -- Reads commands from the communication device
SETOK3   -- Return point from final Supervisor allocation
            to process the SVA command
SYSMVC   -- Generalized move routine
```

Function: IPL common routines:

- Branches to the proper command processing routine.
- Contains subroutines used in common by MACIPLR3 through D.

Called By: MACIPLR2

Phases Called: $$A$IPLE

Data Areas Used:

```
COMREG   -- Partition  communication region
DIRECTRY -- SDL directory entry DSECT (macro MAPDNTRY)
DVCLST   -- List of supported devices (macro VSEDVLST)
            Each entry contains:
            a return displacement used by MACIPLR3,
            a field for:
            the device type code,
            the channel scheduler flags,
            the PUB device type code,
            the length of sense information,
            the PUBX flag
IJBSSID  -- Subsystem ID DSECT (macro MAPSSID)
IOFLD    -- IPL communication area
Low storage for hard wait codes
NONBLANK -- Translate table for non-blank characters
OPTAB    -- Operation code table containing
            an internal command code and a branch index for
            each valid IPL command and one for an
            invalid command
SVICOM   -- IPL to SV communication area DSECT (macro MAPSVIPL)
SYSCOM   -- System communication region DSECT
TRTTAB   -- Translate table to convert command fields into
            internal code
VMAPCRES -- APPC/VM resource table DSECT
```

Messages Caused and Messages Issued:
This CSECT issues, besides the messages caused by its own code, all messages caused by more than one command processor Unique messages are issued by the command processor itself.

0I10	0I87
0I11	0I88
0I18	0I89
0I21	0I94
0I29	0J05
0I36	0J10
0I47	0J14
0I49	0J19
0I50	0J23
0I54	0J24
0I60	0J26
0I64	0J49
0I65	0J66
0I86	

Input: All IPL commands read from the IPL communication device (disk - ASI procedure, printer-keyboard, card-reader, diskette).

Output: None

Exit Normal: to MACIPLR7 to finish command processing

Exit Error: IPLTERM to terminate IPL by a hard wait

Register Use:

```
R 7: Pointer to IOFLD
R 8: Pointer to background COMREG
R14: Base register of common code (IPLDISK)
R15: Base register for command processing routines
```

Sequence of Operation:

Routine	Description
MONITOR:	Processes the IPL commands.
	Issues error messages via error exits.
	Reads ADD, DEL, DEV, SET, DEF, DLA, DLF, DPD, SYS, and SVA commands from SYSUSE. *For blank input*: ------->MONITOR Else
GETADD:	*For ADD* ------->IJBIPL3 (MACIPLR3)
GETDEL:	*For DEL* ------->IJBIPL3 (MACIPLR3)
GETDEV:	*For DEV* ------->IJBIPL3 (MACIPLR3)
GETSYS:	*For SYS* ------->IJBIPLB (MACIPLRB)
GETSET:	*For SET* ------->IJBIPL4 (MACIPLR4)
	For SET ------->IJBIPL4 (MACIPLR4)
	For SET ZONEDEF/ZONEBDY ------->IJBIPLE (MACIPLRE)
	For APPC/VM SET ------->IJBIPLD (MACIPLRD)
	For DEF, DLA, DLF, DPD: Executes 1st part of dynamic storage allocation if it has not already taken place. ------->IJBIPL4 (MACIPLR4)
	Else goes directly to:
FETCH51:	*For DEF* ------->IJBIPL5 (MACIPLR5)
	For DLA ------->IJBIPL8 (MACIPLR8)
	For DLF ------->IJBIPLA (MACIPLRA)
	For DPD ------->IJBIPL9 (MACIPLR9)
GETSVA:	
	Performs 2nd part of dynamic storage allocation. ------->IJBIPLC (MACIPLRC)
	For SVA ------->IJBIPL6 (MACIPLR6)
FETCH7:	
	Performs final IPL processing. ------->IJBIPL7 (MACIPLR7)
MONITRSA:	Controls command processing in stand-alone environment
	For DEL ------->IJBIPL3 (MACIPLR3)
	For SET ------->IJBIPL4 (MACIPLR4)
	Performs 1st part of dynamic storage allocation.

------->IJBIPL4 (MACIPLR4)

Performs 2nd part of dynamic storage allocation.

------->IJBIPLC (MACIPLRC)

Performs SDL preparation

------->IJBIPL6 (MACIPLR6)

Performs final IPL processing.

------->IJBIPL7 (MACIPLR7)

Macro MACIPLR2

Phase Name: $IPLRT2

Module Name: IJBIPL

CSECT Name: IJBIPL2

Entry Point: Begin

Function: Initialization for IPL command processing. The code of this CSECT:

- Checks the diskette header label.
- Checks for TOD support.
- Initializes RAS linkage area and RAS table.
- Detects double bit ECC failures and tries to reconfigure.
- Invokes Device Sensing (Phase $SENSDEV).
- Initializes CVT.

Called By: $$A$IPLR

Phases Called:

```
$SENSDEV for device sensing
```

Data Areas Used:

```
COMREG   -- Partition communication region DSECT (macro MAPCOMR)
CVTMAP   -- Common vector table - MVS (macro CVT)
DKETLABL -- Diskette label record DSECT
DIRECTRY -- Directory entry DSECT
ECVT     -- Common vector table extension - MVS (macro IHAECVT)
IJBSSID  -- Subsystem ID DSECT (macro MAPSSID)
IOFLD    -- IPL communication area
RASLADR  -- RAS linkage area DSECT (macro MAPRASL)
RASTAB   -- RAS table (macro MAPRTAB)
SVICOM   -- IPL to SV communication area DSECT (macro MAPSVIPL)
SYSCOM   -- System communication region DSECT
MAPDTS   -- SENSE-ID information (DSECT)
MAPPUB   -- PUB DSECT
SNSDVIP  -- Interface control block to $SENSDEV
```

Messages Caused:

0I17	0I66
0I18	0J01
0I30	0J47
0I31	0J66
0I32	

Messages Issued: by IPLDISK

Input:

```
CVTMAP
DKETLABL
ECVT
IOFLD
COMREG
RASTAB
SVIPL
RASLADR
SYSCOM
```

Output:

```
CVTMAP
ECVT
IOFLD
COMREG
RASTAB
RASLADR
SYSCOM
```

Exit Normal: to MONITOR in IPLDISK

Exit Error: to IPLTERM in IPLDISK

Register Use:

```
R 7: Pointer to IOFLD
R 8: Pointer to background COMREG
R14: Base register of common code (IPLDISK)
R15: Base register of command processing routine
```

Sequence of Operation:

Routine	Description
$IPLRT2:	Is loaded by $$A$IPLR which calculates the load point below phases and tables to be preserved for command processing.
BEGIN:	Initializes SYSCOM and COMREG base registers.
	Displays some initial messages.
DISKT:	Checks the diskette header label.
TOD:	Checks for TOD support, stores the TOD clock and, depending on the state of the clock, displays messages on SYSLOG.
	Displays the date, time of day, and zone on SYSLOG if the clock is in the 'set' state.
	Sets a switch if the clock is in the 'error' or 'not-set' state.
RASRTN:	Initializes RAS linkage area and RAS table.
	Detects double bit ECC failures and tries to reconfigure.
	Initializes fields and flags in the CVT and ECVT ------->MONITOR
DETIO:	Prepare automatic device identification via device sensing if TYPE=SENSE or TYPE=INSTALL is specified; call Phase $SENSDEV. Auto ADD of SYSLOG and SYSRES device.

Macro MACIPLR3

Phase Name: $IPLRT2

Module Name: IJBIPL

CSECT Name: IJBIPL3

Entry Point: IJBIPL3

Function:

- Adds one or more devices to the system.
- Deletes one or more devices from the system.
- Displays a list of all devices of the system.

Called By: IPLDISK

Phases Called: None

Data Areas Used:

```
CRTSAV  -- DOC save area
CRTTAB  -- DOC table
IOFLD   -- IPL communication area
IJBSSID -- Subsystem ID DSECT (macro MAPSSID)
SYSCOM  -- System communication region DSECT
DVCLST  -- Device list for ADD and DEL (macro VSEDVLST)
PUBFLD  -- Temporary PUB table in IOFLD
SIDFLD  -- Area for SENSE-ID information in IOFLD
```

Messages Caused:

0I09	0I16
0I12	0J22
0I13	0J48
0I15	0J71

Messages Issued: None. The messages caused by this command processor are issued by common code in IPLDISK.

Input:

```
ADD, DEL or DEV commands
IOFLD
DVCLST
PUBFLD
SIDFLD
```

Output: Modified data areas:

```
IOFLD
PUBFLD
SIDFLD
```

Exit Normal: to MONITOR in IPLDISK

Exit Error: to error exits in IPLDISK

Register Use:

```
R 7: Pointer to IOFLD
R 8: Pointer to background COMREG
R14: Base register of common code (IPLDISK)
R15: Base register
```

Sequence of Operation:

Routine	Description
ADDRTN:	Evaluates ADD command operands and adds the PUB(s) to the temporary PUB table. ------->MONITOR (IPLDISK)
ADDCHCK:	Handles ADD command for automatically identified devices. Issues message 0I16I if incorrect device type specified.
DELRTN:	Evaluates DEL command operands and deletes the PUB(s) from the temporary PUB table. ------->MONITOR (IPLDISK)
DELCHCK:	Handles DEL command for automatically identified devices. Issues messages 0J48D.
DELSID:	Deletes a SENSE-ID (SID) entry from SID table (SIDFLD).
BLDPUB00:	Adds a PUB to the temporary PUB table and a SENSE-ID (SID) entry to the SID table (SIDFLD).
FNDTYP:	Uses routine 'DVCLST' in $IPLRT2 to identify a device specified in an ADD or DEL command. Returns control to the calling routine.
R3DEVRTN:	Evaluates DEV command and displays a list of all devices of the I/O configuration. The devices may be sensed and/or ADDed. This routine is also called from the final device allocation routine in MACIPLR4 (BALR R6,R10) in case more devices are present than supported by the supervisor. ------->MONITOR (IPLDISK)

Macro MACIPLR4

Phase Name: $IPLRT2

Module Name: IJBIPL

CSECT Name: IJBIPL4

Entry Point: IJBIPL4

Function:

- Processes the SET command, sets date and clock.
- Passes APPC/VM SET command back to IPLDISK.
- Invokes Phase $SENSDEV to verify a 'added' I/O configuration (TYPE=NORMAL).
- Checks if more devices present than supported. Displays device list and prompts for DEL commands in case of too many devices.
- Invokes Phase DTRIVLD to check whether any duplicate VOLID exists in the configuration (TYPE=INSTALL).
- Invokes Phase DTRICONF to check whether a minimal configuration has been identified via device sensing (TYPE=INSTALL).
- Dynamically allocates supervisor tables at the end of ADD and DEL processing, as signaled by the reading of either a DEF, DLA, DLF, DPD, or SVA command. Final allocation takes place right before SVA processing by MACIPLRC.
  - Builds and allocates final PUBSCAN2 and PUBSCAN3 tables.
  - Creates permanent PUBs for SYSRES, SYSLOG, and the communication device.
  - Moves PUBs and LUBs from area IOFLD into the supervisor.
  - Builds and allocates the AVR table, the PUB2 table, and the PUB extension (PUBX). Moves SENSE-ID (SID) entries into PUBX.
  - Builds CCW chains for DASDFP.
- Allocates the machine check extended logout area, the I/O extended logout area, and the lock file buffer.
- Determines and sets temporary address of background partition.
- Activates clock comparator.

Called By: IPLDISK

Phases Called:

- $$A$IPLE if ASI was specified
- $SENSDEV if TYPE=NORMAL was specified
- DTRICONF if TYPE=INSTALL was specified
- DTRIVLD if TYPE=INSTALL was specified

Data Areas Used:

```
CCW chains for DASDFP
CHNTBL  -- Channel control table
COMREG  -- Partition communication region DSECT (macro MAPCOMR)
DLFADR  -- DLF table DSECT (macro MAPDLF)
DVCLST  -- Used to build permanent PUBs for SYSLOG and SYSRES
IOEL    -- I/O extended logout area
IOFLD   -- IPL communication area
PUBFLD  -- Temporary PUB in IOFLD
SIDFLD  -- Area for SENSE-ID information in IOFLD
LUBTAB  -- LUB table
MCEL    -- Machine check extended logout area
PUB scan tables
PUBTAB  -- PUB table
PUBX    -- Error entry table entry DSECT
PUB2ADR -- PUB2 table entry DSECT (macro MAPPUB2)
RASLADR -- RAS linkage area DSECT (macro MAPRASL)
SVICOM  -- IPL to SV communication area DSECT (macro MAPSVIPL)
SYSCOM  -- System communication region DSECT
```

Messages Caused:

0I19
0I23
0I29
0I35
0I59
0I67
0I71
0I86
0I87
0I88
0I89
0J10
0J46

Messages Issued:

None. The messages caused by this routine are issued by common code in IPLDISK.

Input:

```
SET command
Data areas to be updated, see "Function" above.
```

Output: Updated data areas, see “Function” above.

Exit Normal:

```
From SET processing: to MONITOR in IPLDISK
From allocation: to FETCH51 in IPLDISK
```

Exit Error: to error exits in IPLDISK

Register Use:

```
R 7: Pointer to IOFLD
R 8: Pointer to background COMREG
R 9: Pointer to SYSCOM
R13: 2nd base register
R14: Base register of common code (IPLDISK)
R15: 1st base register
```

Sequence of Operation:

SET Processing:

Routine	Description
SETRTN:	Reroutes APPC/VM SET to MACIPLRD ------->GETSYSX (IPLDISK)
	Reroutes SET ZONEDEF and SET ZONEBDY to MACIPLRE ------->GETSYSZ (IPLDISK)
	Sets the system date. Sets the system zone if required. Resets the contents of the TOD clock, if required.
DATIMERT:	This subroutine converts the date, clock and zone specifications to a 64 bit binary value. This value represents the number of clock units (that is, microseconds times 2 to the 12th power) elapsed since January 1, 1900 at 0.00 a.m. GMT, and is used to set the TOD clock. ------->MONITOR (IPLDISK)

Supervisor Allocation I: This part is executed before the first DEF, DLF, DLA, DPD, or SVA command is processed.

Routine	Description
PASS1BEG:	Entry label to routines doing first part of supervisor allocation
UPDIO:	Calls PUBCHCK for checking the number of devices in the I/O configuration.
	Invokes phase $SENSDEV for verifying the 'added' I/O configuration if TYPE=NORMAL is specified.
	Invokes phase DTRICONF for checking whether the minimal configuration for a VSE/SP installation process has automatically identified (TYPE=INSTALL).
	Invokes phase DTRIVLD for checking whether duplicate VOLIDs are existing in the configuration (TYPE=INSTALL).
	Issues the message 0J47A if a device does not support the SENSE-ID command; the user has to enter a ADD command (TYPE=SENSE).
NODEAC:	Checks the system assignments for SYSRES, SYSLOG and the communication device (SYSUSE) and makes them permanent.
	The system I/O tables are moved from their temporary location in high real storage (IOFLD) to their permanent location in the supervisor area. This move overlays the 3-device IPL I/O tables that were built by $$A$IPLR.
	Stores SYSRES PUB address into SYSCOM and the first PUB of each channel into the channel queue.
BPUBSCN:	Calculates the length of tables to be allocated in order to check whether they fit. The tables must not overlap the IPL console support. The BG address is initialized to a predefined address behind the console support.
BLDTABL:	Builds and allocates the PUBSCAN tables.
DVCDNRTN:	Scans the PUB table for DASD devices. If a non-operational DASD device is found, the PUB is set to indicate “device down” and the LUB is unassigned. Moreover the total number of PUB entries, the number of DASDs and the number of AVR entries are saved.
DYNALLOC:	Allocates space in the dynamic supervisor area for the following tables: • PUBSCAN2 and PUBSCAN3 • Automatic volume recognition table (AVRTAB) • PUB2 table

- PUB extension (PUBX)
- Machine check extended logout area
- I/O extended logout area
- Lock file buffer

Initializes the following tables:

- PUBSCAN2 and PUBSCAN3
- AVRTAB
- PUB2 index table
- PUBX table

Calls DUPVLD to check for duplicate VOLIDs.

Calls SETKEY0 to set the storage key according to the allocation and update PIB with the BG start address. ------->FETCH51 (IPLDISK)

Additional Subroutines:

Routine	**Description**
ADREST:	Establishes the address for the date subfield.
BLDAVR:	Builds an VCTE table.
BLDPUBX:	Allocates and initializes PUB extension.
BLDP2T00:	Builds PUB2 tables.
CALCPXL:	Calculates length of PUB extension (PUBX).
CALCP2L:	Calculates PUB2 table length.
CONSCHCK:	Checks if system console is integrated console. Prompts for an ADD command in case no PUB exists for integrated system console (dummy PUB required).
DATIMERT:	Calculates the clock value.
DEVFF:	Prompts a user to ADD a device which can not be identified by device sensing (TYPE=SENSE).
DUPVLD:	Loads phase DTRIVLD to check for duplicate VOLIDs (TYPE=INSTALL).
MINCONF:	Loads phase DTRICONF to check for minimum I/O configuration. (TYPE=INSTALL).
PBFRTN:	Searches the PUB table for a specified PUB.
PUBCHCK:	Checks if generated PUB is large enough for all devices. Issues message 0J74D, calls DEV command routine in MACIPLR4 to display a list of all devices, and prompts for DEL commands in case of too many devices.
SETKEY0:	Sets the BG storage key and updates save area address in BG PIB.
TMSRTN:	Converts a subfield to decimal.
UIOLOAD:	Loads phases (e.g. DTRIVLD) behind $IPLRT2. Checks if phase to be loaded fits below IOFLD, and checks load conditions.
ZONERT:	Calculates the zone value.

Macro MACIPLR5

Phase Name: $IPLRT2

Module Name: IJBIPL

CSECT Name: IJBIPL5

Entry Point: IJBIPL5

Function: Processes the DEF command, that is:

```
Assigns the VSAM catalog SYSCAT.
Assigns the recorder file SYSREC.
```

Called By: IPLDISK

Phases Called: None

Data Areas Used:

```
AVRADR -- Automatic volume recognition DSECT (macro AVRLIST)
COMREG -- Partition communication region (macro MAPCOMR)
LUBTAB -- LUB table
PUBTAB -- PUB table
```

Messages Caused:

0I41	0I87
0I47	0I88
0I48	0I89
0I86	

Messages Issued:

None. The messages caused by this routine are issued by common code in IPLDISK.

Input: DEF command

Output: Modified LUB table

Exit Normal: to MONITOR in IPLDISK

Exit Error: to error exits in IPLDISK

Register Use:

```
R 7: Pointer to IOFLD
R 8: Pointer to background COMRGN
R14: Base register of common code (IPLDISK)
R15: Base register
```

Macro MACIPLR6

Phase Name: $IPLRT2

Module Name: IJBIPL

CSECT Name: IJBIPL6

Entry Point: IJBIPL6

Function: Processes the SVA command.

Called By: IPLDISK

Phases Called:

```
$INITCON (librarian) to initialize the library
         concatenation tables
$A$SVA  - list of SVA load books
$SVASA  - list of stand-alone SVA phases
$SVA... - SVA load books
```

Data Areas Used:

```
COMREG    -- Partition communication region (macro MAPCOMR)
CONCATPL  -- Parameter list communicating with $INITCON
DIRECTRY  -- System sublibrary or SDL directory entry
             (macro MAPDNTRY)
MAPCLIM   -- Class or system limits
PMCOM     -- Page management control block
SMCOM     -- Storage management control block
SYSCOM    -- System communication region
SVICOM    -- IPL to SV communication area (macro MAPSVIPL)
```

Messages Caused:

0I29
0I83
0I84
0I85
0I86
0I87
0I88
0I90
0I91
0I94
0I95
0J19
0J66

Messages Issued:

None. The messages caused by this routine are issued by common code in IPLDISK.

Input:

- SVA command
- Load Lists:
  - $$A$SVA
    - $SVACSC
    - $SVAICCF
    - $SVALOG
    - $SVASEC
    - $SVABAM
    - $SVAVSAM
    - $SVARCF
    - $SVA0000

  or
  - $SVASA

Output:

```
Sorted SDL entries
Parameter list for phase $INTVIRT
Address table of copy buffer fragments
```

Exit Normal: to FETCH7 in IPLDISK

Exit Error: to error exits in IPLDISK

Register Use:

```
R 7: Pointer to IOFLD
R 8: Pointer to background COMREG
R14: Base register of common code (IPLDISK)
R15: Base register
```

Sequence of Operation:

MACIPLR6 receives control from IPLDISK and is executed for every IPL.

Routine	Description
ANSVA:	Diagnoses the SVA command.
POSTAN:	End of SVA command analysis.
CBALLOC:	Calculates the space needed for copy buffers. Builds an address table of copy buffer fragments for the final formatting done at the end of IPL processing. Determines the end of system area and aligns it to page boundary.
CBSETKEY:	Sets supervisor key for the copy buffer area. Corrects supervisor end address, SDAIDS area address, and BG begin address. BG begin address is not set below a predefined value, in order to prevent the IPL console support from being overloaded. This support has to remain operational until the 31-bit SVA is loaded.
CBALLOCX:	Shifts IPL code adjacent to the BG save area.
IJBIPL62:	Updates GETVIS values by length of library concatenation control tables (24-bit), and by the length of various dynamicly allocated system control blocks (24-bit and 31-bit). Also accumulates PFIX requirements for those system control blocks. Issues GETVIS SVC (GETVIS SVA=YES,IPL=YES) to inquire the length of the system GETVIS control information. Adds the value to the 31-bit System GETVIS requirements.
LOAD1:	Reads phase $$A$SVA, which contains the names of load lists. In stand-alone environment: Reads phase $SVASA, which contains the names of all SVA phases in alphabetical order.
LOADL10:	Loads the load list phases and reads in the directory entries of all phases that are named in the load lists, if they fulfill the requirements for loading.
SORT0:	Sorts the directory entries.
CALCL1:	Calculates for phase $INTVIRT (called by MACIPLR7) the number of SDL entries and the SVA space needed for the text of the phases (including PSIZE); accounts for the additional system GETVIS space (GETVIS, SUBLIB).
MISSLL:	Informs the operator if load lists or phases are missing.
LEAVEPH:	------->FETCH7 (IPLDISK)
SABLDSDL	Builds the SDL for stand-alone systems from loadbook $SVASA

Macro MACIPLR7

Phase Name: $IPLRT2

Module Name: IJBIPL

CSECT Name: IJBIPL7

Entry Point: IJBIPL7

Function: Final processing of IPL

Called By: IPLDISK

Phases Called:

- $$BUFLDR to load print buffers
- $IJBRCFA (supervisor) to establish security exits
- $INITCON (librarian) to initialize library concatenation tables
- $INITSYS (supervisor) to perform final supervisor initialization
- $INTVIRT (supervisor), twice, to allocate virtual storage;
  it calls:
  - $IJBSSM (supervisor) to allocate the BG address space and virtual partition
- INLPSDL (librarian), twice, to load system phases into the 31-bit SVA and 24-bit SVA
- $IJBAR to retrieve SPLEVEL information from SYSLIB

Data Areas Used:

```
CCW copy block area
COMREG    -- Partition communication region (macro MAPCOMR)
DIRECTRY  -- Directory entry (MAPDNTRY)
DSVA      -- SVA header DSECT (macro MAPSHAHD)
INLCSDLE  -- system directory list entry
INLCSDLH  -- system directory list header
INLCSLXE  -- SYSLIB descriptor
IPARML    -- APPC/VM parameter list
JPL       -- Job control parameter list for Access Control (DTSJPL)
SLADCT    -- Label information area
LUBTAB    -- LUB table
MAPCLIM   -- Class or system limits
MAPXPTAB  -- APPC/VM support anchor
NICL      -- number-in-class list
PCEADR    -- Partition control block (macro MAPPCE)
PIBADR    -- PIB table DSECT (macro MAPPIB)
PJBADR    -- POWER job information (macro MAPPOWJB)
PUBTAB    -- PUB table
PTCOM     -- Performance monitor communication area
RASLADR   -- RAS linkage area
SCYVECDS  -- Security vector table DSECT (macro SCYVECTB)
SMCB      -- Storage management control block (macro SMCB)
SMCOM     -- Storage management communication area
SVICOM    -- IPL to SV communication area DSECT (macro MAPSVIPL)
SYSCOM    -- System communication region DSECT
VMAPCVEC  -- DSECT for APPC/VM vector table
VMDSPID   -- DSECT for IUCV/APPC/VM path ID table entry
```

Messages Caused and Issued:

0I20
0I74
0I82
0I94
0J18
0J19
0J39
0J44
0J59
0J63 (not documented)
0J67
0J68
0J69
0J70
0J73

Input:

```
Parameter list for Phase $INTVIRT
Sorted SDL entries
```

Output:

```
CCW copy block area
COMREG
DSVA
INCLSDLE
INLCSDLH
IUCV/APPC/VM path ID table formatted
JPLs for static partitions
Label information area
LUBTAB
Page frame tables
Page tables
PIBADR
SCYVCDS
Segment tables
SLACB
SMCB
SYSCOM
System GETVIS control information
VMAPCRES
```

Exit Normal: SVC 14 (EOJ)

Exit Error:

```
to IPLTERM in IPLDISK to terminate by hard wait
to ISSMSG94 in IPLDISK to terminate by hard wait
```

Register Use:

```
R7:  Pointer to IOFLD
R8:  Pointer to background COMRGN
R9:  Pointer to SYSCOM
R14: Base register of common code (IPLDISK)
R15: Base register
```

Sequence of Operation:

MACIPLR7 receives control from IPLDISK and is executed for every IPL.

Routine	Description
COMPL1:	Calls routine IJBIPLU to support unattended node. Loads print buffers.
MOVINFO:	Checks if phases $IJBSSM and $INTVIRT fit below SDL area and loads them. Allocates virtual space via calling phase $INTVIRT and $IJBSSM. At this first call of $INTVIRT IPL passes all accumulated storage values (SVA, PSIZE, SDSIZE, etc.). All appropriate addresses and counters in the supervisor are initialized, also those referring to 24-bit the shared area. Private space and the 31-bit SVA are allocated, but the space of the 24-bit shared area is not yet allocated. IPL is moved to the virtual BG partition, and it receives control back in virtual mode.
INTVIRTM:	Analyzes return codes from $INTVIRT, and prints error message if applicable.
SETTRAM:	Sets various virtual switches in supervisor.
NOWVIRT1:	Initializes SVA header in the IPL partition copy. Loads phase INLPSDL behind $INTVIRT and $IJBSSM and passes control to it for loading system phases into the 31-bit SVA. The SVA header and the SDL are still in the IPL partition, and INLPSDL updates this copy.
SVACSIO:	Scans the SDL for the router phase and calculates its load point, which is the entry point to the console support initialization.
R7CSIOR:	Reinitializes console support residing in the 31-bit SVA. ------->$IJBCSIO Calls MOVZONE subroutine (MACIPLRE) to allocate the zone boundary table in the SVA and copy the temporary IPL table to it.
R7CSIOOK:	Allocates 24-bit shared area by calling phase $INTVIRT for a second time. No input parameters are passed, a non-zero return code is a system error. Shifts SVA header and SDL entries into 24-bit SVA. Updates the SDL start address and the SDL free list pointer.
SALDSV24:	Passes control to INLPSDL for a second time to have system phases phases loaded into the 24-bit SVA.
LDSVA:	Passes entry points of special phases in the SVA to the supervisor. Initializes label area control block. Quiesces all I/O. Calls INICHANQ subroutine to format the channel queue. Calls ICOPYBL subroutine to format all copy buffers.
BLDJPL:	Initializes dynamic partition support (DYNCLASS ID=INITIAL).
ALLOCJPL:	If SEC = YES was specified, then • allocates job control parameter lists, • anchors JPLs in COMREGs and PJBs (POWER job information blocks), • allocates security work areas, • anchors security work areas in PCB extensions.
APCOPEN:	Opens APPC/VM communication by enabling the IUCV/APPC/VM external interrupt.

RCFOPEN: Initializes communication with security manager. ------->$IJBRCFA

RCFOPEND: Loads $INITSYS (supervisor phase). ------->$INITSYS

BLDCCAT: Allocates and initializes library concatenation control tables. ------->$INITCON

ISSLDSPL: Retrieves SPLEVEL information from SYSLIB ------->$IJBAR special code

Posts attention task to initialize itself.

Issues “IPL complete” message.

Goes into problem state.

EOJ211: ------->EOJ

Macro IPLUNATT

Phase Name: $IPLRT2

Module Name: IJBIPL

CSECT Name: IJBIPUNS

Entry Point: IJBIPLU

Function: Support of unattended nodes

Called By: MACIPLR7

Phases Called: none

Data Areas Used:

```
COMREG   -- Partition communication region (macro MAPCOMR)
AVRADR   -- Automatic volume recognition DSECT (macro AVRLIST)
UNATTCB  -- Control block for unattended node re-IPL processing
```

Messages Caused and Issued:

0J06I

Input:

```
Parameters from SYS command (PRIMIPL, ALTIPL) contained in UNATTCB
Counter record from SYSRES
```

Output:

```
UNATTCB
Counter record from SYSRES
```

Exit Normal: to caller

Exit Error: to caller

Register Use:

```
R 4: Pointer to I/O buffer
R 8: Pointer to background COMRGN
R 9: Pointer to AVRLIST
R10: Pointer to UNATTCB
R11: 2nd base register
R12: 1st base register
R14: Base register of common code (IPLDISK)
```

Sequence of Operation:

Entry point IJBIPLU gets control once for every IPL.

Routine	Description
UNTINI:	Gets device type codes of IPL devices and fills them into UNATTCB.
UNTINI8:	Reads counter record from primary IPL device
UNTFMT:	Initializes counter record when not present or in error.
UNTREWR:	Updates and re-writes counter record
UNTRELUB:	Completes UNATTCB.
UNTMAP10:	Calculates and stores a hash sum over the block of storage consisting of UNATTCB and the supervisor routine DOREIPL. -------> caller
UNTBLD00:	Resets the counter record.
UNTDOIO:	Does I/O.

Macro IPLCGEN

Macro Name: IPLCGEN

Function: To simplify maintenance of:

```
MACIPLR8 (DLA processor)
MACIPLR9 (DPD processor)
MACIPLRA (DLF processor)
```

Contains common routines and data areas: Each operand has its own subroutine whose name begins with either DLA, DPD, or DLF.

```
Subroutine            Operand
----------            -------
for all: xxxEXT00  -  BLK/CYL
         xxxSIZ00  -  NBLK/NCYL
         xxxDSF00  -  DSF
         xxxUNI00  -  UNIT
         xxxVOL00  -  VOLID
for DLA: xxxNAM00  -  NAME
DPD/DLF: xxxTYP00  -  TYPE
for DLF: xxxNCP00  -  NCPU
```

Additionally, IPLCGEN generates the following subroutines:

```
for all: xxxCHK00 -  Calls the appropriate operand subroutine.
         xxxOP00  -  Performs various VTOC handling.
         xxxOPS00 -  Opens the VTOC.
         xxxCL00  -  Closes the VTOC.
         xxxSCR00 -  Scratches the DSCB.
         xxxRSP00 -  Handles operator response.
         xxxED00  -  Edits the data set extent limits.
         xxxCVH00 -  Loads common VTOC handler.
         xxxUSE00 -  Maintains physical usage counters and ownerships.
for DLA: xxxIN00  -  Initializes the label area.
         xxxLIM   -  Checks upper limit for NCYL/NBLK specification.
for DPD: xxxRET00 -  Resets the extent table entries.
         xxxFSN   -  Checks for correct volid specification.
         xxxRENAM -  Renames page data set before and after formatting.
DPD/DLF: xxxFM00  -  Formats the data set.
for DPD: xxxCHFM  -  Checks the data length of a existing page data set.
```

Called By:

```
MACIPLR8
MACIPLR9
MACIPLRA
```

Input: Specification for keyword operand P=(DLA, DLF, or DPD)

Output: Subroutines and data areas needed for processing the command.

Macros Used:

```
CCB          EXCP           LOAD          WAIT
CVTOC        GETVCE         OVTOC
DEVUSE       GENL           PVTOC
```

Macro MACIPLR8

Phase Name: $IPLRT2

Module Name: IJBIPL

CSECT Name: IJBIPL8

Entry Point: IJBIPL8

Function: Processes the DLA command:

Called By: IPLDISK

Phases Called: $IJJHCVH, common VTOC handler

Data Areas Used:

```
AVRADR  -- Automatic volume recognition DSECT (macro AVRLIST)
COMREG  -- Partition communication region (macro MAPCOMR)
DIRECTRY-- Directory entry DSECT (macro MAPDNTRY)
Format-1 label
Format-4 label
IJJHCPL -- Common VTOC handler parameter list
Label information area
LUBTAB  -- LUB table
PUBTAB  -- PUB table
SYSCOM  -- System communication region DSECT
```

Messages Caused:

0I37	0I75
0I38	0I76
0I39	0I77
0I41	0I78
0I42	0I79
0I43	0I80
0I44	0I86
0I45	0I87
0I46	0I88
0I47	0I89
0I51	0I94
0I52	0I97
0I73	0I99

Messages Issued:

None. The messages caused by this routine are issued by common code in IPLDISK.

Input:

```
DLA command
Data areas to be updated
```

Output: Label information area defined

Macros Used:

```
ASYSCOM
IPLCGEN
```

Exit Normal:

```
to FDSRTN in IPLDISK
to MONITOR in IPLDISK
to OPRTN30 in IPLDISK
```

Exit Error: to error exits in IPLDISK

Register Use:

```
R 7: Pointer to IOFLD
R 8: Pointer to background COMREG
R14: Base register of common code (IPLDISK)
R15: Base register
```

Sequence of Operation:

Routine	Description
IJBIPL8:	Establishes addressability
DLAPR00:	Diagnoses the DLA command. Opens label information area.
DLAIN00:	Initializes the label information area if it has not been initialized.
DLAPR51:	------->MONITOR (IPLDISK)

Macro MACIPLR9

Phase Name: $IPLRT2

Module Name: IJBIPL

CSECT Name: IJBIPL9

Entry Point: IJBIPL9

Function: Processes the DPD command.

Called By: IPLDISK

Phase Called: $IJJHCVH, common VTOC handler

Data Areas Used:

```
AVRADR    -- Automatic volume recognition DSECT (macro AVRLIST)
COMREG    -- Partition communication region (macro MAPCOMR)
DIRECTRY  -- Directory entry DSECT (macro MAPDNTRY)
DPDTAB    -- Page data set table (macro MAPDPD)
Format-1 label
Format-4 label
IJJHCPL   -- Common VTOC handler parameter list
LUBTAB    -- LUB table
PUBTAB    -- PUB table
SMCB      -- Storage management control block (macro SMCB)
SYSCOM    -- System communication region DSECT
PMCOM     -- Page management communication region DSECT
Table of constants for load leveller (supervisor)
VOL1 label
```

Messages Caused:

0I37	0I51	0I88
0I38	0I52	0I89
0I39	0I73	0I94
0I41	0I75	0I99
0I42	0I76	0J11
0I43	0I77	0J12
0I44	0I78	0J13
0I45	0I79	0J14
0I46	0I86	0J16
0I47	0I87	0J72

Messages Issued:
None. The messages caused by this routine are issued by common code in IPLDISK.

Input:

```
DPD command
```

Output:

```
DPDTAB
Constant table for load leveller
```

Macros Used:

```
ASYSCOM
IPLCGEN
```

Exit Normal:

```
to FDSRTN  in IPLDISK
to MONITOR in IPLDISK
to OPRTN30 in IPLDISK
```

Exit Error: to error exits in IPLDISK

Register Use:

```
R 7: Pointer to IOFLD
R 8: Pointer to background COMREG
R14: Base register of common code (IPLDISK)
R15: Base register
```

Sequence of Operation:

MACIPLR9 receives control from IPLDISK and is executed for every IPL, unless the NOPDS option of the supervisor parameters command was specified.
The DPD command(s) has to precede the SVA command. If more DPD commands are needed the operator is prompted to enter them until the virtual storage size is mapped.

Routine	Description
IJBIPL9:	Establishes addressability. Retrieves page size from supervisor (PMCOM) and calculates the size of the Page Data Set, the number of FBA-blocks per page or the number of pages per track.
DPDPR00:	Diagnoses the DPD command. Opens page data set extent. Retrieves VSIZE from SYSCOM and page size from PMCOM. Calculates number of pages needed for PDS. Initializes DPD table (DPDTAB) in supervisor.
DPDCHFM:	This routine is invoked only if the DPD specification matches with a existing Page Data Set on a CKD device. In this case the first record of the PDS will be read and the data length will be compared with page size. If there is a mismatch the PDS is to be formatted.
DPDFM00:	Formats the page data set extent if it has not been formatted or if it must be reformatted (for CKD devices only). During formatting the page data set receives a temporary name, so that IPL can check if the previous IPL completed formatting.
DPDFSN00:	Checks if VOLID specification (if any) matches the VOLID on specified unit.
DPDRET00:	Resets the extent table entries.
DPDLL00:	Sets the load leveller control values based on CPUID and device type of first extent definition.
DPDPR91:	------->MONITOR (IPLDISK)

Macro MACIPLRA

Phase Name: $IPLRT2

Module Name: IJBIPL

CSECT Name: IJBIPLA

Entry Point: IJBIPLA

Function: Processes the DLF command:

Called By: IPLDISK

Phases Called: $IJJHCVH, common VTOC handler

Data Areas Used:

```
AVRADR    -- Automatic volume recognition DSECT (macro AVRLIST)
COMRGN    -- Partition communication  region (macro MAPCOMR)
DIRECTRY  -- System sublibrary or SDL directory entry DSECT
             (macro MAPDNTRY)
DLFADR    -- DLF table DSECT (macro MAPDLF)
Format-1 label
Format-4 label
IJJHCPL   -- Common VTOC handler parameter list
Lock file formatting record
LUBTAB    -- LUB table
PUBTAB    -- PUB table
SVICOM    -- IPL to SV communication area DSECT (macro MAPSVIPL)
VOL1 label
```

Messages Caused:

0I37	0I79
0I38	0I80
0I39	0I86
0I41	0I87
0I42	0I88
0I43	0I89
0I44	0I94
0I45	0I97
0I46	0I99
0I47	0J27
0I51	0J28
0I52	0J29
0I73	0J30
0I75	0J31
0I76	0J33
0I77	0J36
0I78	

Messages Issued:

None. The messages caused by this phase are issued by common code in IPLDISK.

Input:

```
DLF command
Data areas (see output)
```

Output:

```
DLF table
Formatted lock file
```

Macros Used:

```
ASYSCOM
CCB
IPLCGEN
UNLOCK
```

Exit Normal:

```
to FDSRTN  in IPLDISK
to MONITOR in IPLDISK
to OPRTN30 in IPLDISK
```

Exit Error: to error exits in IPLDISK

Register Use:

```
R 7: Pointer to IOFLD
R 8: Pointer to background COMREG
R14: Base register of common code (IPLDISK)
R15: Base register
```

Sequence of Operation:

Routine	Description
IJBIPLA:	Establishes addressability.
DLFPR00:	Diagnoses the DLF command.
	Initializes the lock file if this has not been done before.
DLFOR40:	Initializes the supervisor DLF table. Initializes the supervisor DASD sharing support. Unlocks all resources locked for the own CPU ID.
DLFPR50:	------->MONITOR (IPLDISK)

If DASDs with option SHR are added, the DLF command has to precede any DEF, DLA, DPD, or SVA command. The DLF command is ignored if no DASDs with option SHR are added.

Macro MACIPLRB

Phase Name: $IPLRT2

Module Name: IJBIPL

CSECT Name: IJBIPLB

Entry Point: IJBIPLB

Function: Processes the SYS command:

- Analyzes the operands
- Saves the values in SYSPARM

Called By: IPLDISK

Phases Called: None

Data Areas Used:

```
AVRLIST -- Volume recognition table
COMREG  -- Partition communication  region (macro MAPCOMR)
MAPCLIM -- Class/system limits
PMCOM   -- Page manager communication region
PUB     -- Pub table (MAPPUB DSECT)
SVICOM  -- IPL to SV communication area DSECT (macro MAPSVIPL)
SYSCOM  -- System communication region DSECT
SYSPARM -- Save area for SYS operand values (in IPLDISK)
```

Messages Caused:

0I22	0J37
0I25	0J42
0I86	0J43
0I87	0J60
0I88	0J61
0I89	

Messages Issued:

None. The messages caused by this routine are issued by common code in IPLDISK.

Input:

```
SYS command, that is, a pointer to first operand.
```

Output:

```
Flags in
  SYSOPT1 (SEC,JA,DASDFP,VMCF,ATL)
  COMREG.RMSROPEN (BUFLD)
  SYSOPT2 (UNATT,REIPL)
Values in
  SYSCOM.IJBSCTAB.SCYESMPN
  SYSCOM.IJBSCTAB.SCYFLG1.SCYESMS
  SYSCOM.IJBSCTAB.SCYSRVPA
  SYSCHANQ (CHANQ)
  SYSBUFS (BUFSIZE)
  SYSLIB (SUBLIB)
  SYSSDSIZ (SDSIZE)
  SYSPRI and SYSPRCU (PRIMIPL)
  SYSALT and SYSALCU (ALTIPL)
  SPSIZSAV (SPSIZE)
  SYSNPART (NPARTS)
  VIRTPARM.VPPASIZE (PASIZE)
  VIRTPARM.VPRSIZE (RSIZE)
```

Exit Normal: to MONITOR in IPLDISK.

Exit Error: to error exits in IPLDISK.

Register Use:

```
R 7: Pointer to IOFLD
R 8: Pointer to background COMREG
R 9: Pointer to SYSCOM
R14: Base register of common code (IPLDISK)
R15: Base register
```

Sequence of Operation:

MACIPLRB receives control from IPLDISK if the SYS command was given. The SYS command can be entered at any time and repeatedly, but it has to precede the SVA command.

Routine	Description
ANSYS:	Diagnoses the SYS command.
DASDFPRT:	Evaluates the DASDFP specification.
JART:	Evaluates the JA specification.
CHANQRT:	Evaluates the CHANQ specification.
BUFSRT:	Evaluates the BUFSIZE specification.
SUBLIBRT:	Evaluates the SUBLIB specification.
SECRT:	Evaluates the SEC specification.
SDSIZERT:	Evaluates the SDSIZE specification.
BUFLDRT:	Evaluates the BUFLD specification.
UNATTRT:	Evaluates the UNATT specification.
PRIMRT:	Evaluates the PRIMIPL specification.
ALTIPLRT:	Evaluates the ALTIPL specification.
REIPLRT:	Evaluates the REIPL specification.
SPSIZERT:	Evaluates the SPSIZE specification.
NPARTSRT:	Evaluates the NPARTS specification.
VMCFRT:	Evaluates the VMCF specification.
PASIZERT:	Evaluates the PASIZE specification.
RSIZERT:	Evaluates the RSIZE specification.
ATLRT:	Evaluates the ATL specification.
ESMRT:	Evaluates the ESM specification.
SERVPRTN:	Evaluates the SERVPART specification.
ANSYSEX:	------->MONITOR (IPLDISK)

Macro MACIPLRC

Phase Name: $IPLRT2

Module Name: IJBIPL

CSECT Name: IJBIPLC

Entry Point: IJBIPLC

Function: Final dynamic allocation of supervisor tables.

Called By: IPLDISK

Phases Called: None

Data Areas Used:

```
AVRADR   -- Automatic volume recognition DSECT (macro AVRLIST)
CBDSECT  -- Console buffer
CHQADR   -- Channel queue
COMREG   -- Partition communication  region (macro MAPCOMR)
CRTSAV   -- DOC save area
CRTTAB   -- DOC table
External interrupt buffer
Hard copy buffer
LUBTAB   -- LUB table
PUBX     -- PUB extension (macro MAPPUBX)
PHLSTHDR -- Phase load trace table
PIBADR   -- PIB table DSECT (macro MAPPIB)
PUBTAB   -- PUB table
RFTABLE  -- Recorder file table DSECT (macro MAPRFTAB)
SVICOM   -- IPL to SV communication area DSECT (macro MAPSVIPL)
SYSCOM   -- System communication region DSECT
SYSREC buffer
VIOCM    -- VIO communication area
```

Messages Caused:

0I29
0I40
0I58
0I63
0J62

Messages Issued:

None. The messages caused by this phase are issued by common code in IPLDISK.

Input:

```
SVICOM  -- Communication area IPL - SV
SYSPARM -- Save area for SYS operand values
```

Output:

```
Address of background save area in PIB table and SYSCOM
Channel queue
COMREG
Device control blocks
External interrupt buffers for IUCV/APPC/VM
LUB table
Path ID table for IUCV/APPC/VM
PUBX
RFTABLE
SYSCOM
SYSREC buffer
VIO area
```

Exit Normal: to SETOK3 in IPLDISK.

Exit Error: to error exits in IPLDISK.

Register Use:

```
R 7: Pointer to IOFLD
R 8: Pointer to background COMRGN
R 9: Pointer to SYSCOM
R14: Base register of common code (IPLDISK)
R15: Base register
```

Sequence of Operation:

Routine	Description
IJBIPLC:	Establishes addressability.
CATVAL:	Writes the warning message 0158D if SYSCAT was not assigned via the DEF command.
SYSEVAL:	Finally processes YES/NO options of SYS command operands.
SYSUNYES:	Sets unattended node parameters if SYS UNATT=YES was selected.
CYDYNL:	Calculates length of dynamic supervisor area. • Add length of VIO/VPOOL, DEVCB, channel queue, and APPC/VM areas. • Calls subroutine LHCREC to ensure the SYSREC assignment, and to add length of SYSREC buffer. • Allocates an area for SDAID or other monitor functions (SDSIZE specification). Checks for sufficient storage before final allocation. Determines the final background start address and sets the storage key accordingly. The background start address is not set below a predefined minimum value in order to prevent the IPL console support from being overloaded.
UPDPIB:	Updates BG save area address in BG PIB. Allocates space in the dynamic supervisor area for the following constructs: - VIO/VPOOL area (cleared to zero) - device control blocks (cleared to zero)
CALLOC1:	- IUCV/APPC/VM external interrupt buffer and path ID table
CALLOC4:	- SYSREC buffer (cleared to zero)
CALLOC5:	- Channel queue (unformatted) Calls SETZONE routine (MACIPLRE) to finally determine the system time zone. Returns control to IPLDISK ------->SETOK3

Macro MACIPLRD

Phase Name: $IPLRT2

Module Name: IJBIPL

CSECT Name: IJBIPLD

Entry Point: IJBIPLD

Function: Processes the APPC/VM SET command.

Called By: IPLDISK

Phases Called: None

Data Areas Used: VMAPCRES -- APPC/VM resource table

Messages Caused:

0J55	0J57
0J56	0J58

Messages Issued:

```
The messages caused by this routine are issued by common code
in IPLDISK.
```

Input:

```
APPC/VM SET command
Data area for temporary APPC/VM resource table
VM release indicator - if VM guest
```

Output:

```
Temporary APPC/VM resource table
(allocated permanently by MACIPLR7)
```

Exit Normal:

```
to MONITOR in IPLDISK
```

Exit Error:

```
to ILLCD1 in IPLDISK
```

Register Use:

```
R 7: Pointer to IOFLD
R 8: Pointer to background COMREG
R14: Base register of common code (IPLDISK)
R15: Base register
```

Sequence of Operation:

Routine	Description
IJBIPLD:	Establishes addressability. Tests CPUID if system is running as VM guest. Rejects command in native environment. Not VM guest -------> INVCMD Retrieves VM release information.
VMTESTRX:	Checks VM release number. Rejects command if is not VM/ESA*. VM release incorrect -------> VMINVREL
VMSCAN:	Scans input command, checks predefined operands, and passes variable operand specifications to the appropriate subroutine: • TSUBSYS to analyse the XPCC TARGET specification, • TVMRESID to analyse the APPCVM TARGET specification.
VMSCANEX:	Syntax correct - starts update of temporary APPC/VM resource table.
VMSCNTAB:	Compares command to previously submitted and accepted APPC/VM SET commands. The APPC/VM TARGET specification must be unique by itself (XPCC TARGET not specified), and the alias specification must be unique (specified as XPCC TARGET). Rejects command if more than 10 commands issued. -------> VMMAX
VMSCNTB2:	Specification unique - adds specification to temporary APPC/VM resource table. Returns control to IPLDISK to read next card. -------> MONITOR
VMREPL:	Specification not unique - overrides the previously specified duplicate entry in the temporary APPC/VM resource table by current specification. Issues an informational message. Returns control to IPLDISK to read the next card. -------> MONITOR
TSUBSYS:	Checks XPCC TARGET subsystem specification for correct syntax. Returns control to VMSCAN routine: in case of syntax error -------> VMINSPEC to continue processing -------> VMSCANI

TVMRESID: Checks APPCVM TARGET resid specification for correct syntax:

- either resid as single element,
- or resid and SNA network routing information in a list of four elements.

Returns control to VMSCAN routine:
in case of syntax error -------> VMINSPEC
if VM release lower than VM/SP 6, and SNA network
information submitted -------> VMINVREL
to continue processing -------> VMSCANEX

VMINVREL: Issues an informational message. Returns control to the IPLDISK to read the next card. -------> MONITOR

VMINSPEC: Issues a message. Returns control to IPLDISK to have the command corrected. -------> ILLCD1

VMMAX: Issues an informational message. Returns control to the IPLDISK to read the next card. -------> MONITOR

Macro MACIPLRE

Phase Name: $IPLRT2

Module Name: IJBIPL

CSECT Name: IJBIPLE

Entry Point: IJBIPLE

Function: Processes the SET ZONEBDEF and SET ZONEBDY commands.

Called By: IPLDISK

Phases Called: None

Data Areas Used:

```
COMREG   -- Partition communication  region (macro MAPCOMR)
MAPZNBDY -- ZONE definitions and boundaries
SYSCOM   -- System communication region DSECT
TODCOM   -- TOD clock communication area
```

Messages Caused and Issued:

```
0I36                0J81
0I87                0J82
0I88                0J83
0I89
```

Input:

```
SET ZONEDEF command
SET ZONEBDY command
```

Output:

```
Temporary zone definition and boundary table
(allocated permanently by MACIPLR7)
```

Exit Normal:

```
to MONITOR in IPLDISK
```

Exit Error:

```
to DUPKEYW in IPLDISK
to INVSEQ  in IPLDISK
to INVSPEC in IPLDISK
to MISKEYW in IPLDISK
```

Register Use:

```
R 1: Pointer to translated command
R 5: Base register of subroutines also called by external modules
R 7: Pointer to IOFLD
R 8: Pointer to background COMREG
R 9: Pointer to system COMREG
R14: Base register of common code (IPLDISK)
R15: Base register
```

Sequence of Operation:

SET ZONEBDY and SET ZONEDEF Command Processing:

Routine	Description
IJBIPLE:	Establishes addressability.
ZONERTN:	Checks if command may still be given - is rejected after a SET DATE or SET ZONE command ------->INVSEQ (IPLDISK) Routes command to the appropriate command processing routine. Command is SET ZONEBDY ------->ZONEBDY Command is SET ZONEDEF ------->ZONEDEF
ZONEBDY:	Checks the command syntax.
ZONBDYPR:	This subroutine converts the date and clock specifications to a 64 bit binary value. This value represents the number of clock units (that is, microseconds times 2 to the 12th power) elapsed since January 1, 1900 at 0.00 a.m. GMT.
ZBDYBLD:	Uses the 64-bit binary clock value and the zone_id - which has to be previously defined by a SET ZONEDEF command - to build an zone boundary entry. Puts it in chronological order into the time zone boundary table. Equal time values are overridden. Returns to IPLDISK to read the next card. ------->MONITOR
ZONEDEF:	Checks the command syntax.
ZONDEFPR:	This subroutine converts the zone specification into minutes, positive for east, negative for west of Greenwich.
ZDEFBLD:	Uses the zone value and the zone_id to build an zone definition entry. Puts it in the entered sequence into the time zone definition table. Equal zone values or zone_ids are overridden. Returns to IPLDISK to read the next card. -------> MONITOR

Time Subroutines: The subroutines are called internally and externaly by other other command processors to evaluate or retrieve the current system zone.

Routine	Description
GETZONE	Makes zone printable for messages. The zone value, in minutes, is input to the routine. It is called by: • MACIPLR2 - to prepare message 0I30I • MACIPLRE - SETZONE subroutine to prepare message 0J83I
MOVZONE	Allocates the zone boundary table (ZONEBDY) in the 31-bit SVA. Builds the SVA ZONEBDY table by tranforming the local binary time values as collected in the IPL ZONEBDY table into TOD clock values. (TOD clock is GMT.) Copies the zone definitions into the SVA ZONEBDY table. It is called by: • MACIPLR7 - to move the local ZONEBDY table to the SVA.
SETZONE	Determines and sets system zone value in TODCOM and job zone value in the partition COMREG. It is called by: • MACIPLRC - to finally initialize TODCOM and BG COMREG.

SETZDEF Determines and sets the system zone id in TODCOM. Input is the system zone value in TODCOM.

It is called by:

- MACIPLR4 - SET ZONE processing to determine an associated zone id.

SETZVAL Checks if the specified zone_id has been defined previously. Sets the zone id and the system zone value in TODCOM, and the job zone in the partition COMREG, if requested.

It is called by:

- MACIPLR4 - SET zone_id processing to determine if zone id has been previously defined. Updates TODCOM and COMREG with actual zone value.
- MACIPLRE - SET ZONEBDY processing to determine if zone id has been previously defined.

Description of Master Procedure $ASIPROC

The type of IPL (“INSTALL,” “SENSE” or “NORMAL”) and selection criteria can be passed to IPL in the ASI master procedure $ASIPROC.

Two formats of the ASI master procedure are supported. The formats are alternative and must not be merged.

Format 1 of ASI Master Procedure

The format 1 of the master procedure is described in *VSE/ESA Guide to System Functions,* SC33-6611.

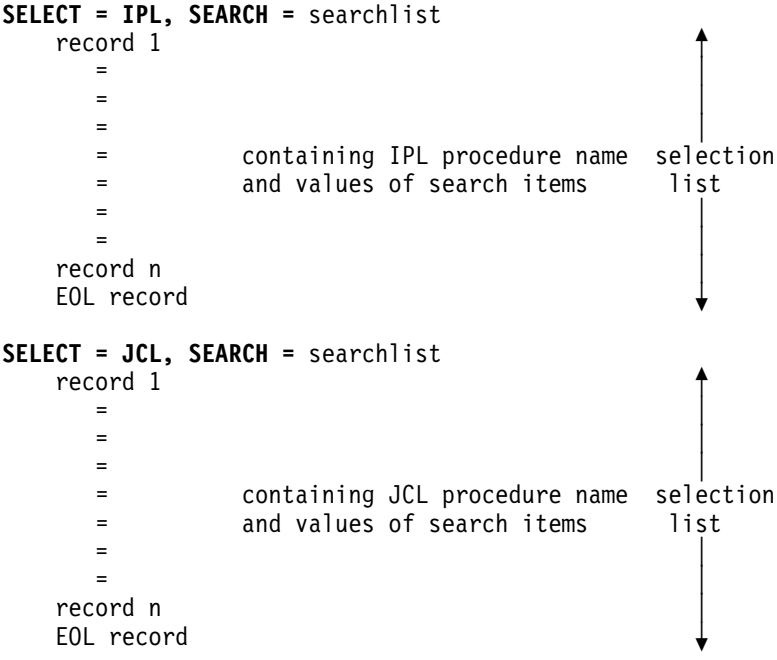
Format 2 of ASI Master Procedure

The format 2 consists of a TYPE specification and a set of records describing which ASI-IPL procedure is applicable and a set of records describing which ASI-JCL procedure is applicable. If one of this record sets is omitted a default procedure name will be taken by IPL.

The first record within such a set of records must contain the information which sort of procedure (IPL or JCL) is defined by the following records (SELECT specification) and how to select a procedure name via search items (SEARCH specification). The next records (selection records) of such a record set are a selection list and consist of a procedure name and values of these search items which are specified in the SEARCH statement.

The last record of a selection list must be a end-of-list record.

Layout of a Format 2 Master Procedure: An ASI master procedure format 2 has the following layout:



TYPE Specification:

The TYPE specification has the following format:

TYPE = {INSTALL/NORMAL/SENSE}

The first record is optional, if omitted TYPE = NORMAL will be assumed.

TYPE = INSTALL — IPL will perform an "installation" process. IPL identifies the I/O configuration via device sensing. For that purpose VSE will issue a SSCH instruction to each subchannel address, until a condition is received indicating that no more subchannels are defined. The returned device number and device type code will be used to generate a PUB entry.
For devices which do not support the SENSE-ID command (or which were not operational) the user has to enter an ADD command or to specify the device lateron in the VSE/ESA base installation process. If special options (e.g. SHR) are to be defined, they also have to be specified lateron.
IPL invokes a routine provided by VSE/ESA to check whether a minimal configuration for a base installation process has been identified automatically. If not, the user will be prompted to enter ADD commands. Additionally IPL invokes a phase provided by VSE/ESA to check whether duplicate VOLIDs are existing.

TYPE = NORMAL — IPL does not automatically identify the I/O configuration but for each device which has been added the device type specification will be verified. If an incorrect device type was specified an information message will be issued and the correct device type will be put into the PUB table.

TYPE = SENSE — IPL identifies the I/O configuration via device sensing as described for the option INSTALL. For devices which

- do not support the SENSE-ID command,
- are not operational,
- are dummy devices,
- need special options, e.g. tape modes or SHR,

the user has to enter an ADD command. Running under VM IPL can determine the device type of all devices which are attached to a virtual machine.

SELECT and SEARCH Specification

SELECT and SEARCH specifications are only allowed in a master procedure of format 2. The syntax is as follows:

SELECT = {IPL/JCL}, SEARCH = searchlist

After finding a SELECT and SEARCH keyword IPL assumes that the next records of the ASI Master Procedure are records containing a selection list.

SELECT = IPL — The following records contain a selection list, which determine the ASI-IPL Procedure depending on the specified search items in the search list.

SELECT = JCL — The following records contain a selection list, which determine the ASI-JCL Procedure depending on the specified search items in the search list.

SEARCH = searchlist — List of search items. If more than one is specified, the items must be enclosed in parenthesis and separated by comma. The search items are defined keywords and the sequence defined in the searchlist must be identical to the sequence of the search item values in the following records defining a selection list. IPL deter-

mines the actual values of the search items, and selects a procedure name if the values within a record of the selection list match with the actual values of the search items.

Note: The number of search items in a search list is currently restricted to one, but it may be extended.

Valid search item: SYSRES

Selection List: Records of a selection list determine a ASI-IPL procedure or a JCL procedure depending on the search items specified in the previous SELECT and SEARCH statement. The last record must be a record which indicates end of the list.

Each record in a selection list is up to 80 bytes in length and has the following format:

```
◄ procedure-name ►,◄    value 1      ►,..,◄     value n     ►
◄---- 1-8 bytes ---► ◄--- 1-8 bytes ---► .. ◄--- 1-8 bytes ---►
or
|◄end of list►|          not applicable                      |
```

Each record entry is up to 8 bytes in length and must be separated by comma. The first entry defines a procedure name and all other entries are values of search items. The entries specifying a value of a search item correspond to the search items defined in the SEARCH statement.

```
        SEARCH  = ( search-item 1,..,search-item n )
                          |               |
                          |               |
                          ▼               ▼
◄ procedure-name ►,◄    value 1   ►,..,◄   value n   ►
```

The values of a search item are fixed terms defined by IPL and maximal 8 characters in length.

Specification of SYSRES in a Selection List

The search item SYSRES specifies that one selection criterion is the SYSRES DASD device type. The DASD device type consists of the device type number and the device model number. If SEARCH=SYSRES is specified IPL scans the records of the selection list. If the corresponding entry within a record matches with the actual SYSRES DASD TYPE IPL stops scanning. Valid values for the search item SYSRES in a selection list are:

```
CKD: the DASD device number e.g. 3380, 3390, ...
FBA: 0671, 9332, 9335, and FBA (for all other FBA DASDs)
```

Additional information to the device number such as a model group number can be specified as a suffix:

```
3390-02    DASD device 3390 model group 1
3390-06    DASD device 3390 model group 2
```

If no model group number is specified IPL selects this record which first match the device type number.

End of List Identifier

The last record of a selection list is the “End of list” record. These record must contain the string “EOL” in the first three bytes, the rest of the record will be ignored.

Example of ASI Master Procedure

The following cards give an example how the ASI Master Procedure of a Installation Process can look like.

```
$ASIPROC - Procedure
TYPE=INSTALL                          first card defines IPL type
SELECT=IPL,SEARCH=SYSRES            selection criteria for
                                      ASI-IPL procedure
$IPLESA,3390                        SYSRES=3390  ----> $IPLESA
$IPLESAB,3380                       SYSRES=3380  ----> $IPLESAB
$IPLESAC,FBA                        SYSRES=FBA   ----> $IPLESAC
EOL                                 end of list

SELECT=JCL,SEARCH=SYSRES            selection criterion for
                                    for ASI-JCL procedure
$$JC3390,3390                       SYSRES=3390  ----> $$JC3390
$$JC3380,3380                       SYSRES=3380  ----> $$JC3380
$$JCFBA,FBA                         SYSRES=FBA   ----> $$JCFBA
EOL                                 end of list
/+                                  end of data
```

The $$BUFLDR Function

This function comprises 3 phases: $$BUFLDR, $$BUFLD1, and $$BUFLD2. It also makes use of a number of device adapted phases called $$BUCBxx and $$BFCBxx.

Phase $$BUFLDR

Module Name: $$BUFLDR

Entry Point: IJBBUFLDR + 8, START2

Function: Loads printer control buffers.

(For more detail see "Sequence of Operation" below.)

Called By: $IPLRT2 (MACIPLR7)

Phases Called:

```
$$BUFLD1 for FCB loading
$$BUFLD2 for error handling
$$BUCBxx for actual UCB loading for
         individual printers (see "Sequence of Operation")
$$BFCBxx for actual FCB loading for
         individual printers (see "Sequence of Operation")
```

Data Areas Used:

```
BUFFRIN  -- Buffer where the UCB image gets loaded
COMRGN   -- Partition communication region (macro MAPCOMR)
PRNTUNIT -- For conversion of cuu addresses to unpacked format
PUBTAB   -- PUB table
SYSUSE LUB
TABLE    -- For translation of cuu addresses into characters
```

Messages Caused and Issued:

```
0I26
0I28
0I69
0I70
```

Input:

```
PUB address of SYSUSE in SYSLOG LUB
LUB is saved and restored when returning to $IPLRT2 (MACIPLR7)
Phases $$BUCBxx and $$FCBxx for the various printers
```

Output: UCB and/or FCB of each PRT1/4248/1403U is loaded.

Exit Normal: to $IPLRT2 (MACIPLR7) via SVC 11

Exit Error: to $$BUFLD2

Register Use:

```
R 0: Buffer load address
R 1: EXCP, FETCH phase, etc.
R 2: Pointer to current PUB
R 3: Pointer to start of PUBTAB
R 5: Pointer to COMREG
R 6: Base register of SYSUSE LUB
R 7: Subroutine linkage register
R 8: Pointer to SYSLOG LUB
R13: Base register of $$BFCBxx phase
R15: Base register
```

Sequence of Operation:

Routine	Description
IJBUFR:	Initializes the phase.
START2:	Entry point for overlay phases $$BUFLD1 and 2.
RELO:	Relocates CCWs, saves SYSUSE LUB.
NEXT:	The phase scans the PUB table for 1403U, 4248, PRT1, and 3800.
INIT3800:	If 3800, gives initialize command and prints message 0I69I.
	If no more entries ------->$IPLRT2 (MACIPLR7)
FOUND:	Else points SYSUSE LUB to printer.
	If dummy device ------->NEXT
	If printer not ready, and the flag COMREG.RMSROPEN.IJBBUFIG is not set, issues message 0I28D to allow the operator to skip buffer load request for that printer. If the COMREG flag is set (BUFLD=IGNORE specified in IPL-SYS command), no message is issued and the buffer load request is ignored.
SETLUB:	Loads $$BUCBxx. Figure 6 shows the standard buffer image phases for the various printers.

Printer	To Load FCB	To Load UCB
1403U	————	$$BUCB4
(3211,4248 PRT1= (3203–4,–5 (3289–4 (3262 (4245	$$BFCB $$BFCB00 $$BFCB10 $$BFCB22 $$BFCB23	$$BUCB $$BUCB00 $$BUCB10 $$BFCB22 —
4248 native	$$BFCWM	—

Figure 6. Standard Buffer Image Phases

The 3800 printer buffer load is initialized via a special command.

If an error occurs ------->$$BUFLD2 If an error occurs during 3800 initialization, prints message 0I70I and ------->NEXT If printer 4245 or 4248, go to FETCH1 without loading UCB.

FETCH1: Loads $$BUFLD1 to label START2.

ENDMSG: If 1403U, prints message 0I26I and ------->NEXT

Phase $$BUFLD1

Module Name: $$BUFLDR

Entry Point: IJBUF1

Function: Prepares loading of the FCB. For the PRT1 printers, this phase performs a skip to the first printline of the form.

$$BUFLD1 is an overlay phase to $$BUFLDR:

IJBUF1: Starts processing.

RELO: Relocates CCWs, saves SYSUSE LUB.

BEGIN1: Aligns the paper.

LFCB: Loads $$BFCBxx.
See the table of standard buffer image phases in “Sequence of Operation” of phase $$BUFLDR. If an error occurs ------->$$BUFLD2 Else ------->ENDMSG ($$BUFLDR)

Called By: Phase $$BUFLDR

Phases Called:

```
$$BFCBxx (loaded)
$$BUFLD2
$$BUFLDR
```

Data Areas Used:

```
BUFF2, a buffer where the FCB image gets loaded.
```

Messages: None

Input:

```
Printer device information
Phase $$BFCBxx
```

Output: None

Exit Normal: to $$BUFLDR to load the FCB

Exit Error: to $$BUFLD2

Register Use:

```
R13: Base register
R15: Base register of root phase
```

Phase $$BUFLD2

Module Name: $$BUFLDR

Entry Point: IJBUF2 + 8

Function: Tests load type of the failing buffer load request and issues error messages with the load type and name of the failing phase.

Called By:

```
$$BUFLDR
$$BUFLD1
```

Phases Called: $$BUFLDR

Data Areas Used: None

Messages Caused and Issued:

```
0I26
0I27
```

Input: Uses common data fields of $$BUFLDR and $$BUFLD1, which are not overlaid, by direct addressing.

Output: Error messages with names of phases that failed to be loaded.

Exit Normal: None

Exit Error:

- to label ENDMSG in $$BUFLDR for FCB failure
- to label FETCH in $$BUFLDR for UCB failure

Register Use:

```
R 9: CCW save register
R10: Return linkage
R14: Base register
R15: Base register of root phase
```

The SYSBUFLD Program

The program:

- Reads control statement.
- Checks for errors and incompatibilities.
- Loads the UCB buffer.
- Checks number of print positions and phase length.
- Loads FCB buffer.
- Reads the next control statement.

The SYSBUFLD program loads the control buffer image into the forms control buffer (FCB) for output in a nonstandard page layout and into the universal character set buffer (UCB) for output with a UCS print train, chain, or belt.

SYSBUFLD is invoked by a // EXEC SYSBUFLD statement and can be executed any time after IPL within the user job stream. With one invocation of SYSBUFLD, the FCB and UCB of a single printer, or any combination of buffers on several printers, can be loaded.

The use of the SYSBUFLD program is described in *VSE/ESA System Control Statements*, SC33-6713.

Phase SYSBUFLD

Module Name: IJBSBUFF

Entry Point: ALPHA

Function: Loads the print control buffers UCB and/or FCB, or does BAND-ID verification:

```
UCB    for PRT1(not 4245)/1403U printers.
FCB    for PRT1/4248 printers.
BANDID for 4248 (native mode) printers.
```

PRT1 is the common name of all printers which are 3211-compatible.

Except for 4245 and 4248, the forms are aligned to line 1 and message 1B21A allows the operator to change the forms if necessary.

(For details see “Sequence of Operation” below.)

Called via: EXEC statement

Phases Loaded:

```
$$BUCBxx
$$BFCBxx
```

These phases contain only data. See the standard buffer image phase table in “Sequence of Operation” of phase $$BUFLDR.

Data Areas Used:

```
BUFFR and DATA -- Same area used to read the fields
                  of the statement into DATA,
                  then the buffer and messages are built
                  into it.
TRTABLE        -- Used to analyze the control statement.
ADCONS         -- Eight 2-byte constants which show the
                  binary value of the FCAR counter of
                  PRT1 printers.
BUFF2          -- FCB read-in area
BUFFRIN        -- Buffer-build area
TABLE          -- Used for translating FCB load records.
```

Messages Caused and Issued:

```
1B01
1B02
1B03
1B04
1B05
1B06
1B09
1B10
1B11
1B20
1B21
1B22
```

Input:

- UCB or FCB definition phases from system sublibrary. (See table in “Sequence of Operation” of IPL phase $$BUFLDR.)
- Control statements from SYSIPT to identify printer and buffer load type. (See *VSE/ESA System Control Statements*, SC33-6713.)
- An FCB definition 'data string' can be read in the form of 80-byte records from SYSIPT instead of from the system sublibrary. (This is not possible for a 4248 printer.)

Output:

- A loaded UCB or FCB and, optionally, a message
- If the device is a PRT1 or 4245 and the FCAR (forms control address register) is not at position 1, a skip to channel 1 is issued after doing an FCB load to position the forms properly.

Exit Normal: EOJ - SVC14

Exit Error: CANCEL - SVC6

Register Use:

```
R 4: Linkage register
R 6: Pointer to COMREG
R 7: Pointer to CCB
R 8: Pointer to PUB
R10: Base register
R12: FCB definition record start
R13: Linkage register
R14: FCB definition record end
```

Sequence of Operation:

Routine	Description
ALPHA:	Initializes the phase.
Statement Scanning:	
READIN:	Reads a control statement. If end-of-file ------->SVC14 (EOJ)
TYPRTURN:	Check for BANDID, FCB or UCB and the logical unit. If o.k. ------->CHKTYPE *For invalid operation*: If non-log system ------->SVC6 (CANCEL) If UCB specified for 5203 or FCB for 1403U issues message 1B10. ------->SVC6 (CANCEL) Else issues IBO2A and waits for an operator response.
SYSRTURN:	*For invalid SYSxxx*: If non-log system ------->SVC6 (CANCEL) Else issues 1B02A and waits for operator response.
CHKTYPE:	Checks the physical unit for a supported printer. If o.k. ------->LOAD3
NOPHYS:	*For invalid physical unit*: If non-log system ------->SVC6 (CANCEL) Else issues message 1B11D and waits for operator response.
LOAD3:	Checks phase name. If o.k. ------->NEXTUP
NOPHSE:	If invalid phase name or none, with UCB load, issues message 1B03I.
NEXTUP:	Checks optional operand.
BAD2:	If FOLD or NOCHK and FCB LOAD, issues message 1BOnI. n indicates the position of the invalid operand. If this is BANDID VERIFICATION ------->BANDIDRT If this is FCB LOAD ------->SETUP2 If this is UCB LOAD ------->UCBLOAD
UCB Load:	
UCBLOAD:	Alter CCW string to add optional operands.
GETUCB:	Loads UCB image phase. If it has the wrong length, issues message 1B03I. ------->NOPHSE Wrong length if: LENGTH ◄ 512 and PRT1 or ◄ 240
UCBLOAD1:	Else loads UCB and ------->READIN

FCB Load:

SETUP2: Here begins FCB loading.

GETCC: *For card image load*: Reads until end of buffer. If more than:

112 bytes for a 5203 printer,
180 bytes for a 4245 printer,
192 bytes for a 3203 printer,
255 bytes for a PRT1 printer
are read, issues message 1B03I and ------->SVC6 (CANCEL)

LOADFCB: *For phase load*:
Loads the FCB phase.
If phase length is not one of the following device-specific lengths:
192 bytes for a 5203 printer,
260 bytes for a 4245 printer,
336 or 261 bytes for a PRT1 printer with indexing on
272 bytes for a 3203 printer,
335 bytes for a PRT1 printer,
340 bytes for a 4248 printer,
issues message 1B03I.
Ensures that forms are positioned at line 1.

DOIO: Loads FCB buffer.

BAL: Prints message 1B21A if not under POWER. ------->READIN

BANDID Verification:

BANDIDRT: BAND-ID Verification routine.

BANDID20: Read BANDID from 4248.
If no BANDID in control card ------->READIN
If given BANDID = read BANDID ------->READIN
Set up for msg 1B22.

BANDID30: Issue message and read response.
If response = CANCEL ------->SVC6 (CANCEL)
If response = EOB ------->BANDID50
If response ► 4 char's, set up msg 1B20 ------->BANDID30

BANDID50: If response = read BANDID ------->READIN
Signal Attention to 4248.
Goto ------->BANDID20

Subroutines:

READIPT Reads input.

TYPEXCP Does I/O for messages.

PRINTMSG Does I/O for messages.

UNITPUB Finds PUB entry.

ISSUMESG Puts out error message.

Chapter 3. Organization Information -- Initial Program Load

Figure 7 and Figure 8 on page 102 show the control flow from phase to phase.

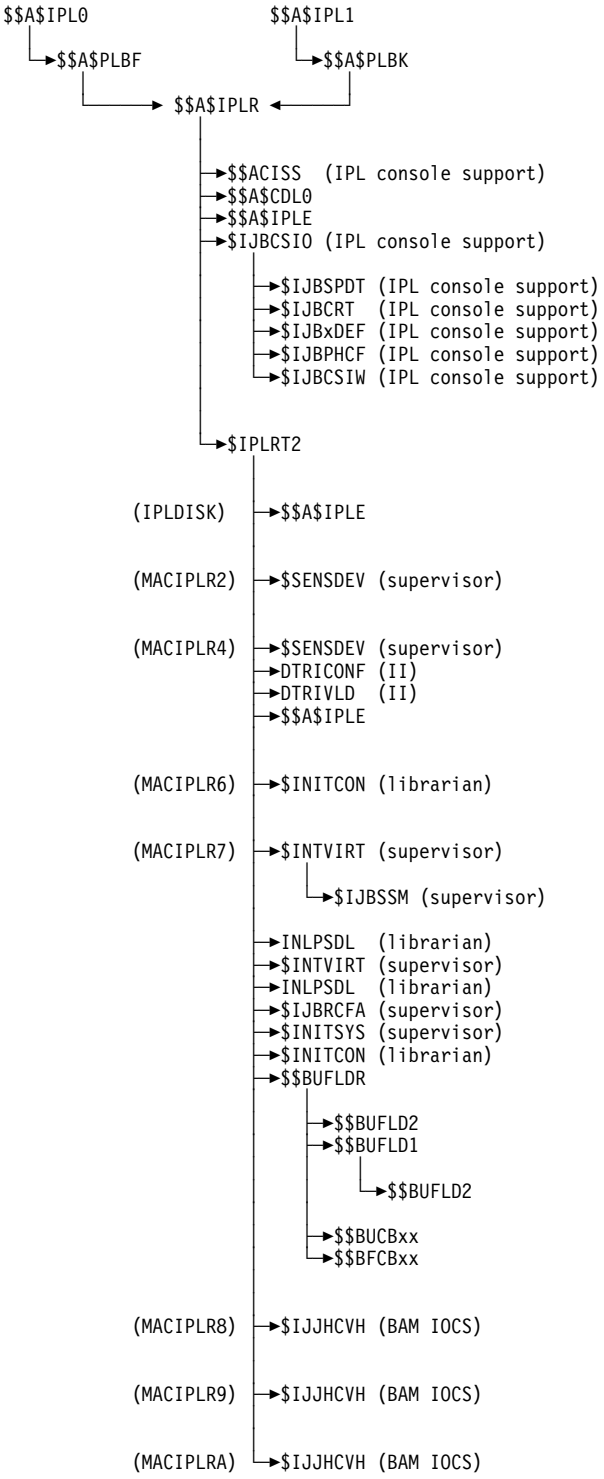


Figure 7. Control Flow from Phase to Phase for Online Environment

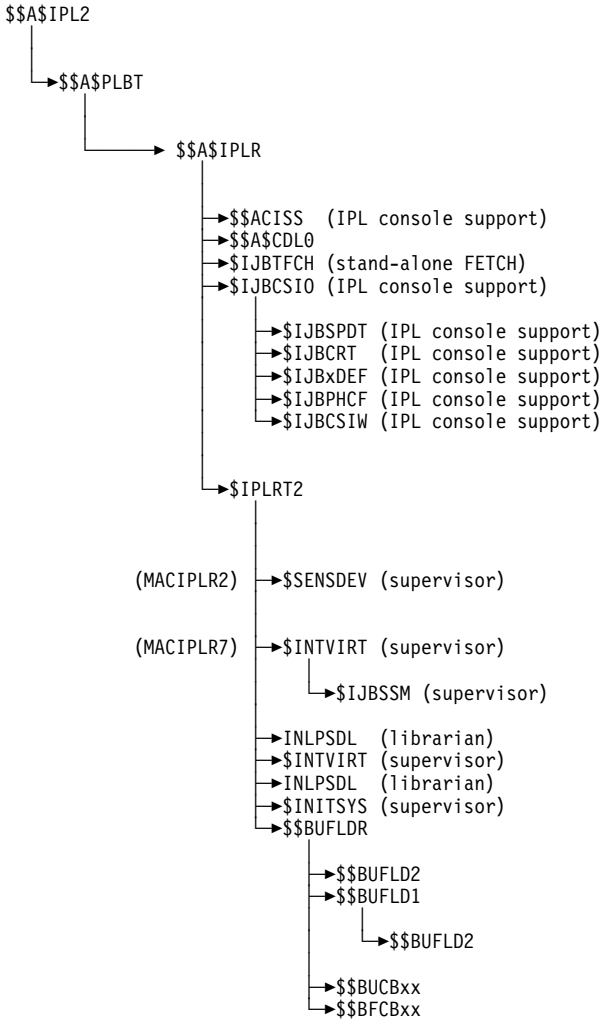


Figure 8. Control Flow from Phase to Phase for Stand-Alone Environment

Chapter 4. Data Areas -- Initial Program Load

Communication Areas

SVIPL -- Communication Area Between IPL and the Supervisor

This area has the name SVICOM as a DSECT in all IPL phases.

It is described via the MAPSVIPL macro; it contains the entry points of routines invoked from IPL for supervisor initialization and pointers to tables that have to be generated or initialized through IPL processing.

```
Location:        Dynamic supervisor area (temporarily)

Pointed to by:   IJBSVIPL in SYSCOM during IPL.

Initialized by:  MAPSVIPL macro at supervisor generation.

Changed by:      $$A$IPLR when IPL retrieves the supervisor

Used by:         $$A$IPLR when IPL retrieves the supervisor
                 $IPLRT2
                 $INTVIRT
```

Layout of MAPSVIPL

Loc.	Source Statement			
000000	SVICOM	DSECT		
000000	IPLSVLVL	DS	0F	
000000		DS	CL7	PRODUCT NAME
000007		DS	CL1	
000008		DS	CL3	RELEASE ID
00000B		DS	CL1	
00000C		DS	CL3	CLC CONSTANT
00000F		DS	CL1	
000010		DS	CL8	DEPENDENCY LEVEL
000018	SVISPGEN	DS	A	END OF GENERATED SUPV AREA
00001C	SVISPCON	DS	A	END OF DYNAMIC SUPERVISOR AREA - UPDATED SET BY IPL
000020	SVISPEND	DS	A	BEGIN OF IPL CONSOLE SUPPORT - UPDATED SET BY IPL
000024	SVISC1	DS	A	A(PUBSCAN TABLE1)
000028	SVIASC23	DS	A	A(A(PUBSCAN TABLE2,3))
00002C	SVICHTB	DS	A	A(CHANNEL CONTROL TABLE)
000030		DS	A	L(CHANNEL CTL TABLE ENTRY)
000034	SVIAVRT	DS	A	A(A(AVR-TABLE))
000048	SVIACBQ	DS	A	A(1ST COPY BUFFER AREA)
00003C	SVIACBQ2	DS	A	A(2ND COPY BUFFER AREA)
000040	SVIAICTL	DS	A	A(INITIALIZE CTL-REGS)
000044	SVIARCHQ	DS	A	A(A(RESERVED CHANQ ENTRIES))
000048	SVIRCHQ#	DS	F	#(RESERVED CHANQ ENTRIES)
00004C	SVISVACO	DS	XL4	SVA LOAD FLAGS
000050	SVIRQPMR	DS	A	A(PAGING RESOURCE QUEUES)
000054	SVIRQTBE	DS	A	A(A(END OF RQTAB))
000058	SVIPASIZ	DS	F	DEFAULT/MINIMUM PASIZE IN BYTE
00005C	SVIRSIZE	DS	F	DEFAULT RSIZE IN BYTES

Loc.	Source Statement			
000060	SVIBGMIN	DS	A	MINIMUM START OF IPL PARTITION
000064	SVIIOELL	DS	H	L(I/O EXT LOGOUT AREA)
000066	SVIMCELL	DS	H	L(MC EXT LOGOUT AREA)
000068		DS	A	RESERVED
00006C	SVIASSID	DS	A	A(SUPERVISOR SUBSID ENTRY)
000070	SVIAISHR	DS	A	A(DASD SHARING INIT ROUTINE)
000074	SVIADLFT	DS	A	A(DLF-TABLE)
000078	SVIVSIZE	DS	F	DEFAULT VSIZE VALUE IN K
00007C	SVIVSMIN	DS	F	MINIMUM VSIZE VALUE IN K
000080	SVIVSMAX	DS	F	MAXIMUM VSIZE VALUE IN K
000084	SVISPSIZ	DS	F	DEFAULT SPSIZE VALUE IN BYTES
000088	SVISPMIN	DS	F	MINIMUM SPSIZE VALUE IN K
00008C	SVIACCW	DS	A	A(A(BEGIN OF CCW AREA))
000090	SVIECCW	DS	A	A(A(END OF CCW AREA + 1))
000094	SVIADVCB	DS	A	A(A(DEVICE CONTROL BLOCKS))
000098	SVIVPOOL	DS	F	DEFAULT VALUE OF VPOOL IN K
00009C	SVIVIOMX	DS	F	MAXIMUM VALUE OF VIO IN K
0000A0	SVIENDAT	DS	A	A(INSTR MODIFIED BY INITDAT)
0000A4	SVIJCBSZ	DS	H	SIZE OF JC WORK AREA IN BYTES
0000A6	SVISLBSZ	DS	H	SIZE OF SLA WORK AREA IN BYTES
0000A8	SVIXPYAV	DS	A	A(A(APPC/VM VECTOR TABLE))
0000AC	SVILPIDT	DS	H	L(PATH ID TABLE ENTRY)
0000AE	SVICBL	DS	H	L(COPY-BUFFER)
0000B0	SVIACBE	DS	A	A(ADDRESSES OF COPY BUF FRAGM)
0000B4	SVIBFMIN	DS	F	MINIMUM BUFSIZE
0000B8	SVIBFDFL	DS	F	DEFAULT BUFSIZE
0000BC	SVIAPUBX	DS	A	A(A(PUBX VECTOR))
0000C0	SVISGVIS	DS	F	GENERATED SYSTEM GETVIS, BYTES
0000C4	SVISGDFP	DS	F	SYSTEM GETVIS FOR DASDFP, BYTE
0000C8	SVISGV31	DS	F	31 BIT SYSTEM GETVIS, BYTES
0000CC	SVIACHFP	DS	A	A(2 BYTE CHANQ FREE LIST PTR)
0000D0	SVIAPUB2	DS	A	A(A(PUB2 VECTOR))
0000D4	SVIAVPBG	DS	A	A(A(VPOOL BEGIN)
0000D8	SVIAVPND	DS	A	A(A(VPOOL END)
0000DC	SVIPFX24	DS	F	24 BIT SYSTEM PFIX
0000E0	SVIPFX31	DS	F	31 BIT SYSTEM PFIX
0000E4	SVIPAMIN	DS	F	MINIMUM PASIZE IN BYTES
0000E8	SVIVSLIM	DS	F	VSIZE>=THIS VALUE REQUIRES LARGER PASIZE MINIMUM
0000EC	SVIASFT	DS	F	A(MVS SFT)
0000F0		DS	6F	RESERVED

IOFLD -- IPL Communication Area

```
Location:       High storage end (before decompress phase and
                phase $$A$IPLE)

Initialized by: $$A$PLBK for CKD-SYSRES
                $$A$PLBF for FBA-SYSRES

Pointed to by:  A register

Changed by:     $$A$IPLE when ASI function is requested
                $$A$IPLR when IPL retrieves the supervisor
                $IPLRT2 (MACIPLR2, 8, 9, A) when the
                IPL communication device SYSUSE is temporarily
                switched
                $IPLRT2 (MACIPLR4) when temporary supervisor
                tables are set up

Used by:        $$A$IPLR when IPL retrieves the supervisor
                $$A$IPLE when ASI function is requested
                $IPLRT2 (IPLDISK, MACIPLR2, 3, 4, 5, 8, 9, A, B, C and D)
```

Use of IOFLD in ASI:

If one of the internal ASI functions is to be performed, the caller places the function's code number and the necessary parameters into IOFLD and branches to the entry point of this phase.

If it is the first call of the phase, the default names for the ASI procedures are set and the PSWs and CCWs are relocated. The function code is checked and if it is valid, the appropriate function is performed. An invalid function code will result in a NOP operation. A return code to the caller informs him of the completion of the function.

Layout of IOFLD

Loc.	Source Statement			
000000	IOFLD	DSECT		DSECT FOR IPL PHASES
000000	IPLRCOM	DS	0X	COMM. ARES FOR $$A$IPLR
000000	XACUUDA	DS	A	CUU-SSCH TABLE FOR ESA
000004	IPLIWK1	DS	CL2	CUU OF AUTOINST.SYSWK1
000006	IPLICON	DS	CL2	CUU OF AUTO INSTALL CONS
000008	IPLPGID3	DS	CL2	PATH GROUP ID SAVE AREA
00000A		DS	CL2	RESERVED
00000C	IPLINSTR	DS	F	ADDRESS OF TESTMON AREA
000010	IPLREOC	DS	F	END OF STORAGE ADDRESS
000014	IPLREOCS	DS	F	EOR SAVE AREA FOR RESTRT
000018	IPLSCCB	DS	A	SCLP SCCB ADDRESS
00001C	IPLRRES	DS	1H	X'CUU' OF SYSRES
00001E	IPLRVID	DS	CL6	SYSRES VOLUME-ID
000024	IPLRDVTP	DS	X	SYSRES PUB DEVICE TYPE
000025	IPLRDEV#	DS	XL2	SYSRES DEV TYPE NUMBER
000027	IPLRMOD#	DS	X	SYSRES DEV TYPE MODEL #
000028	IPLRFLG	DS	X	FLAG-BYTE
	#CONIC	EQU	X'80	INDICATE INTEGRATED CONS
	#CONLOC	EQU	X'40	INDICATE LOCAL CONS
	#SAMODE	EQU	X'20	STAND-ALONE MODE
	#ASILOAD	EQU	X'10	PHASE $$A$IPLE LOADED
	#FBARES	EQU	X'08	SYSRES IS FBA-DEVICE
	#CKDRES	EQU	X'04	SYSRES IS CKD-DEVICE
000029	IPLRFLG1	DS	X	FLAG-BYTE (DEV-SENSING)
	#INSTALL	EQU	X'80	INSTALLATION IPL
	#SENSE	EQU	X'40	SENSE ALL DEVICES
	#HDLXTIN	EQU	X'20	HANDLE EXT INTERRUPT (INTERRUPT KEY PRESSED) IF SOFT WAIT
	#BFYACT	EQU	X'10	BFY ACTIVE ON PROCESSOR
	#ESAACT	EQU	X'08	ESA PRESENT ON HARDWARE
	#SCLP	EQU	X'04	SCLP SUPPORTED ON PROC.
	#IPLSTOP	EQU	X'02	STOP FOR IPL PROMPTING
	#BLKFAIL	EQU	X'01	STORAGE DEFECT DETECTED
00002A	IPLRFLG2	DS	X	FLAG-BYTE (HW SUPPORT)
	#CUSE	EQU	X'80	CUSE INSTR SUPPORTED
	#CMPSC	EQU	X'40	CMPSC INSTR SUPPORTED
	#LOADPER	EQU	X'20	INCORRECT IPL LOAD PARM
	#NOMSG	EQU	X'08	SUPRESS IPL MESSAGES
	#STARTUP	EQU	X'04	PROMPT FOR STARTUP MD
00002B	IPLVMODE	DS	X	VIRTUAL MACHINE MODE
	VMVV	EQU	X'80	VIRT=V
	VMVR	EQU	X'40	VIRT=R OR VIRT=F
00002C	IPLRDEVC	DS	CL48	SYSRES DEVICE CHARACT
00005C		ORG	IPLRDEVC	CKD DEV CHARACTERISTICS
00002C	IPLRVTOC	DS	XL5	CC.HH.R OF BEG. OF VTOC
000031		DS	XL5	RESERVED
000036		DS	0H	
000036	IPLRRESC	DS	0XL10	SYSRES CAP. CHARACTER
000036	CYLPALT	DS	AL2	# OF CYLS+ALTERNATES
000038	TRKCYL	DS	AL2	# OF TRACKS/CYLINDER
00003A	TRKCAP	DS	AL2	TRACK CAPACITY
00003C	BLKTRK	DS	AL1	LB BLOCKS/TRACK
00003D	RESTYPE	DS	XL1	SYSRES PUB DEVICE TYPE
00003E		DS	XL2	RESERVED
000040		DS	XL28	RESERVED

Loc.	Source Statement			
00005C		ORG	IPLRDEVC	FBA DEV CHARACTERISTICS
00002C	IPLRFVTO	DS	XL4	FBA BEGIN BLK-NO (VTOC)
000030		DS	XL4	RESERVED
000034		DS	XL4	RESERVED
000038		DS	XL4	RESERVED
00003C		DS	XL4	RESERVED
000040		DS	XL4	RESERVED
000044		DS	XL4	RESERVED
000048	IPLRFRDC	DS	0XL18	FBA READ DEV CHAR. INFO
000048	IPLRFEAT	DS	XL4	VARIOUS FEATURES
00004C	IPLRFBSZ	DS	XL2	FBA-BLKSIZE
00004E		DS	XL4	#(BLKS PER CYCL. GROUP)
000052	IPLRFBFH	DS	XL4	#(BLKS UNDER FIX HEADS)
000056	IPLRFBMH	DS	XL4	#(BLKS UNDER MOV HEADS)
00005A		DS	XL2	RESERVED
00005C		ORG	IPLRDEVC	TAPE DEV CHARACTERISTICS
00002C	IPLRTAPE	DS	XL4	DEVICE TYPE AND MODE
00002E		DS	XL2	RESERVED
000030		DS	XL4	RESERVED
000034		DS	XL4	RESERVED
000038		DS	XL4	RESERVED
00003C	IPLRTBLK	DS	XL8	SA BLOCK HEADER
000044	IPLRTAP8	DS	XL4	USER RETURN REGISTER
000048	IPLRTIOS	DS	XL1	SAVED IOSW INFO
000049		DS	XL1	RESERVED
00004A	IPLRTRTY	DS	XL2	RETRY LIMIT
00004C	IPLRTCAW	DS	XL4	USER ORIGINAL CAW
000050		DS	XL12	RESERVED
00005C		ORG	,	
00005C	IPLDATA	DS	CL8	IPL LOAD PARAMETER
000064	IPLCPULS	DS	XL((TUM#ENTR+1)*2)	CPU LIST
	IPLRCOM#	EQU	(*-IPLRCOM+3)/4*4	LN OF COM AREA 1 ALIGNED TO FULLWORD
	ENDIPLR1	EQU	*	END OF REL 2.1 IPLRCOM
00007A	IOLOAD	SVC	4	LOAD $IPLRT2
00007C		BR	1	TRANSFER CTL TO $IPLRT2
00007E	IPLLOG	DS	1H	DEVADR X'CUU' OF CONS
000080	DOCFLG	DS	0H	SCREEN SUPPORTFLAG
000080	DOCTYPE	DS	X	SYSLOG DEVICE TYPE
000081	DOCSUP	DS	X	SYSLOG SCREEN SUPPORT
	LOGFLAG	EQU	X'08	LIST IPL CMDS AT SYSLOG
000082	IPLCOM	DS	1H	STORAGE FOR COMMUN DVC
000084	ASIPARM	DS	0F	AUTOM.SYSTEM.INIT. HOOK
000084	ASIFLAG	DS	AL1	ASI-FLAG-BYTE
	ASI#PROC	EQU	X'80	ASI IN PROCESS
	ASI#REST	EQU	X'40	ASI TO BE RESTARTED
	ASI#KEEP	EQU	X'20	RESERVED
	ASI#TERM	EQU	X'10	ASI TERMINATED
	ASI#ERR	EQU	X'08	ASI INPUT ERROR
	ASI#SUP	EQU	X'04	ASI SUPVR AVAILABLE
	ASI#DEAC	EQU	X'02	ASI RESTART DEACTIVATED
	ASI#RELO	EQU	X'01	ASI CCW'S & PSW'S RELOC.
000085	ASIFLAG1	DS	AL1	ASI-FLAG-BYTE
	ASI#KEYW	EQU	X'80	ASI KEYWORD IN OP-REPLY
	ASI#JCLF	EQU	X'40	ASI JCL PROC FAULT
	ASI#STOP	EQU	X'20	ASI STOPS AT IPL PROC.
	ASI#SPRM	EQU	X'10	ASI STOPS AT SUP PARM C
	ASI#STOV	EQU	X'08	FLAG OVERRIDE ASI STOPS

Loc.	Source Statement			
	ASI#TYOV	EQU	X'04'	FLAG OVERRIDE ASI TYPE
	ASI#ATTN	EQU	X'02'	REMEMBER SYSLOG ATTNTN
	ASI#PN	EQU	X'01'	REMEMBER P/N OPERAND
000086	ASIFUNC	DS	AL1	ASI-FUNCTION
	ASIFINIT	EQU	X'00'	ASI INITIALIZATION
	ASIFCHCK	EQU	X'04'	ASI CHECK FOR PROCEDURE
	ASIFGET	EQU	X'08'	ASI GET NEXT RECORD
	ASIFENA	EQU	X'0C'	ASI ENABLE RESTART
	ASIFDEAC	EQU	X'10'	ASI DEACTIVATE RESTART
000087	ASIRC	DS	AL1	ASI-RETURN-CODE
	ASIRC0	EQU	X'00'	ASI SUCCESSFUL COMPLET.
	ASIRC4	EQU	X'04'	ASI END OF DATA
	ASIRC8	EQU	X'08'	ASI PROCEDURE NOT FOUND
	ASIRCC	EQU	X'0C'	ASI SYSRES I/O-ERROR
000088	ASIENTRY	DS	A	ASI-ENTRYPOINT
00008C	ASIIPLR	DS	A	ASI-$$A$IPLR ENTRY POINT
000090	ASISTOPN	DS	H	ASI NO OF STOP'S
000092	ASISTOPC	DS	CL12	ASI IPL STOP COMMANDS
00009E	ASIFREE	DS	XL2	ASI FREE AREA
0000A0	ASIXNPSW	DS	XL8	ASI-SUPVR EXT. NEW PSW
0000A8	ASIIPLN	DS	CL8	ASI IPL PROCEDURE NAME
0000B0	ASIJCLN	DS	CL8	ASI JCL PROCEDURE NAME
0000B8	ASIWORK	DS	XL80	ASI-WORKAREA
000108		ORG	ASIWORK	ASI PROC FIRST RECORD
0000B8	ASILOG	DS	CL3	ASI-SYSLOG-CUU
0000BB		DS	CL1	COMMA
0000BC	ASISUP	DS	0C	ASI-SUPNAME,OPTIONS
0000BC		ORG	ASIWORK+L'ASIWORK	RESET LOCATION COUNTER
000108	PUBSCT1	DS	A	A(1.LEV PUBSCAN-TBLE)
00010C	PUBSCT2	DS	A	A(2.LEV PUBSCAN-TBLE)
000110	PUBSCT3	DS	A	A(3.LEV PUBSCAN-TBLE)
000114	IPLCIADR	DS	F	ADDRES OF $$ACISS
000118	DKETLAB	DS	CL80	DISKETTE LABEL AREA
000168		DS	XL5	RESERVED
00016D	FOCLFLD	DS	XL(MAXCHN)	FIRST ON CHANNEL
00017D	FICLFLD	DS	XL(MAXPART+1)	FIRST IN CLASS
00018D	NICLFLD	DS	XL(MAXPART+1)	NO IN CLASS
00019E		DS	0H	FORCE BOUNDARY
00019E	TABAVR	DC	XL(VCTCLEN)'00'	DUMMY AVRTBLE FOR IPL
00020C	FTTRES	DC	XL(VCTCLEN)'00'	FETCH GETVCE-AREA
00027A	SAVIOELL	DS	H	L(I/O EXT LOGOUT AREA)
00027C	SAVMCELL	DS	H	L(MCH EXT LOGOUT AREA)
	****			SAVED FOR RESTART
00027E	ASIJCLSV	DS	CL8	ASI JCL PROCEDURE NAME
000286	ASISTCSV	DS	CL12	ASI IPL STOP COMMANDS
000292	ASISTNSV	DS	H	ASI NUMBER OF STOPS
000294	ASITYPSV	DS	X	ASI IPL TYPE SAVE AREA
000295	ASISVFLG	DS	X	ASI PARM OVERRIDE FLAG
	ASI#SAVE	EQU	X'80'	VALID SAVE INFORMATION
000298	IPLRCOM2	DS	0F	
000298	BEGIPLR2	DS	0F	BEGIN OF 2ND PART OF $$A$IPLR COMM AREA
000298	LIBFLAG	DS	XL1	LIBRARY IN ERROR FLAG
	HORIZONR	EQU	X'80'	HORIZONTAL READ
	NOPHVIF	EQU	X'40'	PHASE VIF MISSING
000299	LOG2LBSZ	DS	XL1	LOG2(LBSIZE)
00029A	LBSIZE	DS	H	LIBRARY BLOCK SIZE
00029C	LBCFL	DS	H	LENGTH OF LBCF

Loc.	Source Statement			
00029E	LBCFDISP	DS	H	LBCF OFFSET IN LB
0002A0	ADDRIPLE	DS	A	$$A$IPLE LOAD ADDR
0002A4	ACOMPRES	DS	A	A(COMPRESS ROUTINE)
0002A8	LCOMPRES	DS	H	L(COMPRESS ROUTINE)
	LVIFPHAS	EQU	UPHAPLN+4-UPHAELEN	L(REDUCED PHASE VIF)
	LPHDESC	EQU	DENTDLE1+LVIFPHAS	REDUCED PHASE DESCRIPTOR LENGTH
0002AA	CMPRDESC	DS	XL(LPHDESC)	COMPR RTN RED MEMB DESC
0002E0	RDESC	DS	XL(LPHDESC)	$$A$IPLR RED MEMB DESC
000316	EDESC	DS	XL(LPHDESC)	$$A$IPLE MEMBER DESCR
00034C	SYSLIB1D	DS	XL(SLXELEN)	SYSLIB DESCRIPTOR
	LPLRCOM1	EQU	(*-BEGIPLR2)/256*256	0 IF 2ND PART LENGTH 256 256 IF 256 LENGTH 512 ELSE ASSEMBLY ERROR
	LPLRCOM2	EQU	*-BEGIPLR2-LPLRCOM1	ND PART LENGTH MODULO 256
	ONEFUNCT	EQU	(LPLRCOM2+2)/(LPLRCOM2+1)-1	0 IF LPLRCOM2>0 1 IF LPLRCOM2=0
000394		DS	(ONEFUNCT)X	ZERO BYTE IF LPLRCOM2=0
	ENDIPLR2	EQU	*	END OF 2ND PART OF $$A$IPLR COMM AREA
	IOFLD1	EQU	*	
000394		ORG	IOFLD1	
000394	VPOOLSIZ	DS	F	VPOOL SIZE IN K-BYTES
000398	VIOSIZ	DS	F	VIO SIZE IN K-BYTES
		SNSDVIP		MACRO: PARAMETERS FOR INVOKING DEVICE SENSING
0003B8	MSG04SAV	DS	CL52	SAVE AREA FOR MSG 0I04I
0003EC	MSSUPSAV	DS	CL76	SAVE A FOR SUP PARM CMD
	IOFLD2#	EQU	((*-IOFLD)/1024+1)*1024-4-64	ADDR OF ASISAVE
000438		ORG	IOFLD+IOFLD2#	
000440	ASISAVE	DS	XL64	SAVE AREA FOR $$A$IPLE
000480	APUBFLD	DS	F	ADDRESS OF SHADOWPUB
000484	ASIDFLD	DS	F	ADDR OF SENSE ID TABLE
000488		DS	H	PADDING
00048A	LUBLNG	DS	1H	STORAGE FOR LUB LNG
00048C	LUBFLD	DS	1024C	STORAGE FLD FOR LUBS
00088C	PUBLNG	DS	1H	STORAGE FOR LUB LNG
	#PUBS	EQU	65535	NUMBER OF PUB ENTRIES
000890	PUBFLD	DS	(#PUBS+1)CL(PUBWIT)	STORAGE FOR PUBS
	LPUBFLD	EQU	*-PUBFLD	LENGTH OF TEMP. PUB
080890	SIDFLD	DS	(#PUBS+1)CL(DTSLNGTH)	STORAGE FOR PUBS
	LSIDFLD	EQU	*-SIDFLD	LENGTH OF SID TABLE
	IOFLD#	EQU	*-IOFLD	LENGTH ROUNDED TO 1K

IPLRCOM -- $$A$IPLR Communication Area

Location: At bootstrap time within $$A$PLBK/$$A$PLBF/$$A$PLBT; then, two parts of IOFLD, the IPL communication area, labelled IPLRCOM and IPLRCOM2

Pointed to by: A register

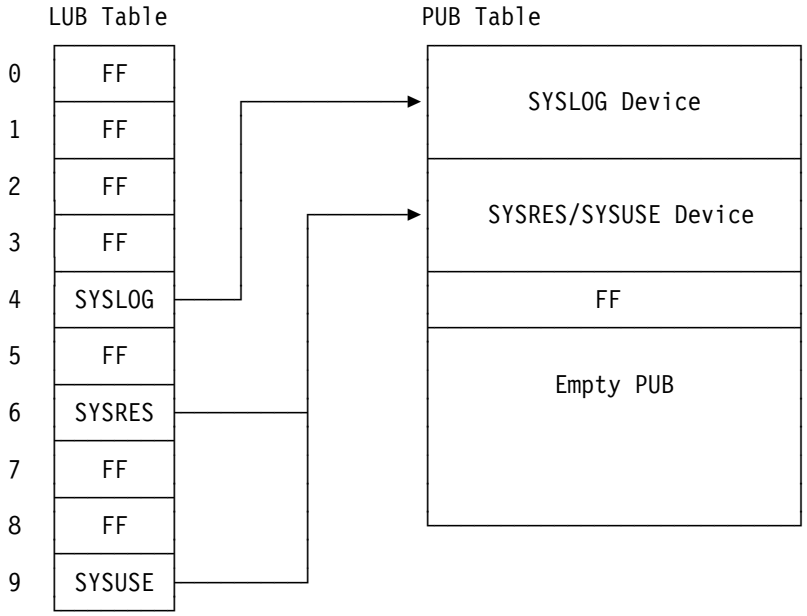
Initialized by: $$A$PLBK for CKD-SYSRES
$$A$PLBF for FBA-SYSRES
$$A$PLBT for tape-SYSRES

Changed by: $$A$IPLR when IPL retrieves the supervisor

Used by: $$A$IPLR when IPL retrieves the supervisor
$$A$IPLE when an ASI function is requested

The I/O Tables

Figure 9 shows the I/O tables for the 3-device system generated at the beginning of IPL.



Note: It is assumed that the IPL command source (SYSUSE) is an IPL ASI procedure.

Figure 9. I/O Tables for the 3-Device System

Chapter 5. Diagnostic Aids -- Initial Program Load

Interface Information

Interfaces of ASI with Other Components

External interrupts with interrupt code X'40' (interrupt key pressed) are handled by the ASI program, while all other external interrupts are passed to the supervisor: the preserved supervisor external new PSW is copied into RESTART old PSW; RESTART old PSW is made the current PSW.

Each partition communication region COMREG serves as an interface area between ASI and job control and receives the following values:

```
COMREG.PROCNAM:      Name of JCL ASI procedure, 1-8 bytes.

COMREG.JCLSW5.ASIPL: Flag is set, if ASI was started.
```

Interface Between IPLDISK ($IPLRT2 common code) and Its Callers

```
(The names of the registers vary.)

R 7: Pointer to IOFLD
R 8: Pointer to background COMREG
R14: Base register of common code (IPLDISK)
R15: Base register of command processing phase
```

Registers Passed from $$A$PLBF to $$A$IPLR

```
R10: $$A$IPLR entry point
R14: Pointer to IOFLD
```

Interface Between $$A$IPLE and Its Callers

For the interface between $$A$IPLE and its callers, see in Chapter 4, “Data Areas -- Initial Program Load” on page 103 the description of IOFLD. The registers used in this interface are:

```
R 6: Return register
R 7: Pointer to IOFLD
```

Interfaces Between MACIPLR7 and $INTVIRT

First call of $INTVIRT

Input:	R1	Address of parameter list

```
Format of parameter list (storage sizes in bytes):
  VPSDL    DS  F   NUMBER OF SDL ENTRIES
  VPSIZE   DS  F   SIZE OF 24-BIT SVA MODULE AREA
  VPGETVIS DS  F   24-BIT SYSTEM GETVIS REQUIREMENT
  VSPSIZE  DS  F   SIZE OF SHARED PARTITIONS
  VPSIZE31 DS  F   SIZE OF 31-BIT SVA MODULE AREA
  VPGETV31 DS  F   31-BIT SYSTEM GETVIS REQUIREMENT
  VPPAMBG  DS  F   MINIMUM PA START
  VPPASIZE DS  F   SIZE OF PRIVATE AREA
  VPRSIZE  DS  F   SIZE OF ALLOC R AREA
  VPIJBSSM DS  F   ENTRY POINT OF IJBSSM
```

	R13	Address of save area
Output:	R15=0	Normal return - 24-bit SVA undefined
	R1=Size of private area in K	
	R15=4	Insufficient real storage below 16M for RSIZE
	R0=Reduced RSIZE in K	
	R1=Size of private area in K	
	R15=8	Insufficient virtual storage
	R0=Minimum virtual storage in K	
	R15=12	Insufficient processor storage
	R0=minimum real storage in K	
	R15=16	24-bit shared area larger than 15 M
	R0=Actual size of 24-bit shared area in K	
	R15=20	Private area too small to accomodate BG partition
	R15=24	IPL code too large to fit into BG partition
	R15=28	IPL code overlaps private area

Second call of $INTVIRT

Input:	R1=0	
Output:	R15=0	Normal return - 24-bit SVA initialized

Interfaces Between MACIPLR7 and INLPSDL

First call of INLPSDL

Input:	R0	Address of SVA header
	R1	Address of SDL header
	R13	Address of 24K work area
	STOW	'I' for online environment
	STOW	'S' for stand-alone environment
Output:	R15=0	Normal return, 31-bit SVA loaded
	R15=8	SDL full, message issued by INLPSDL
	R15=16	SVA full (stand-alone environment), message issued by INLPSDL

Second call of INLPSDL

Input:	R0	Address of SVA header
	R1	Address of SDL header
	R13	Address of 24K work area
	STOW	'O' for online environment
	STOW	'T' for stand-alone environment
Output:	R15=0	Normal return, 24-bit SVA loaded
	R15=8	SDL full, message issued by INLPSDL
	R15=16	SVA full (stand-alone environment), message issued by INLPSDL

Note: STOW is the librarian term for the type of action to be taken

Interface Between IPL and the Console Support

IPL Bootstrap Processing - Console Support Stage I

IPL determines if the system console is a local device or an integrated console.

System console is a CRT or line mode console:

- I/O is performed by the IPL stand-alone I/O routine.

System console is the Integrated Console:

- IPL messages are sent to and received from the integrated console by the synchronous IC support $$ACISS.

  Note: The synchronous version of the Integrated Console support is active until the supervisor is loaded, and the FETCH service is initialized.

Phase $$A$IPLR loads phase $$ACISS and then initializes the integrated console support with the first SEND request (message 0I04I IPLDEV=...).
All SEND, RECEIVE requests are synchronous. Control is not returned to IPL before the messages are actually sent or the responses are present. During IPL bootstrap processing the messages are not yet buffered.

The interface between IPL phase $$A$IPLR and IC support phase $$ACISS is described below:

- System state when control transferred to $$ACISS:

```
A-mode 31
Supervisor state
PSW key zero
External interrupts disabled
I/O interrupts disabled
Storage is V=R
DAT off
```

- Input to $$ACISS:

```
R1  --> Parameter list CIRPL
        CIRSCCB --> A(SCCB allocated by IPL)
        CIRFCT  --> CIRSEND
        CIRADAT --> A(message)
        CIRLDAT --> l(message)  1
        CIRSAVE --> 72-byte save area
        or
        CIRSCCB --> A(SCCB allocated by IPL)
        CIRFCT  --> CIRRECV
        CIRADAT --> A(reply area)
        CIRLDAT --> l(reply area)  2
        CIRSAVE --> 72-byte save area
R15 --> Load point of $$ACISS
BASSM   R14,R15     transfer control
```

  1 The maximum length of IPL messages is 76 bytes.
  2 The reply area supplied by IPL is 128 bytes long.

- Output from $$ACISS:

```
R15     --> Return code
CIREC   --> Non-zero return code: error code
CIRLDAT --> Receive: actual reply length
Control is returned in callers AMODE (BSM  0,R14)
```

- A non-zero return code causes the following action:
  - first call (initialization): addressing of a CRT or line mode console,
  - all subsequent calls: termination with a hard wait code.
    The hard wait codes are documented in Figure 13 on page 122.

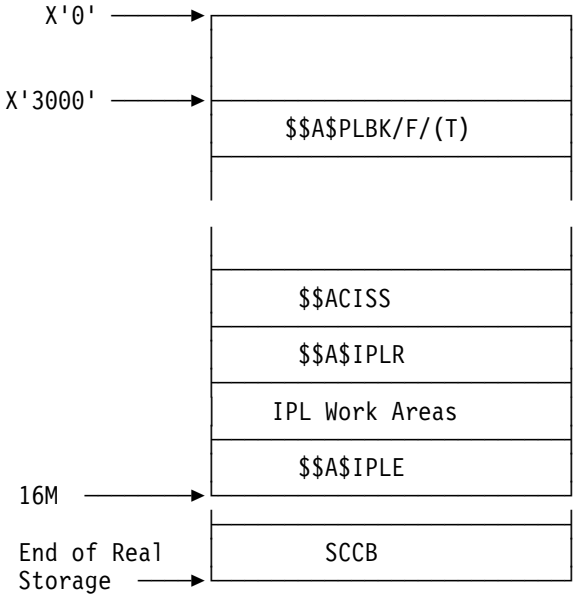


Figure 10. Map during IPL Bootstrap Processing - Console Support Stage I, V=R

IPL Command Processing - Console Support Stage II: When the supervisor FETCH/LOAD service is initialized, the console router, the NLS support, the hard copy support, and either the CRT support or the integrated console support depending on the type of system console, are loaded at a fixed location in storage, from 2M downwards. IPL will control, that the supervisor - excluding the channel queue and copy buffers - does not grow beyond the start of the console support. During IPL the supervisor operates with a small temporary channel queue and with a few preallocated copy buffers. Channel queue and copy buffers as specified during IPL are formatted at the end of IPL, and the space may overlay the IPL console support.

At this time the console support is activated with pre-allocated buffers, and it remains operational until the console support is finally initialized in the 31-bit SVA. The code and data areas are located within 24-bit shared space, because no other address space remains addressable during initialization of virtual storage (first call of $INTVIRT). The console support finally resides in the 31-bit SVA.

When the supervisor is enabled for I/O, but the stage II console support is not yet activated, the supervisor has to be prevented from handling console requests. This is achieved by initially setting the SYSLOG assignment to 'IGNORE'.

If the system console is a CRT or line mode console:

- IPL loads CRT phase $IJBCRT,
- anchors the entry address into SUPAVT.SVASVDL.AIJBCRT.

If the system console is an integrated console:

- IPL creates a dummy PUB for the integrated console for use by the system.
  - A device number not defined in the IOCDS, and a subchannel number regarded invalid by VSE (X'FFFF') are selected.
  - The PUB is built with the following values:
    - device type 3277
    - dummy device indication in option flag
  - SYSLOG assignment and IJBOCDEV needed by the supervisor SVC 0/15 intercept initially point to this PUB.
- The integrated console 'cuu' is put into SYSCOM.IJBCIDEV.
- Flag 'integrated console exists'is set: SYSCOM.IJBFLG08.IJBCIEXI.
- The integrated console support phase $IJBSPDT is loaded.
- Its entry address is anchored into SUPAVT.SPDTTAB.SPSGVTAB.
- The SCCB address is passed: SUPAVT.SPDTTAB.SPSGSCBO.

For both types of system console:

- NLS phase $IJBxDEF is loaded.
  Character x is:
  - G for German,
  - S for Spanish,
  - J for Japanese.
  - E for English,
    Since $IJBEDEF is initially shipped with the system, it may be in the library in addition to another NLS module. Therefore IPL tries to retrieve the non-English NLS phases $IJBGDEF, $IJBSDEF or $IJBJDEF first.
- The determined NLS identifier is put into SYSCOM.IJBNLSIC.
- Hard copy phase $IJBPHCF is loaded.
- Hard copy support is anchored in SUPAVT.SVASVDL.AIJBPHCF.
- The console router work area $IJBCSIW is loaded.
- The console router phase $IJBCSIO is loaded.
- The console router initialization (entry point: load point of $IJBCSIO) is called.
  - System state when control transferred to console router initialization:

```
A-mode 31
Supervisor state
PSW key zero
External interrupts disabled
I/O interrupts disabled
Storage is V=R
DAT off
```

  - Input to console router initialization:

```
SYSLOG assigned to system console
R1  --> Parameter list MAPCORIP
        CORIFUNC --> Initialize from scratch
        CORIWRKA --> A(CORIWA)
R13 --> CORIRSAV --> 72-byte save area
R15 --> Load point of console router phase $IJBCSIO
BASSM   R14,R15     Transfer control
                    Router will activate either CRT or IC, and HC tasks.
```

– Output from console router:

```
R15     --> Return code
COREC   --> Non-zero return code: error code
Control is returned in callers AMODE (BSM 0,R14)
```

– A non-zero return code causes termination with a hard wait code. The hard wait codes are documented in Figure 13 on page 122.

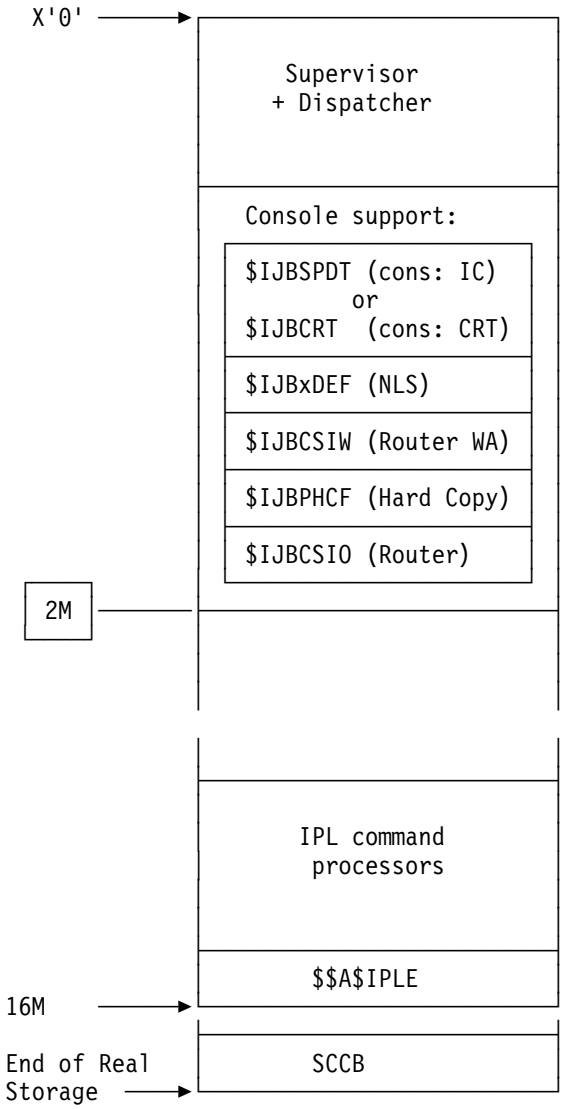


Figure 11. Map at the End of IPL Bootstrap Processing - Console Support Stage II, V=R

Final Console Support Initialization - Console Support Stage III: Virtual storage is initialized in two stages and the 31-bit SVA and the 24-bit SVA are loaded seperately. First all storage excluding the 24-bit shared area is initialized, and IPL is moved to a virtual partition, that may not start below 3M. Then the 31-bit SVA is loaded. Until this time the IPL console support is operational. The console support in the 31-bit SVA receives control, and it performs the transition from the IPL console support to its SVA copy. The storage occupied by the console support during IPL becomes available. After sucessful reinitialization of the console support the 24-bit shared storage is initialized, the 24-bit SVA is loaded, and the areas reserved for the channel queue and for copy buffers are formatted.

Loading occurs tranparently by the normal SVA load procedure. The following modules are included in the system load book $SVACSC

- Console router ($IJBCSIO)
- Integrated console ($IJBSPDT)
- CRT console ($IJBCRT)
- CMS console ($IJBVMCF) - on condition that VMCF is requested
- NLS support ($IJBxDEF) - x being replaced by the NLS identifier G, S, J or E from the SYSCOM
- Hard copy support ($IJBPHCF)

IPL calls the console router initialization (entry is the load point of phase $IJBCSIO) after the 31-bit SVA is loaded.

- State when control transferred to console router initialization:

```
A-mode 31
Supervisor state
PSW key zero
External interrupts enabled
I/O interrupts enabled
Storage is V=V (page faults may occur)
DAT on
```

- Input to console router initialization:

```
R1  --> Parameter list MAPCIRIP
        CORIFUNC --> Initialize from existing queue
        CORIWRKA --> Zero
R13 --> CORIRSAV --> 72-byte save area
R15 --> Load point of console router
BASSM   R14,R15     Transfer control
                    Router will then post CRT, IC, and HC tasks.
```

- Output from console router:

```
R15     --> Return code
COREC   --> Non-zero return code: error code
Control is returned in callers AMODE (BSM  0,R14)
```

- A non-zero return code causes termination with a hard wait code. See Figure 13 on page 122.

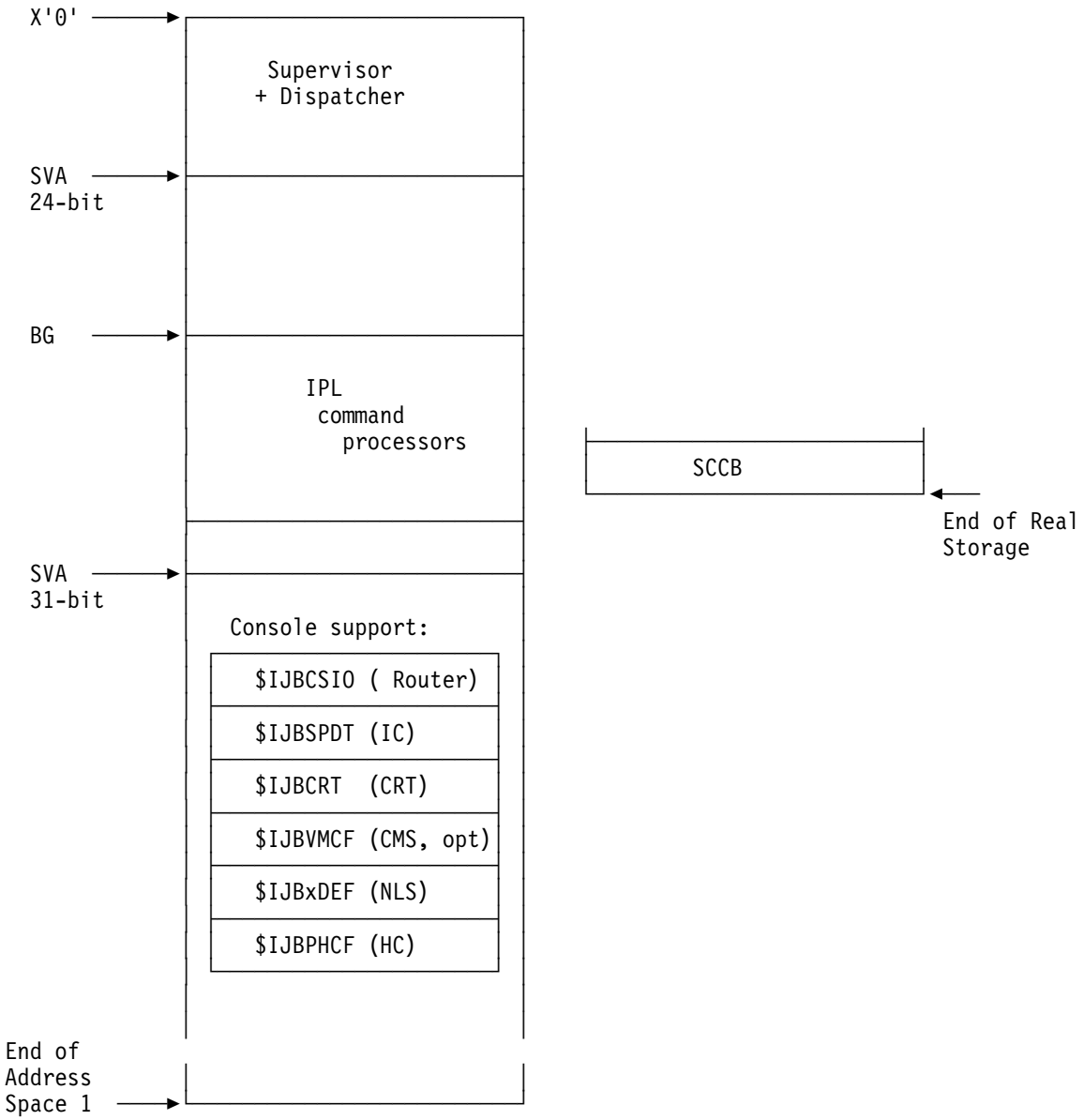


Figure 12. Virtual Storage Map at the End of IPL - Console Support Stage III

Wait Codes for IPL in Low Storage

The table below lists the hard and soft wait codes that are caused by IPL. All other wait codes that may occur during IPL are described in *VSE/ESA Messages and Codes*.

If there is an equipment malfunction during IPL, or if the IPL program cannot be loaded, a message is placed in storage starting at location 0. In this state, all interrupts are disabled and IPL must be repeated after these bytes have been displayed.

Byte 0	Byte 1	Byte 2	Byte 3	Explanation
X'07'	X'E6'	Device number		IPL input/output error: - I/O error on SYSRES - I/O error on communication device[1]
X'07'	X'E6'	X'C3'	X'E2'	Console router error 'CS': Byte 4,5: 2 bytes return code Byte 6,7: 2 bytes error[2]
X'07'	X'E6'	X'C9'	X'C3'	Integrated Console error 'IC': Byte 5: 1 byte return code Byte 6,7: 2 bytes error[2]
X'C1'	X'E2'	not used	not used	Unrecoverable machine check
X'cc'	X'00'	X'0F'	X'D0'	Error during IPL. IPL canceled. (cc=cancel code)[3]
X'F0'	X'C9'	X'F0'	X'F0'	See message 0I00
X'F0'	X'C9'	X'F0'	X'F1'	See message 0I01
X'F0'	X'C9'	X'F0'	X'F6'	The device type of SYSRES can not be identified. The volume label (VOL1) or format–4 record contains invalid information. The pack was not initialized correctly.
X'F0'	X'C9'	X'F0'	X'F7'	See message 0I07
X'F0'	X'C9'	X'F1'	X'F4'	Unexpected return from SCLP See message 0I14

Figure 13 (Part 1 of 2). Wait Codes for IPL

Byte 0	Byte 1	Byte 2	Byte 3	Explanation
X'F0'	X'C9'	X'F5'	X'F4'	Phase not found, phase name is appended. See message 0I54
X'F0'	X'C9'	X'F6'	X'F8'	Unsupported Hardware. See message 0I68, RC=2
X'F0'	X'D1'	X'F1'	X'F7'	Too many devices in IOCDS. See message 0J17
X'F0'	X'D1'	X'F5'	X'F0'	Unsupported SYSLOG device. See message 0J50[4]

Figure 13 (Part 2 of 2). Wait Codes for IPL

---

1 When IPL procedure reaches the normal IPL wait state, and the IPL communications device is to be SYSLOG, press the request key on the console keyboard.

2 The internal Cancel and Error Codes are documented in the appropriate integrated console or console router modules (OCO).

3 For description of the Cancel Codes see *VSE/ESA Messages and Codes*, Appendix 3.

4 The device type of the SYSLOG device cannot be identified. Press END/ENTER on a console keyboard which is supported by VSE/ESA.

Cross-References

The following cross-references are provided to assist in problem determination.

- Command to phase
- Message to phase
- Phase to module

Command-to-Phase Cross-Reference

In the following list only the final command processors are mentioned. All IPL commands are first evaluated in macro IPLDISK. The commands are listed here in the approximate sequence of their entry.

Command	Phase	Macro
ADD	$IPLRT2	MACIPLR3
DEL	$IPLRT2	MACIPLR3
SET	$IPLRT2	MACIPLR4 (SET DATE)
	$IPLRT2	MACIPLRD (SET APPC/VM)
	$IPLRT2	MACIPLRE (SET ZONEBDY/ZONEDEF)
DLF	$IPLRT2	MACIPLRA
DEF	$IPLRT2	MACIPLR5
DLA	$IPLRT2	MACIPLR8
DPD	$IPLRT2	MACIPLR9
SYS	$IPLRT2	MACIPLRB
SVA	$IPLRT2	MACIPLR6

Message-to-Phase/Macro Cross-Reference

For easier reading, the message numbers after the first of each subcomponent group are listed by their two digit numerals only. For the same reason, the phase names are listed by their last character or numeral only if the rest of the name is the same as for the preceding phase name.

Message	Phase/Macro
0I00	$$A$IPLR, $$A$PLBK,F,T
01	$$A$PLBK,F
02 - 05	$$A$IPLR
06	$$A$PLBK
07	$$A$IPLR, $$A$PLBK,F,T
08	$$A$IPLR
09	MACIPLR3
10 - 11	IPLDISK
12 - 13	MACIPLR3
14	$$A$PLBK,F,T
15	MACIPLR3
16	MACIPLR3
17	MACIPLR2
18	IPLDISK
19	MACIPLR4
20	MACIPLR7
21	IPLDISK
22	MACIPLRB
23	MACIPLR4
24	$$A$IPLR
25	MACIPLRB
26	$$BUFLDR,2
27	$$BUFLD2
28	$$BUFLDR
29	MACIPLR4,6,C
30 - 32	MACIPLR2
36	IPLDISK
37 - 39	MACIPLR8,9,A
40	MACIPLRC
41	MACIPLR5,8,9,A
42 - 46	MACIPLR8,9,A
47	MACIPLR5,8,9,A
48	MACIPLR5
49 - 50	IPLDISK
51 - 52	MACIPLR8,9,A
54	$$A$IPLR, MACIPLR2
58	MACIPLRC
59	MACIPLR4
60	IPLDISK
63	MACIPLRC
64 - 65	IPLDISK
66	MACIPLR2
68	$$A$IPLR, $$A$PLBK,F,T
0I69 - 70	$$BUFLDR
71	MACIPLR4
73	MACIPLR8,9,A
74	MACIPLR7
75 - 79	MACIPLR8,9,A
80	MACIPLR8,A
82	MACIPLR7

Message	Phase/Macro
83 - 85	MACIPLR6
86	MACIPLR4,5,6,8,9,A,B,D
87	MACIPLR2,4,5,6,8,9,A,B,E
88	MACIPLR4,5,6,8,9,A,B,E
89	MACIPLR4,5,8,9,A,B,E
90 - 91	MACIPLR6
93	MACIPLRC
94	MACIPLR2,6,7,8,9,A
95	MACIPLR6
96	$$A$IPLR
97	MACIPLR8,A
99	MACIPLR8,9,A
0J01	MACIPLR2
02 - 04	$$A$IPLR
05	$$A$IPLR, IPLDISK
06	MACIPLR7
07 - 09	$$A$IPLR
10	IPLDISK, MACIPLR4
11 - 13	MACIPLR9
14	IPLDISK, MACIPLR9
16	MACIPLR9
17	$$A$IPLR
18	MACIPLR7
19	MACIPLR6,7
20	MACIPLR2,4
21	MACIPLRB,C
22	MACIPLR3
23 - 24	IPLDISK
26	IPLDISK
27 - 31	MACIPLRA
32	$$A$IPLR
33	MACIPLRA
34	$$A$IPLR
36	MACIPLRA
37	MACIPLRB
39	MACIPLR7
42 - 43	MACIPLRB
44	MACIPLR7
45	MACIPLR7
46	MACIPLR4
47	MACIPLR2
48	MACIPLR3
49	IPLDISK
50	$$A$IPLR
51	MACIPLR3
55 - 58	MACIPLRD
59	MACIPLR7
61	MACIPLRB
62	MACIPLRC
64	MACIPLRB,C

Message	Phase/Macro
65	$$A$IPLR
66	MACIPLR2,6
67 - 70	MACIPLR7
71	MACIPLR3
72	MACIPLR9
73 - 74	MACIPLR7
76	MACIPLR7
77	$$A$IPLR
78	MACIPLR4
79	MACIPLR3
80 - 83	MACIPLRE
1B01 - 06	SYSBUFLD
10 - 11	SYSBUFLD
20	SYSBUFLD

Phase-to-Module Cross-Reference

This is a complete list of all phases of the IPL component. Those phases which are not documented in this manual have a reference to the manual in which they are described.

Phase	Module
'IPL Function'	
$$A$IPLE	$$A$IPLE
$$A$IPLR	$$A$IPLR
$$A$PLBF	$$A$PLBF
$$A$IPL0	$$A$PLBF
$$A$PLBK	$$A$PLBK
$$A$IPL1	$$A$PLBK
$$A$PLBT	$$A$PLBT
$$A$IPL2	$$A$PLBT
$IPLRT2	IJBIPL
'$BUFLDR Function'	
$$BATTF0[5]	$$BATTF0
$$BATTF2[5]	$$BATTF2
$$BATTF3[5]	$$BATTF3
$BFCB[6]	$$BFCB
$$BFCB00[6]	$$BFCB00
$$BFCB10[6]	$$BFCB10
$$BFCB22[6]	$$BFCB22
$$BFCB23[6]	$$BFCB23
$$BFCB3[6]	$$BFCB3
$$BFCB5[6]	$$BFCB5
$$BUCB[6]	$$BUCB
$$BUCB00[6]	$$BUCB00
$$BUCB10[6]	$$BUCB10
$$BUCB22[6]	$$BUCB22
$$BUCB3[6]	$$BUCB3
$$BUCB4[6]	$$BUCB4
$$BUCB5[6]	$$BUCB5
$$BUFLDR	$$BUFLDR
$$BUFLD1	$$BUFLDR
$$BUFLD2	$$BUFLDR
SYSBUFLD	IJBSBUFF
'SVA Loadlists'	
$$A$SVA	$$A$SVA
$SVABAM	$SVABAM
$SVACSC	$SVACSC
$SVALOG[7]	
$SVASEC	$SVASEC
$SVA0000[7]	

Macros

IPLBMAC
IPLCGEN
IPLDISK
IPLUNATT
LFCB[8]
MACIPLR2
MACIPLR3
MACIPLR4
MACIPLR5
MACIPLR6
MACIPLR7
MACIPLR8
MACIPLR9
MACIPLRA
MACIPLRB
MACIPLRC
MACIPLRD
MACIPLRE
MAPSALPL
MAPSVIPL
SVALLIST

---

5 See *VSE/Advanced Functions Diagnosis Reference: Logical Transients* for documentation.
Used by the macro LFCB.

6 See *VSE/ESA System Control Statements* for a description.

7 These phases are dummy phases to be replaced later by phases of the same name which then contain code and belong each to a different component.

8 See *VSE/ESA System Macros Reference*.

Debugging Aid

The IPL Debugging Aid is a tool to debug dynamically a VSE System running during IPL.

The IPL Debugging Aid uses the PER (program-event recording) feature by executing a NI instruction with a direct operand X'FF'. IPL doesn't run at fixed locations (the addresses depend on the real machine size and the state of IPL), so the IPL Debugging Aid can help you in analyzing a problem during IPL.

The following figures give a summary of all points where the execution of IPL can be stopped.

How to Use the IPL Debugging Aid

To use the Debugging Aid you have to activate the processor's display/alter function as described in the processor's Operating Procedures manual, or the trace function of VM.

The appropriate addresses can be found in the first column of the following tables.

Example

```
You want to stop the execution of IPL when allocating the PUBX table.
Running under VM you have to enter:

       TRACE STORE INTO 14C NORUN

Executing the instruction

       NI  14C,X'FF'

IPL stops and you can display the registers or you can invoke
the instruction step trace to analyze a certain problem.
```

Summary of Stop Points

Address	Function	Phase	Routine:	Description
100	Retrieve	$$A$IPLR	TESTLP:	Check IPL Load Parameter
101	Supervisor			Check SV generation options
102	Dispatcher		OKSUP:	Check VSIZE,VPOOL,VIO value
103	+ Console			Init. control register
104	support			Call INITDAT to initialize 16M IPL address space
105			INITSYS:	Initialize the system
106			PSUP:	Extract SV name, paging opt., list opt.,VSIZE,VPOOL,VIO
107				scan Keywords
108				Init. IPL partition (BG)
109			REQLOG:	Request a SYSLOG device
10A			REQSUP:	Request IPL parameters (IPL=proc_name,...)
10B				Check input for IPL param.
10C			WTORLOG:	Write to/read from CRT SYSLOG
10C			WTORIC:	Send to Integr. Con. SYSLOG Receive from IC SYSLOG
10D			WRITELOG:	Write to CRT SYSLOG
10D			WRITEIC:	Send to IC SYSLOG
10E			LOADIC:	Load SA Integr.Cons. support
10F			ATTNCHCK:	Check SYSLOG device type
12C			LOADCS:	Load IPL console support
12D				Call Console Support initial.
12E			REQIC:	Initialize Integr.Con support
12F			REQCRT:	Determine system cons type
201	Common	$$A$PLBK	ACCSLXE:	Searches a LB for SYSLIB
	Bootstrap	$$A$PLBF		
202	Routines	$$A$IPLR	CMDESC:	Reads system lib. descriptor
203			SSLXE:	Searches sublib. index for SYSLIB entry
204			ACCLMX:	Scan through level n LB
205			ACCPHMR:	Read member of type phase
206			GETLB:	Read LBs into storage
207			NXTREC:	Point to next record in LB
208			SMDESC:	Searches member index
209			XPRBA:	Converts PRBA to physical Adr
20A			EXTRVIF:	Extracts pointer to phase VIF
20B			IORTN:	I/O routine
20F			ACCALT:	Special code - alternate IPL
20C	Tape	$$A$PLBT	SATFCH:	Entry point to tape FETCH
	Bootstrap	$$A$IPLR		- in stand-alone environment
	Routines			

Figure 14 (Part 1 of 8). Debugging Aid - Stop Points

Address	Function	Phase	Routine:	Description
1E0	ASI	$$A$IPLE	BEGIPLE:	ASI processing entry point
1E1	Processing		INITASI:	ASI processing initialization
1E2			CHCKASI:	Check if proc. in SYSLIB
1E3			GETASI:	Get next 80–bytes proc record
1E4			CALLIOS:	I/O subroutine
1E5			ACCPRMR:	Read member rec. and decompr.
1E6			DECOMPR:	Decompress member record
1E7				Exit decompress routine
1E8			PROCSRCH:	Process search list
1E9			SELDATA:	Process a selection list

Figure 14 (Part 2 of 8). Debugging Aid - Stop Points

Address	Function	Macro	Routine:	Description
110	Root	IPLDISK	MONITOR:	Read and evaluate a command
111	Phase		FDSRTN:	Find operand entry
112			FDSRTN2:	Find operand exit
113			OPRTN:	Operation scan routine
114			ILLOG:	Buffer command
115			SYSMVC:	Generalized move Rtn entry
116			RESIDL1:	Execute move
117			DOWTO:	Write suppressed msg to HC
118			QSCEIO:	Quiesce I/O
119			CHECKSHR:	Check DASDs for HW sharing
120	Initializ.	MACIPLR2	BEGIN:	Initialization entry point
121			DETIO:	Routine entry point
122				Return from phase $SENSDEV
123			DETDUMMY:	Insert Integ. Cons. dummy PUB
124			DETIO:	Routine exit point
125			TOD:	STCK and calc date/time
126				Set MVS CVT flags and addr
130	ADD/DEL	MACIPLR3	IJBIPL3E:	Process ADD/DEL command
131	Command		ADDCHCK:	Check ADD commands for sensed devices
132			ACKRET:	Routine Exit Point
133			ADDCHCK2:	Special 3380/ECKD check
135			DELRTN25:	Check DEL commands for sensed devices
136			DELPUB:	Delete a PUB entry
137			DELSID:	Delete a SID entry
138			BLDPUB40:	Insert a SID entry
139			DVCCONS:	Special CONS type handling
13A			R3DELSA:	Delete devices not needed in stand-alone environment

Figure 14 (Part 3 of 8). Debugging Aid - Stop Points

Address	Function	Macro	Routine:	Description
140	SET Command	MACIPLR4	PASS1BEG:	Complete tables Permanent PUBS for SYSRES and SYSLOG
141	First		BPUBSCN:	Build PUBSCAN2 and PUBSCAN3
142	Allocation			Dynamic SV area length entry Length of AVR table Length of PUB2 and PUBX Length of CCW chain Length of MCEL Length of IOEL max FBA blksize for DLF area
143			UPDIO:	Routine entry point
144			NODSHR1:	Dynamic SV area length exit
145			UPDPPBEG:	Update IJBPPBEG
146			DYNALLOC:	Allocation of dyn. SV areas initialization of PUB2 Init.ERROR ENTRY TABLE (PUBX) formatting CCW chains (ECKD) MCEL IOEL I/O Area for DASD sharing
147			NODSHR2:	Init. AVR table
148				Set storage key Set clock comparator Allocation exit
149			BLDAVR:	Build AVR table entry
14A			UIOSHR:	Preserve DASD SHR if ADDed and sensed dev type not equal
14B			CALCPXL:	Calculate length of error entry table (PUBX)
14C			BLDPUBX:	Alloc., Init. of PUBX
14D			UIOSCN0:	Special 3380/ECKD checking
14E			DVCDNRTN:	Check for not operat. DASDs
14F			UPDIO:	Return from phase $SENSDEV
210			MINCONF:	SP minimal configuration
211			UIOLOAD:	Loads a phase
212			DUPVLD:	Checks duplicate VOLIDs
213			DUPVLD:	Return from phase DTRIVLD
214			PUBCHCK:	Check for enough PUB space
215			MINCONF:	Return from phase DTRICONF
216			DEVFF:	Prompting if any dvc code=FF
217			CONSCHCK:	Request ADD if IC not ADDed
219			BLDP2T00:	Allocation/initializat. PUB2

Figure 14 (Part 4 of 8). Debugging Aid - Stop Points

Address	Function	Macro	Routine:	Description
150	DEF	MACIPLR5	DEFPR00:	Process DEF command entry
151	Command		DEFSYS40:	Handle SYSxxx operand
				Check SYSxxx operand
152			DEFRET:	Handle SYSxxx operand exit
153			DEFSYS50:	Handle SYSxxx CUU or VOLID
154			DEFERR48:	Message 0I48
155			DEFERR47:	Message 0I47
156			DEFVC00:	GETVCE CUU
157			DEFVC01:	GETVCE VOLID
158			DEFVC02:	Error exit of GETVCE
159			DEFASS00:	Assgn defined system unit
160	SVA	MACIPLR6	CBALLOC:	Calculate copy buffer areas
161			IJBIPL62:	Update GETVIS value by length
				of concatenation tables space
162			SWVIRT10:	Load $$A$SVA
163			LOADL10:	Load loadlists routine
164				At LOAD SVC
165			CALCL1:	Accumulate length of phases
166			PSIZERT:	Check PSIZE specification
167			GETVISOP:	Check GETVIS specification
168			BLDSASDL:	Build SDL in stand-alone
				environment
170	Final	MACIPLR7	IJBIPL7E:	Final processing entry point
171	Processing			1st call to $INTVIRT
				2nd call to $INTVIRT
172				Return from $INTVIRT 1st call
				Return from $INTVIRT 2nd call
173			NOWVIRT1:	Initialize SVA header
174			R7CSIOR:	Call final console support
				reinitialization
175			RCFOPEN:	Initialize security manager
176				1st call to INLPSDL
				2nd call to INLPSDL
177				Move SDL entries into SVA
178			R7SVALC:	Save entry points of special
				SVA phases into SUPV
179			LDSVEXIT:	Exit special SVA phase handl.
17A			APCOPEN:	Open APPC/VM communication
17B			RCFOPEND:	Call: INITSYS
17C				Call to DELREPA to release
				copy buffers
17D			BLDCCAT:	Alloc. and Init. of library
				concatenation control tables
				Call: INITCON
17E			EOJ211:	EOJ—SVC, exit of IPL
17F			INICHANQ:	Format channel queue
			ICOPYBL:	Chain CCW copy blocks

Figure 14 (Part 5 of 8). Debugging Aid - Stop Points

Address	Function	Macro	Routine: Description
21A 21B 21C 21D 21E 21F	Unattended node	IPLUNATT	IJBIPUNS: Initialize unattended node GETVCE - primary IPL device GETVCE - alternate IPL device IPL count record read IPL count record initialized IPL count record updated
180 181	DLA Command	MACIPLR8	DLAPR00: DLA command entry point DLAPR22: Check DLA size upper limit
190	DPD Command	MACIPLR9	DPDPR00: DPD command entry point
1A0 1A1 1A2	DLF Command	MACIPLRA	DLFPR00: DLF command entry point DLFSU00: Set up DLFITAB and L.F.header Entry point DLFRET: Exit
1D0 1D1 1D2 1D3 1D4 1D5 1D6 1D7 1D7 1D8	Common Routines	IPLCGEN	xxxVOL00: Check and handle VOLID opernd xxxVC00: GETVCE CUU xxxVC02: GETVCE VOLID xxxVC12: GETVCE capacity after GETVCE xxxDFM00: Format the extent xxxDFLT: Set defaults xxxCYL00: CKD: Check CYL specification xxxBLK00: FBA: Check BLK specification xxxCHFM: Check record size of an existing PDS
1B0 1B9 1B1 1B2 1B3 1B4 1B5 1B6 1B7 1B8 1BA 1BB 1BC 1BD 1BE 1BF 1D9 1DA 1DB 1DC	SYS Command	MACIPLRB	ANSYS: SYS Command entry point ANSYSEX: SYS Command exit DASDFPRT: Handle DASDFP operand JART: Handle JA operand CHANQRT: Handle CHANQ operand BUFSRT: Handle BUFSIZE operand SUBLIBRT: Handle SUBLIB operand SECRT: Handle SEC operand SDSIZERT: Handle SDSIZE operand BUFLDRT: Handle BUFLD operand UNATTRT: Handle UNATT operand PRIMRT: Handle PRIMIPL operand ALTIPLRT: Handle ALTIPL operand REIPLRT: Handle REIPL operand SPSIZERT: Handle SPSIZE operand NPARTSRT Handle NPARTS operand VMCFRT: Handle VMCF operand PASIZERT Handle PASIZE operand RSIZERT: Handle RSIZE operand ATLRT: Handle ATL operand

Figure 14 (Part 6 of 8). Debugging Aid - Stop Points

Address	Function	Macro	Routine:	Description
1C0	Final Allocation	MACIPLRC	IJBIPLCE:	Final dynamic allocation of supervisor tables — entry
1C1			SYSEVAL:	Evaluate SYS operand values
1C2			CDYNL:	Determine final BG begin Length of VIO dynamic area Length of VPOOL dynamic area Length of device CB area Length of ext.interrupt buff.
			CDYNL3:	Length of SYSREC buffer
1C3			CDYNL62:	Length of channel queue
1C4			CDYNL7:	Alignment of BG start
1C5			CSSKXA:	Set storage key for BG
1C6			CDYNL8:	Final allocation of VIO/VPOOL dynamic area Device control blocks
1C7			CALLOC1:	Ext. interrupt buffer + form.
1C8			CALLOC4:	SYSREC buffer
1C9			CALLOC5:	Channel queue (unformatted)
1CB				Exit final allocation
1CC			LHCREC:	Ensure SYSREC assignment
1CD			ADDIUCVL:	Length of IUCV/APPC/VM tables
1CE			IPLIUCV:	Retrieve length of external interrupt buffer and paths
1DD	APPC/VM SET command	MACIPLRD	IJBIPLDE:	APPC/VM SET command - entry
1DE			VMSCANEX:	Add entry to temporary APPC/VM resource table
1DF			TVMRESID:	Check SNA network information

Figure 14 (Part 7 of 8). Debugging Aid - Stop Points

Address	Function	Macro	Routine: Description
			The following values are reused. (1st usage: MACIPLR4 I/O allocation) Set STOPI before entering commands SET ZONEBDY or SET ZONEDEF.
140	SET command	MACIPLRE	ZONERTN: SET ZONEBDY and SET ZONEDEF commands - common entry
141	SET ZONEBDY		ZONEBDY: Entry to SET ZONEBDY processing
142			ZONBDYPR: Calculate binary local clock value from specified DATE and CLOCK
143			ZBDYBLD: Build table entry and put it into zone boundary table - entry
144			Exit
145	SET ZONEDEF		ZONEDEF: Entry to SET ZONEDEF processing
146			ZONDEFPR: Calculate sytem zone in minutes from specified ZONE
147			ZONESTOR: Build table entry and put it into zone definition table - entry
148			Exit
	Start of sub-routines		
149			GETZONE: Transforms zone value in minutes into printable format (e.g. EAST/02/00)
14A			SETZONE: Searches applicable zone in zone boundary table using TOD value. Updates system zone and zone id in TODCOM, and job zone in COMREG
14B			SETZDEF: Gets applicable zone id from zone definition table using TODCOM zone value and updates zone id in TODCOM
14C			SETZVAL: Checks if zone id on command is defined, and updates zone and zone ID in TODCOM if requested
14E			MOVZONE: Allocates zone boundary and definition table in 31-bit system GETVIS. Moves entries from table built by IPL. Transforms binary local time into TOD value (GMT).

Figure 14 (Part 8 of 8). Debugging Aid - Stop Points

Chapter 6. Introduction -- Job Control

The job control component comprises the job control program and some service programs which are used by job control.

The Job Control Program within the VSE System

Job control provides job-to-job transition for all user programs.

Job control is automatically loaded into the background partition after IPL and after the end of a jobstep. In order to get it into a foreground partition for the first time, a START command must be issued for that partition. The job control program is then loaded into the partition if the following minimum requirements are met:

- Virtual storage size must be at least 128K including GETVIS space.
- The partition must have separate system files.

Before, between, and after jobs and job steps, the job control program is invoked automatically. One or more programs can be executed with a single job. Such an execution is called a job step and is preceded each by a special set of job control statements. (The term 'statement' is used here indifferently for statements and commands. For the external differentiation, see *VSE/ESA System Control Statements*.) Job control reads those and accordingly prepares the system for each execution, that is, it:

- sets up I/O device tables,
- calls procedures,
- initializes fields in the communication regions,
- accepts label information,
- sets up library concatenation chains,
- accepts and generates input statements for linkage editor and MAINT programs,
- prepares restart of checkpointed programs,
- clears the program space (partition or pages).

The job control program is executed in the storage of the partition it is preparing and runs in virtual mode.

Service Programs and Phases

The programs listed below are part of the job control component. They can be used by job control and other system components.

$IJBASGN
: Is the dynamic device assign program. It assigns and unassigns an I/O unit identified by the calling program as a cuu address to any free logical unit or an unassigned tape to any free programmer logical unit and does the necessary housekeeping at the end.

$IJBCJC
: Handles information about conditional job control.

$IJBJCCN
: Is used to clear the partition, store the PARM value of the EXEC statement, to provide a 72 byte save area, and to load the program to be executed.

$IJBMAP
: Is used to produce a storage map.

$IJBPROC
Is used for symbolic parameters and nested procedures.

$IJBPRTY
Is used to control the priority sequence of the partitions.

$IJBSDSP
Is used to set and query information related to data spaces and to the Turbo Dispatcher. Is also used to display the current setting of both standard and temporary options.

$IJBSHCF
Handles the hardcopy file.

$IJBSLA
Is used for symbolic label access.

$IJBSLIB
provides control functions for the IBM 3494 Tape Library Dataserver.

$IJBSTRT
Is used to dynamically start a partition.

$IJBVDII
Is used to create/remove virtual disk devices.

$IJBVTAP
Is used to register/unregister virtual tape devices.

The phases $IJBSLA, $IJBSTRT, and $IJBSHCF are documented in *VSE/Advanced Functions Diagnosis Reference: Logical Transients*.

Chapter 7. Design Information -- Job Control

Function

The job control program is invoked automatically before, between, and after job steps and does one or more of the following on the basis of information provided in job control statements:

- Evaluates control statements. Reads cataloged SYSRDR and SYSIPT data for cataloged procedures and calls the appropriate processing programs.
- Assigns device addresses to symbolic units.
- Initializes fields in the communication regions.
- Edits and stores volume and file label information.
- Supports procedure nesting up to level 15.
- Substitutes symbolic parameters and passes parameters to procedures.
- May skip parts of the job stream according to conditional job control.
- Builds library concatenation chains.
- Prepares the system for restarting checkpointed programs.
- Clears the program area to binary zeros between job steps if the job step is in real mode, otherwise the pages of virtual partitions are cleared as they are used.
- Prepares input for the linkage editor program if the LINK option has been specified. The statements ENTRY, ACTION, PHASE, MODE and INCLUDE, when present in the input stream, are copied to SYSLNK as card images. An INCLUDE statement with a blank operand causes the contents of SYSIPT to be copied to SYSLNK until a /* statement is read from SYSIPT. Blank statements from SYSIPT are ignored.
- Prepares input for the librarian by writing a CATALR statement from the SYSRDR job stream and any following PHASE statement to SYSPCH. If a compilation or assembler run follows, the SYSPCH output can be used as input (on SYSIPT) for a subsequent librarian job step. It is, however, only possible if the LINK option is not in effect.

A JOB statement in the input stream marks the beginning of a job and a /& statement marks the end of a job. An EXEC statement calls for execution of a job step or of a cataloged procedure. A job step is normally ended with the EOJ macro. A cataloged procedure is ended by a /+ statement.

Function-to-Phase List

The following list shows the functions, first of all phases of the job control program and then of the service programs attached to the job control component.

Figure 15. Phase-to-Function Cross-Reference

Phase	Function	Mod Name
$JOBCTLA	Root phase, processes //, /*, *, /., OVEND, PAUSE, :READ, IGNORE.	IJBJC1
$JOBCTLB	Prepares a checkpointed job for restart (RSTRT).	IJBJCB
$JOBCTLC	Opens the hard-copy file.	IJBJCC
$JOBCTLD	Processes ASSGN and CLOSE.	IJBJC2
$JOBCTLE	Processes EXEC, SETPARM, PROC, PWR and MSECS.	IJBJC9
$JOBCTLF	Processes DVCDN, DVCUP, LISTIO, RESET, UNBATCH.	IJBJC5
$JOBCTLG	Processes CANCEL, EOJ, EOP, JOB, OPTION, ID, START.	IJBJC3
$JOBCTLH	Processes LIBDEF, LIBDROP, LIBLIST.	IJBJCH
$JOBCTLI	Processes IF, ON, GOTO.	IJBJCI
$JOBCTLJ	Processes ALLOC, ALLOCR, CATALR, DATE, HOLD, LOG, MTC, NOLOG, NPGR, RELSE, SET, SIZE, STDOPT, STOP, UCS, UPSI, ZONE and writes the linkage editor statements (ACTION, ENTRY, PHASE, MODE, INCLUDE) to SYSLNK if the link bit is on.	IJBJC4
$JOBCTLK	Processes DLBL, EXTENT, RSTRT, SETPRT, TLBL.	IJBJC6
$JOBCTLM	Processes ROD, opens recorder file.	IJBJC7
$JOBCTLN	Job accounting and VSE/POWER interface.	IJBJC8
$JOBCTLO	Precesses JCLEXIT, MAP, SYSDEF, QUERY, SETPFIX, VDISK, LIBSERV, PRTY and passes (during BG ASI) selected AR commands to the attention routine.	IJBJCO
$JOBEXIT	Exit for a user phase between job control statements.	$JOBEXIT
$SYSOPEN	Exit for a user phase after IPL is complete.	$SYSOPEN
$IJBASGN	Assigns devices dynamically (called via macro ASSIGN).	IJBASGN
$IJBATTN	Interface between $IJBAR on the one hand and the SVA phases $IJBMAP, $IJBPRTY, $IJBSDSP and $IJBSLIB on the other.	IJBATTN
$IJBCJC	Handles information of conditional JC (called via macro CONDJC).	IJBCJC
$IJBJCCN	Clears partition, stores PARM value of EXEC, provides save area and loads user program specified in the EXEC statement.	IJBJCLCN
$IJBMAP	Produces a map of the current amount of storage allocated to program areas. Called during MAP processing.	IJBMAP
$IJBPROC	Handles parameters and nested procedures (called via macro PROCMAC, PARMMAC).	IJBPROC
$IJBPRTY	Processes the PRTY and TPBAL commands.	IJBPRTY
$IJBSDSP	Sets data space information into supervisor table, produces list of information about data spaces and about standard options. Allows to activate, terminate and query VSE/ESA's multiprocessing capabilities. Called during SYSDEF, QUERY processing.	IJBSDSP
$IJBSHCF	Services the hard-copy file (called via macro POINTHC, WRITEHC).	IJBIOHCF
$IJBSLA	Reads, writes, and modifies label information (called via macro LABEL, LPL).	IJBSLA
$IJBSLIB	Provides control functions for IBM 3494 Tape Library Dataserver. Called during LIBSERV processing.	IJBSLIB
$IJBSTRT	Starts a partition (called via macro STARTP).	IJBSTRT
$IJBVDII	Creates/removes virtual disk devices. Called during VDISK processing.	IJBVDII
$IJBVTAP	Registers/unregisters virtual tape devices. Called during VTAPE processing.	IJBVTAP

Storage Map

During end-of-task (SVC14) processing, the terminator routine $IJBSEOT (label EOT2000 in IJBEOTSK) will load the *job control root phase* $JOBCTLA into the partition. The entry point is label JOBCTL. In VSE/ESA Version 2 this entry point is located at X'9000' relative to the partition start address. This code initializes job control and is only executed, when a new copy of $JOBCTLA is loaded into the partition.

When initialization has completed and job control commands are being executed then this initialization code is overlaid by different job control phases $JOBCTLx. Since these other job control phases $JOBCTLx highly depend on the root phase $JOBCTLA, these $JOBCTLx phases are called *job control subphases*.

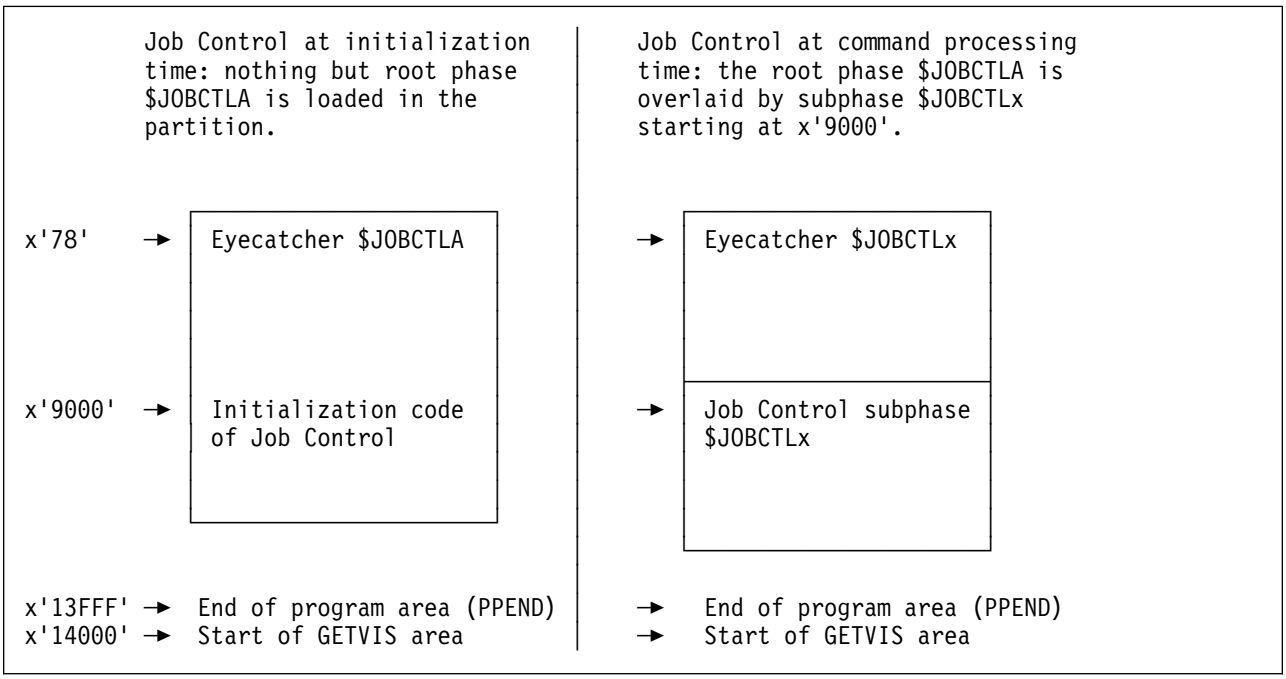


Figure 16. Job Control Storage Layout

Job Control command processing is controlled by $JOBCTLA (subroutine CONTROL). This CONTROL routine

- Reads the next job control command into the command input buffer. This buffer is located at X'1BC' relative to the partition start address.
- Scans the command input buffer for the first operand (i.e. the command name).
- Loops through the command table for a match. This command table contains a 14-byte entry for each available job control command. Each entry contains the letter x identifying the proper job control subphase $JOBCTLx, which will process the command.
- Passes the command to one or more job control user exit routine(s).
- Invokes subroutine EXIT, which will load the command processing subphase at X'9000' relative to the partition start address (provided it is not already in storage) and pass control to the subphase. The subphase will continue to process the command. It will use services provided by the root phase, such as operand scanning, substitution of symbolic parameters, concatenation of continuation lines, writing of messages to SYSLOG/SYSLST etc. When command processing has completed, the subphase will return to CONTROL.

Figure 17 on page 142 illustrates, which subphases are loaded during execution of a simple job.

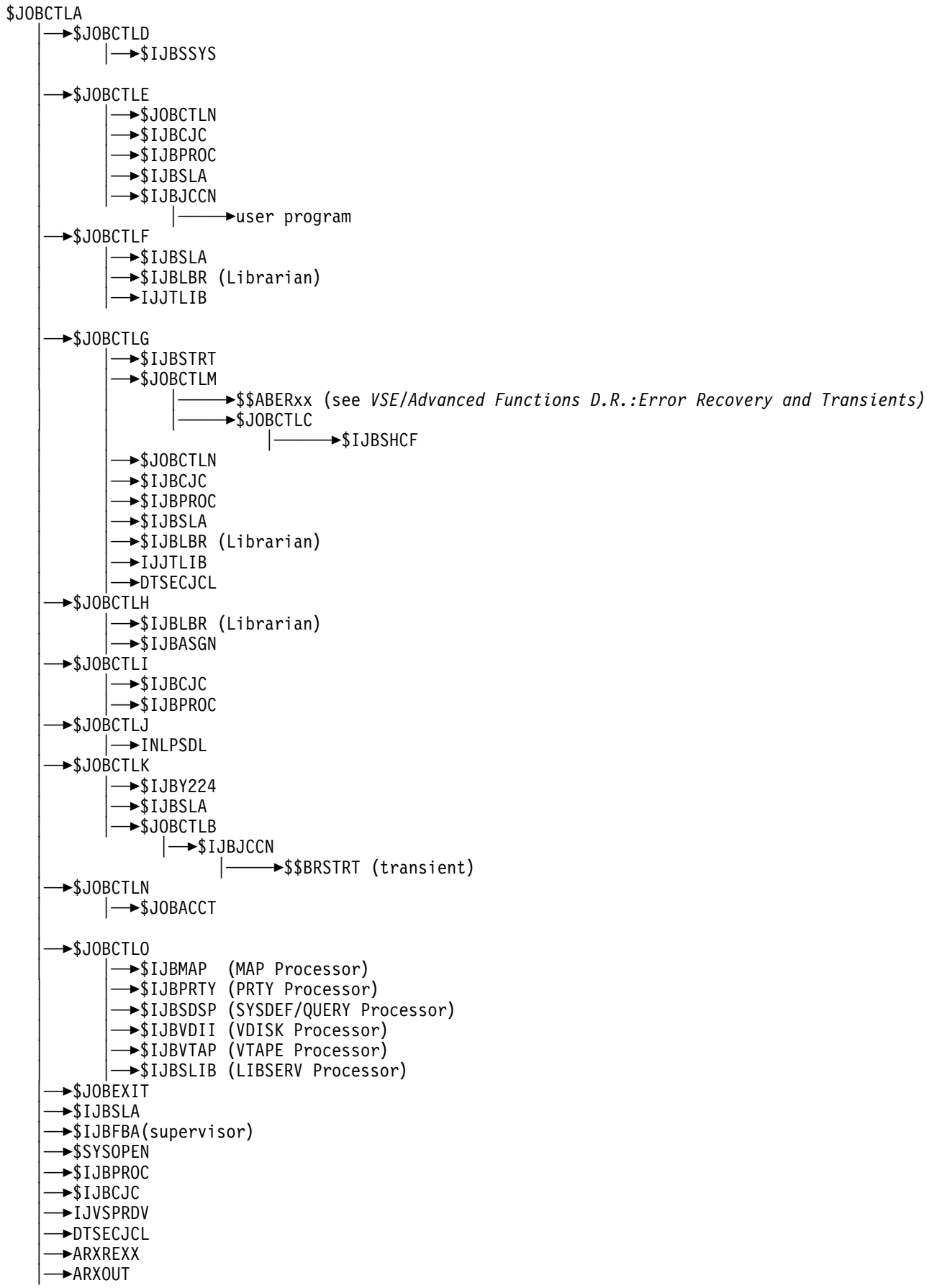
```
// JOB EXAMPLE                  (load $JOBCTLG)[9], pass control to $JOBCTLG
// OPTION PARSTD=ADD                             pass control to $JOBCTLG
// DLBL EXAMPLE,'EXAMPLE.FILE'  load $JOBCTLK, pass control to $JOBCTLK
// EXTENT ,VSE200,,,1500,30                      pass control to $JOBCTLK
// OPTION USRLABEL              load $JOBCTLG, pass control to $JOBCTLG
// ASSGN SYSLST,00E             load $JOBCTLD, pass control to $JOBCTLD
// EXEC LSERV,PARM='PARSTD=BG'  load $JOBCTLE, pass control to $JOBCTLE
/&                              load $JOBCTLG, pass control to $JOBCTLG
```

Figure 17. Sample Job to Demonstrate Subphase Loading

Job Control Calling Structure (Phases)

The following table shows the control flow of the job control program.

[9] At this point a load of subphase $JOBCTLG is not required, if the preceding command (that is the last command in the preceding job) was a /& and hence $JOBCTLG is still in storage.


$JOBCTLA
$JOBCTLD
$IJBSSYS
$JOBCTLE
$JOBCTLN
$IJBCJC
$IJBPROC
$IJBSLA
$IJBJCCN
user program
$JOBCTLF
$IJBSLA
$IJBLBR (Librarian)
IJJTLIB
$JOBCTLG
$IJBSTRT
$JOBCTLM
$$ABERxx (see VSE/Advanced Functions D.R.:Error Recovery and Transients)
$JOBCTLC
$IJBSHCF
$JOBCTLN
$IJBCJC
$IJBPROC
$IJBSLA
$IJBLBR (Librarian)
IJJTLIB
DTSECJCL
$JOBCTLH
$IJBLBR (Librarian)
$IJBASGN
$JOBCTLI
$IJBCJC
$IJBPROC
$JOBCTLJ
INLPSDL
$JOBCTLK
$IJBY224
$IJBSLA
$JOBCTLB
$IJBJCCN
$$BRSTRT (transient)
$JOBCTLN
$JOBACCT
$JOBCTLO
$IJBMAP (MAP Processor)
$IJBPRTY (PRTY Processor)
$IJBSDSP (SYSDEF/QUERY Processor)
$IJBVDII (VDISK Processor)
$IJBVTAP (VTAPE Processor)
$IJBSLIB (LIBSERV Processor)
$JOBEXIT
$IJBSLA
$IJBFBA(supervisor)
$SYSOPEN
$IJBPROC
$IJBCJC
IJVSPRDV
DTSECJCL
ARXREXX
ARXOUT

Description of Job Control Phases

The job control component consists of job control phases and some SVA-resident service programs of which $IJBASGN, $IJBCJC, $IJBJCCN, $IJBMAP, $IJBPROC, $IJBPRTY, $IJBSDSP, $IJBSLIB, $IJBVDII, and $IJBVTAP are described here (see Chapter 6, “Introduction -- Job Control” on page 137).

Every phase is described in a summary, first by listing its important characteristics in the same way as the code does. The last item, however, “Sequence of Operation,” gives an overview of the control flow with the most important labels and indicates where the main exits occur, by means of an arrow (------->). The exits indicate either entries in the same phase or in the root phase, or a branch to another phase.

Phase $JOBCTLA

Module Name: IJBJC1

Entry Points:

- JOBCTL
- For the entry points to the service routines, see “Sequence of Operation” below.

Function:

- Initial entry into the job control program and its root phase.
- Reads each control statement into a common buffer and determines which of the processing phases is to be called.
- Contains routines to be used by all job control phases and a common area.

(For details see “Sequence of Operation” below.)

Called By:

- $IJBSEOT (See *VSE/Advanced Functions Diagnosis Reference: Logical Transients*.)

Phases Called:

- $JOBCTLD,E,F,G,H,I,J,K,N,O
- $IJBFBA - For FBA SYSFIL support (supervisor). For details on the interface, see “Interface Between $JOBCTLA and Supervisor Phase $IJBFBA” on page 253.
- $IJBSLA (via LABEL macro)
- $JOBEXIT
- $SYSOPEN
- $IJBCJC (via CONDJC macro)
- $IJBPROC
- DTSECJCL
- ARXREXX (interface to REXX)
- ARXOUT

Data Areas Used:

```
TBLADR   -- Phase vector table
DFBs     -- Data file blocks in BASVCT
BASVCT   -- Common job control area
COMDST   -- Partition communication region DSECT (macro MAPCOMR)
SYSCOM   -- System communication region DSECT
LUBTAB   -- LUB table in supervisor
PUBTAB   -- PUB table in supervisor
DIBs and PIBs in supervisor
PCE      -- Table of partition related control block addresses
PJBADR   -- DSECT for job information control block (macro MAPPOWJB).
PODS     -- POWER Partition control blocks
CRTTAB   -- CRT Support root table
CRTSAV   -- CRT Support table
```

Messages Caused:

1A81I	1F2nD	1M3nD	1S79D
1C00A	1F3nD	1N00I	1S9nD
1C10D	1I00D	1N91I	1SA0D
1C70D	1I50I	1S0nt	1T80I
1C80D	1I70I	1S40t	1U72I
1C90I	1M10A	1S50D	1U75D
1D01t	1M20D	1S55I	

Messages Issued: All job control error messages are issued via $JOBCTLA routines. See routines NVSERR, ERRRTN, ERRRTN0, OERRRTN, WAITERR.

Input:

- Job control statements from SYSLOG or SYSRDR depending on the input switch (COMREG+56, bit 2), or from the procedure library.
- Linkage editor data from SYSIPT

Output:

```
Messages on SYSLOG and SYSLST
Linkage editor data on SYSLNK or SYSPCH
Label information
Linkage editor input statements
Control fields
```

Exit Normal: to processing phase from label SUBPHASE

Exit Error: Hard wait from label WAITERR

Register Use: See “Diagnostic Aids -- Job Control” for passing and return of registers.

Sequence of Operation:

Initialization:

JOBCTL: Initializes the job control program as follows: Loads base registers.

FICNIC: Get number of partitions in system. Flag current partition as static or dynamic.

DSKINT: Resets conditions in $IJBSLA for that partition by specifying an ENDLBL function.

PFT17: Checks available record counts if system units are supported on a DASD device.

PRCINIT: Initializes the job control program for procedure processing if required.

The first time after IPL, $SYSOPEN is loaded and executed.

ACCTIGN4: Checks if an ON condition is pending. Generates an 'EXEC LISTLOG' statement if a job was canceled and the LISTLOG is requested by the DUMP routines. ------->EXIT

Clears time of day (PJBJTIME) in the job information control block (PJB). Saves maximal return code (PJBJRET) in the job information control block (PJB) (see VSE/ESA Supervisor Diagnosis Reference). Prints last return code (message 1S55I) if it was not zero.

For dynamic partitions it is checked whether SYSRDR is assigned in the ASI proc. and whether the partition is to be cancelled. Then an UNBATCH statement is generated and passed to the processing routine.

ACCTIGN5: Generates an 'EXEC PROC=mmm' statement to call the ASI procedure if the ASI bit is on. ------->CONTROL3

ACCTIG55: Generates an 'EXEC LNKEDT' or 'EXEC' statement if the 'GO' function of an EXEC statement is in effect. ------->EXIT

Reading of Control Statements:

CONTROL: Checks the SYSRDR assignment to see if the next statement can be read.

PTEST: Tests for end-of-job (/&) and end-of-procedure (/+) statements.

Checks if procedure modification is necessary.

Reads overwrite statements if requested and tests for overwrite end.

CONTROL3: Checks if column 1 is blank.

CONT7: Checks if the statement starts with //.

Scans the operation code of the first entry.

Initializes for the phase vector table lookup. ------->BTLOOP

Phase Vector Table Lookup: The phase vector table is used by $JOBCTLA for initial processing of all job control statements. See the description in “TBLADR -- Phase Vector Table in $JOBCTLA” on page 249.

BTLOOP: The phase vector table is searched for the operation code specified in the statement to be processed.

The entry thus found identifies the phase wanted and, in this phase, the displacement of the branch instruction to the processing routine wanted. The entry also contains two bytes of condition switches which determine details of the execution. (See “TBLADR -- Phase Vector Table in $JOBCTLA” on page 249.)

Register R3 then points to the first character of the operation field. If the statement has continuation cards these cards are read via subroutine CONCAT. If a real phase $JOBEXIT is present in the SVA, an exit is made to check each statement. A return code of zero must be set in register 15. Otherwise the statement is treated as a comment. $JOBEXIT returns control via register 14. The operand is checked for validity.

Processing of the statement is skipped if an appropriate ON, IF, GOTO or CANCEL condition exists.

The statement is logged and listed according to the condition switches of the entry found in the phase vector table.

EXIT: If the processing phase for the statement is not in virtual storage it is loaded (SVC4). Processing for //, /*, /., *, IGNORE, OVEND or PAUSE is in the root phase and they are processed as follows:

```
//      - The statement is scanned  to make the operation
          field available for processing. ------------>CONT7
/*      - Ignored ---------------------------------->CONTROL
*       - Ignored _____|                      |
IGNORE  - The JC input switch is set to       |
          indicate SYSRDR as input device-----|
OVEND   - Switches are set to prevent any     |
          more overwrite statements from      |
          being read for this procedure.------|
PAUSE   - If the statement is used without //, the job-control-pause
          switch (bit 5 in JCSW1 of COMREG) is set on to cause
          a pause before the next job-step.
          If the statement is used with //, the job control
          input-on-SYSLOG switch (bit 2 in JCSW1 of COMREG) is set on.
          This forces a pause on the next active job control
          read operation. ---------------------------->CONTROL
/.      - If an ON, or GOTO condition exists the statement is
          scanned for the label. If the label is the one which
          was specified in the ON or GOTO statement the condition
          is reset. ---------------------------------->CONTROL
```

SUBPHASE: Control is given to the processing phase as indicated by the following table:

$JOBCTLD: ASSGN CLOSE	$JOBCTLE: EXEC MSECS PROC PWR SETPARM	$JOBCTLF: DVCDN RESET DVCUP UNBATCH LISTIO
$JOBCTLG: CANCEL OPTION EOJ ID EOP START JOB	$JOBCTLH: LIBDEF LIBDROP LIBLIST	$JOBCTLI: IF ON GOTO

$JOBCTLJ: ACTION	DATE	LOG	NPGR	STDOPT	ZONE
ALLOC	ENTRY	MODE	PHASE	STOP	
ALLOCR	HOLD	MTC	SET	UCS	
CATALR	INCLUDE	NOLOG	SIZE	UPSI	

$JOBCTLK: DLBL EXTENT RSTRT SETPRT TLBL	$JOBCTLM: ROD	$JOBCTLO: JCLEXIT MAP SYSDEF QUERY SETPFIX (pass AR cmds) VDISK LIBSERV PRTY

Subroutines:

- The subroutines in this phase, which are common to all job control subphases, are placed in the job control common area BASVCT which is discussed in “BASVCT -- Common Job Control Area” on page 247. The area BASVCT is generated by the macro MAPJCLA and contains the following subroutines:

  CONTROL Main control routine, normal return from all command processors. Used to input a statement and determine the proper phase and routine to process the statement.

  CONTROL2 Entry point in CONTROL to process generated statements in the buffer.

  NVSERR Logs and prints message 1S0nt (INVALID STATEMENT) and branches to routine ERRRTN.

  ERRRTN

  Logs and prints statement in the buffer, puts message into the buffer, puts number of currently scanned parameter into byte 3 of the buffer, logs and prints the message and requests reading of the next statement.

  ERRRTN0 Entry point in ERRRTN with the only difference, that byte 3 of the message remains unchanged.

  OERRTN Clears buffer, puts message into the buffer, branches to routine LOGOUT.

  FETCHRF Calls a specified JC subphase and branches to it.

  WAITERR Branches to WAITERRX. This subroutine issues a severe error message and causes a hard wait afterwards. In order to synchronize the system and guarantee that the message is seen by the operator, a WTOR macro is issued and the operator has to hit the ENTER key or to re-IPL immediately.

  SYSERR This routine is called in case of a system failure. It forces end-of-procedure, writes message 1M10D (JOB CONTROL FAILURE) and cancels the job.

  OVEND This routine handles *, /* or OVEND statement.

  - In case of *, control is given to subroutine MSGCP, which examines, whether *, “* CP” or “* $$” is given.
    In case of “* CP,” subroutine MSGISCP, the CP command handler, is called. Checking for master authority is done (message 1SA0t: COMMAND NOT ALLOWED, INSUFFICIENT AUTHORITY), and the CP command is passed to AR via the MGCRE macro.
    In case of “* $$” (this can only happen, if either POWER is not active or there is a writer-only system), subroutine MSGISPW, the POWER command handler, is called. In case of a writer-only system, the POWER statement is communicated to the SVC0 appendage of console I/O (JCL itself uses WTO instead of SVC0 to write to the console, therefore a 'dummy' SVC 0 on behalf of POWER is issued).
  - In case of /*, control is given to subroutine PSLSAST. If REXX is active and sub-phase $JOBCTLK (DLBL/EXTENT processor) is in storage, then subroutine LBLOUTF in $JOBCTLK is called to write label information to the label area.
  - If neither * nor /* has been encountered, then OVEND processing is performed.

  LABELCHK This routine processes the label statement (/.).

  CHKASG This routine computes the LUB address for a logical unit number contained in R1. If assigned, it also computes the corresponding PUB.

  CHKASG3 Entry point in CHKASG. For a given LUB address of a system logical unit it calculates the corresponding PUB address (provided the logical unit is assigned).

MTNCNT This routine is called to seize (SVC 22) the system if it is currently released or to release it if it is currently seized.

IGNORE This routine is called to switch input from console to current SYSRDR and suppress logging. If a temporary console was connected it is released.

RLIND(T) This routine is called to read job control statements either from SYSLOG or SYSRDR, depending on bit 2 of JCSW1 in COMREG. If a procedure is being processed, subroutine INCLUDE is called. In case of a new POWER job, subroutine SPREPARE is called to pass security information from POWER to the security processor.

EXCPRG This routine performs I/O (SVC 0) depending on the DFB.

SCANR2 This routine scans for the next parameter in the buffer, skipping leading blanks. It calls subroutine PAREPL to replace symbolic parameters by the value which was assigned to them via SETPARM, EXEC PROC or PROC statements.

SCANR3 This routine scans for the next parameter in the buffer, but without skipping leading blanks.

SCANRS This routine scans for the next parameter in the buffer, if it is enclosed in apostrophes. If not, control is given to SCANR3.

LOGOUT This routine logs the content of the buffer on SYSLOG, if not already done. The writing to the console is done via the WTO macro, either in subroutine WTO1 (writes one line) or in subroutine WTO2 (writes two lines for JOB and END-OF-JOB information). In case of REXX (address JCL), control is being given to phase ARXOUT.

LSTOUT This routine lists the content of the buffer on SYSLST, if not already done. The writing to SYSLST is done with the EXCPRG routine described above.

MSGOUT Identical to LOGOUT.

LOGCHK This routine is used to set a switch to indicate the device type of SYSLOG (1052 or line printer).

SETAPRT This routine invokes macro SETPRT (phase IJVSPRDV) to supply information for a 3800 printer.

INITDIB There is one DIB (Disk Information Block) table per partition. Each DIB contains entries for SYSIN, SYSPCH, SYSLST and SYSLNK. The routine initializes the DIB for the specified FBA system file with values obtained by the OPEN macro issued against that file. Buffers are requested by GETVIS and the buffer addresses for both DIB extension and I/O buffer are also inserted into the DIB. Then the IORB (Input Output Request Block) and the FBA CCW chain are constructed in the DIB extension table.

CLOSEDTF Invokes the CLOSE macro against the SYSLNK DTFCP (IJSYSLN).

STMNTOUT This routine logs/lists the statement on SYSLOG/SYSLST depending on the following bits:

- Bit 3 (LOGMSK) in JCSW4: determines whether the statement is written to SYSLST or not. This bit is set via the // OPTION LOG or // LOG statements.
- Bit 3 (LOGMSK) in JCSW1: determines whether the statement is written to SYSLOG or not. This bit is set via the LOG command.
- Bit 7 (LGDONE) in JCSW7: don't log on SYSLOG, under no circumstances
- Bit 5 (PRDONE) in JCSW7: don't list on SYSLST, under no circumstances
- Bit 6 (LGEVER) in JCSW7: do log on SYSLOG, in any case
- Bit 4 (LOGUNCON) in JCSW8: do log/list on SYSLOG and SYSLST, in any case

The routines LOGOUT and LSTOUT are called to log or list the statement on SYSLOG or SYSLST, respectively.

BUFDEV This routine processes buffered devices like printers (e.g. 4248) and tape (3480). The following functions are provided:

- Reset horizontal copy (4248)
- Clearprint for buffered printers
- Synchronize for tape 3480

JCGETCOM This routine obtains the COMREG address for the specified partition ID (via GETFLD macro) and indicates whether the partition is dynamic.

HREXXCNT This is the main routine for REXX support. Control is transferred from $JOBCTLE (EXEC REXX=...) after all parameters are processed. REXX is called via phase ARXREXX. When REXX returns to job control, HREXXCNT processes the command stack, if any.

PAUSE This routine handles the PAUSE command/statement. The DLBL buffer (if any) is written out.

- Subroutines which cannot be called by job control subphases are:

PAREPL: Scan a symbolic parameter and replace it by the value assigned to it by a previous SETPARM, EXEC PROC or PROC-statement.

CONCAT: Read continuation statement(s) into a save area.

CHKCONT: Put another continuation portion into BUFFER if one exists and the previous portion is finished.

REMOV120: Print out a part of BUFFER to make space to continue scanning of a statement.

NEWSTOUT: List and log a statement before it is processed.

ADRJCLRX During execution of address JCL, REXX branches to COMREG.IJBRXJCL, which contains the address of ADRJCLRX. Reg1 points to 2 fullwords. The first fullword points to the JCL command, which was specified via address JCL. The seconds fullwords points to a fullword containing the length of the command to be executed.

In case of a command without SYSIPT data, this length is x'50'. In case of a command with n lines of SYSIPT data SYSIPT data, this length is (n+1)*x'50'. In the following example this length is x'190':

```
address JCL
call rexxipt input.
input.0 = 4
input.1 = '$IJBSDSP,SVA'
input.2 = '$IJBSLIB,SVA'
input.3 = '$IJBSLA,SVA'
input.4 = '/*'
'SET SDL'
```

Branch to CONTROL2 gets the command executed.

REXXGOBK This subroutine returns to REXX after an address JCL command has been completed. The following return codes are passed via R15:

- R15 = 0 : JCL command completed successfully.
- R15 =-3 : JCL command not found
- R15 =-4 : JCL command has invalid syntax
- R15 =-7 : JCL command not allowed via ADDRESS JCL

Phase $JOBCTLB

Module Name: IJBJCB

Entry Point: IJBJOBRS

Function: Prepares a partition for restarting a checkpointed job:

- Finds the checkpoint records.
- Checks if the resources are still the same.
- Restores control blocks and areas.
- Calls transient phase $$BRSTRT or cancels with a message.

(For details see “Sequence of Operation” below.)

Called By: $JOBCTLK

Phases Called: $$BRSTRT (transient via $IJBJCCN) to restore the problem program.

Data Areas -- Job Control Used:

```
BASDST   -- Job control common area DSECT
PIBTAB   -- PIB table (in supervisor)
PUBTAB   -- PUB table (in supervisor)
LUBTAB   -- LUB table (in supervisor)
LUBEXT   -- LUB extension table (in supervisor)
XTNTAR   -- Extent area in SVA
SYSCOM   -- System communication region DSECT
PBECIN   -- Partition save area in supervisor
COMDST   -- Partition communication region DSECT (macro MAPCOMR)
@RSTSYS  -- Restart parameter list
@DEUTAB  -- Checkpoint/restart device type table
JCBMSGTB -- Branch table for messages
```

Messages Caused:

0R00I	0R06I	0R10I	0R15I
0R01I	0R07I	0R11I	0R16A
0R03I	0R08I	0R12I	0R17I
0R04I	0R09I	0R13I	1S40t
0R05I			

Input:

- Statement RSTRT
- Parameter list prepared by checkpoint in the buffer of BASDST:

```
DC  H     Logical unit for checkpoint device
DC  CL4   Checkpoint ID
DC  CL8   Checkpoint file name
DC  Y(0)  PUB address of the checkpoint device
```

Output:

- Cleared partition
- Restart information in a system GETVIS area pointed to by register 5.

Exit Normal: JCB3520: to $IJBJCCN to clear the partition for $$BRSTRT, the restart transient

Exit Error: JCBERMSG: CANCEL with a message

Register Use:

```
R 3: Work register, interface to scan routine, addr. of ERRRTN
R 4: Work register and interface to scan routine
R 5: Work register and interface to scan routine
R 7: First base register of root phase and address of BASDST
R 8: First base register
R 9: Second base register
R10: Pointer to COMDST
R14: Return register for common subroutines
```

Sequence of Operation:

RESTART Statement Processing:

IJBJOBRS: Establishes addressability.

JCB0020: Checks if valid RESTART device is specified.

JCB0500: Builds and opens a DTFPH if the CHKPT file is on DASD.

JCB0600: Builds the necessary channel programs.

JCB0670: Searches the specified CHKPT on the file.

JCB0900: Checks the supervisor options of the restarted system.

JCB0980: Checks the storage allocations of the checkpoint/ restart system.

JCB1200: Rewrites 3800 information as saved at checkpoint time.

JCB2000: Restores COMDST and PIB as necessary.

JCB2400: Restores extent information to the label information area if required.

JCB3500: Requests system GETVIS space to build the parameter list used by the transient phase $$BRSTRT.
If error found ------->JCBERMSG

JCB3520: Else ------->$IJBJCCN

JCBERMSG: If error conditions are encountered issues a message and ------->SVC6 (CANCEL)

Phase $JOBCTLC

Module Name: IJBJCC

Entry Points:

```
JOBCJOB, to open the hard-copy file.
JOBCROD, to write the hard-copy buffer to the hard-copy file.
```

Function: Initializes the hard-copy file.

(For details see “Sequence of Operation” below.)

Called By: $JOBCTLM

Phases Called: $IJBSHCF (SVA), to service the hard-copy file.

Data Areas -- Job Control Used:

```
BASDST -- Job control common area DSECT:
    BUFFER  -- Message buffer
    CONTROL -- Return control point
    WAITERR -- Issue-message-and-wait exit
    NVSERR  -- Invalid statement message exit
    JBCSW5 (WRITE) -- Message control byte
    TYPCON1 (READ) -- Translate table for hexadecimal
COMDST -- Partition communication region DSECT (macro MAPCOMR):
     JCSW1 (WRITE) -- Job control switch
     LUBPT (READ)  -- LUB pointer
     PUBPT (READ)  -- PUB pointer
SYSCOM -- System communication region DSECT:
    IJBCONSP (READ) -- Address of CRTTAB
    IJBRFTAB (READ) -- Address of RFTABLE
    IJBFLG01 (READ) -- Flag byte

In the supervisor:

CRTSAV -- C-transient save area:
    CRTFLGHC (READ,WRITE) -- Flag-byte for hard-copy file
    Flags    (READ)       -- Flag byte
    HCFDEVTP (READ)       -- Device type of hard-copy file device
CRTTAB -- DOC table:
    ACRTSAV  (READ) -- Address of C-transient save area
    AHCFIOMD (WRITE)-- Address of HCF I/O module in SVA
LUBTAB -- LUB table:
    SYSLOG-LUB entry (READ)
PUBTAB -- PUB table:
    PUBDEVTY (READ) -- Device type code
    PUBOPTN  (READ) -- Device characteristic codes
RFTABLE -- RMSR communication table in supervisor:
    RFFLAGS1 (READ) -- Status byte 1
```

Message Issued:

1I94I
1I97E

Messages Caused:

1I95A	1I97E
1I96A	1I99A

Input: HC=YES or CREATE option of SET statement

Output: To the hard-copy file: (Done by POINTHCF)

- Dummy blocks to initialize the hard-copy file if HC=CREATE.
- First hard-copy file record indicating a new IPL complete.

Exit Normal: to $JOBCTLA entry CONTROL

Exit Error: to $JOBCTLA error routines

Register Use:

```
R 3: Work register, interface to scan routine, addr. of ERRRTN
R 4: Work register and interface to scan routine
R 5: Work register and interface to scan routine
R 7: First base register of root phase and address of BASDST
R 8: First base register
R10: Pointer to COMDST
R14: Return register for common subroutines
```

Sequence of Operation: Phase $JOBCTLC is called when a JOB or a ROD statement is read, provided the option HC=YES or HC=CREATE has been specified in a SET statement.

Hard-Copy File Initialization:

JOBCJOB: When the first JOB statement is read, the hard-copy file is opened for WRITE using the POINTHCF macro with the parameters:

- 'CONTINUE' if HC=YES was specified or
- 'CREATE' if HC=CREATE was specified

The POINTHCF macro initializes the WRITEHCF control block and writes the message 'DOS/VSE IPL SUCCESSFULLY COMPLETED' as the first hard-copy record to indicate that here a new IPL took place. The control block is used by later WRITEHCF macro calls. ------->CONTROL (root phase)

If OPEN is not successful and the DOC does not have a printer attached, message 1I95A is issued and the system enters hard wait. ------->WAITERR (root phase)

If OPEN is not successful but DOC has a printer attached, message 1I97I is issued and ------->NVSERR (root phase)

Writing the Hard-Copy Buffer to the Hard-Copy File:

JOBCROD: When the ROD statement is read the current hard-copy buffer is written to the hard-copy file using the WRITEHCF macro with the option FORCE=YES. ------->CONTROL (root phase)

Phase $JOBCTLD

Module Name: IJBJC2

Entry Points: JOBCTLD + displacement for ASSGN and CLOSE.

Function: Contains the processing routines for the statements ASSGN and CLOSE:

ASSGN: Builds DIBs, modifies LUBs and PUB ownership entries.

CLOSE: For tape, writes the EOV tape mark in trailer label. For an alternate tape, calls the BAM EOV routine. For disk, writes EOF and puts zeros for the current address in DIB.

(For details see "Sequence of Operation" below.)

Called By: $JOBCTLA

Phases Called:

```
$IJBSSYS   to check if a specified mode is allowed for a tape
```

Data Areas -- Job Control Used:

```
BASDST   -- Job control common area DSECT
COMDST   -- Partition communication region DSECT (macro MAPCOMR)
DIBs and LUBs in supervisor
INFOTAB  -- Information table for ASSGN operands
OPERTAB1 -- Operand scan table
XATABLE  -- Cross-assignment table
```

Resource Control: Access to the LUB, PUB, and PUBOWN tables by other partitions or tasks is controlled by the supervisor macros LOCK and UNLOCK. Resource name is CIOBLOCKS. The UNLOCK is eventually done in the root phase $JOBCTLA, near label CONTROL or CONTROL2.

Messages Caused:

1A0nD	1A70D	1A89D	1T20I
1A1nD	1A80t	1A9nt	1T40D
1A2nt	1A82D	1C92D	1T50A
1A4nD	1A83t	1C93D	1T60A
1A5nt	1A87D	1C94D	1T70A
1A60t	1A88D	1S40t	

Input: ASSGN and CLOSE statements

Output: See "Function" above.

Exit Normal: to $JOBCTLA entry CONTROL

Exit Error: to $JOBCTLA error routines

Register Use:

```
R 3: Work register, interface to scan routine, addr. of ERRRTN
R 4: Work register and interface to scan routine
R 5: Work register and interface to scan routine
R 7: First base register of root phase and address of BASDST
R 8: First base register
R 9: Second base register
R10: Pointer to COMDST (partition COMREG)
R11: Base register for internal CSECTS
R14: Return register for common subroutines
```

Sequence of Operation: This phase has two main routines labelled ASSGN and CLOSE which are both called from the root phase and return both to it. CLOSE has a second entry point for the case that the closed unit is to be reassigned. It leads into ASSGN. The rest of the phase consists of subroutines in the form of CSECTS which are called by the main routines.

JOBCTLD: Establishes addressability.

ASSGN: Initializes fields and switches, processes OPERSCN.

ASS0: Processes ERRCHK.

If UA or IGN is not specified ------->ASS1

Else processes ASSIGN from entry ASSGN20. ------->EXIT

ASS1: Processes CSECTs FINDPUB and ASSIGN.

If PROGRAM UNIT or ALT assigned ------->EXIT

Else processes OPENSYS

EXIT: Informs the operator of the device assigned if necessary. Gives file protect message for system output units on tape if necessary.
------->CONTROL (root phase)

CLOSE: Initializes fields and switches.

Processes CSECT CLOSESYS.
------->CONTROL (root phase)

CLOS0: Entry point which is used as return address by CLOSESYS if the closed unit is to be reassigned. Processes CSECT OPERSCN from entry ASSGN0. ------->ASS0

Subroutines:

OPERSCN: Scans and checks the statement: Tests statement for permanence and sets switch for temporary or permanent assign.

Checks and converts SYSxxx to the hex logical unit address.

- The unit cannot be SYSRES,SYSREC,SYSVIS,or SYSCAT.
- Assignment of SYSLOG is allowed only in the background.
- The unit cannot be SYSLOG if any foreground program is loaded or if the attention routine is active.

SCNOPRS: The second and following operands are scanned, and information is stored in INFOTAB:

- MODE (X'ss') can only be specified for tapes.
- SHR can only be specified for disks.
- ALT can only be specified for tapes.
- VOL= can only be specified for tape, disks, and diskettes.

- SYSyyy cannot be SYSOUT. If SYSyyy is SYSIN, checks if SYSIPT and SYSRDR are assigned to the same device. ------->CONTROL (root phase)

FINDPUB: Selects the PUB that can be assigned:

- Checks the status, the volume serial numbers, and the ownership to find a device that matches the specifications in the ASSGN (CLOSE) statement.
- Requests to mount a volume with the specified volume serial number, if not already mounted.
- Informs the operator of the cuu of the device to be assigned. ------->CONTROL (root phase)

ERRCHK: Performs additional error checking on specifications:

- If the second operand is UA or IGN, no other operands except TEMP and PERM are allowed.
- Only programmer logical units can be assigned to a 2245 printer.
- If SYSRDR, SYSIPT, or SYSIN is unassigned, SYSLOG must be assigned to a 1052 or CRT device.
- If SYSLOG is assigned to a non-1052/CRT device, SYSRDR must be assigned.
- The SYSLOG assignment,
- An assignment for SYSOUT, or
- A system I/O unit assignment to disk, must be permanent.
- If a system I/O unit is assigned and the old assignment was to a disk, the system file must be closed.
- If an ALT assignment is made, the logical unit must already be assigned to tape.
- If an ALT assignment is made for SYSOUT, SYSLST and SYSPCH must be assigned to the same device.
- Of the system units only SYSPCH, SYSLST, and SYSOUT can be assigned ALT.
- If the ALT assignment is TEMP, the logical unit must be temporarily assigned.
- If the ALT assignment is PERM, the logical unit must be permanently assigned.
- It is not allowed to give a TEMP assignment for SYSPCH or SYSLST if they are both assigned to the same device (SYSOUT). The proper CRT support must have been generated when SYSLOG is assigned.
- Checks for physical compatibility between the devices and the logical unit to which they are to be assigned. ------->CONTROL (root phase)

ASSIGN: Prepares and does the assignment:

Initializes MODE byte.

TSTMNT: If a volume has to be mounted a message is given and the partition goes into wait state.

After a NEWVOL statement has been issued, the volume serial number is checked again.

If NEWVOL IGNORE is specified
------->CONTROL (root phase)

Else if assignment is ALT, issues MSAT ID=ALP/ALT for permanent/temporary alternate assignment.

Alternate assign to SYSOUT is only allowed if SYSOUT has been assigned before.
------->CONTROL (root phase)

Else goes to

ASSGN20: Entry point from ASSGN

ASSGN20A: If the assignment is not permanent ------->ASSTEMP

Else goes to

ASSPERM: Prepares permanent assignment.

Completes the CLOSE for the file if one is in progress.

ASSGN23: If SYSOUT is assigned and assignment is for SYSOUT, issues MSAT ID=DEL for both SYSPCH and SYSLST.

ASSGN23C: Issues MSAT ID=RTL request for scan of all stored assignments including ALT assignments. If the assignment is for SYSIPT or SYSRDR, resets the 80/81 byte indicators.

ASSGN23K: If assignment is to UA or IGN ------->ASSGEN24A

Else

ASSGN23H: If the new assignment is for MFCM/MFCU, the hopper information is set in PUB+5.

ASSGN23G: If the new assignment is for tape and a MODE was specified, the standard MODE in PUB+7 is changed.

ASSGN25: Updates the SET MODE byte of the PUB (PUB+5). ------->ASSGEN24A

ASSTEMP: Prepares assignment. Completes the CLOSE for the file if one is in progress.

ASSGN28: If the device is a tape and MODE was specified, updates the SET MODE byte in PUB+5.

ASSTEM0: Issues a MSAT ID=RTL request and scans all matched elements in order to reset the PUB-ownership bits if possible.

ASSTEM4: If the assignment is for SYSRDR or SYSIPT, saves and resets the 80/81 byte indicators.

ASSGN28C: If the assignment is for MFCU/MFCM, sets hopper specification in PUB.

ASSGN24: If it is not a temporary assign ------->ASSGEN24A

Else saves permanent assignment by MSAT ID=PER.

ASSGN24M: Makes actual assignment by moving the new LUB from the work area to the LUB table. If the ASSIGN statement is for SYSOUT, an assignment must actually be performed for both SYSPCH and SYSLST.

ASSGN24B: If the assignment was for SYSLOG and the device is CRT or 1052, cuu is set into the system COMREG.

ASSGNB7: Inserts the ownership bit into the PUBOWNER table.

ASSGN29: If the assignment was for SYSIN, SYSIPT must also be assigned.

ASSGN32: If SYSLST is assigned, resets the line count to one. ------->CONTROL (root phase)

OPENSYS: Opens system I/O units on DASD or tape:

Updates the DTFs for DASD devices.

If it is not for a system device ------->CONTROL (root phase)

Else sets up the DTF for OPEN.

Opens the file.

Initializes a DIB entry if the system file is on disk.

Sets the 80/81 byte indicators (a record is read to find the record size) for SYSIPT/SYSRDR.
------->Return

CLOSESYS: CLOSE statement processing:

1. Closes a logical unit:
   - The unit may optionally be reassigned to another device made UA or IGN, or if a tape it may be specified as an alternate (ALT) unit. If the unit is a system unit, one of the optional operands must be specified.
   - If an optional operand is not specified, the programmer logical unit is closed and the assignment remains unchanged.
   - If ALT has not been specified ------->CLOSE
2. Causes the logical unit to be closed and an alternate unit to be opened:
   - The ALT operand is valid only for the system output units SYSPCH, SYSLST, or SYSOUT when they are assigned to tape.
     ------->CONTROL (root phase)

Phase $JOBCTLE

Module Name: IJBJC9

Entry Points:

```
JOBCTLE, to process EXEC statement
JOBCTLE + 4, back to CONTROL (formerly PRTY statement)
JARETURN, return point from $JOBCTLN
JOBCTLE + 12, to process PROC statement
JOBCTLE + 16, to process SETPARM statement
JOBCTLE + 20, to process PWR statement
JOBCTLE + 24, to process MSECS statement
```

Function: EXEC, PROC, SETPARM, PWR and MSECS statement processing

(For details see “Sequence of Operation” below.)

Called By: $JOBCTLA

Phases Called:

```
$IJBJCCN
$JOBCTLN
$IJBSLA
$IJBCJC
$IJBPROC
```

Data Areas -- Job Control Used:

```
ACCTABLE --  Partition accounting table DSECT
AVRADR   --  Volume characteristics table DSECT
BASDST   --  Job control common area DSECT
COMDST   --  Partition communication region DSECT (macro MAPCOMR)
DIRECTRY --  DSL directory entry DSECT
LBRACB   --  Librarian access control block (in librarian,
             phase $IJBLBR)
PHLST    --  Phase load trace table DSECT (pointed to be COMDST,
             entry IJBPHLST)
PIB      --  Program information block in supervisor
SYSCOM   --  System communication region DSECT
```

Messages Caused:

0S35I	1F4nD	1S3nt	1S6nt
1C91I	1M7nD	1S40t	1S76I
1F02D	1M80D	1S43D	1S77D
1F03D	1M81D	1S44t	1T71t
1F04D	1M9nt	1S47I	1U00t
1F05D	1N10D	1S48D	1U40t
1F07D	1N11t	1S49D	1U5nt
1F08D	1N2nt	1S53D	1U6nt
1F09D	1N92D	1S56D	1U70A
1F1nD	1P04D	1S57D	1YGnt
1F2nD	1P2nt	1S58D	1Y4nt
1F3nD	1S1nt	1S59D	

Input: EXEC, PROC, SETPARM, PWR and MSECS statements

Output: Register contents when calling phase $IJBJCCN

See “Interface from and to Phase $IJBJCCN” on page 256.

Exit Normal:

```
to $JOBCTLA  entry  CONTROL
to $JOBCTLN
```

Exit Error: to $JOBCTLA error routines

Register Use:

```
R 3: Work register, interface to scan routine, addr. of ERRRTN
R 4: Work register and interface to scan routine
R 5: Work register and interface to scan routine
R 6: Third base register
R 7: First base register of root phase and address of BASDST
R 8: First base register
R 9: Second base register
R10: Pointer to COMDST (partition COMREG)
R14: Return register for common subroutines
```

Sequence of Operation:

JOBCTLE: Establishes addressability

EXEC: EXEC Processing: If the EXEC statement is for a cataloged procedure ------->CALL

EXIIO: Checks presence and length of program name.

ACCTNTI: If job accounting is supported and if this statement is not the first after JOB ------->$JOBCTLN

JARETURN: Return point from $JOBCTLN

EXBLNK: If there is no phase name in the EXEC statement the name of the phase to be executed is obtained via the GETPN macro.

EXTRCT: Gets partition boundaries for the following operations:

MOREOP: Processes the operands:

```
TESTREAL     REAL
TESTSIZE     SIZE=AUTO
             nK
             (AUTO, nK)
             phasename
             (phasename,nK)
TESTGO       GO
TESTPARM     PARM='value'
TESTDSPC     DSPACE=nK|mM
TESTTRCE     TRACE
TESTNPA      NPA
TSTOS390     OS390
```

EXCEDT: If it is EXEC LNKEDT, an ENTRY statement is written to SYSLNK.

GETDIR: Checks via DIRECTRY if the phase or, if necessary, the largest of several phases, fits into the available partition.

CLPHLST: Clears the phase load list.

REGLOAD: Loads registers as input to $IJBJCCN. ------->$IJBJCCN

CALL: EXEC PROC=Processing: If the procedure name starts with $$, replace

- (in a static partition) the second $ by the partition number
- (in a dynamic partition) the first $ by the class character

CALLNORM: Gets and checks procedure name as well as additional operands, if any.

SETPARM: Get and check operands of statement.

Put parameter and its value into parameter list.

Store parameter list via PARMMAC macro into SVA.

EXECP1: Process expressions in EXEC PROC statement.

Prepare access to wanted procedure via PROCMAC macro.

Store parameters and values of EXEC PROC statement via PARMMAC macro into SVA.

PROCST: Process expressions in PROC statement. Store parameters and values of PROC statement via PARMMAC macro into SVA.

SCNPRC: Scan next expression and put it into PARMLE.

PUTLST: Put parameter list entry PARMLE into parameter list.

GETRC: Get last or maximum return code.

CHKPN: Check parameter name and store it in PARMLE.

MSECS: MSECS statement processing.
Test for ASI and for operand n.

MSECS1: Check operand n and convert it to binary. Set new time slice value.
------->CONTROL (root phase)

Phase $JOBCTLF

Module Name: IJBJC5

Entry Points: IJBJOF + displacement for each statement

Function: DVCDN, DVCUP, LISTIO, RESET, UNBATCH processing

(For details see “Sequence of Operation” below.)

Called By: $JOBCTLA

Phases Called:

- $IJBLBR to drop LIBDEF chain
- $IJBSLA (via LABEL macro) to clear group of labels during UNBATCH processing.
- IJJTLIB (via LBSERV macro) to release 3490E tape drives during UNBATCH processing.

Data Areas -- Job Control Used:

```
ACCTCOMN -- Common accounting table
AVRENTRY -- Volume characteristics table DSECT
BASDST   -- Job control common area DSECT
BUFFER   -- Message buffer in BASDST
COMDST   -- Partition communication region DSECT (macro MAPCOMR)
LBCONREL -- Library concatenation table DSECT
LOUTBUF  -- Logical LISTIO output buffer
LUBTAB   -- LUB table in supervisor
MSAT table in supervisor
PIBTAB   -- PIB table in supervisor
POUTBUF  -- Physical LISTIO output buffer
POWPCB   -- POWER program control block
PUBOWN   -- PUB owner table in supervisor
PUBTAB   -- PUB table in supervisor
SYSCOM   -- System communication region DSECT
XTNTAR   -- Extent area in SVA
```

Resource Control: Access to the LUB, PUB, and PUBOWN tables by other partitions or tasks is controlled by the supervisor macros LOCK and UNLOCK. Resource name is CIOBLOCKS. The UNLOCK is done in the root phase $JOBCTLA, near label CONTROL or CONTROL2.

Messages Caused:

1A4nD	1A84D	1A9nt	1AA1t
1A5nt	1A85I	1AA0D	1S40t
1A70D	1A86I		

Input:

```
Job control statement
SYSCOM  System communication region fields
COMDST  Partition communication region fields
Results of the macros: MSAT, EXTRACT, GETVCE
PIB table
LUB table
POWER tables
PUB table
```

Output:

in fields of:

```
LUB table
PIB table
Accounting table
MSAT table
Extent area
PUB owner table
Chain of concatenated libraries
AVR table
LISTIO table
```

Exit Normal: to $JOBCTLA entry CONTROL

Exit Error: to $JOBCTLA error routines

Register Use:

```
R 3: Work register, interface to scan routine, addr. of ERRRTN
R 4: Work register and interface to scan routine
R 5: Work register and interface to scan routine
R 6: Third base register
R 7: First base register of root phase and address of BASDST
R 8: First base register
R 9: Second base register
R10: Pointer to COMDST (partition COMREG)
R14: Return register for common subroutines
```

Sequence of Operation:

IJBJOF: External entry point

DVCUP: Makes a device available after it has been down:

Computes the PUB address of the device specified by the operand X'cuu'.

Sets the control flags in the PUB to indicate the device is up.

If the device is a tape, the standard mode is restored.

TEST3800: If the device is a 3800, the print control buffer is reset to standard. -------->CONTROL (root phase)

LISTIO: Lists the I/O assignment of the system as specified by one of the following operands:

LIOSYS: SYS

LIOPROG: PROG

LIOBGFG: BG, Fn=F1,F2,F3,..., etc.

LIOALL: ALL

LIOSYX: SYSxxx

LIOUNITS: UNITS

LIOASGN: ASSGN

LIODOWN: DOWN

LIOUA: UA

LIOCUU: cuu

LIONPGR: NPGR ------->CONTROL (root phase)

DVCDN: Indicates that a device in the PUB table is no longer available to the system.

DVCDNA: Refuses DVCDN for devices owned by VSE/POWER.

DVCDN14: Refuses DVCDN for SYSRES, SYSREC, SYSDMP, SYSCAT or the page data set. Indicates in the PUB that the device is down.

DVCDUA: Unassigns all standard or permanent assignments to the device in the LUB and MSAT tables.

Unassigns all temporary assignments to the device in the LUB.

Removes all alternate assignments to the device.

Logs all assignments that have been released.

MDR3895R: Causes records to be written on SYSREC for the 3895 and 3800 devices. ------->CONTROL (root phase)

RESET: Resets logical unit assignments to permanent or standard as specified by one of the following operands:

RESALL: *ALL*: resets all assignments in current partition.

RESSYS: *SYS*: resets all system logical unit assignments in current partition.

RESPROG: *PROG*: resets all programmer logical unit assignments in current partition.

RESSYX: *SYSxxx*: resets a single logical unit assignment.

Note: SYSRDR, SYSIPT, and SYSIN cannot be reset during procedure processing.

RSTXTNT: De-queues all EXTENTs to the LUB specified.

ALLXTNT: De-queues all extents to all programmer units in the partition. ------->CONTROL (root phase)

UNBATCH: Housekeeping when detaching a foreground partition:

Refuses UNBATCH and issues a message if SYSRDR, SYSIPT, SYSPCH, or SYSLST is left assigned to tape or disk.

RELLIBS: Resets all permanent library definitions except when the HOLD statement was given.

UNA1A: Resets all assignments to UA except assignments for SYSLOG, SYSRES, SYSREC, SYSDMP, and SYSCAT. Releases all tape devices mounted via LIBSERV (via BAL to RSTMOUNT, which resets permanent mount information by means of MODFLD and LBSERV macro).

NORELS: Resets SETPFIX limits. Issues an EOJ macro to have job control re-initialize. Final unbatch is done in $JOBCTLA (via TSTOP COND=UNBATCH).

Phase $JOBCTLG

Module Name: IJBJC3

Entry Points: JOBCTLG + displacement for each statement

Function: JOB, /&, /+, OPTION, CANCEL, ID, START processing.

(For details see “Sequence of Operation” below.)

Called By: $JOBCTLA

Phases Called:

```
$JOBCTLM
$JOBCTLN
$IJBSTRT
$IJBCJC
$IJBLBR (to drop temporary LIBDEF chain)
$IJBPROC
$IJBSLA
DTSECJCL
IJJTLIB (via LBSERV macro) to release 3490E tape drives during /& processing.
```

Data Areas -- Job Control Used:

```
ACCTABLE -- Partition accounting table DSECT
BASDST   -- Job control common area DSECT
BUFFER   -- Message buffer in BASDST
COMDST   -- Partition communication region DSECT (macro MAPCOMR)
CONCCTL  -- Library concatenation table DSECT
IJJLBSER -- Interface area for LBSERV macro
LABELBUF -- Buffer for interface with $IJBSLA
LUBTAB   -- LUB table in supervisor
MSAT table in supervisor
Partition GETVIS area
PJBADR   -- DSECT for job information control block (macro MAPPOWJB).
PUBOWN   -- PUB owner table in supervisor
PUBTAB table in supervisor
RSVARA   -- Partition save area DSECT
SYSCOM   -- System communication region DSECT
XTNTAR   -- Extent area in SVA
```

Messages Caused:

1C60D	1L61I	1L70D	1S1nt
1C90I	1L62D	1M21D	1S40t
1F00I	1L63I	1M4nD	1S72D
1F01I	1L64D	1M82I	1S73t
1I00D	1L65t	1N7nD	1S78I
1I20I	1L66D	1N80I	1S8nt
1I21I	1L67D	1N90I	1U00t
1L1nD	1L68D	1P1nD	1U71I

Input: Any of the control statements listed under “Function”

Output:

- A generated /+ statement for CONTROL2
- Fields in:

```
- accounting tables
- COMDST -- Partition communication region DSECT
- LUBTAB -- LUB table
- MSAT table
- label information area
- extent area
- partition GETVIS area
- chain of concatenated libraries
- PUB owner table
```

Exit Normal: to $JOBCTLA entry CONTROL

Exit Error:

```
to $JOBCTLA  error routines
CANCEL
```

Register Use:

```
R 3: Work register, interface to scan routine, addr. of ERRRTN
R 4: Work register and interface to scan routine
R 5: Work register and interface to scan routine
R 6: Third base register
R 7: First base register of root phase and address of BASDST
R 8: First base register
R 9: Second base register
R10: Pointer to COMDST
R14: Return register for common subroutines
```

Sequence of Operation:

JOBCTLG: External entry point.

CANCEL: This statement is ignored if a job is not in process.

Displays the message “JOB job name CANCELED DUE TO OPERATOR INTERVENTION.”

Sets Operator Cancel Bit in JCSW7 to indicate operator cancel. ------->EOJ

EOJ: Resets conditional job control information.

Cleanup of procedure and parameter tables.

Resets the chain of concatenated libraries.

Calls the EOP processing routine if a procedure is being executed. ------->EOP

Sets off SYSRDR EOF in PUB.

RSTLBL: Clears user label area.

NOIPT7: Restores all LUB assignments to permanent.

Resets job control options to standard.

Resets the job name field in the communications region to NO NAME.

Sets job status bit off.

Displays EOJ message and logs the time and job duration.

Cleans up LOGON authorization.

Resets link control bits in COMREG + 57.

Resets the buffers of assigned 3800 printers.

Transfers control to JOB statement processing at label RSTCOM (or at label SIMEND if EOJ processing was caused by a JOB statement). ------->RSTCOM

JOB: Clears user label area.

Resets the chain of concatenated libraries.

Resets conditional job control information.

Cleanup of procedure and parameter tables.

Resets the link control bits in COMREG + 57.

If the previous job has not been completed (no /& statement read) goes to EOJ and returns at SIMEND.

SIMEND: Fetches $JOBCTLN if job accounting is active. ------->$JOBCTLN

JOBPROC: Restores all options to standard.

Moves the job name to the communication region.

Saves job start time in register save area of the partition.

Logs the JOB statement on SYSLST/SYSLOG.

Logs the time on SYSLOG, if TOD is supported.

RSSASG: Restores all LUB assignments to permanent or standard (by branching to RSTASG).

RSTCOM: Resets job zone in communication region system zone.

Resets the EOF indicators in all PUBs.

Sets switches to initialize, check, or bypass the recorder file.

RSTOPT: Resets temporary options to standard options. Resets maximal return code (PJBJRET) in the job information control block (PJB) (see VSE/ESA Supervisor Diagnosis Reference).

RSTCONC: Resets temporary LIBDEF chain by means of LBRCTUPD request. Performs a UNLOCK ALL to free all resources locked by the partition.

RSTPFX: Resets temporary SETPFIX limits by means of SETLIMIT macro.

RSTMOUNT: Resets temporary mount information established by the LIBSERV command by means of MODFLD and LBSERV macro.

RSTID: Resets job authorization acquired by an ID command. This is done by means of a BAL to the address contained in SYSCOM.IJBSCTAB.SCYSCJCL (JCL-logoff linkage), which is the start address of phase DTSECJCL.

OPTION1: Sets the job control options requested by the programmer.

Scans the parameters of the OPTION statement, one at a time, and passes control to the correct processing routine as shown in “Job Control Option Bytes in COMDST” on page 244.

If last parameter was processed
------->CONTROL (root phase)

Else ------->OPTION

DELLABEL Deletes labels in a label subarea (when executing // OPTION STDLABEL=DELETE or PARSTD=DELETE or CLASSTD=(class,DELETE). Subroutine DELREC is called which sets a LOCK for resource name CJC3LBL.

EOP: Most code of this routine is also executed if a /& statement or CANCEL statement is issued while a procedure is being processed and no $CANCEL ON-condition is in effect.

EOPENT: Entry point for End Of Job and CANCEL.

NORESET: If a JCL statement is found in the comment part of the /+ statement, executes it via CONTROL2 in the root phase. Return via register 14 . ------->CONTROL (root phase)

FREEVIS: Performs a FREEVIS ALL. ------->CONTROL (root phase)

START: Executes a STARTP macro for the partition specified. ------->CONTROL (root phase)

ID: If security is used and a job is in process, the routine executes the LOGON authorization routing. This is done by means of a BAL to the address contained in SYSCOM.IJBSCTAB.SCYSCJCL (JCL-logon linkage), which is the start address of phase DTSECJCL.
------->CONTROL (root phase)

Phase $JOBCTLH

Module Name: IJBJCH

Entry Point: IJBJOH

Function: LIBDEF, LIBDROP, and LIBLIST statement processing which does the following:

- Syntax checking
- Validation
- Update of control tables (LOT, LDT, SDT, EDT, DDT) for library concatenation
- Listing of library control information

(For details see “Sequence of Operation” below.)

Called By: $JOBCTLA

Phases Called:

```
$IJBLBR, library concatenation services
$IJBASGN
```

Data Areas -- Job Control Used:

```
ASGLIST -- Dynamic assign control block
BASDST  -- Job control common area DSECT
BLDCBMAP-- Control block for build resource name
COMDST  -- Partition communication region DSECT (macro MAPCOMR)
INLCDDTE-- Device definition table (in librarian, phase $IJBLBR)
INLCEDTE-- Extent definition table (in librarian, phase $IJBLBR)
INLCLDTE-- Library definition table (in librarian, phase $IJBLBR)
INLCSDTE-- Sublibrary definition table (in librarian, phase $IJBLBR)
LBRLCTDS-- Control block for access to library control table
LOTENTRY-- Library offset table (in librarian, phase $IJBLBR)
SYSCOM  -- System communication region DSECT
TOLCRQL -- Control block for library definition update
           (in librarian, phase $IJBLBR)
```

Messages Caused:

1D02t	1D09t	1D3nt	1D8nI
1D03t	1D10t	1D4nt	1E1nt
1D04t	1D12t	1D5nt	1E4nt
1D06I	1D13t	1D6nt	1U00t
1D07t	1D14t	1D7nt	

Input:

```
LIBDEF, LIBDROP, and LIBLIST statements
Concatenation table information
```

Output:

```
Library concatenation list
Library concatenation table updates
```

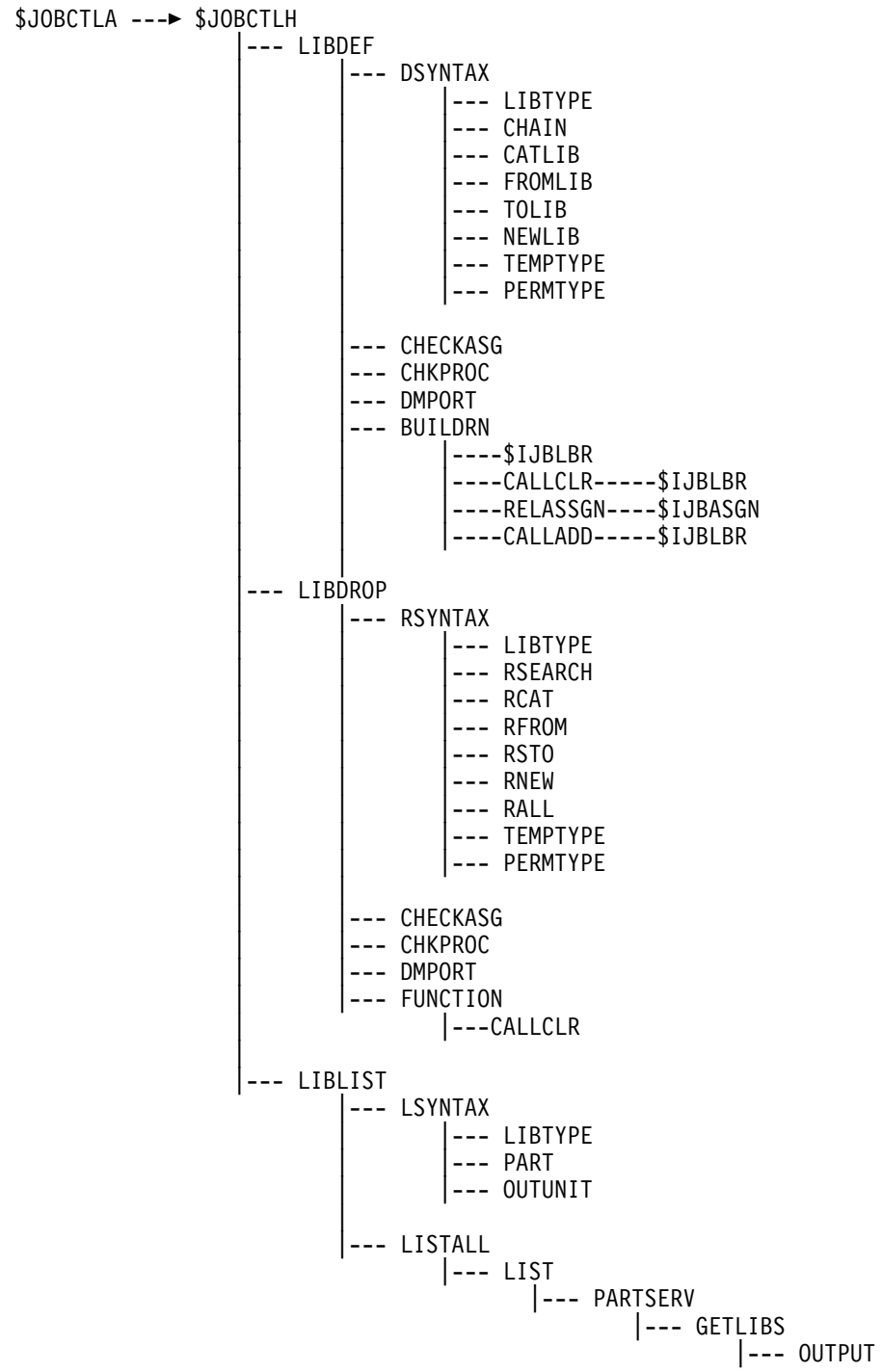
Exit Normal: to $JOBCTLA entry CONTROL

Exit Error: to $JOBCTLA entry ERRRTNO

Register Use:

```
R 3: Work register, interface to scan routine, addr. of ERRRTN
R 4: Work register and interface to scan routine
R 5: Work register and interface to scan routine
R 6: Third base register
R 7: First base register of root phase and address of BASDST
R 8: First base register
R 9: Second base register
R10: Pointer to COMDST
R14: Return register for common subroutines
```

Control Flow within Phase $JOBCTLH: The phase is structured in a way that it seems useful to show the internal control flow for fast orientation:



Sequence of Operation:

IJBJOH: Establishes addressability.

LIBDEF Processing:

LIBDEF: Defines access sequence of libraries in the library concatenation tables:

Checks the syntax of the LIBDEF statement.

If a LIBDEF statement is given in the old form (VSE/Advanced Functions, Release 2 or 3) the library names will be converted to the new format.

Builds a resource name and adds the specified library definition to the library control tables for each library specified. ------->CONTROL (root phase)

LIBDROP Processing:

LIBDROP: Removes library definitions from the library concatenation tables:

Checks the syntax of the LIBDROP statement.

Removes the specified library definitions from the library control tables. ------->CONTROL (root phase)

LIBLIST Processing:

LIBLIST: Lists the library access sequence for a specified library type and a specified partition:

Checks the syntax of the LIBLIST statement.

Accesses and lists the requested library access sequence from the library control tables. ------->CONTROL (root phase)

Phase $JOBCTLI

Module Name: IJBJCI

Entry Points: IJBJCI + displacement for each statement

Function: IF, ON, GOTO processing. Facilitates conditional job control.

(For details see “Sequence of Operation” below.)

Called By: $JOBCTLA

Phases Called:

- $IJBCJC Retrieve and update conditional job control information, especially $RC and $MRC
- $IJBPROC Substitute symbolic parameter in the condition clause of an IF statement

Data Areas -- Job Control Used:

```
BASDST   -- Job control common area DSECT        (macro MAPJCLA)
BUFFER   -- Message buffer in BASDST
COMDST   -- Partition communication region DSECT (macro MAPCOMR)
```

Messages Caused:

1F02D	1F3nD	1F6nD	1S40t
1F06D	1F4nD	1F7nD	1S46I
1F1nD	1F5nD	1M80D	1S49I
1F2nD			

Input:

```
Job control statement
BASDST  Job control common area
COMDST  Partition communication region fields
Results of the macros: CONDJC,PROCMAC
```

Output:

in fields of:

```
BASDST  Job control common area
Results of the macro : CONDJC
```

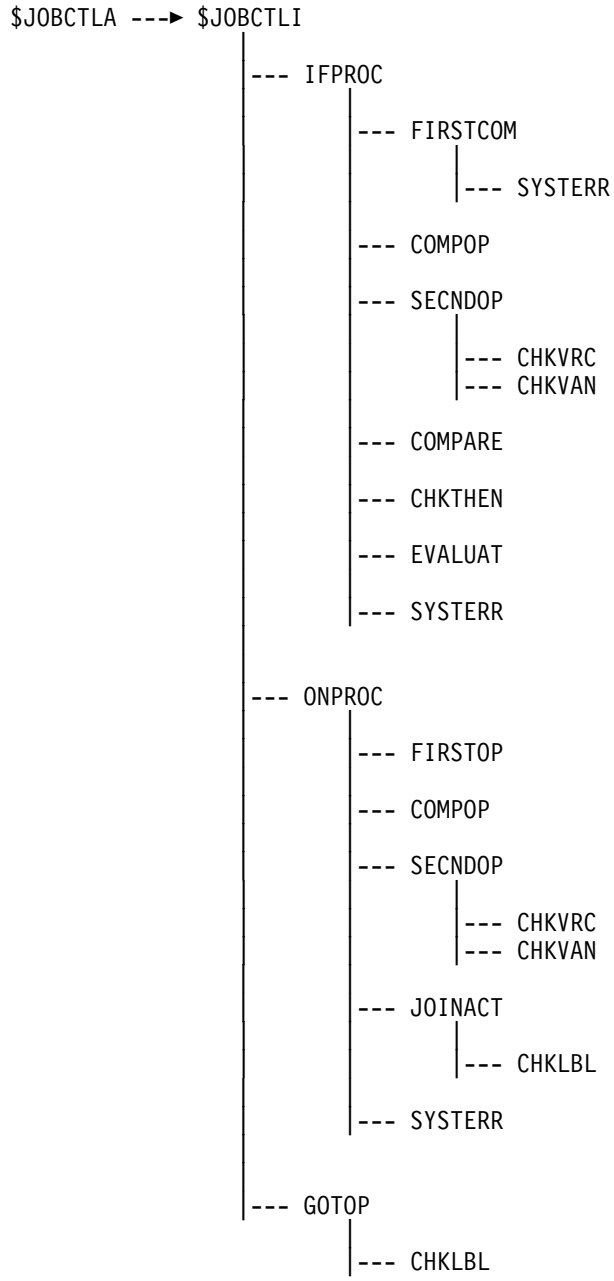
Exit Normal: to $JOBCTLA entry CONTROL

Exit Error: to $JOBCTLA error routines

Register Use:

```
R 3: Work register, interface to scan routine, addr. of ERRRTN
R 4: Work register and interface to scan routine
R 5: Work register and interface to scan routine
R 6: Work register
R 7: First base register of root phase and address of BASDST
R 8: First base register
R 9: Second base register
R10: Pointer to COMDST
R14: Return register for common subroutines
```

Control Flow within Phase $JOBCTLI: Internal control flow for fast orientation:



Sequence of Operation:

IJBJCI: External entry point.

IF Processing:

IFPROC: Decides whether the next statement is to be executed dependent on the condition(s) specified in the IF-statement.

FIRSTCOM: Reads and evaluates first comparand.

COMPOP: Reads and checks comparator.

SECNDOP: Reads and checks second comparand.

COMPARE: Compares first and second comparand.

CHKTHEN: Tests 'THEN' or join operator to second condition.

EVALUAT: Evaluates conditions. Indicates 'SKIP NEXT STATEMENT' in Job Control Common Area if condition is FALSE.
------->CONTROL (root phase)

ON Processing:

ONPROC: Sets a condition and specifies an action. Condition(s) and action are stored by macro CONDJC for subsequent usage.

FIRSTOP: Reads and checks first operand.

COMPOP: Reads and checks comparator in case of '$RC'.

SECNDOP: Reads and checks second comparand in case of '$RC'.

JOINACT: Tests Join Operator to second condition or action.
------->CONTROL (root phase)

GOTO Processing:

GOTOP: Reads and checks the label. Indicate 'GOTO-MODE' and move label name to Job Control Common Area.
------->CONTROL (root phase)

Common Routines:

CHKLBL: Performs syntax check for a label.

CHKVAN: Check string for containing only alpha- or numeric characters.

CHKVRC: Check string for containing a valid return code.

SYSTERR: Issues message 1S40D after unexpected return code of a system macro.

Phase $JOBCTLJ

Module Name: IJBJC4

Entry Points: JOBCTLJ + displacement for each statement

Function:

- Processes the following control statements:

```
ALLOC           MTC             STOP
ALLOCR          NOLOG           UCS
CATALR          NPGR            UPSI
DATE            SET             ZONE
HOLD            SIZE
LOG             STDOPT
```

- Reads the linkage editor statements: ACTION, ENTRY, INCLUDE, PHASE, MODE.
- Writes them to SYSLNK if the link bit in the linkage control byte, COMDST, displacement 57, is on.

(For details see “Sequence of Operation” below.)

Called By: $JOBCTLA

Phases Called: INLPSDL (for SET SDL processing)

Data Areas -- Job Control Used:

```
APL      -- ALLOCATE parameter list
COMDST   -- Partition communication region DSECT (macro MAPCOMR)
CRTSAV   -- C-transient save area in the supervisor
DIRECTRY -- CIL or DSL directory entry DSECT
LUB of SYSUSE in LUBTAB
PIB      -- Program information block in supervisor
SDL      -- System directory list in the SVA
SLPL     -- Parameter list for SETLIMIT request
```

Messages Caused:

1A2nt	1I39t	1S0nt	1S52D
1A4nD	1I98I	1S1nt	1S71D
1A5nt	1P01D	1S2nI	1T10I
1A70D	1P02D	1S40t	1T1nD
1B08I	1P03I	1S42A	1U1nD
1C10D	1P04D	1S45D	1U3nD
1C30t	1P05D	1S51D	1YLnt

Input:

```
Linkage editor statements from SYSIPT
The job control statements mentioned under “Function” above
```

Output:

- User phase (print buffer image) is loaded.
- UCS print buffer is loaded.
- ACTION, ENTRY, MODE, and PHASE statements to SYSLNK.

To COMDST:

- Data moved from input statement into buffer.
- Date bit (X'02' in COMDST+59) set on if TOD active switch is set in UPSI byte (COMDST+23).
- ZONE stored in COMDST extension + 70.
- HOLD assignment bit in PIB X'08' byte 13 set on.
- LOG switch turned off.

Exit Normal: to $JOBCTLA entry CONTROL

Exit Error: to $JOBCTLA error routines

Register Use:

```
R 3: Work register, interface to scan routine, addr. of ERRRTN
R 4: Work register and interface to scan routine
R 5: Work register and interface to scan routine
R 6: Third base register
R 7: First base register of root phase and address of BASDST
R 8: First base register
R 9: Second base register
R10: Pointer to COMDST
R14: Return register for common subroutines
```

Sequence of Operation:

JOBCTLJ: Establishes addressability.

NOLOG: If the NOLOG statement is used without //, the log-on-SYSLOG switch (COMDST + 56, bit 3) is set off. If the statement is used with //, the log-on-SYSLST switch (COMDST + 59, bit 3) is set off. ------->CONTROL (root phase)

STOP: A WAIT macro pointing to a dummy CCB is issued to force entry into the supervisor task selection routine. This removes the partition from the task selection mechanism. ------->CONTROL (root phase)

ALLOC(R): This routine reallocates the virtual address area for an ALLOC statement, or the real address area for an ALLOCR statement, according to the number of 2K blocks specified for one or more partitions. Therefore, the routine does the following:

```
  If no space-id is specified, reject a statement with more than
  1 partition specified.

  Builds a table (APL) containing the number of
  2K blocks for each specified partition.

  Indicates in APL: Virtual allocation, if the ALLOC
  statement was specified, or real allocation, if the
  ALLOCR statement was specified. Default the space-id, if none is given.

  Issues the 'ALLOCATE' supervisor macro. Checks
  the return code and issues a message if it is unequal
  to zero.
------->CONTROL (root phase)
```

LOG: If the LOG statement is used, the log-on-SYSLOG switch (COMDST + 59, bit 3) is set on. ------->CONTROL (root phase)

SIZE: This routine is used for dynamic storage limit specification. It does the following:

```
  Builds a table (SLPL) containing the number of 2K
  blocks of PFIX and SIZE.

  Issues the SETLIMIT supervisor macro using the
  SLPL as interface.

  Checks the return code after execution of the
  SETLIMIT macro and issues a message if the code
  is not zero.
------->CONTROL (root phase)
```

UCS: The statement is scanned and the phase specified by the operand “phasename,” is loaded into the buffer of the logical unit (operand “SYSxxx”) assigned to the respective printer. The statement is ignored for a 3800 printer. The specification of the optional operands FOLD, BLOCK, and NULMS is stored in the second and third CCWs. ------->CONTROL (root phase)

HOLD: All I/O assignments for the foreground partitions specified are to stay in effect from job step to job step. The operands of the HOLD statement, Fx, can appear in any sequence. The hold flag, bit 4 of PIB + 12, is set on. ------->CONTROL (root phase)

ACTION: This statement is invalid if the link option (JCSW2, bit 0) is off. If valid, this statement is copied to SYSLNK. ------->CONTROL (root phase)

The other linkage editor statements ENTRY, PHASE and MODE are processed in the same way as ACTION.

STDOPT: Sets standard job control options in all partitions. Uses LOCK name $IJBSTDOPT to serialize concurrent update requests.

UPSI: The operand of this statement is converted to a single byte and stored in COMDST + 23:

- A character 1 sets the bit on.
- A character 0 sets the bit off.
- A character X leaves the bit unchanged.

->CONTROL (root phase)

SET: Used to change the UPSI byte, the system date, the line count, the system time, and the remaining disk capacity of SYSLST or SYSPCH when they are assigned to disk; also to define to the system the status of the recorder file SYSREC. Processing is as follows when the operand is:

DATE= No longer supported.

LINECT= The parameter is converted to binary and stored in COMDST + 78. The job control line count for the remaining lines on the current page is adjusted.

RF= After the specified parameter has been checked for validity, the proper option bit is set in the RF table status flag byte (byte 1):

```
CREATE  - X'08'
YES     - X'04'
NO      - X'10'
```

HC= After the specified parameter has been checked for validity, the proper option bit is set in the CRTSAV table status flag byte:

```
CREATE  - X'08'
YES     - X'00'
NO      - X'04'
```

UPSI= The operand of this statement is converted to a single byte and stored in COMDST + 23.

- A character 1 sets the bit on.
- A character 0 sets the bit off.
- A character X leaves the bit unchanged

->CONTROL (root phase)

RCLST= The parameter of this operand is converted to binary and stored in the SYSLST DIB.

RCPCH= The parameter of this operand is converted to binary and stored in the SYSPCH DIB.

SVA= (nK,nK)

This operand is no longer processed. For compatibility reasons it is read and flushed.

SDL Uses LOCK name $IJBSETSDL to serialize concurrent update requests. Building of the system directory list in the SVA is started and statements are read from the input device. If a phase starting with $JOBEX is detected (job control exit routine), then the high order bit in JUEFLAG1 is set on. This requests to build a new control block for the job control exit routines.
->CONTROL (root phase)

MRCZERO Invokes the CONDJC macro with function RESETMRC to reset the maximal return code.

RCZERO Invokes the CONDJC macro with function RESETRC to reset the return code of the last job step.

CATALR: This statement is invalid if the LINK option (JCSW2, bit 0) is on. If valid, CATALR/SYSPCH mode switch (JBCSW5, bit 0) is set for handling the PHASE statement. The CATALR statement is checked for correct syntax. The version and modification number (V.M.) is checked for valid characters. A valid statement is punched or written on SYSPCH. If the statement is invalid, the job is canceled.
------->CONTROL (root phase)

DATE: If specified in the old format (8-byte operand), the operand is moved to COMREG.JOBDATE without any checking. The old date bit in COMDST + 233 (COMREG.JCSW9.IJBDATEO) is set on.

If specified in the new format (10-byte operand), the operand is checked for correct date format. If check is negative, message 1YLnt is issued. If check is positive, the operand is moved to the appropriate parts of structure COMREG.JOBDATWC. The old date bit in COMDST + 233 (COMREG.JCSW9.IJBDATEO) is turned off.

In both cases the date bit in COMDST + 59 (COMREG.JCSW4.DATEBIT) is set on.
------->CONTROL (root phase)

INCLUDE: This statement is invalid if the LINK option (JCSW2, bit 0) is off. If the statement contains an operand, the complete statement is copied to SYSLNK. If the statement does not contain an operand, SYSIPT data is copied to SYSLNK until a /* statement is read on SYSIPT. INCLUDE and /* are not copied to SYSLNK. ------->CONTROL (root phase)

MTC: Controls specified tape operations on specified logical units. Checks the operation code, computes or locates the PUB pointer, assigns it to the SYSUSE LUB, and executes the operation. The duplication factor determines the number of times a specified operation is to be repeated.

ZONE: Checks validity of operands. Converts value to minutes and moves it into the partition communication region zone field (COMREG.JOBZON). ------->CONTROL (root phase)

NPGR: Changes the number of programmer LUBs for one or more partitions:

- builds a table called NPGLIST which is the interface to the NPGR macro
- the NPGR parameter values are checked for correct syntax and converted into the NPGLIST
- the NPGR macro is performed
- the return code of the macro is checked and, if not zero, an error message is issued.

Phase $JOBCTLK

Module Name: IJBJC6

Entry Points:

IJBJOK	RSTRT
IJBJOK + 4	DLBL
IJBJOK + 8	EXTENT
IJBJOK + 12	TLBL
IJBJOK + 16	SETPRT
IJBJOK + 20	LBLOUTF

Function: Processes the following statements:

- DLBL
- EXTENT
- RSTRT
- SETPRT
- TLBL

(For details see “Sequence of Operation” below.)

Called By: $JOBCTLA

Phases Called:

- $IJBSLA (in case of DLBL, EXTENT or TLBL statement)
- $IJBY224 (if the date operand in a DLBL/TLBL statement has the format yy/ddd with yy not equal 00)
- $JOBCTLB (in case of RSTRT statement)

Data Areas -- Job Control Used:

BASDST	Job control common area DSECT
COMDST	Partition communication region DSECT (macro MAPCOMR)
DSKWT	DSECT of keyword table for SETPRT
DSPRM	DSECT of parameter table for SETPRT
LBLDST	DSECT of tape and disk labels (DLBL/TLBL)
LPLDCT	DSECT of label parameter list, required by LABEL macro
PCEADR	DSECT of partition related control blocks, used to retrieve PCEALUB (partition's LUBTAB address) and PCEPIB (partition's PIB address). This is required to process the SYSxxx operand of the SETPRT and RSTRT commands.
PIBADR	DSECT of partition information block, used to retrieve PIBLUBID and PIBLUBNO. This is required to process the SYSxxx operand of the SETPRT and RSTRT commands.
SPPLIST	SETPRT parameter list
SYSCOM	System communication region DSECT

Assembler Macros Used:

- ASYSCOM
- CDLOAD
- DFB
- DTL
- LABEL
- LOAD
- LPL
- LPLDCT

- MAPCOMR
- MAPJCLA
- MAPPCE
- MAPPIB
- SPLIST
- SYSCOM
- YEAR224

Messages Caused:

1A4nD	1L1nD	1L5nD	1S40t
1A5nt	1L2nt	1L60D	1S70D
1L0nt	1L30D	1S1nt	

Input: The statements listed under “Function” above

Output: Label records in label information area

Exit Normal: to $JOBCTLA entry CONTROL

Exit Error: to $JOBCTLA service routines

Register Use:

```
R 3: Work register, interface to scan routine, addr. of ERRRTN
R 4: Work register and interface to scan routine
R 5: Work register and interface to scan routine
R 6: Third base register
R 7: First base register of root phase and address of BASDST
R 8: First base register
R 9: Second base register
R10: Pointer to COMDST
R14: Return register for common subroutines
```

Sequence of Operation:

IJBJOC: External entry point.

TLBL: Inserts default values for all operands.

Tests filename and moves it to the label output area.

Checks, converts, and moves the file ID and date operands to the output area. If the century information is missing in the date operand, the YEAR224 macro with WINDOW=99 is used to provide the proper expiration date. The creation date is determined with WINDOW=20 (expiration dates are supposed to be future-oriented while creation dates are not).

Checks the remaining operands and moves them to the output area overlaying the default values.

Inserts default values for file access control, block count, and system code.

EXTENT: Checks the sequence of the EXTENT statement.

Sets default values from the default table.

Processes the operands overlaying the default values where specified. ------->CONTROL (root phase)

RSTRT: Computes the LUB and PUB pointers for SYSxxx.

Saves the checkpoint number.

Ensures that all label processing is done and closes the user label area.

Causes $JOBCTLB to be loaded by $JOBCTLA. ------->CONTROL (root phase)

DLBL: Branches to LBLOUTF (writes any pending label information record from the output area into the label information area).

Inserts default values for all operands.

Builds a default table for EXTENT processing.

Tests the filename operand and moves it to the output area.

Checks, converts, and moves the file ID and date operands to the output area. If the century information is missing in the date operand, the YEAR224 macro with WINDOW=99 is used to provide the proper expiration date.

Inserts the filename as default for file ID.

Inserts the proper file type code for SD, DA, DU, VSAM, or IS.

Sets on the data security indicator if DSF was specified.

Checks BUFSP=n and CAT=filename and puts their values into the output area.

Issues an “invalid statement” message if the filetype code is not VSAM.

Sets an indicator that only EXTENT statements may follow.

Checks and moves the BLKSIZE operand.

Issues an “error” message if it is wrong or the filetype is VSAM.

Checks and moves the CISIZE operand.

Checks and moves the DISP, RECORDS, and RECSIZE operands.

Issues an error message if filetype is not VSAM. ------->CONTROL (root phase)

SETPRT: Initializes SPPLIST to zero.

Computes the LUB pointer for SYSxxx and moves it to SPPLIST.

Scans the keyword table for the keywords defined in the statement.

Checks the value and its delimiters on finding a match.

Gives control to the routine defined in the keyword table which sets fields or flags in the SPPLIST.

Sets the undefined parameters of DFLT to default values if DLFT=Y was specified.

Gives control to SETAPRT in $JOBCTLA where the SETPRT macro is issued.

On return from $JOBCTLA ------->CONTROL (root phase)

LBLOUTF: Writes the label information record from the output area into the label information area.

Invokes the LABEL macro with functions CLRGRPL, ADDLBL and ADDNXL.

Phase $JOBCTLM

Module Name: IJBJC7

Entry Points:

```
JOBMJOB when a JOB statement is read
JOBMROD when a ROD statement is read
```

Function:

- Initializes the recorder file.
- Processes the ROD statement.
- Reads the MCH and CCH records from the service console and writes them to SYSREC to support common EREP1 for VSE/Advanced Functions and OS/VS. (Routine FRAMPROC)

(For details see “Sequence of Operation” below.)

Called By: $JOBCTLG

Phases Called:

```
$JOBCTLC
$$ABERAR, builds 3895 MDR records
$$ABERBE, builds DASD MDR records
$$ABERBC, builds 3800 MDR records
$$ABERAK, builds 8809 MDR records
$$ABERAT, builds 3890 MDR records
```

Data Areas -- Job Control Used:

```
AVRADR   -- Volume characteristics table DSECT
BASDST   --  Job control common area DSECT
Channel-ID field (low storage)
CCBADR   --  DSECT for CCBs
COMREG   --  Partition communication region DSECT
Program check new PSW (low storage)
PIBCOMRA --  PIB communication region DSECT
PUBADR   --  Physical unit block DSECT
PUB2ADR  --  PUB extension DSECT
PUBADR   --  DSECT for PUBs
RFTABLE  --  RMSR communication table DSECT
SYSCOM   --  System communication region DSECT
```

Messages Caused and Issued:

1I81I	1I83A	1I86A	1I92D
1I82t	1I84A	1I90D	1I93t

Input: None

Output: Recorder file header and IPL, EOD, MDR, and OBR records are written to the recorder file.

Exit Normal: to $JOBCTLC: CRT hard-copy file initializer

Exit Error: to $JOBCTLA error routines

Register Use:

```
R 1: Pointer to CCB
R 2: Pointer to SYSCOM
R 3: Pointer to PUB
R 4: Pointer to PUB2
R 5: Pointer to AVR Table
R 7: First base register of root phase and address of BASDST
R 8: First base register
R 9: Second base register
R10: Pointer to COMREG
R11: Pointer to RFTABLE
R12: Pointer to PIBCOMRA
R14: Return register for common subroutines
```

Sequence of Operation:

JOBMJOB: External entry point.

MAIN00: Recorder File Initialization:

Initializes the phase via INIT00. If it is not the first execution ------->$JOBCTLC

OPENFILE: Opens the recorder file. If not successful issues message 1I84A. ------->Hard Wait

BLDHDR00: Builds recorder file header record.

WRTHDR00: Writes the recorder file header record.

BLDIPL00: Builds and writes IPL record. ------->$JOBCTLC

JOBMROD: External entry point.

ROD00: Processes the ROD statement.

Initializes the phase via INIT00.

NEXTP00: Scans PUB table and builds OBR and MDR records according to the device type via the $$ABERxx phases. Calls the appropriate record writer transients via SVC44.

RODCMP: If ROD function is completed ------->$JOBCTLC

Phase $JOBCTLN

Module Name: IJBJC8

Entry Points:

GO: For normal, simulated, and PAUSE EOJ and for EXEC

CANCEL: From $JOBCTLA for housekeeping if $JOBACCT itself was canceled

Function:

- Provides interface between the VSE/Advanced Functions system and the job accounting phase $JOBACCT to let the user access job accounting information.
- If VSE/POWER is in control of the partition, the phase also provides an interface between VSE/Advanced Functions and VSE/POWER to allow the creation of an execution account record. (For more information on VSE/POWER, see the documentation of that Licensed Program).
- Finalizes time fields for job accounting.
- Loads and executes user phase $JOBACCT.
- Reinitializes accounting tables for next job(step).

(For details see “Sequence of Operation” below.)

Called By:

- $JOBCTLA -- When the $JOBACCT user routine was canceled or
  for PAUSE EOJ.
- $JOBCTLE -- For all but the first EXEC statements of a job.
- $JOBCTLG -- At real or simulated end of job.

Phases Called: $JOBACCT, the user phase for job accounting.

For the interface to $JOBACCT see “Diagnostic Aids -- Job Control” below.

Data Areas -- Job Control Used:

```
ACCTABLE -- Partition accounting table DSECT
ACCTCOMN -- Common accounting table
BASDST   -- Job control common area DSECT
```

Messages: None

Input:

```
Task interrupt key
Both accounting tables
```

Output: Updates for both accounting tables

Exit Normal: Return to calling phase except at PAUSE EOJ.

Exit Error: SVC14 if called at PAUSE EOJ or if $JOBACCT was canceled.

Register Use:

```
R 3: Work register, interface to scan routine, addr. of ERRRTN
R 4: Work register and interface to scan routine
R 5: Work register and interface to scan routine
R 6: Third base register
R 7: First base register of root phase and address of BASDST
R 8: First base register
R 9: Second base register
R10: Pointer to COMDST
R14: Return register for common subroutines
```

Sequence of Operation:

GO: Stores registers. Prepares the exit to the user routine $JOBACCT. Calculates and stores final CPU time. Calls GETJA macro to update information in the partition's job accounting tables. Issues a LOCK for resource name $JOBACCT. Loads and gives control to $JOBACCT.

CANCEL: This routine is entered from $JOBCTLA if $JOBACCT itself is canceled by the operator or due to an error so that accounting information is saved.
Saves root phase registers.

RETURN: Restores base registers.

RETCNCL1: Issues SVC91 to pass control to the VSE/POWER Accounting Appendage Routine (SVC90/91), and waits for posting of ECB by VSE/POWER.

RETUNLCK Unlocks resource $JOBACCT.

BACK: Re-initializes job control after return from the user's accounting routine to the partition. Calls GETJA macro to reset information in the partition's job accounting tables.

EXECBACK: If entry was from EXEC ------->$JOBCTLE

If entry was from EOJ ------->$JOBCTLG

SEOJBACK: If entry was from simulated EOJ (JOB statement processing) ------->$JOBCTLG (label FREEVIS)

EOJFRVS: Else ------->SVC14 (EOJ)

Phase $JOBCTLO

Module Name: IJBJCO

Entry Points:

JOBCTLO	to process JCLEXIT command
JOBCTLO+x'04'	to pass AR commands to the Attention Routine
JOBCTLO+x'08'	to process MAP command
JOBCTLO+x'0C'	to process SYSDEF command
JOBCTLO+x'10'	to process QUERY command
JOBCTLO+x'14'	to process SETPFIX command
JOBCTLO+x'18'	to process VDISK command
JOBCTLO+x'1C'	to process LIBSERV command
JOBCTLO+x'20'	to process PRTY command

Function:

- Processes JCLEXIT, MAP, SYSDEF, QUERY, SETPFIX, VDISK LIBSERV and PRTY commands and passes (during BG ASI processing) selected AR commands to the Attention routine (for details see 'Sequence of Operation' below).

Called By:

- $JOBCTLA

Phases Called:

$IJBMAP	to process the MAP command
$IJBSDSP	to process the SYSDEF/QUERY commands
$IJBVDII	to process the VDISK command
$IJBSLIB	to process the LIBSERV command
$IJBPRTY	to process the PRTY command
$IJBVTAP	to process the VTAPE command

Assembler Macros Used:

- AMODESW
- ASYSCOM
- DFB
- DTL
- JCLARIF
- LOAD
- LOCK
- LPL
- MAPCOMR
- MAPJCLA
- MAPUE
- MODDTL
- PMCOM
- SETIME
- SETLIMIT
- SPFXPL
- SYSCOM
- TECB
- UNLOCK
- WAIT

Data Areas -- Job Control Used:

```
BASDST   -- Job control common area DSECT (macro MAPJCLA)
PMCOM    -- Page Management Communication Region DSECT
JCARAREA -- Interface area between JCL/AR and SVA routines
            (macro JCLARIF)
SYSCOM   -- System Communication Region DSECT
MAPUE    -- JCL User Exit Control Block DSECT
COMDST   -- Partition Communication Region DSECT (macro MAPCOMR)
```

Messages:

1S40t	1U75D	1U80t	1U81I
1U73D	1U76I		1U82I

Input / Output:

Command (Input)	*Output*
JCLEXIT	Information on the status of JCL exit routines or a modified status of them
AR commands	output from AR commands
MAP	Information on storage layout and usage
SYSDEF	Data space information (counters, limits) in supervisor table (DSPACE operand). Start, stop and quiesce additional CPU's (TD operand).
QUERY	Information on data space usage, permanent and temporary job control options, and information related to the VSE/ESA Turbo Dispatcher.
SETPFIX	modified PFIX limits (via SETLIMIT macro) and changed COMREG bits for permanent PFIX limits.
VDISK	Created/deleted virtual disk
LIBSERV	Control functions for the IBM 3494 Tape Library Dataserver.

Exit Normal: to $JOBCTLA entry CONTROL

Exit Error: to $JOBCTLA error routines

Register Use:

```
R 1: Work
R 2: Work
R 3: Work, SCANR2/3 result
R 4: Work
R 5: Work, SCANR2/3 result
R 6: 3rd base register
R 7: DSECT of root phase
R 8: 1st base register
R 9: 2nd base register
R10: Partition COMREG
R11: Work
R12: Work
R13: Work
R14: Work
R15: Work
```

Sequence of Operation:

JOBCTLO: External entry point, establish addressability

JCLEXIT Processing:

JCLEXIT: Syntax check of JCLEXIT command

VALPHAS: Enable/disable all exit routines or a single one ------->CONTROL (root phase)

DISPLAY: Display the state of the exit routines ------->CONTROL (root phase)

AR command processing:

ATTENTIO: Check that the AR command is entered in BG partition during ASI processing time; if not ------->NVSERR (Invalid statement)

ATTLOOP: Resolve all symbolic parameters and copy command to buffer.

ATTPROC: Pass command to AR routine via SVC 30.

MAP command processing:

MAP: Set up interface area to SVA routine ($IJBMAP) and load $IJBMAP.

CALLSVA: Switch to the addressing mode of the SVA routine and branch to it.

On return, check the return code in register 15:

- RC = 0 generate statement if required ------->CONTROL2
- RC = 0 else ------->CONTROL
- RC = 1 issue message as passed in buffer, insert operand number ------->ERRRTN
- RC = 2 issue message as passed in buffer, don't insert operand number ------->ERRRTN0
- RC = 4 ------->NVSERR (Invalid statement)
- RC = 8 issue msg 1S40 (SYSTEM ERROR, GETVIS RET.CODE=0C) ------->ERRRTN0
- RC = 16 issue message as passed in buffer, don't insert operand number, indicate message is 120 bytes long ------->ERRRTN0

SYSDEF,QUERY,VDISK processing:

SYSDEF: Set up interface area to SVA routine ($IJBSDSP) and load $IJBSDSP
-------->CALLSVA

QUERY: Set up interface area to SVA routine ($IJBSDSP) and load $IJBSDSP
-------->CALLSVA

VDISK: Set up interface area to SVA routine ($IJBVDII) and load $IJBVDII
-------->CALLSVA

LIBSERV: Set up interface area to SVA routine ($IJBSLIB) and load $IJBSLIB
-------->CALLSVA

PRTY: Set up interface area to SVA routine ($IJBPRTY) and load $IJBPRTY
-------->CALLSVA

VTAPE: Set up interface area to SVA routine ($IJBVTAP) and load $IJBVTAP
-------->CALLSVA

SETPFIX processing:

SETPFIX: Syntax check of SETPFIX statement. Call PFXCNV to convert input value(s) into number of pages and store them into SPFXPL (Parameter list of SETLIMIT macro).

PFXISS: If MODE=VM|VMESA ignore complete statement; if MODE=370 ignore the PFIX(ABOVE) limits.

Use SETLIMIT macro to pass request to supervisor.

- RC = 0 -------->PFXALT
- RC = 12 -------->PFXM1U81
- RC = 16 issue msg 1U82 -------->ERRRTN0
- other RCs issue msg 1S40 -------->ERRRTN0

PFXALT: For permanent SETPFIX requests alter flags IJBPFXAP/IJBPFXBP in COMREG.JCSW9.
-------->CONTROL (root phase)

PFXCNV: This subroutine

- checks for correct 'K' or 'M' specification
- checks that the 'K' or 'M' values are not negative
- checks that 127 MB or 131068 KB are not exceeded (131068 KB is the highest value accepted by SETLIMIT)
- converts Mbyte to Kbyte
- rounds Kbytes to a multiple of PAGE-SIZE
- converts Kbytes into number of pages
- stores number of pages into SETLIMIT parameter list

PFXM1U81: For RC=12 form SETLIMIT macro, translate error flags in SPFXPL into message(s) 1U81/1U80. -------->ERRRTN0

Phase $IJBASGN

Module Name: IJBASGN

Entry Point: IJBASGN - called via the ASSIGN macro

Function: Dynamic device assign/unassign/assignment change:

- Assigns a specified disk cuu to any free programmer or system logical unit.
- Assigns any unassigned tape device to any free programmer logical unit.
- Unassigns an internal system or programmer logical unit that was assigned by function 1 or 2 above.
- Changes the temporary assignment of a unit which was assigned by function 1 or 2 above to permanent.
- Assigns a specified tape unit (CUU) or another device (non-type or non-disk) temporarily to any free programmer logical unit.
- Assigns a specified tape unit (CUU) temporarily to a specified free programmer logical unit.

(For details see “Sequence of Operation” below.)

Called By:

- $JOBCTLH
- $IJBVDII
- $IJBSLA

Phases Called: $IJBSSYS to validate tape mode

Assembler Macros Used:

- ASYSCOM
- COMRG
- EXTENT
- FREEVIS
- GENDTL
- GETFLD
- GETVCE
- GETVIS
- IPW$DDE
- IPW$DPD
- LOAD
- LOCK
- MAPPUB
- MAPPUBX
- MODFLD
- MSAT
- UNLOCK

Data Areas -- Job Control Used:

```
ASPL     -- Input parameter list in user area (addressed by
            register ASPLREG)
COMDST   -- Partition communication region DSECT (macro MAPCOMR)
PATCHASG -- A 128-byte patch area
SYSCOM   -- System communication region DSECT
```

In supervisor:

```
LUBTAB   -- LUB table
PUBTAB   -- PUB table
PIB      -- Program information block
```

Resource Control: Access to the LUB, PUB, and PUBOWN tables by other partitions or tasks is controlled by the supervisor macros LOCK and UNLOCK. Resource name is CIOBLOCKS.

Messages: None

Input: Parameter list specifying the function to be performed. **Output:**

- For disk, logical unit in bytes 1 and 2 of PARMLIST.
- For tape, physical unit in bytes 3 and 4, and logical unit in bytes 1 and 2 of PARMLIST.

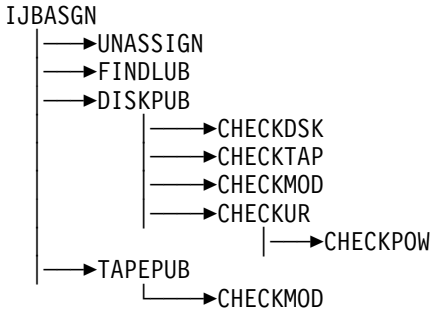
Exit Normal: Return to caller

Exit Error: None

Register Use:

```
R 3: Pointer to input parameter list (ASPL)
R 7: Base register
R 8: Pointer to automatic work space
R11: Pointer to COMDST
R12: Pointer to SYSCOM
```

Control Flow within Phase $IJBASGN



Sequence of Operation:

IJBASGN: Establishes addressability.

INIT: Obtains partition GETVIS space.

GETVISOK: Establishes addressability for supervisor control blocks.

Gets key 0 if necessary.

Evaluates input parameter list.

Enqueues LUB, PUB, etc. via the LOCK macro for resource name CIOBLOCKS.

To change or unassign a device, checks for the correct logical unit. ------->UNASSIGN

FINDLUB: Searches the LUB table for free LUB entry.

DISKPUB: For disk assignment, searches the PUB table for the specified unit.

TAPEPUB: For tape assignment, searches the PUB table for a free tape unit or for the specified unit.

CHECKUR: For unit record assignment, searches the PUB table for the specified unit.

ASSIGN: Completes assignment by updating LUB entry and PUBOWN entry. Fills in input parameter list.

EXIT: Releases I/O blocks via UNLOCK macro. Releases GETVIS space. Sets return code in register 15. ------->Return

UNASSIGN: To unassign a device, releases extents via the EXTENT macro (see *VSE/Advanced Functions Diagnosis Reference: Supervisor).* If the assignment was temporary, resets to stored permanent assignment. Resets the PUBOWN bit unless the device is assigned to another logical unit. ------->Return

To change an assignment, restores LUB entry to make the assignment permanent.

Phase $IJBCJC

Module Name: IJBCJC

Entry Point: IJBCJC

Function: Stores and retrieves conditional job control information such as last return code of a job step and maximum return code of a job. Saves ON-statements and retrieves applicable ON-conditions.

ON-related information is stored in a part of the job control work area, where COMREG.IJBJCWA points to. This area starts with eyecatcher PARCJCCB and provides enough space to store 10 ON-conditions. If required, $IJBCJC acquires space GETVIS storage to store additional ON-conditions. The subpool name is IJBCJC.

Called By: Called via CONDJC macro by other JC phases.

Phase Called: None.

Data Areas Used:

```
MAPCOMR -- Partition Communication Region
```

Messages: None.

Input:

```
Register 0  contains the function code of CONDJC.
Register 1  contains the address of the parameter field of CONDJC.
Register 14 contains the return code.
```

Output:

```
Parameter field of macro CONDJC,
Register 15 contains the return code.
```

Exit Normal: Return to caller.

Exit Error: None.

Register Use:

```
Register  7: 1st base register
Register  9: 2nd base register
Register 12: 3rd base register
Register 13: points to partition work area.
```

Sequence of Operation:

IJBCJC: Set Register 13 to partition work area. Branch according of function code. Return to caller.

GETLRC: Get last return code.

GETMRC: Get maximum return code.

RESETMRC: Reset maximum return code to zero.

RESETRC: Reset last return code to zero.

PUTRC: Save return code.

PUTONS: Put ON-statement into ON-list.

GETONA: Get appropriate ON-action from ON-list.

ENDCJC: Clear all conditional JC information.

ENDLEV: Indicate end of current level in ON-list.

GOTOEJ: Indicate GOTO $EOJ happened.

ENDPRC: Clear all ON-statements in ON-list of just ended procedure.

INCRLPTR: Set listpointer to next entry of ON-list.

DECRLPTR: Decrease listpointer by one entry of ON-list.

CHECKAB: Check for ON $ABEND-condition.

CHECKCA: Check for ON $CANCEL - condition.

CHECKRC: Check for ON $RC - condition.

CHECKAND: Check for ON - conditions connected by AND.

Phase $IJBJCCN

Module Name: IJBJCLCN

Entry Points:

IJBJCLCN If called by $JOBCTLE or $JOBCTLB

RETURN If called by EOJ macro

Function: If called by $JOBCTLE:

- Clears the partition.
- Puts the PARM value of the EXEC statement into the job control work area, where COMREG.IJBJCWA points to.
- Provides an 18 fullword save area in the job control work area, where COMREG.IJBJCWA points to.
- switches to the addressing mode of the user program.
- Passes control to user program

If called by $JOBCTLB:

- Passes control to phase $$BRSTRT.

(For details see “Sequence of Operation” below.)
If called after return from user program:

- Switch to 24-bit addressing mode.
- Save return code and issue DUMP macro if DUMP macro with return code was issued by user program
- Save return code and issue EOJ macro otherwise

Called By:

- $JOBCTLE, to process EXEC
- $JOBCTLB, to prepare job restart
- EOJ macro, when RC operand has been specified.

Phases Called:

- Program specified in the EXEC statement
- $$BRSTRT, transient to restore the problem program

Data Areas -- Job Control Used:

- COMDST -- Partition communication region DSECT (macro MAPCOMR)
- Save area and PARM buffer in job control work area (COMREG.IJBJCWA)
- SYSCOM -- System Communication Region
- PJBADR -- DSECT for job information control block (macro MAPPOWJB).

Messages: None

Input: PARM value from EXEC statement

Output: See “Interface from and to Phase $IJBJCCN” on page 256.

Exit Normal: to program specified in EXEC

Exit Error: None

Register Use: R 8: Base register

Sequence of Operation:

IJBJCLCN: Establishes addressability.

BEG1 Gives control to the user program by means of SYSSERV macro.

BEGINN: Invalidates all pages of the partition by means of INVPART macro.

PARM0: If called by $JOBCTLB: Loads $$BRSTRT. ------->$$BRSTRT

SETUP: Sets up registers, sets the storage key of the partition, resets the job control active bit, sets the user mode bit, sets time of day (PJBJTIME) in the job information control block (PJB) (see VSE/ESA Supervisor Diagnosis Reference) and loads the user program specified in the EXEC statement.

SETUP3: Switch back to 24-bit addressing mode. Save return code given back in register 15 by user program in SVA via CONDJC macro and issue EOJ macro if no DUMP macro.
Save return code and issue DUMP macro if user program issued DUMP macro with return code.

Phase $IJBMAP

Module Name: IJBMAP

Entry Point: IJBMAP

Function: MAP command processor.

Called By: Phases $JOBCTLO and $IJBATTN

Phases Called: None.

Data Areas Used:

```
MAPCOMR   -- Partition Communication Region
SYSCOM    -- System Communication Region
MAPPIB    -- Program Information Block
SLADCT    -- SLA DSECT to access supervisor name
SIDSTR    -- Table of valid space IDs
JCARAREA  -- Interface area to calling module (macro JCLARIF)
MAPCLIM   -- Mapping of Class and System limits
SGLOWC    -- Storage Layout of Supervisor Low Core
SMCOM     -- Storage Management Communication Area
PMCOM     -- Page Management Communication Area
MAPPCB    -- Partition Control Block
MAPJACCT  -- Job Accounting Control Block
SYSDEF    -- DSECT used with SYSDEF macro
MAPSCB    -- Space control block
MAPEXTR   -- DSECTs for EXTRACT services
```

Messages Caused:

1Y02I
1Y2nt
1Y8nt

Input:

- MAP command with operands.
- Register 1 contains pointer to interface area (JCARAREA). For a description of this interface area refer to “JCARAREA - Interface Area Between JCL/AR and SVA Routines” on page 251.

Output: Messages in output buffer which describe the storage layout.

Exit Normal: Return to caller.

Exit Error:

```
- Register 15 contains 1 if an error msg has been placed in the I/O buffer
  and the caller has to show this msg with the operand number inserted.
- Register 15 contains 2 if an error msg has been placed in the I/O
  buffer and the caller has to show this msg unchanged.
- Register 15 contains 4 if input was incorrect.
- Register 15 contains 8 if MAP is not possible because no
  System GETVIS space is available.
```

Register Usage:

```
R1  : Pointer to interface area.
R8  : Base register for workarea.
R10 : 3rd base register for code.
R11 : 2nd base register for code.
R12 : 1st base register for code.
R13 : Address of save area.
R14 to R0 are destroyed.
R15 to Return code.
```

Control Flow within Phase $IJBMAP: Internal subroutine call structure for fast orientation:

```
$IJBMAP
   |
   |--> SCANX                      /* scan operand(s) of MAP command       */
   |
   |--> MAPVIRT                    /* show virtual storage information     */
   |       |--> MAPHDRV
   |       |--> CONVERT
   |       |--> BUFFOUT
   |       |--> MAPSVA             /* show SVA-24 and SVA-31               */
   |       |       |--> MAPHDRV
   |       |       |--> BUFFOUT
   |       |       |--> CONVERT
   |       |       |--> TRANSLAT
   |       |       |--> MAPHDRR1
   |       |       |--> MAPHDRR2
   |       |
   |       |--> MAPALLST           /* show all static partitions           */
   |       |       |--> MAPONEST   /* show a single static partition       */
   |       |       |       |--> CONVERT
   |       |       |       |--> TRANSLAT
   |       |       |--> BUFFOUT
   |       |       |--> CONVERT
   |       |
   |       |--> MAPSUMDP           /* show summary on dynamic partitions   */
   |               |--> CONVERT
   |               |--> BUFFOUT
   |
   |--> MAPREAL                    /* show real storage information        */
   |       |--> MAPHDRR1
   |       |--> MAPHDRR2
   |       |--> BUFFOUT
   |       |--> CONVERT
   |       |--> MAPSVA             /* show SYS-24 and SYS-31               */
   |       |       |--> MAPHDRV
   |       |       |--> BUFFOUT
   |       |       |--> CONVERT
   |       |       |--> TRANSLAT
   |       |       |--> MAPHDRR1
   |       |       |--> MAPHDRR2
   |       |
   |       |--> MAPALLST           /* show all static partitions           */
   |       |       |--> MAPONEST   /* show a single static partition       */
   |       |       |       |--> CONVERT
   |       |       |       |--> TRANSLAT
   |       |       |--> BUFFOUT
   |       |       |--> CONVERT
   |       |
   |       |--> MAPSUMDP           /* show summary on dynamic partitions   */
   |               |--> CONVERT
   |               |--> BUFFOUT
   |
   |
```

```
$IJBMAP (continued)
  |
  |--> MAPREALV                    /* show real storage information under VM*/
  |      |--> BUFFOUT
  |      |--> TRANSLAT
  |      |--> CONVERT
  |
  |--> MAPSVA                      /* show SVA-24, SVA-31, SYS-24, SYS-31   */
  |      |--> MAPHDRV
  |      |--> BUFFOUT
  |      |--> CONVERT
  |      |--> TRANSLAT
  |      |--> MAPHDRR1
  |      |--> MAPHDRR2
  |
  |--> MAPCLASS                    /* show information on a or all class(es)*/
  |      |--> BUFFOUT
  |      |--> CONVERT
  |      |--> TRANSLAT
  |
  |--> MAPPART                     /* show information on a single partition*/
         |--> CONVERT
         |--> TRANSLAT
         |--> BUFFOUT
```

Sequence of Operation:

Main routine:

- Perform GETVIS for workarea space.

  If RC > 0, set Reg15 = 8 and return to caller.

- Get system related storage information:

  ```
  MAPEXTR ID(BDY) MODE(PERM)
  MAPEXTR ID(BDY) MODE(SYSP)
  ```

- Analyze input and call appropriate subroutine:

Command	Action
MAP or MAP VIRTUAL	Call MAPVIRT
MAP REAL	If MODE=VM|VMESA, call MAPREALV; if MODE=370|ESA, call MAPREAL
MAP CLASS=ALL|c	Call MAPCLASS
MAP SVA	Call MAPSVA
MAP xx	Call MAPPART

- If an error occurred, set up a return code in register 15, store (if applicable) an error msg into the output buffer, do FREEVIS and return to caller.

Phase $IJBPROC

Module Name: IJBPROC

Entry Point: IJBPROC

Function: Supports all functions of the macros GETSYMB, PARMMAC and PROCMAC. The GETSYMB macro allows user programs and job control exit routines to substitute symbolic parameters. The PARMMAC-functions store and retrieve symbolic parameters, create new parameter tables, and define default values for parameters. The PROCMAC-functions get ACCESS to a required cataloged procedure, get next sequential record of a cataloged procedure, handle end-of-procedure requests and cancel - or end-of-job (/&) requests.

Called By: PARMMAC - and PROCMAC - macro calls.

Phases Called: $IJBLBR for librarian services.

Data Areas Used:

```
COMREG    -- Partition Communication Region, based structure,
             macro MAPCOMR
INLCLAMB  -- Library Access Method Block, based structure,
             macro INLMLAMB MF=MAP
INLCLARG  -- Find-Argument, based structure, macro INLMLAMB MF=MAP
INLCLRPL  -- Request Parameter List, based structure,
             macro INLMLAMB MF=MAP
INLCNTPT  -- Note/Point field, based structure, macro INLMLAMB
             MF=MAP
PARHDR    -- Header of central control block PARLIBCB, based structure
PARDSECT  -- Description of central control block PARLIBCB for
             parameterized procedure, based structure PARDSECT
             contains one entry per partition. Each partition entry
             contains as substructure a table descriptor TABDESC
             with 80 byte length containing:
             -  pointer to level 0 header PHDR0
             -  pointer to active header of current parameter table
                PHDRA
             -  pointer to previous (level n-1) header PHDRB
             -  pointer to end of block PEOBC = end of used system
                GETVIS space
TABHDR    -- Description of header of parameter table, based structure
PARENTRY  -- Description of parameter entry in parameter table, based
             structure
PARMAR    -- Description of parameter list, used as input for
             PARMMAC-call, based structure
PROCAR    -- Description of parameter list, used as input for
             PROCMAC-call, based structure
TABLBUF   -- Description of buffer passed for the PARMMAC-requests:
             SETPAR, SETPARS, SETPARP, SETPEX, SETPDF. Based structure
SYSCOM    -- System Communication Region, based structure, macro
             SYSCOM
```

Messages Caused: None.

Input: Contents of parameter list for PARMMAC and PROCMAC calls plus buffer contents.

Output:

PARMMAC-function Output

GETVAL	parameter value
SETPAR	parameter table update for symbolic parameters at nesting level n
SETPARS	parameter table update for symbolic parameters at system level
SETPARP	parameter table update for symbolic parameters at POWER job level
RELPARP	release parameter table for symbolic parameters at POWER job level (during processing of * $$ EOJ)
SETPEX	creation of new parameter table on new nesting level
SETPDF	update of current parameter table with default parameter values

PROCMAC-function Output

ACCESS	open status of cataloged procedure
GETREC	next sequential record of cataloged procedure
EOPREQ	deletion of currently active parameter table
CANCRQ	disconnected status of level 1 procedure, total cleanup of all parameter tables in the partition
INCORE	sets low, current and high address of incore procedure that should be executed instead of the current procedure. This is required to have JCL process the statements left on the REXX data stack (see label HREXXCNT in $JOBCTLA).
INCOREX	sets low, current and high address of incore procedure that should be executed instead of the current procedure. This is required to support the address JCL command environment provided by REXX (see label ADRJCLRX in $JOBCTLA).
QUERYI	returns the addresses of remaining statements in the in-core procedure to the caller (REXX). Thus the in-core procedure can be overwritten by another one and later be restored when it is needed again
QUERYP	returns the name of a n-1 level procedure. Thus a REXX program can determine from which procedure it was called (if any).

Exit Normal: To calling phase with return code zero

Exit Error: To calling phase with return code non-zero

Register Usage:

```
R1  : Pointer parameter list of PARMMAC or PROCMAC call
R6  : 3. Base Register
R7  : 1. Base Register
R9  : 2. Base Register
R11 : Base for partition COMREG
R12 : Base for SYSCOM
R13 : Pointer to register save area
R14 : Return Register
R15 : Return Codes
```

Control Flow within Phase $IJBPROC

```
$IJBPROC
   |-->INITPROC
   |
   |-->(PARMMAC-call)
   |         |
   |         |-->(GETVAL-request)
   |         |       |-->PARGET
   |         |               |-->TABSRCH
   |         |               |-->LOCK_GLOBAL_VAR
   |         |               |-->UNLOCK_GLOBAL_VAR
   |         |
   |         |-->(SETPAR-request)
   |         |       |-->PARPAR
   |         |               |-->UPDPTLST
   |         |               |-->TABSRCH
   |         |               |-->CHKSPACE
   |         |                       |-->UPDPTLST
   |         |
   |         |-->(SETPARS or SETPARP-request)
   |         |       |-->LOCK_GLOBAL_VAR (SETPARS only)
   |         |       |-->SET_GLOBAL_VAR
   |         |       |       |-->SPACE_GETVIS
   |         |       |       |-->TABSRCH
   |         |       |-->UNLOCK_GLOBAL_VAR (SETPARS only)
   |         |
   |         |-->(RELPARP-request)
   |         |       |-->RELEASE_GLOBAL_VAR
   |         |
   |         |-->(SETPEX-request)
   |         |       |-->PARPEX
   |         |               |-->UPDPTLST
   |         |               |-->TABSRCH
   |         |               |-->CHKSPACE
   |         |                       |-->UPDPTLST
   |         |
   |         |-->(SETPDF-request)
   |         |       |-->PARPDF
   |         |               |-->TABSRCH
   |         |               |-->CHKSPACE
   |         |                       |--UPDPTLST
   |
   |-->(PROCMAC-call)
             |
             |-->(ACCESS-request)
             |       |-->ACCPROC
             |               |-->FINDTAB
             |               |-->DUPNMCHK
             |               |-->(INLMFIND-macro)---$IJBLBR
             |               |-->(INLMNOTE-macro)---$IJBLBR
             |
             |-->(GETREC-request)
             |       |-->GETPREC
             |               |-->FINDTAB
             |               |-->(INLMGETR-macro)---$IJBLBR
             |
             |-->(CANCRQ or EOPREQ-request)
             |       |-->EOPROUT
             |               |-->FINDTAB
             |               |-->(INLMLDIS-macro)---$IJBLBR
             |               |-->(INLMPOIN-macro)---$IJBLBR
             |
             |-->(INCORE or INCOREX-request)
             |       |-->INCOREP
             |
             |-->(QUERYI-request)
             |       |-->QUERYRT
             |
             |-->(QUERYP-request)
             |       |-->QUERYNAM
```

Sequence of Operation:

INIT: Establish addressability for compiler generated data. Get storage key 0. Lock against subtasks within the calling partition. The LOCK resource name used is DLBPRO concatenated with the partition identifier (PIK). Save function code and current PIK. Check validity of function code.

INITPROC: If pointer JCWPRPT in the job control work area (where COMREG.IJBJCWA points to) is zero, procedure INITPROC is called. INITPROC issues system GETVIS request for control block PARLIBCB. The subpool name is IJBPRC. PARLIBCB is then initialized.

GETVAL Make control block PARLIBCB addressable via FINDTAB. Get pointer to active parameter table from PARLIBCB. Search parameter table sequentially for requested parameter.

SETPAR Find correct entry in PARLIBCB. Create parameter table on level 0 if first request. Search through active parameter table. If parameter found and length of character string is equal in old and new value, do update in place, otherwise create new entry in table. This process is continued until all parameters defined in SETPARM-statement are handled. Check if space sufficient in table before creating any new entry, by CHKSPACE procedure.

SETPARS or SETPARP Similar to SETPAR, stores symbolic parameters at system or POWER job level. For SETPARS requests lock resource name $IJBSETPARM is used to serialize concurrent SETPARM SYSTEM commands.

RELPARP Releases GETVIS space and pointer chain associated with symbolic parameters at POWER job level, see Figure 18 on page 210.

SETPEX Find correct entry in PARLIBCB. Create parameter table on level 0 if first request. Build new table header for new parameter table. Fill new parameter table with parameters defined in EXEC PROC statement. Check if space sufficient in table before creating any new entry, by CHKSPACE procedure.

SETPDF Find correct entry in PARLIBCB. Get pointer to active parameter table from PARLIBCB. Handle default definitions in PROC statement by adding parameter entries in parameter table. Check if space in table sufficient before creating any new entry, by CHKSPACE procedure.

ACCPROC: Handles the PROCMAC-request ACCESS. Find the correct entry in PARLIBCB. Check for duplicate name in nested stack. Issue Find-request via macro IMLMFIND. Issue Note-request via macro INLMNOTE.

GETPREC: Handles the PROCMAC-request GETREC. Find the correct entry in PARLIBCB. Issue GET-request for next sequential record of cataloged procedure via macro INLMGETR.

EOPROUT: Handles the PROCMAC-requests EOPREQ and CANCRQ.

- EOPREQ Find correct entry in PARLIBCB. Delete current parameter table. Reset Job Control flags. Issue Member-Disconnect (macro INLMLDIS) in case of end-of-procedure request on nesting level 1, issue point (macro INLMPOIN) on nesting level * 2.
- CANCRQ Find correct entry in PARLIBCB. Delete all parameter tables except level 0 for the requesting partition. Make level 0 parameter table empty. Reset Job Control flags. Issue Member-Disconnect (macro INLMLDIS) for level 1 procedure.

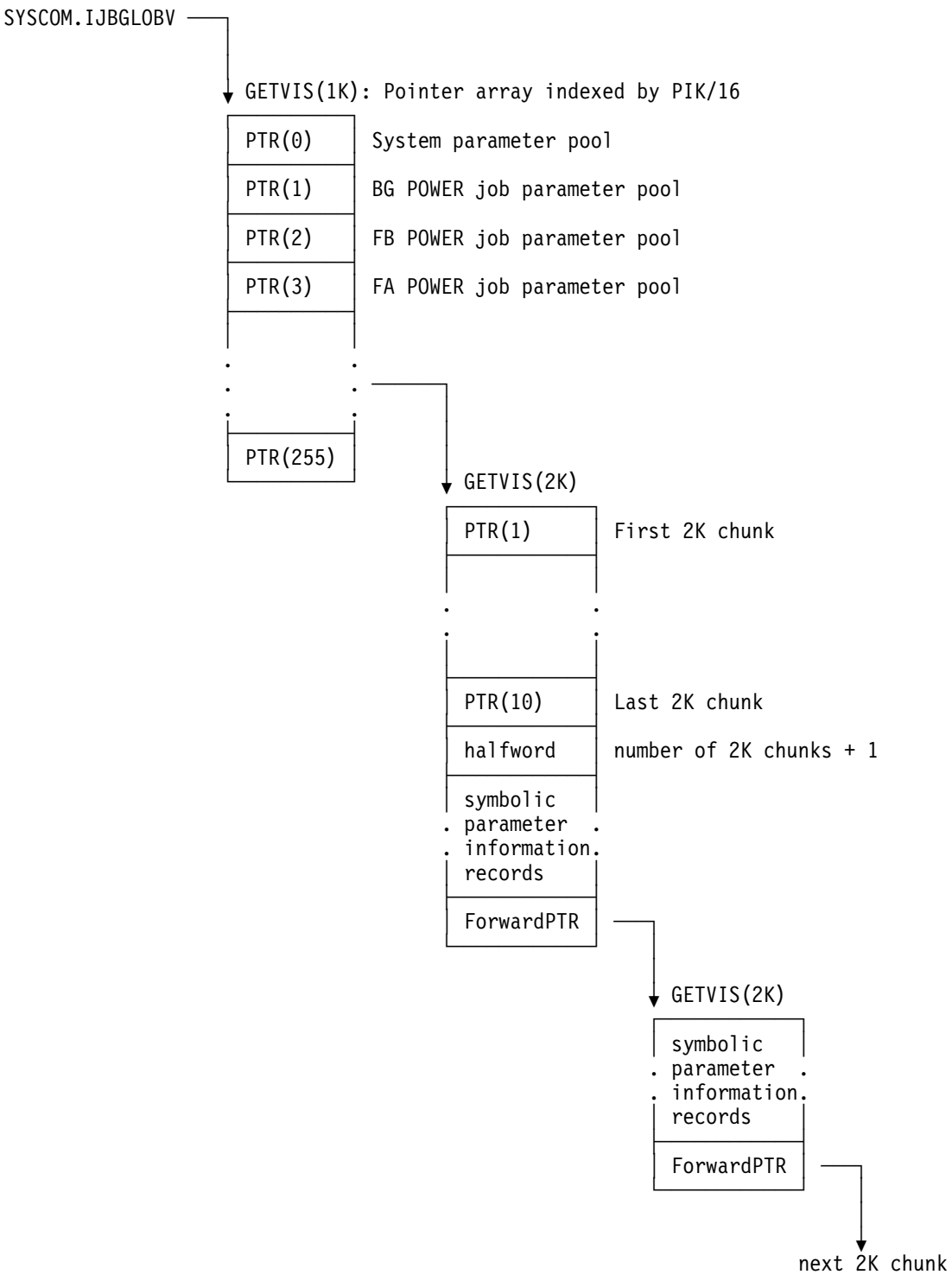


Figure 18. Pointer Array for Parameters at System or POWER Job Level

Phase $IJBPRTY

Module Name: IJBPRTY

Entry Point: IJBPRTY

Function: Processing of PRTY and TPBAL command. Reentrant SVA phase.

Called By: Phases $JOBCTLO and $IJBATTN.

Phases Called: None.

Data Areas Used:

CLIMADR	Dynamic class and system limits table, needed to access CMPBNO, the maximal possible number of balanced groups
COMREG	Partition Communication Region
JCARAREA	Interface area to $JOBCTLO and $IJBATTN
SYSCOM	System Communication Region

Messages Caused:

1P76I	1S75I	1Y5nt	1Y8nt
1P77I	1Y2nt	1Y6nt	1YK1t
1S40t	1Y4nt	1Y7nt	

Input:

- PRTY or TPBAL command with or without operands.
- Register 1 contains pointer to interface area (JCARAREA). For a description of this interface area refer to “JCARAREA - Interface Area Between JCL/AR and SVA Routines” on page 251.

Output: Messages in output buffer which contain

- Information related to the priority sequence - obtained via GETPRTY macro.
- Share values for the partitions/classes belonging to a balanced group (only if the VSE/ESA Turbo Dispatcher is active) - obtained via TDSERV macro.
- Partitions delayed by teleprocessing balancing (only if the TPBAL command has been issued before) - obtained via IJBTPBAL in SYSCOM.

Exit Normal: Return to caller.

Exit Error:

- Register 15 contains 1 if an error message has been placed in the I/O buffer and the caller has to show this message with the operand number inserted (JCARRTN).
- Register 15 contains 2 if an error message has been placed in the I/O buffer and the caller has to show this message unchanged (JCARRTN0).
- Register 15 contains 8 if PRTY/TPBAL is not possible because no partition GETVIS space is available.

Register Usage:

R1	Pointer to interface area.
R3	Restricted work register
R8	Base register for workarea.
R10	1st base register for code.

R11 2nd base register for code.
R13 Address of save area.
R15 Return code.

Phase $IJBSDSP

Module name: IJBSDSP

Entry point: IJBSDSP

Function: Processing of SYSDEF and QUERY commands

Called by: Phases $JOBCTLO and $IJBATTN

Messages caused:

1S40t	1Y4nt	1YAnt	1YK2I
1UV7t	1Y5nt	1YH5t	1YK3t
1Y01t	1Y6nt	1YH6I	1YK4t
1Y1nt	1Y7nt	1YH7I	1YK6t
1Y2nt	1Y8nt	1YK1t	1YK7I
1Y3nt			

Input:

- SYSDEF/QUERY commands with operands.
- Register 1 contains pointer to interface area (JCARAREA). For a description of this interface area refer to “JCARAREA - Interface Area Between JCL/AR and SVA Routines” on page 251.

Output: Messages in output buffer which contain information related to data spaces, to standard or temporary options and to the VSE/ESA Turbo Dispatcher.

Exit Normal: Return to caller.

Exit Error:

- Register 15 contains 1 if an error message has been placed in the I/O buffer and the caller has to show this message with the operand number inserted.
- Register 15 contains 2 if an error message has been placed in the I/O buffer and the caller has to show this message unchanged.
- Register 15 contains 4 if input was incorrect.
- Register 15 contains 8 if SYSDEF/QUERY is not possible because no system GETVIS space is available.

Register Usage:

R1 Pointer to interface area.
R3 Restricted work register
R8 Base register for workarea.
R10 3rd base register for code.
R11 2nd base register for code.
R12 1st base register for code.
R13 Address of save area.
R15 Return code.

Phase $IJBSLIB

Module Name: IJBSLIB

Entry Point: IJBSLIB

Function: Processing of LIBSERV command, which provides control functions for the IBM 3494 Tape Library Dataserver.

Called By: Phases $JOBCTLO and $IJBATTN

Messages Caused:

1A5nt	1Y5nt	1YBnt	1YH2I
1S40t	1Y6nt	1YCnt	1YH3t
1Y05t	1Y7nt	1YEnI	1YH4I
1Y09t	1Y8nt	1YFnI	1YH8t
1Y1nt	1Y9nt	1YH0I	1YK0t
1Y2nt	1YAnt	1YH1I	1YK5t
1Y3nt			

Input:

- LIBSERV command with operands.
- Register 1 contains pointer to interface area (JCARAREA). For a description of this interface area refer to “JCARAREA - Interface Area Between JCL/AR and SVA Routines” on page 251.

Output: A tape device attached to or removed from the VSE system.

Exit Normal: Return to caller.

Exit Error:

- Register 15 contains 1 if an error message has been placed in the I/O buffer and the caller has to show this message with the operand number inserted.
- Register 15 contains 2 if an error message has been placed in the I/O buffer and the caller has to show this message unchanged.
- Register 15 contains 4 if input was incorrect.
- Register 15 contains 8 if LIBSERV is not possible because no system GETVIS space is available.

Register Usage:

R1 Pointer to interface area.
R3 Restricted work register
R8 Base register for workarea.
R10 1st base register for code
R11 2nd base register for code.
R12 3rd base register for code.
R13 Address of save area.
R15 Return code.

Phase $IJBVDII

Module Name: IJBVDII

Entry Point: IJBVDII

Function: Processing of VDISK Job Control command

Called By: $JOBCTLO

Messages Caused:

1A5nt	1UV4t	1UV9D	1Y3nt
1S40t	1UV5t	1Y05t	1Y4nt
1UV1t	1UV6t	1Y07t	1Y5nt
1UV2t	1UV7t	1Y08t	1Y6nt
1UV3t	1UV8t	1Y2nt	1Y7nt

Input:

- VDISK command with operands.
- Register 1 contains pointer to interface area (JCARAREA). For a description of this interface area refer to “JCARAREA - Interface Area Between JCL/AR and SVA Routines” on page 251.

Output: Error messages

Exit Normal: Return to caller.

Exit Error:

```
- Register 15 contains 1 if an error msg has been placed in the I/O buffer
  and the caller has to show this msg with the operand number inserted.
- Register 15 contains 2 if an error msg has been placed in the I/O
  buffer and the caller has to show this msg unchanged.
- Register 15 contains 4 if input was incorrect.
- Register 15 contains 8 if VDISK is not possible because no
  System GETVIS space is available.
```

Register Usage:

```
R1  : Pointer to interface area.
R3  : Restricted work register
R8  : Base register for workarea.
R11 : 2nd base register for code.
R12 : 1st base register for code.
R13 : Address of save area.
R15 : Return code.
```

Phase $IJBVTAP

Module Name: IJBVTAP

Entry Point: IJBVTAP

Function: Processing of VTAPE command, which enables (START) or disables (STOP) a tape cuu to act as virtual tape.

Called By: Phase $JOBCTLO

Phases Called: $IJBTAPE (Virtual Tape Simulator) and $IJBSLA (for SRCLBL)

Messages Caused:

1A5nt	1Y3nt	1YM1t	1YM8t
1S40t	1Y5nt	1YM2t	1YM9t
1S70t	1Y6nt	1YM3I	1YN1t
1U75t	1Y7nt	1YM4I	1YN2t
1UV4t	1YCnt	1YM5t	1YN3t
1Y05t	1YK8t	1YM6I	1YN4t
1Y2nt	1YK9t	1YM7t	

Input:

- VTAPE command with operands.
- Register 1 contains pointer to interface area (JCARAREA). For a description of this interface area refer to “JCARAREA - Interface Area Between JCL/AR and SVA Routines” on page 251.

Output: For START: virtual tape capabilities enabled for UNIT=cuu. For STOP: virtual tape capabilities disabled for UNIT=cuu.

Exit Normal: Return to caller.

Exit Error:

- Register 15 contains 1 if an error message has been placed in the I/O buffer and the caller has to show this message with the operand number inserted.
- Register 15 contains 2 if an error message has been placed in the I/O buffer and the caller has to show this message unchanged.
- Register 15 contains 4 if input was incorrect.
- Register 15 contains 8 if LIBSERV is not possible because no system GETVIS space is available.

Register Usage:

R1 Pointer to interface area.
R8 Base register for workarea.
R10 2nd base register for code.
R11 1st base register for code.
R13 Address of save area.
R15 Return code.

Sequence of Operation for VTAPE START:

- Check and validate keywords, operands and delimiters
- Special checking for VSAM file name (LABEL SRCLBL)
- Special checking for cuu via GETVCE

- Append DefineCuu element to request queue starting at SYSCOM.IJBSPAVT.ASUPAVTE.AVTAPANC.VTAWTODO. Request elements are mapped via DSECT TADMCBADR provided in macro MAPTACN.
- If Tape Data Handler is not active: release POWER job TAPESRVR
- Invoke $IJBTAPE to change UNIT=cuu status from physical to virtual
- Pass control to Tape Data Handler to process DefineCuu request.

  For LOC=VSAM the Tape Data Handler will open the VSAM ESDS file containing the tape image. For LOC=ipaddr the Tape Data Handler will establish a TCP/IP connection to the Virtual Tape Server and have the Virtual Tape Server open the tape image file.
- Remove DefineCuu element from request queue
- If DefineCuu request failed: invoke $IJBTAPE again to change UNIT=cuu status back to physical

Sequence of Operation for VTAPE STOP:

- Check and validate keywords, operands and delimiters
- Special checking for cuu via GETVCE
- Invoke $IJBTAPE to change UNIT=cuu status from virtual to physical
- If Tape Data Handler is not active: exit
- Append UndefineCuu element to request queue starting at SYSCOM.IJBSPAVT.ASUPAVTE.AVTAPANC.VTAWTODO. Request elements are mapped via DSECT TADMCBADR provided in macro MAPTACN.
- Pass control to Tape Data Handler to process UndefineCuu request.

  For LOC=VSAM the Tape Data Handler will close the VSAM ESDS file containing the tape image. For LOC=ipaddr the Tape Data Handler will have the Virtual Tape Server close the tape image file. Then the TCP/IP connection to the Virtual Tape Server is closed.
- Remove UndefineCuu element from request queue

Chapter 8. Job Control Macros

CONDJC Macro

A routine is provided to support Conditional Job Control. This routine performs the saving and retrieving of return codes, the saving of the ON-statements and the checking of the ON conditions. The routine is resident in the SVA (re-entrant). Modules that want to read or store any information about return codes and ON-statements must communicate with this routine via the CONDJC macro.

The CONDJC macro has the following format:

```
[name] CONDJC  CJCADDR= address | (register)
                        ┌        ┐
                        │ GETLRC │
                        │ RESETRC│
                        │ GETMRC │
                        │RESETMRC│
                ,FUNCT= │ PUTRC  │
                        │ PUTONS │
                        │ GETONA │
                        │ ENDCJC │
                        │ ENDPRC │
                        │ ENDLEV │
                        │GOTOEOJ │
                        └        ┘
               [,ENTRY]
```

The parameters have the following meaning:

CJCADDR: Specifies the address of a parameter field either directly or as a register contents.

FUNCT: See detailed description given below.

ENTRY: Specifies that the address of the loaded SVA-module is already in reg.15.

CONDJC Macro Functions

GETLRC: This function is used to fetch the last return code of a job step. After completion the last return code is stored in the parameter field with a length of four bytes,e.g. x'F0F0F0F8'.

If there is no last return code stored, a last return code of zero is put into the parameter field, and register 15 contains X'04' after completion.

RESETRC: This function is used by the SET RCZERO command to reset the last return code of a job step.

GETMRC: This function is used to fetch the maximum return code within a job. After completion the max. return code is stored in the parameter field with a length of four bytes,e.g. x'F0F0F1F6'.

If there is no max. return code stored, a max. return code of zero is put into the parameter field, and register 15 contains X'04' after completion.

RESETMRC: This function is used by the SET MRCZERO command to reset the maximum return code within a job.

PUTRC: This function is used to store the last return code of a job step, and to update the maximum return code if appropriate. The return code has to be in the parameter field with a length of four bytes,e.g. X'F4F0F0F0'.If the return code is greater than 4095, a return code of 4095 is stored.

PUTONS: This function is used to store the condition(s) and action given by an ON-statement.

The parameter field - pointed to by CJCADDR - has to contain in a length of 15 or 21 bytes:

Byte 0: The mode of the action: X'00' if CONTINUE, X'0F' if GOTO.

Bytes 1-8: The label (only if GOTO mode): e.g. X'E6C1D3E3C5D94040'.

Bytes 9-10: The 1st comparand: EQ or NE or GT or LT or GE or LE, or AB for $ABEND, or CA for $CANCEL.

Bytes 11-14: The 1st number in string format, e.g. X'F0F0F0F4'.

Byte 15: The operand: A if AND, O if OR, E if end of condition.

Only if 2nd condition available:

Bytes 16-17: The 2nd comparand,

Bytes 18-21: The 2nd number in string format.

GETONA: This function is needed to check the ON-conditions, and to get the action of the appropriate ON-statement if the condition(s) is/are true. Upon completion the action and the according run-mode are stored in the parameter field pointed to by CJCADDR:

Byte 0: Run-mode: X'00' if normal run-mode, X'0F' if GOTO-mode.

Bytes 1-8: The label (only if GOTO-mode).

Byte 9: The level (only if GOTO-mode).

If no appropriate ON-statement is found, the default value is set.

ENDCJC: This function is needed to reset:

the last return code, the maximum return code, all stored ON-statements.

ENDLEV: This function is used if an EXEC PROC occurs to indicate the end of the current level.

ENDPRC: This function is used if EOP occurs in order to clear all stored ON statements which have been specified in the procedure just ended.

GOTOEOJ: This function is used to indicate a GOTO $EOJ happened.

Note: The parameter field is not used by RESETRC, RESETMRC, ENDCJC,ENDLEV,ENDPRC, and GOTOEOJ.

Register Usage:

When the CONDJC macro is executed general registers 14 to 1 are destroyed. Register 13 must contain the address of a register save area (18 fullwords). After execution of the macro register 15 contains a return code:

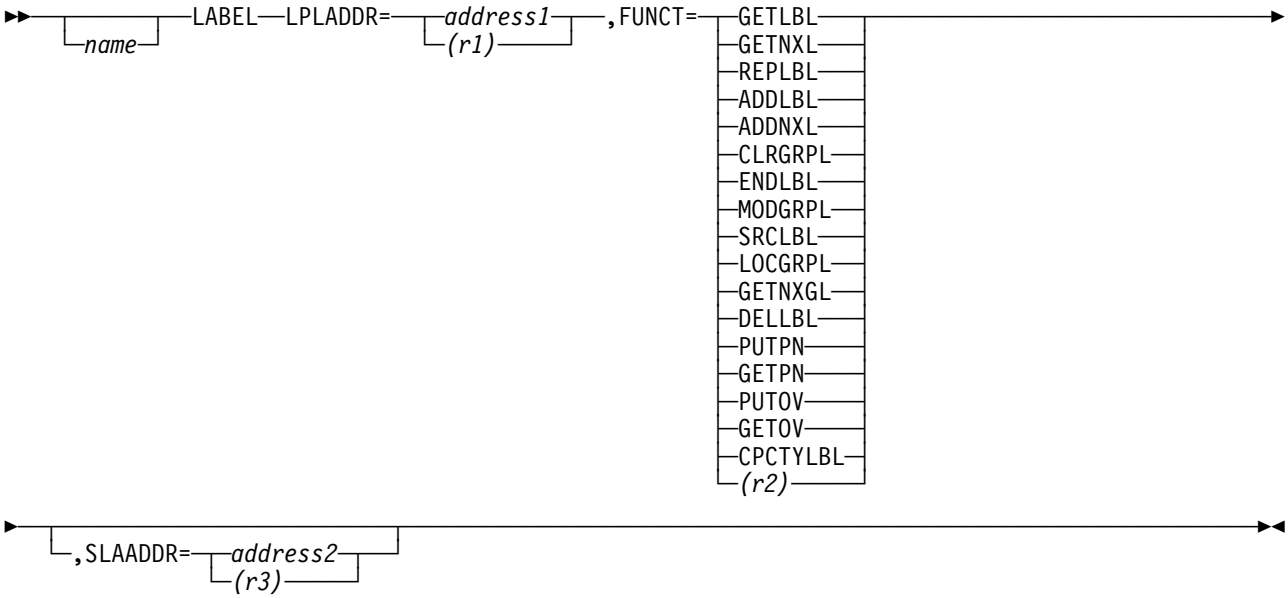
```
X'00': function successfully executed,
X'04': the wanted information is not available,
X'08': invalid parameter field, function not done,
X'0C': no GETVIS space available, function not done.
```

LABEL Macro

This macro provides the interface to the Symbolic Label Access (SLA) routine to read and/or update label information.

For detailed information on $IJBSLA and $IJBSLAD (the Symbolic Label Access routines) refer to *VSE/Advanced Functions Diagnosis Reference: Logical Transients and $IJBSxxx Phases*.

The LABEL macro has the following format:



LPLADDR Specifies the address of the Label Parameter List (LPL). The parameter LPLADDR can only be omitted if FUNCT=ENDLBL.

FUNCT Specifies the function to be performed. The parameter FUNCT is mandatory. Each function is described in detail later in this section.

```
1.GETLBL      Get a label.
  GETNXL      Get next label.
  REPLBL      Replace a label.
2.ADDLBL      Insert a label into a label subarea.
  ADDNXL      Insert additional label information into a label subarea.
  CLRGRPL     Reset a group of labels.
  ENDLBL      End of adding label information to a label area.
  MODGRPL     Reactivate a group of labels.
  SRCLBL      Find out whether a label already exists
              (permanent labels only).
  DELLBL      Delete label (from partition's free usage, temporary, or
              permanent label subarea).
3.LOCGRPL     Locate a group of labels.
  GETNXGL     Get next label from located group of labels.
4.PUTPN       Save phase name.
  GETPN       Retrieve phase name.
  PUTOV       Save overwrite statement.
  GETOV       Retrieve overwrite statement.
  CPCTYLBL    Return capacity information into user supplied buffer.
```

SLAADDR The parameter SLAADDR is optional and specifies the start address of SVA phase $IJBSLA. The SLAADDR keyword operand may be a performance benefit for (vendor) programs with the need to execute the LABEL macro code often and/or repeatedly.

Note: The first group of functions is designed to be used by OPEN, the second group is for job control and the third group is designed to be used by LSERV and the last group is used by several components (for example Utilities).

Return Codes from $IJBSLA (Overview)

Return code	What does it mean?
X'00'	Successful request
X'04'	Label does not exist
X'08'	Buffer length too small
X'0A'	Only used by ICCF: wrong length
X'0B'	Only used by ICCF: CDLOAD not OK
X'0C'	Wrong sequence of macros
X'0D'	Dyn.Assign did not work
X'0E'	GETVCE did not work
X'14'	Contents of LPL invalid
X'18'	No space in label area
X'1C'	No GETVIS space available
X'20'	Label subarea in progress
X'24'	Wrong extent sequence number
X'28'	Same label found in subarea
X'63'	Label function code is invalid

LOCK Requests issued by $IJBSLA

1. Right at the beginning $IJBSLA locks against subtasks within the calling partition. The LOCK resource name used is DLBSLA concatenated with the partition identifier (PIK). So, as far as different tasks in one and the same partition are requesting label services, there can be only one task at a given time accessing the label area via $IJBSLA.
2. When updating system standard labels $IJBSLA uses LOCK resource name DLBSLASL.
3. In all other cases, especially when modifying the LAS master bit table, $IJBSLA uses LOCK resource name DLBSLAQU.

LABEL Macro Functional Description:

GETLBL Function:

This function is used to fetch a label from a label subarea. The label is returned in the buffer specified in the LPL macro, and it is either a label of the free usage subarea, the partition's temporary label subarea, the partition's permanent label subarea, a class standard label subarea (if called from a dynamic partition) or the system label area, wherever the label is found first.

LPL parameters required:
AREA, AREALEN, and FILENAM.

After completion register 15 contains one of the following return codes:

```
X'00'  Successful request. The buffer contains the specified label.
X'04'  The specified label does not exist.
X'08'  The buffer length is smaller than the length of the label, but
       part of the label in the length of the buffer was moved into the
       buffer.
X'14'  Contents of the LPL invalid.
X'1C'  No GETVIS storage available.
```

For return codes 0 and 8 the two-byte field LPLLBLEN in the LPL contains a binary value, specifying the length of the label. Field LPLSTORE specifies, in which subarea (LPLFREE, LPLTEMP, LPLPERM, LPLCLASS) the label was found.

GETNXL Function:

This function is needed if label information of a sequential file consists of several parts (extents). It can only follow a GETLBL function or another GETNXL function; i.e. the first or only part of a label is always fetched by a GETLBL function and only if the extent information in the label indicates that more extents exist, that are not part of the label just fetched, the additional information can be obtained by the GETNXL function (see also description of ADDNXL function).

GETNXL and preceding GETLBL function have to be issued within the same task.

LPL parameters required:
AREA, AREALEN, and FILENAM.

After completion register 15 contains one of the following return codes:

```
X'00'  Successful request. The buffer contains the specified label.
X'04'  No additional information does exist for this label.
X'08'  The buffer length is smaller than the length of the label, but a
       part of the label in the length of the buffer was moved into the
       buffer.
X'0C'  Within the task the function is not preceded by a GETLBL or
       another GETNXL function for the same file label.
X'14'  Contents of the LPL invalid.
```

REPLBL Function:

This function is used to modify a label that has been fetched by a previous GETLBL or GETNXL function. It is not possible to change the length of the label or its file name.

The REPLBL function must be preceded by a GETLBL or GETNXL function for the same file label. REPLBL and its preceding function have to be issued within the same task.

LPL parameters required:
AREA, AREALEN, and FILENAM.

After completion register 15 contains one of the following return codes:

```
X'00'  Successful request, the label has been replaced.
X'0C'  The replacement is not for a label just retrieved, or it changes
       fields which are not allowed to be changed.
X'14'  Contents of the LPL invalid.
```

ADDLBL Function:

This function inserts a label stored in the buffer into a label subarea. Before any label can be added to a label subarea a CLRGRPL or MODGRPL function must have been issued for this label subarea to clear or

to reactivate it. The CLRGRPL or MODGRPL function has to be issued in the same partition as the ADDLBL function.

LPL parameters required:
AREA, LABLEN, FILENAM, GROUP, and STORE. STORE is not required if GROUP=SYSTEM.

After completion register 15 contains one of the following return codes:

```
X'00'  Successful request. The label has been moved to the specified
       area.
X'0C'  The function does not follow a CLRGRPL or a MODGRPL function,
       or another ADDLBL function for the same label subarea.
X'14'  Contents of the LPL invalid.
X'18'  No space available in the label area.
```

ADDNXL Function:

This function is used to insert additional label information into the label area. VSE job control produces one label for each // EXTENT statement for sequential files. These labels contain the same 'DLBL' information (for example filename) but differ in their 'EXTENT' information. To enable VSE to support this function, additional label information is inserted to a label area by the ADDNXL function and accessed by the GETNXL function.

An ADDNXL function must be preceded by an ADDLBL or may follow another ADDNXL function.

LPL parameters required:
AREA, LABLEN, and FILENAME.

After completion register 15 contains one of the following return codes:

```
X'00'  Successful request. The label information is stored.
X'0C'  Within the partition this function is not preceded by an ADDLBL
       or ADDNXL function with the same file name specified.
X'14'  Contents of the LPL invalid.
X'18'  No more space available in the label area.
```

CLRGRPL Function:

With this function a group of labels can be deleted from the label area.

LPL parameters required:
GROUP and STORE. STORE is not needed if GROUP=SYSTEM.

After completion register 15 contains one of the following return codes:

```
X'00'  Successful request. All labels of the specified label subarea
       have been deleted.
X'14'  Contents of the LPL invalid.
```

MODGRPL Function:

This function is needed to reactivate a group of labels. It is used to prepare the adding of labels to a label subarea which is already closed. It has to be preceded by an ENDLBL function in the same partition.

LPL parameters required: GROUP=SYSTEM or PARTITION, STORE=PERM. STORE is not needed if GROUP=SYSTEM.

After completion register 15 contains one of the following return codes:

```
X'00'  Successful request. The specified label subarea is reactivated.
X'0C'  The function does not follow an ENDLBL function.
X'14'  Contents of the LPL invalid.
```

ENDLBL Function:

In case of an FBA device the label blocks are written on the device only if the block is completely filled up with labels. If the block is to be written before it is filled up (because no more labels will be added), the SLA has to be informed by using the ENDLBL function.

LPL parameters required:
None, or STORE=DEL|CLDEL (only used by JCL after the deletion of labels).

After completion register 15 contains the following return code:

```
X'00'  Successful request, the labels were written on the FBA device.
```

LOCGRPL Function:

This function is used to indicate to the SLA that the calling program (normally LSERV) wants to display the contents of the label subarea specified in the LPL. This function cannot be invoked by more than one task within the same partition.

No actual data transfer occurs. After this function has been issued the calling program can retrieve labels of the specified label subarea in their physical sequence by issuing GETNXGL functions.

LPL parameters required:
GROUP and STORE. STORE is not needed if GROUP=SYSTEM. STORE=DEL|CLDEL is only used by JCL in connection with the deletion of labels.

If LOCGRPL is used for partition labels, LPLGRP (GROUP parameter in LPL) must contain the partition number, even if issued for partition labels of the issuing partition. Partition numbers are calculated for static partitions: x'00' for BG, x'01' for F1, ... x'0A' for FA, x'0B' for FB; for dynamic partitions: PIK/16, i.e. x'0D' for the first allocated dynamic partition, x'0E' for the second one and so forth.
After completion register 15 contains one of the following return codes:

```
X'00'  Successful request. The SLA is ready to accept GETNXGL functions
       for the label subarea specified in the LPL.
X'04'  The specified label subarea is empty.
X'14'  Contents of the LPL invalid.
X'20'  Updating in progress, area not available.
```

GETNXGL Function:

This function treats a label subarea as a sequential file of labels. For each request the function retrieves the label next in sequence from the label subarea which was specified by a preceding LOCGRPL function, and stores it into the buffer specified in the LPL.

LPL parameters required:
AREA, AREALEN, GROUP, and STORE. STORE is not required if GROUP=SYSTEM.

After completion register 15 contains one of the following return codes:

```
X'00'  Successful request. The buffer contains the label next in
       sequence.
X'04'  The label subarea is exhausted.
X'08'  The buffer length is smaller than the length of the label.
X'0C'  The function is not preceded by a LOCGRPL function or another
       GETNXGL function for the same label subarea.
X'14'  Contents of the LPL invalid.
X'20'  Updating in progress, area not available.
```

DELLBL Function:

This function deletes partition labels (permanent, temporary or free-usage) of the partition under which the macro is executed. LPLAREA in the LPL points to a user supplied buffer. This buffer is to contain a table with maximal 32 8-byte entries: the first 7 bytes of each entry contain the filename of the LIR to be deleted, the last byte is used as a flag byte, where $IJBSLA indicates to the caller, whether one (or more) LIRs corresponding to filename were found and successfully deleted. The table's end must be specified with a X'FF' byte.

LPL parameters required:
AREA, and STORE. All other LPL parameters are ignored.

After completion register 15 contains one of the following return codes:

```
X'00'  Successful request. The last byte in each 8-byte table entry
       indicates whether corresponding label information was not
       found (x'00') or successfully deleted (x'80').
X'04'  The specified label-information subarea is empty.
X'14'  Contents of the LPL invalid for one of the following reasons:
       - Only LPLTEMP, LPLPERM or LPLFREE allowed for LPLSTORE
       - Only 32 table entries allowed in user supplied buffer
       - User supplied table must end with byte x'FF'
       - Table must not contain duplicate filenames
X'18'  No more space available in the label area.
X'1C'  No GETVIS storage (neither partition GETVIS nor system GETVIS)
       available to temporarily store label information records.
x'63'  Label-information area must reside in native data space
       (VDISK...USAGE=DLA) to process DELLBL request.
```

SRCLBL Function:

This function is used to find out whether a label already exists in a label subarea. It may be used from static partitions only and for permanent labels only. No label is returned but only the existence of the label is indicated in register 15.

LPL Parameters required:
FILENAME, GROUP, STORE (if GROUP ¬= SYSTEM)

After completion register 15 contains one of the following return codes:

```
X'00'  Label has not been found
X'28'  Same label found in subarea
```

PUTPN Function:

This function is used by the librarian for a CATAL-GO or LINK-GO job to save the name of the phase cataloged or linked in the system GETVIS area.

GETPN Function:

This function is used by the EXEC processor to retrieve the phasename stored by the librarian from the system GETVIS area. This name is needed to determine the phase to be executed, if an EXEC statement is encountered following a CATAL or LINK step.

PUTOV Function:

This function is used by job control to save a pending OVERWRITE statement, if an EXEC statement is encountered during procedure processing.

GETOV Function:

This function is used to retrieve the saved OVERWRITE statement if job control is reloaded after execution of a program, specified in the preceding EXEC statement.

CPCTYLBL Function:

(Capacity of label area). The capacity information is returned into the user supplied buffer specified in the LPL macro (AREA=address) and can be mapped with a new DSECT contained in the LPLDCT macro:

```
LPLCPCTY DSECT               MAPPING OF CAPACITY INFORMATION
LPLNLASA DS    H             NUMBER OF LAS'S AVAILABLE
LPLNLASU DS    H             NUMBER OF LAS'S CURRENTLY IN USE
LPLNLASM DS    H             HIGHWATERMARK: MAX NUMBER IN USE
```

For return code 0 the two-byte field LPLLBLEN (see LPLDCT macro) contains a binary value, specifying the number of bytes returned (currently 6).

If the label area resides on physical disk LABEL FUNCT=CPCTYLBL returns with x'63', thus indicating that the function is invalid.

Register Usage:

When the LABEL macro is executed general registers 14 to 1 are destroyed. Register 13 must contain the address of a register save area (18 fullwords). After execution of the macro register 15 contains a return code as described for each LABEL function.

Requirements for the caller:

- If the label area resides on data space (VDISK...USAGE=DLA in $0JCL.PROC - this is the default since VSE/ESA 2.4):

  AMODE 24 or 31
  RMODE 24 or ANY

- If the label area resides on physical disk:

  AMODE 24
  RMODE 24

The address of the user supplied buffer (LPLAREA) in the LPL must be a 24-bit address, because byte 0 of the LPL is reserved.

Label Parameter List (LPL):

The LPL contains all information necessary to execute the LABEL macro. For more information refer to LPL macro.

Note: After completion of GETLBL and GETNXL functions LPLPNUM and LPLSTORE (see “LPLDCT Macro” on page 230) are updated - if the specified label is found - according to the label subarea where the label is found.

LPL Macro

This macro is used to define a label parameter list (LPL) and has the following format:

```
[name]    LPL  [AREA=address]
               [,AREALEN=integer]
               [,LABLEN=integer]
               [,FILENAM=characters]
               [,GROUP={PARTITION|pn|SYSTEM|class}]
               [,STORE={TEMP|PERM|FREE|CLASS|DEL|CLDEL}]
```

AREA=address — Specifies the address of the buffer from which the SLA is to fetch a label, or where the SLA should store a label depending on the SLA macro specified. The buffer must reside below the 16MB line, that is AREA=address must denote a 24-bit address, because byte 0 is used by vendor products (namely disk or tape management systems) as a flag byte in conjunction with VS COBOL II.

AREALEN=integer — A number from 1 to 4 digits specifying the length of the buffer described above.

LABLEN=integer — A number from 1 to 4 digits specifying the length of the label in the buffer.

FILENAM=characters — Specifies the file name (1 to 7 characters) which is contained in the label that is to be fetched or stored.

GROUP=PARTITION — Specifies that the macro handles partition specific labels. It is always the partition under which the macro is executed. This is the default specification of GROUP.

GROUP=pn — Specifies the partition number if it is to be different from the partition in which the macro is executed.

GROUP=SYSTEM — Specifies that the macro handles system labels.

GROUP=class — Specifies a class value if STORE=CLASS|CLDEL.

STORE=TEMP — Specifies that the label belongs to the temporary label subarea. This is the default specification of STORE.

STORE=PERM — Specifies that the label belongs to a permanent partition label subarea.

STORE=FREE — Specifies that the label belongs to the free-usage label subarea.

STORE=CLASS — Specifies that the label belongs to a class label subarea.

STORE=DEL — Specifies that the function is used before or after deleting labels (DEL is only used by JCL for LOCGRPL and ENDLBL for permanent labels except class standard labels.)

STORE=CLDEL — Specifies that the function is used before or after deleting labels (CLDEL is only used by JCL for LOCGRPL and ENDLBL for class standard labels.)

LPLDCT Macro

The layout of the LPL is described by a mapping DSECT, generated by the macro LPLDCT. The macro has the following format:

```
[name]      LPLDCT
```

The macro has no operands and its expansion looks as follows:

```
         DS    0F             GO TO WORD BOUNDARY
LPLDCT   DSECT
LPLAREA  DS    F              ADDRESS OF BUFFER
LPLBFLEN DS    H              LENGTH OF BUFFER
LPLLBLEN DS    H              LENGTH OF LABEL
LPLNAM   DS    CL8            FILE NAME
LPLGRP   DS    0X             GROUP OF LABELS
LPLPNUM  DS    X              PARTITION NUMBER
LPLSYS   EQU   X'FF'          1. BYTE OF LPLGRP IF SYSTEM LABELS
LPLPART  EQU   X'00'          1. BYTE OF LPLGRP IF PART. LABELS
         DS    X              RESERVED
LPLSTORE DS    X              OPTION CODE
LPLTEMP  EQU   0              OPTION CODE FOR TEMPORARY LABELS
LPLPERM  EQU   1              OPTION CODE FOR PERMANENT LABELS
LPLFREE  EQU   2              OPTION CODE FOR FREE USAGE LABELS
LPLDEL   EQU   3              OPTION CODE TO DELETE LABELS
LPLCLASS EQU   4              OPTION CODE FOR CLASS LABELS
LPLCLDEL EQU   5              OPTION CODE FOR DELETE CLASS LABELS
LPLINDIC DS    X              INDICATORS IN LPL
LPLSQNO  EQU   0              NO CHECKING OF EXTENT SEQUENCE NUMBER
LPLSQCHK EQU   1              CHECKING OF EXTENT SEQUENCE NUMBER
LPLGETNP EQU   2              GETLBL/GETNXLB PARALLEL
LPLSEQNM DS    X              EXTENT SEQUENCE NUMBER
LPLGETLB EQU   X'01'          FUNCTION CODE FOR GETLBL
LPLGETNL EQU   X'02'          FUNCTION CODE FOR GETNXL
LPLREPLB EQU   X'03'          FUNCTION CODE FOR REPLBL
LPLADDLB EQU   X'04'          FUNCTION CODE FOR ADDLBL
LPLADDNL EQU   X'05'          FUNCTION CODE FOR ADDNXL
LPLCLRGL EQU   X'06'          FUNCTION CODE FOR CLRGRPL
LPLENDLB EQU   X'07'          FUNCTION CODE FOR ENDLBL
LPLLOCGL EQU   X'08'          FUNCTION CODE FOR LOCGRPL
LPLGETNG EQU   X'09'          FUNCTION CODE FOR GETNXGL
LPLMODGL EQU   X'10'          FUNCTION CODE FOR MODGRPL
LPLSRCLB EQU   X'11'          FUNCTION CODE FOR SRCLBL
LPLSUCC  EQU   X'00'          SUCCESSFULL REQUEST
LPLNOINF EQU   X'04'          LABEL DOES NOT EXIST
LPLBUFLG EQU   X'08'          BUFFER LENGTH TOO SMALL
LPLETSSL EQU   X'0A'          ONLY USED BY ETSS:WRONG LENGTH
LPLCDLDF EQU   X'0B'          ONLY USED BY ETSS:CDLOAD NOT OK
LPLWRSEQ EQU   X'0C'          WRONG SEQUENCE OF MACROS
LPLDASGN EQU   X'0D'          DYN.ASSIGN DID NOT WORK
LPLGETVC EQU   X'0E'          GETVCE DID NOT WORK
LPLINVAL EQU   X'14'          CONTENTS OF LPL INVALID
LPLSPACE EQU   X'18'          NO SPACE IN LABEL AREA
LPLGETVS EQU   X'1C'          NO GETVIS SPACE AVAILABLE
LPLPRGSS EQU   X'20'          LABEL SUBAREA IN PROGRESS
LPLWREXT EQU   X'24'          WRONG EXTENT SEQUENCE NUMBER
LPLSLBFD EQU   X'28'          SAME LABEL FOUND IN SUBAREA
LPLFCINV EQU   X'63'          LABEL FUNCTION CODE IS INVALID
```

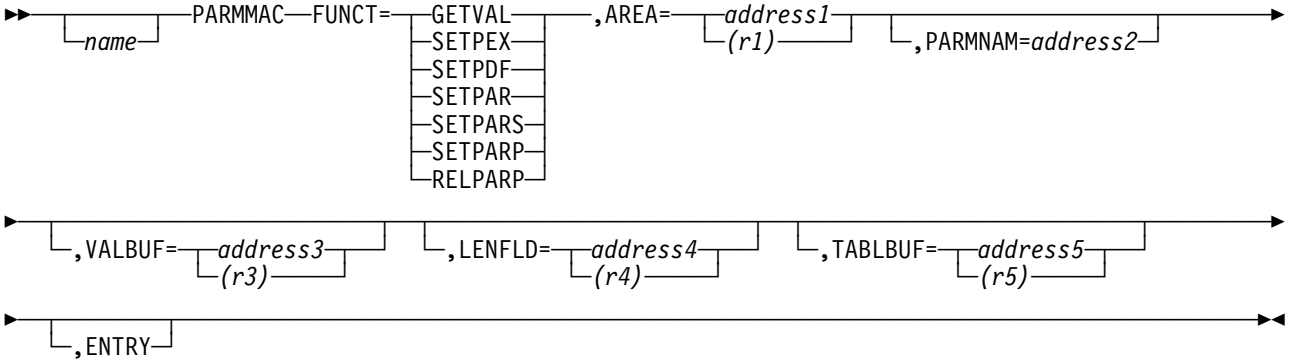
```
LPLLEN   EQU   *-LPLDCT       OBTAIN LENGTH OF LPL
LPLCPCTY DSECT               MAPPING OF CAPACITY INFORMATION
LPLNLASA DS    H             NUMBER OF LAS'S AVAILABLE
LPLNLASU DS    H             NUMBER OF LAS'S CURRENTLY IN USE
LPLNLASM DS    H             HIGHWATERMARK: MAX NUMBER IN USE
```

PARMMAC Macro

The PARMMAC macro allows to request service functions from the SVA - module IJBPROC. The possible functions which can be requested by PARMMAC are:

- Retrieve the value of a specified procedure parameter.
- Define or change a parameter or a set of parameters due to EXEC PROC, PROC, or SETPARM statement.

Format



Operands FUNCT describes the requested function of the macro.

FUNCT=GETVAL: Get the value of the procedure parameter specified in PARMNAM from the parameter tables and put the character string which represents the value into the buffer specified by VALBUF. The following parameter tables are searched:

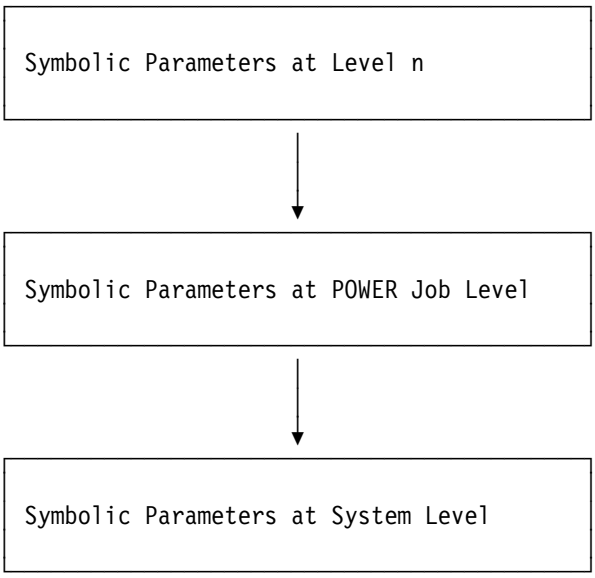


Figure 19. Search Sequence for Substitution of Symbolic Parameters

The buffer length of VALBUF should be 5o bytes.

FUNCT=SETPEX: An EXEC PROC statement has been processed with parameter definitions. The output of the statement processor is a buffer containing a set of table entries. The pointer to this buffer is specified with the TABLBUF parameter. The macro service fills the parameter table with the table entries contained in the buffer.

FUNCT=SETPDF: A PROC statement of a cataloged procedure has been processed with default parameter definitions. The output of the statement processor is of the same type as for EXEC PROC and is contained in a buffer specified via TABLBUF parameter. The macro service puts the table entries into the parameter table as far as EXEC PROC definitions do not override the PROC default definitions.

FUNCT=SETPAR: A SETPARM JOB statement has been processed. This results in a service call with the SETPAR function. The SETPAR function defines either new parameters for the currently active cataloged procedure or changes the value of already existing parameters. The output of the statement processor (IJBJC9) is of the same type as for EXEC PROC and PROC. The output is contained in a buffer specified via TABLBUF parameter. The macro service puts the new entries into the appropriate parameter table (symbolic parameters at level n).

FUNCT=SETPARS: A SETPARM SYSTEM statement has been processed. Similar to SETPAR this macro service puts the new entries into the appropriate parameter table (symbolic parameters at system level).

FUNCT=SETPARP: A SETPARM PWRJOB statement has been processed. Similar to SETPAR this macro service puts the new entries into the appropriate parameter table (symbolic parameters at POWER job level).

FUNCT=RELPARP: A POWER EOJ (* $$ EOJ) statement has been processed. GETVIS space and pointer chain associated with symbolic parameters at POWER job level are released, see Figure 18 on page 210.

AREA: The AREA parameter is required. By means of the AREA parameter the user provides a 24 byte block in the user program which is used as control block for saving all information related to the macro call. AREA is specified either by a symbolic address or by a pointer in a register.

PARMNAM: This parameter is required for GETVAL. It provides the symbolic address of a 7 - byte field containing the parameter name. A parameter name shorter than 7 bytes must start with the first position from the left, while the unused bytes must be blank.

VALBUF: This parameter is required for GETVAL. It provides a pointer to the buffer which will receive the character string for the GETVAL function. The buffer length should be 50 bytes.

LENFLD: LENFLD specifies a two byte field containing the length of the character string assigned to the parameter. LENFLD is required for GETVAL. The length in LENFLD must not exceed 50 bytes. For GETVAL the length is set by the system as part of the response.

TABLBUF: This parameter is required for SETPEX, SETPDF, SETPAR, SETPARS, and SETPARP. It points to the table buffer which contains a set of pre-formatted parameter table entries which are related either to the assignments of parameters in an EXEC PROC statement (SETPEX function) or to default assignments of parameters in a PROC statement (SETPDF) or to the SETPARM statement (SETPAR, SETPARS, and SETPARP functions). The precise format of the table entries which have to be put into the table buffer has been defined internally.

ENTRY: This parameter specifies that the address of the loaded SVA-module is already in Reg. 15.

Register Usage:

```
Reg. 0 and Reg. 1   Internal work registers
Reg. 13             Address of 72 - byte area
Reg. 14             Return address
Reg. 15             Return codes
```

Return Codes from Language Processor

4 Invalid function code

8 Invalid macro parameter

12 Required parameter not specified, generation suppressed

Return Codes from Service Function

0 Request was successful

8 Invalid length in LENFLD

C Invalid pointer for a buffer parameter

10 Parameter not defined in GETVAL request

14 SETPDF request occurred twice

18 SETPDF request occurred after second GETREC

1C Space block chain exhausted

20 No system GETVIS space available

2C No partition GETVIS space available

40 Invalid function.

PROCMAC Macro

The PROCMAC macro allows to request service functions from the SVA-resident routine $IJBPROC. The possible functions which can be requested by PROCMAC are:

- Establish access path to the cataloged procedure activated by an EXEC PROC statement.
- Get next sequential record from the currently active cataloged procedure.
- Do all necessary librarian or table cleanup if end of procedure (/+) occurs.
- Do complete librarian and table cleanup if /& or cancel occurs.
- Get information about so-called incore procedures. These incore procedures are established by REXX via address JCL or when commands are queued on the REXX data stack.
- Query functions for REXX.

Format:

```
[LABEL] PROCMAC FUNCT=ACCESS|GETREC|EOPREQ|CANCRQ|INCORE|INCOREX|QUERYI|QUERYP
               ,AREA={NAME|(R)}
               [,PROCNAME=NAME]
               [,BUF={NAME|(R)}]
               [,BUFLEN={NAME|(R)}]
               [,ENTRY]
```

Operands: FUNCT describes the requested function of the macro.

FUNCT=ACCESS: Procedure library is opened and access path is established to cataloged procedure specified in PROCNAME so that records can be read on GET level. If nesting is required (Level 2 or higher) the stack of note information fields of macro INLMNOTE is also maintained.

FUNCT=GETREC: Get next sequential record of the currently active cataloged procedure from disk and put it into the buffer specified by BUF. GETREC is not allowed on Level 0 (see under nested procedures).

FUNCT=EOPREQ: When the currently active cataloged procedure has encountered end of procedure (/+), the service request EOPREQ does the necessary cleanup of the librarian control blocks as well as of the parameter tables. EOPREQ is not allowed on Level 0.

FUNCT=CANCRQ: Either a /& occurred or a cancel request was given. The macro service issues FREEVIS for all system GETVIS space held by the affected partition for parameter tables.

FUNCT=INCORE: Sets low, current and high address of incore procedure that should be executed instead of the current procedure. This is required to have JCL process the statements left on the REXX data stack (see label HREXXCNT in $JOBCTLA).

FUNCT=INCOREX: Sets low, current and high address of incore procedure that should be executed instead of the current procedure. This is required to support the address JCL command environment provided by REXX (see label ADRJCLRX in $JOBCTLA).

FUNCT=QUERYI: Returns the addresses of remaining statements in the in-core procedure to the caller (REXX). Thus the in-core procedure can be overwritten by another one and later be restored when it is needed again

FUNCT=QUERYP: Returns the name of a n-1 level procedure. Thus a REXX program can determine from which procedure it was called (if any).

AREA: The AREA parameter is required. By means of the AREA parameter the user provides a 24 byte block in the user program which is used as control block for saving all information related to the macro call. AREA is specified either by a symbolic address or by a pointer in a register.

PROCNAM: It provides the name of the field which contains the procedure name of the currently requested procedure. PROCNAM is required if FUNCT=ACCESS is specified. The procedure name must start with the first left position, while the rest of the buffer must contain blanks.

BUF: This parameter is required if GETREC is specified. The user provides the pointer to the 80 byte buffer in the user program, where the procedure record will be placed.

BUFLEN: This parameter is optional for a GETREC request. The user provides a pointer to a two-byte field which contains the desired length of the procedure record. The length must be greater than zero. If the length is greater than 79 an 80-byte record will be put into the buffer specified by buf.

ENTRY: This parameter specifies that the address of the loaded SVA-module is already in Reg. 15.

Register Usage:

```
Reg. 0 and 1 Internal work registers
Reg. 13      Address of 72-byte area
Reg. 14      Return address
Reg. 15      Return codes
```

Return Codes from Language Processor

4 Invalid function code

8 Invalid macro parameter

12 Required parameter not specified, generation suppressed

Return Codes from Service Function

0 Request has been successful

4 Procedure not found

8 EOPREQ was given on level 0

C GETREC was given on level 0

10 ACCESS exceeds nesting level of 15

14 Duplicate procedure name in nested stack

18 Request outside member

1C Invalid pointer to buffer

20 No system GETVIS space available

24 One of the following:

- Insufficient SYSTEM GETVIS SPACE for librarian,
- Security violation found by librarian, or
- Inconsistency found by librarian.

28 Conflict in nested stack related to data=yes/no option (must be all of same type)

2C No partition GETVIS space available.

30 Error in LABEL request.

34 Partition FREEVIS failed.

40 Invalid function.

Chapter 9. Data Areas -- Job Control

Format of TAPE Labels in the Label Information Area

Figure 20. Format of Tape Labels in the Label Information Area

Displacement	Bytes	Description
0	1	Reserved
1	7	File name
8	1	Auxiliary flag byte
9	17	File ID
26	6	File-serial-number
32	4	Volume-sequence number (EBCDIC) or File-section number (ASCII)
36	4	File-sequence number
40	4	Generation number
44	2	Version number
46	6	Creation date: initialized to X'404040404040'. If an expiration date ccyy/ddd was specified in the date operand of the TLBL statement, then the creation date contains Cyyddd, where C is the century identifier: C=X'40' for cc=19, C=X'F0' for cc=20.
52	6	Expiration date or retention period: Initialized to X'000000000000'. If a retention period was specified in the date operand of the TLBL statement, then the field contains the hexadecimal representation of this retention period. If an expiration date ccyy/ddd was specified in the date operand of the TLBL statement, then the field contains Cyyddd, where C is the century identifier: C=X'40' for cc=19, C=X'F0' for cc=20.
58	1	File Security
59	6	Block Count
65	13	System Code: character string 'IBMDOSVS' concatenated with 5 blanks
78	2	Flag byte: • X'0080' DISP=NEW • X'0040' DISP=OLD • X'0020' DISP=MOD

Note: A record length of 80 bytes is typical of a tape label record.

Format of DASD Labels in the Label Information Area

Figure 21 (Page 1 of 2). Format of DASD Labels in the Label Information Area

Displace-ment	Bytes	Description
0	1	DLBL-EXTENT Indicator • SAM: – bit0: 1 = Next extent on a new pack – bit1: 1 = Last extent – bit2: 1 = Bypass extent – bit3: 1 = New volume on same unit – bit4: 1 = Extent limits omitted – bit5: 1 = Extent converted to DASD address – bit6: 1 = No EXTENT card – bit7: 1 = Data secured file • DAM, ISAM or VSAM: Number of extents
1	7	Filename
8	1	Switch byte • bit0: 1 = Additional DLBL information record follows • bit1: 1 = CISIZE specified in DLBL • bit2: 1 = BLKSIZE specified in DLBL • bit3: 1 = FBA indicator for OPEN • bit4: 1 = Extent limits omitted • bit5: 1 = Extent converted to DASD address • bit6: 1 = Additional DLBL information record • bit7: 1 = Data secured file
9	44	File ID
53	1	Format ID: Numeric 1 is inserted
54	6	File Serial Number: Volume Serial Number from first extent
60	2	Volume Sequence Number: Initialized X'0001'
62	3	Creation Date: Initialized X'000000'
65	3	Expiration Date: Initialized to X'000000'. If specified, the input from DLBL (ccyy/ddd) is converted to YDD, where Y denotes the hexadecimal offset to the year 1900 and DD the hexadecimal representation of ddd. For example X'63016D' corresponds to 1999/365.
68	2	Retention Period: Initialized to X'0007'. If specified, the input from DLBL (1 to 4 decimal characters) is converted to a 2-byte hexadecimal number and inserted in this field.
70	1	Open Code: • A = VSAM • S = Sequential access method • D = Direct access method • C = ISAM Load create function • E = ISAM Load extent function

Figure 21 (Page 2 of 2). Format of DASD Labels in the Label Information Area

Displacement	Bytes	Description
71	13	System Code: " • For VSAM: – bytes 71-77: Filename of owning catalog, as specified in DLBL CAT – bytes 78-79: Not used by VSE – bytes 80-83: Buffer space for this file, as specified in DLBL BUFSP • For BLKSIZE and CISIZE parameters: – bytes 71-75: Reserved – bytes 76-79: CISIZE – bytes 80-83: BLKSIZE • Otherwise: character string 'IBMDOSVS' concatenated with 5 blanks
84	6	Volume Serial Number
90	1	EXTENT Type: Same code as in format-1 label • X'00' = Next three fields do not indicate any extent. • X'01' = Prime data are ISAM, data area (SAM,DAM), or data space (VSAM) • X'02' = Overflow area of an ISAM file • X'04' = Cylinder index or master index of an ISAM file. • X'40' = User label track area • X'80' = Data area with split cylinder SAM • X'8n' = Shared cylinder indicator, where n=1,2,or 4.
91	1	Extent Sequence Number
92	8	If Byte 102, bit 0 is 0: • 2 bytes: Start track/block (binary) • 2 bytes: Number or tracks/blocks (binary) • 4 bytes: X'00000000' If Byte 102, bit 0 is 1: • 4 bytes: Start track/block (binary) • 4 bytes: Number or tracks/blocks (binary)
100	2	Logical (Symbolic) Unit Address: Identifies the logical unit with the same code as the one used in the CCB. • 1st byte: – X'00' = System logical unit – X'01' = Programmer logical unit • 2nd byte identifies the logical unit within its class. – X'04' = Cylinder index or master index of an ISAM file. – X'40' = User label track area – X'80' = Data area with split cylinder SAM – X'8n' = Shared cylinder indicator, where n=1,2,or 4.
102	2	Indicators: • bit0: 1 = Block address or block number > 64k-1. • bits 1-15: Reserved
Note: For SAM files, a complete 104-bytes block is repeated for each new EXTENT. For DAM, VSAM, and ISAM files, only displacements 84 through 103 are repeated for each EXTENT.		

For additional label records, (Byte 8 bit 6 is '1') the layout differs from the upper one. Only byte 0 up to byte 8 are used identically.

Figure 22. Additional Label Record Layout

Displace-ment	Bytes	Description
9	1	Disposition • bit0: 1= Open disposition is NEW • bit1: 1= Open disposition is OLD • bit2: 1= Abend disposition is KEEP • bit3: 1= Abend disposition is DELETE • bit4: 1= Termination disposition is KEEP • bit5: 1= Termination disposition is DELETE • bit6: 1= Termination disposition is DATE • bit7: reserved
10	2	reserved
12	4	Primary allocation (number of records)
16	4	Secondary allocation (number of records)
20	4	Record size
24	4	Number of data buffers
28	4	Number of index buffers

Format of Symbolic Parameter Information Records

For each procedure nesting level n (0 ≤ n ≤ 15) $IJBPROC allocates up to 10 2KB-chunks of GETVIS storage to store symbolic parameter information records. This GETVIS storage is allocated with the attribute SPACE=YES, the pool identifier is IJBPRC, concatenated with a 2-byte partition identifier. Hence for a dynamic partition the required storage areas are taken from the dynamic space GETVIS area, whereas for static partitions they are allocated in the system GETVIS area.

It takes n+10 bytes to store a symbolic parameter information record, where n denotes the length of the character string assigned to the parameter name.

Figure 23. Format of Symbolic Parameter Information Records

Displacement	Bytes	Description
0	2	Length of symbolic parameter value
2	1	Flag byte
3	7	Name of symbolic parameter
10	≤ 50	Value of symbolic parameter

Communication Areas

SYSCOM, the system communication area, described in the *Supervisor Diagnosis Reference* manual, has the name SYSCOM also as a DSECT in each job control phase.

The partition communication areas, called COMREG in the supervisor, have the name COMDST as a DSECT in each job control phase except in $JOBCTLM.

Job Control Option Bytes in COMDST

One part of the COMDST, the job control option bytes are essential for job control option processing and therefore they are shown here.

```
Name:               Option fields

Location:           Fields in COMDST

Initialized by:     IPL

Changed by:         The statements OPTION, STDOPT, and /&

Used by:            Linkage editor, FETCH, compilers, job control
```

Table of job control option bytes in partition COMDST which have a standard option equivalent:

Temporary Option Field	JCSW2 X'39'	JCSW3 X'3A'	JCSW4 X'3B'	TEMOPT X'8D'	JCSW5 X'AC'	TEMOPT2 X'AF'	JCSW6 X'AD'	JCSW7 X'3C'	IJBTMOP3 X'104'
Standard Option Field	-	SOB1 X'36'	SOB2 X'37'	STDOPT X'8C'	-	STDOPT2 X'AE'	-	-	IJBSTOP3 X'103'
Options |		Bit definition and setting within byte. In parentheses: additional functions.							
-- V -------				V					
ACANCEL				X'01'=1					
--NOACANCEL				X'01'=0					
ALIGN				X'40'=1					
--NOALIGN				X'40'=0					
CATAL -)	X'10'=1								
CATAL -)	X'80'=1								
CATAL -)	X'20'=0				(X'20'=1)				
DECK		X'80'=1							
--NODECK		X'80'=0							
DSPDUMP						X'02'=1			
--NODSPDUMP						X'02'=0			
DUMP			X'40'=1	(X'20'=0)					
--NODUMP			X'40'=0	(X'20'=0)		(X'02'=0)			
EDECK				X'80'=1					
--NOEDECK				X'80'=0					
ERRS		X'04'=1							
--NOERRS		X'04'=0							
JCANCEL						X'08'=1			
--NOJCANCEL						X'08'=0			
LINK -)	X'80'=1								
LINK -)	X'20'=0				X'20'=1				
--NOLINK	X'80'=0				(X'20'=0)				
LIST		X'40'=1							
--NOLIST		X'40'=0							
LISTX		X'20'=1							
--NOLISTX		X'20'=0							
LOG			X'10'=1						
--NOLOG			X'10'=0						
NOFASTTR						X'80'=1			
ONLINE							X'40'=1		
--NOONLINE							X'40'=0		
PARTDUMP			(X'40'=1)	X'20'=1					
RLD				X'10'=1					
--NORLD				X'10'=0					
SCANCEL						X'01'=1			
--NOSCANCEL						X'01'=0			
SUBLIB=AE				X'02'=0					
SUBLIB=DF				X'02'=1					
SXREF		(X'08'=0)		X'08'=1					
SYM		X'10'=1							
--NOSYM		X'10'=0							

Temporary Option Field	JCSW2 X'39'	JCSW3 X'3A'	JCSW4 X'3B'	TEMOPT X'8D'	JCSW5 X'AC'	TEMOPT2 X'AF'	JCSW6 X'AD'	JCSW7 X'3C'	IJBTMOP3 X'104'
Standard Option Field	-	SOB1 X'36'	SOB2 X'37'	STDOPT X'8C'	-	STDOPT2 X'AE'	-	-	IJBSTOP3 X'103'
Options		Bit definition and setting within byte. In parentheses: additional functions.							
SYSPARM	MOVE PARAMETER INTO SYSPARM FIELD								
SYSDUMP						X'40'=1			
--NOSYSDUMP						X'40'=0			
XREF		X'08'=1		(X'08'=0)					
--NOXREF		X'08'=0		(X'80'=0)					
TERM				X'04'=1					
--NOTERM				X'04'=0					
48C		X'02'=1							
64C		X'02'=0							
LOGSRC								X'40'=1	
--NOLOGSRC								X'40'=0	
IGNLOCK									X'80'=1
--NOIGNLOCK									X'80'=0
ACL									X'40'=1
--NOACL									X'40'=0
OLDASSEM									X'20'=1
--NOOLDASSEM									X'20'=0
SYSDUMPC									X'10'=1
--NOSYSDUMPC									X'10'=0
SLISKIP									X'08'=1
--NOSLISKIP									X'08'=0

TEMPORARY OPT. FIELD	PCBSADMP IN PCB	PCBSADM1 IN PCB
SADUMP	FIRST NUMBER	SECOND NUMBER

STDLABEL	CLOSE THE CURRENT LABEL AREA AND OPEN THE STANDARD LABEL AREA. IF STDLABEL=DEL IS SPECIFIED, DELETE THE LABELS SPECIFIED.
PARSTD	CLOSE THE CURRENT LABEL AREA AND OPEN THE PARSTD LABEL AREA. IF PARSTD=DEL IS SPECIFIED, DELETE THE LABELS SPECIFIED.
CLASSTD	CLOSE THE CURRENT LABEL AREA AND OPEN THE SPECIFIED CLASS LABEL AREA. IF CLASSTD=(x,DEL) IS SPECIFIED, DELETE THE LABELS SPECIFIED.
USRLBL	CLOSE THE CURRENT LABEL AREA AND OPEN THE USER LABEL AREA.

BASVCT -- Common Job Control Area

The job control common area called BASVCT in the job control root phase, has the name BASDST as a DSECT in each other job control phase.

```
Name:                Common job control area

Label or Identifier: BASVCT

Location:            At the beginning of $JOBCTLA. Each processing
                     phase contains a DSECT of this area which starts
                     with the label BASDST.

Initialized:         Via the macro MAPJCLA DSECT=NO for the root phase
                     and DSECT=YES for the processing phases.

Used by:             All job control phases

Contains:         1. A Branch table with entries for all common
                     subroutines of JCL. These subroutines are
                     located in the root phase $JOBCTLA and can be
                     invoked by every JCL Overlay Phase.
                  2. Buffers for the statement input and message output
                  3. DFBs as described below.
                  4. Common constants and control fields as listed.
```

DFBs -- Data File Blocks in Phase $JOBCTLA

```
Names:                  RDRDFB  for SYSRDR
                        PCHDFB  for SYSPCH
                        IPTDFB  for SYSIPT
                        LSTDFB  for SYSLST
                        LNKDFB  for SYSLNK
                        LGIDFB  for SYSLOG input
                        LGODFB  for SYSLOG output

                        PRCDFB  for procedure library
                        ORDRDFB for overwrite statements on SYSRDR
                        OLOGDFB for overwrite statements on SYSLOG

Labels or Identifiers: See "Names"

Location:               In BASDST, the last three elsewhere in $JOBCTLA

Used by:                Job control for I/O file handling.
                        Each DFB consists of a CCB, control information,
                        and CCW chains to handle the appropriate system
                        files on disk, tape, or unit record.

                        The SYSLNK DFB has no CCWs as it is handled via
                        LIOCS.
```

Job Accounting Interface Tables

The Job Accounting Common Table (ACCTCOMN) is used by the supervisor and job control to save and to communicate general job accounting information.

The Job Accounting Partition Tables (ACCTxx) are maintained by the supervisor and by job control. They contain job accounting information for the partitions.

DSECT ACCTABLE symbolically addresses the JAI partition tables. Each partition in which JAI is supported has its own JAI partition table labelled ACCTBG for back ground, ACCTF1 for the active partition F1, ACCTF2 for F2 etc.

A detailed list of both types of Job Accounting Interface tables is given in *VSE/Advanced Functions Diagnosis Reference: Supervisor*.

TBLADR -- Phase Vector Table in $JOBCTLA

Names: Phase vector table

Label: TBLADR

Location: In phase $JOBCTLA

Used by: Phase $JOBCTLA

The phase vector table contains a 14-byte entry for each job control command or statement. This entry is generated by means of the COMND macro. Figure 24 shows the format and contents of this 14-byte field:

byte 0 to 6	byte 7	byte 8	byte 9	byte 10-11	byte 12	byte 13
operation field	condition switches	branch vector displacement	phase-ID character	reserved	condition switches	reserved

Figure 24. Entry of the Phase Vector Table

The meaning of the fields is as follows:

Operation Field:
EBCDIC representation of the statement/command name.

Condition Switches (byte 7)

Bit 0 continuation lines are allowed for this statement/command (for example: EXEC, PROC, SETPARM, IF, DLBL, TLBL, SETPRT, LIBDEF, LIBLIST, LIBDROP, LIBSERV).

Bit 1 statement/command is to be processed even though job control is in skip mode (for example: /&, /+, /., JOB, OVEND). Skip mode may be entered for one of the following reasons:

- A `GOTO label` statement or command has been given. All statements until label (or end-of-job) are skipped.
- An `ON condition GOTO label` statement or command has been given. If the condition is true, all statements until label (or end-of-job) are skipped.
- An `IF condition THEN` statement or command has been given. If the condition is not true, the next statement is skipped.
- The job has been canceled by the operator. All statements until end-of-job are skipped.

Bit 2 off: unconditional logging of statement/command on SYSLOG even when NOLOG has been specified (for example: *, IGNORE, PRTY, DVCUP, UNBATCH, HOLD, STOP, ALLOC, ALLOCR, SIZE, JCLEXIT, MAP).

Bit 3 off: log/list statement/command after reading
on: log/list statement/command just before reading next statement.

Bit 4 statement starts with // (for example JOB, DLBL, EXTENT, OPTION, DATE, UPSI, ZONE, TLBL).

Bit 5 command starts without // (for example START, JCLEXIT, DVCDN, DVCUP, SET, UNBATCH).

Bit 6 statement must start in column 1 (for example /&, /+, //, /*, *).

Bit 7 statement/command may start in other than column 1.

Branch Vector Displacement:
: Displacement within the phase to find the address of a branch instruction which transfers control to the correct processing routine.

Phase Identification Character:
: Contains the last EBCDIC character of the name of the job control phase containing the processing routine.

Condition Switches (byte 12)

Bit 0 This bit is on if the command/statement must not be given in a dynamic partition (for example: DVCUP, DVCDN, UNBATCH, HOLD, STOP, ALLOC, ALLOCR, SIZE, NPGR, RSTRT, START).

Bit 1 This bit is on if the command/statement must not be given together with `address JCL` in a REXX procedure (for example: PAUSE, EXEC, PROC, IF, ON, GOTO, UNBATCH, /&, /+, /., JOB, ALLOC, RSTRT, ROD,OVEND). These job control commands result in a return code of -7 when specified together with the REXX `address JCL` command.

Bit 2 This bit is on if the command/statement requires special authorization (for example: ALLOC, DVCDN, DVCUP, HOLD, JCLEXIT, MSECS, NPGR, PRTY, ROD, SET, SIZE, START, STOP, SYSDEF, UCS, UNBATCH, * CP).

Bit 3-7 Reserved

Example of an Entry in the Phase Vector Table: The JOB statement may have the following entry in the phase vector table:

```
DC    CL7'JOB'
DC     B'01111010'
DC    AL1(8)                  DISPLACEMENT 8 IN PROCESSING PHASE
DC    C'G'                    PROCESSING PHASE IS $JOBCTLG
DC    AL2(0)                  RESERVED
DC    B'01000000'             NO REXX
DC    X'00'                   RESERVED
```

The condition switches indicate:

- The JOB statement does not allow any continuation statement.
- The JOB statement is processed even if job control is in skip mode.
- Both logging and listing are suppressed.
- The JOB statement must start with // and in column 1.
- The JOB statement is allowed in dynamic partitions.
- The JOB statement is not allowed in REXX.
- The JOB statement does not require special authorization.

Bytes 8 and 9 of the entry say:

The branch-vector table entry is located at a displacement of 8 bytes from the beginning of the phase with suffix 'G' ($JOBCTLG).

JCARAREA - Interface Area Between JCL/AR and SVA Routines

This interface area is used for communication between the JCL phase $JOBCTLO and the AR phase $IJBATTN on the one hand, and the SVA routines $IJBMAP, $IJBPRTY, $IJBSDSP, $IJBSLIB and $IJBVDII on the other hand:

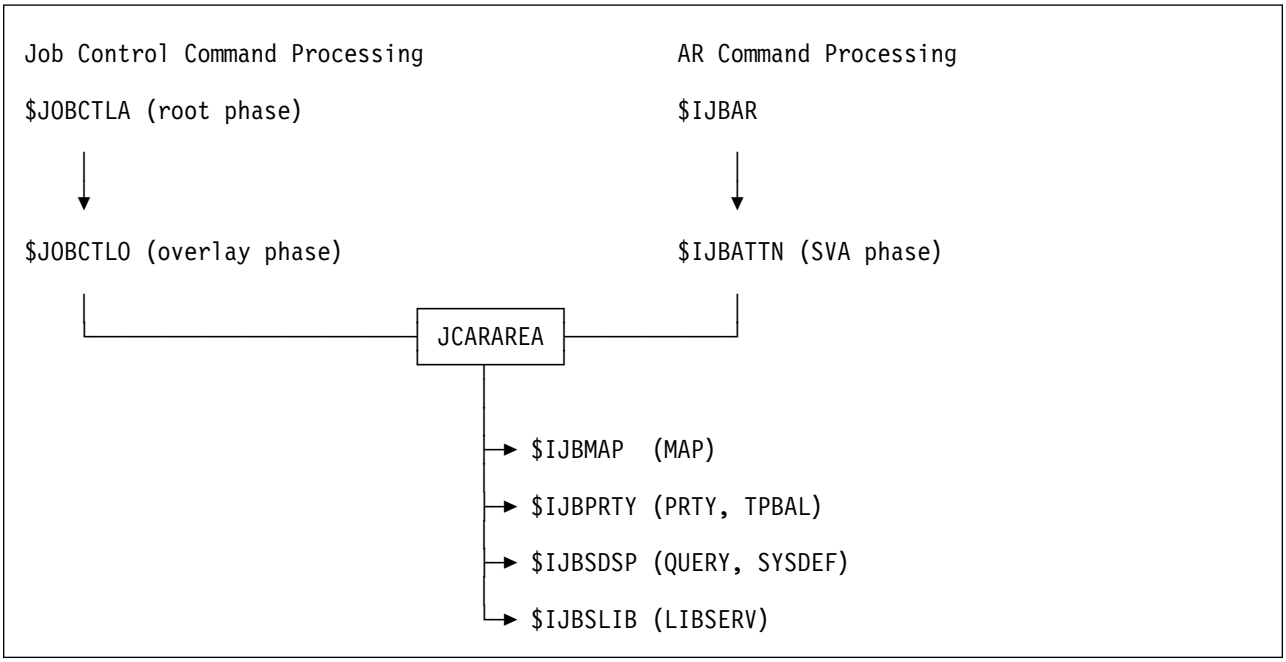


Figure 25. LIBSERV, MAP, PRTY, QUERY, SYSDEF, and TPBAL Command Handling

Macro JCLARIF generates a DSECT for this interface area:

```
DCL 1 JCARAREA      BASED,           /*INTERFACE AREA                  */
      2 JCARBUFF    PTR(31),         /*ADDRESS OF I/O BUFFER           */
      2 JCAROUTR    PTR(31),         /*ADDRESS OF OUTPUT ROUTINE       */
      2 JCARSCNR    PTR(31),         /*ADDRESS OF SCAN/LOCK ROUTINE    */
      2 JCARITPT    PTR(31),         /*ADDRESS OF SCANNED PARAMETER    */
      2 JCARITLN    BIN(15) SIGNED,  /*LENGTH OF SCANNED PARAMETER - 1 */
      2 JCARNSTP    CHAR(1),         /*SCAN STOP CHARACTER             */
      2 JCARREQC    BIT(8),          /*REQUEST BYTE FOR FUNCTION       */
      2 JCARSV13    PTR(31),         /*SAVE AREA FOR REGISTER 13       */
      2 JCARSAVE(16) PTR(31),        /*SAVE AREA CALLING PHASE REGS.   */
      2 JCARSVE2(18) PTR(31),        /*SAVE AREA FOR SVA ROUT. REGS.   */
      2 JCARMSGF    CHAR(72);        /*MESSAGE AREA FOR OUTPUT FROM SVA ROUTINE */
      2 JCARMSGE    CHAR(48),        /*EXTENDED MESSAGE AREA           */
      2 JCARLPL     PTR(31);         /*JCL Label Parameter List        */

REQUEST CODES FROM CALLING PHASES TO IJBSDSP

DCL JCARSYSD BIT(8) CONSTANT('01'X);  /* PROCESS SYSDEF                */
DCL JCARQERY BIT(8) CONSTANT('02'X);  /* PROCESS QUERY                 */

REQUEST CODES FROM SVA ROUTINE TO CALLING PHASES

DCL JCARSCN2 BIT(8) CONSTANT('11'X); /* CALL SCANR2                    */
DCL JCARSCN3 BIT(8) CONSTANT('12'X); /* CALL SCANR3                    */
DCL JCARSCNS BIT(8) CONSTANT('13'X); /* CALL SCANRS                    */
DCL JCARLOCK BIT(8) CONSTANT('21'X); /* LOCK RESOURCE                  */(BG only)
```

```
DCL JCARUNLC BIT(8) CONSTANT('22'X); /* UNLOCK RESOURCE                */    "
DCL JCARGEND BIT(8) CONSTANT('31'X); /* REQUEST GENERATED STMNT        */    "
DCL JCARSTMN BIT(8) CONSTANT('41'X); /* OUTPUT JCL STATEMENT ONLY      */

Return CODES FROM SVA-PHASES

DCL JCARRTN  BIT(8) CONSTANT('01'X); /* SHOW OPERAND NUMBER IN MSG     */
DCL JCARRTN0 BIT(8) CONSTANT('02'X); /* SHOW MSG UNCHANGED             */
DCL JCARRTNX BIT(8) CONSTANT('03'X); /* SHOW OPERAND NUMBER IN JCARNSTP*/
DCL JCARRINV BIT(8) CONSTANT('04'X); /* INVALID STATEMENT/COMMAND      */
DCL JCARRGVS BIT(8) CONSTANT('08'X); /* GETVIS PROBLEM                 */
DCL JCARLMSG BIT(8) CONSTANT('16'X); /* LONG MESSAGE                   */
```

Chapter 10. Diagnostic Aids -- Job Control

Interface Information

Interface Between $JOBCTLA and the Processing Phases

When calling a processing phase:

```
R 6: Second base register of root
R 7: First base register and return to root
R 8: Start address of overlay area
R 9: Third base register of root
R10: Pointer to COMDST
```

In the common job control area:

```
TMPAR1: Pointer to first byte behind operation code
TMPAR2: Remaining number of bytes of the statement
```

In the partition COMDST:

```
JBCSW4: Bit X'80': off if statement starts with //
                   on if it does not
```

When returning from processing phase:

```
R 7: Pointer to beginning of this DSECT
R 8: Start address of overlay area
R10: Pointer to COMDST
```

The use of other registers is described with the processing phases.

Interface Between $JOBCTLA and Supervisor Phase $IJBFBA

The supervisor call 103 performs the input/output operations from and to system files on FBA. The code of the SVC103 is split into two parts:

- a resident part contained in the supervisor,
- a pageable part loaded into the SVA (phase $IJBFBA).

For performance reasons, however, $JOBCTLA calls $IJBFBA directly via SVC4 (LOAD).

Whenever one of these files is to be opened or initialized, $JOBCTLA requests GETVIS space for I/O buffers, IORB, and CCW chains, and it updates the DIB and DIB extension for the file INITDIB.

Only when during blocking or de-blocking a buffer is exhausted, I/O is performed via SVC103.

Entry Conventions from $JOBCTLA to $IJBFBA: The phase $IJBFBA is loaded into the SVA by IPL. Job control issues a LOAD macro (SVC4) for the phase $IJBFBA to get its load address returned and transfers control to the phase $IJBFBA by a branch-and-link instruction. Register use at entry to $IJBFBA:

```
R 0: The phase $IJBFBA was entered via LOAD, if the
     register contains 0
R 1: Pointer to CCB
R 5: Pointer to DIB
R13: Pointer to a 64 byte save area for the
     caller's registers
R14: Linkage between job control and IJBFBA
```

Note: Output to the linkage editor file (SYSLNK) is written via LIOCS (DTFCP) which avoids the need for checking if SYSLNK is on CKD, FBA, or in VSAM-managed space.

Interface Between $JOBCTLA and the Job Control Exit ($JOBEXIT)

Registers passed at entry to $JOBEXIT:

R 0: System identification characters 'SDOS' (x'E2C4D6E2')

R 1: Address of partition COMREG

R 2: Address of SYSCOM

R 3: Address of current statement's vector table entry (see Figure 24 on page 249 for the format and contents of this 14-byte entry)

R 4: Address of buffer that contains the currently processed statement

R 5: Number of continuation lines if read from SYSRDR, otherwise 0. If register 5 is not zero, the 80-byte long continuation card(s) is (are) located adjacent to the 80-byte long statement.

R 6: Anchor field. At the very first call (after IPL) JCL will load x'00000000' into register 6 before passing control to the exit routine. For all subsequent calls register 6 will contain the value that was returned from the last preceding call. This will allow an exit routine for example to acquire GETVIS storage and get its address saved from call to call. In case of multiple job control exit routines, a SET SDL for one single $JOBEX0n will cause all anchor fields to be re-initialized to x'00000000'.

R13: Skip mode indicator:

R13=X'00000000'	Job Control will process the statement.
R13=X'000000FF'	Job Control will ignore the statement (that is job control is in skip mode, the statement does not come from SYSLOG and the statement is not JOB, /&, /+ or /.). See page 249 for information on skip mode.

R14: Return address

R15: Entry address of $JOBEXIT

Return Codes from $JOBEXIT

R15=X'00000000'	requests job control to continue processing the current statement.
R15=X'D5D3D6C7'	requests job control to ignore the current statement. The statement will NOT be logged (NLOG), neither on SYSLST nor on SYSLOG.
R15=X'C3D3D6C7'	requests job control to ignore the current statement. The statement will be logged conditionally (CLOG), that is depending whether LOG and/or // OPTION LOG are currently in effect or not.
any other non-zero value	requests job control to ignore the current statement. The statement will be logged unconditionally both on SYSLST and on SYSLOG.

Registers Passed from $JOBCTLN to $JOBACCT

```
R11: Length of partition accounting table
R12: Base register
R13: Pointer to user extent save area
R14: Link return
R15: Pointer to user section of a partition accounting table
```

Return Codes from Phase $IJBASGN in Register 15

```
00 - Assignment successful
04 - No free LUB entry found
08 - Device cuu not found in PUB table
0C - Device cuu is not a disk
10 - Device cuu is down
18 - No free tape unit found
1C - Invalid logical unit for unassignment
20 - Device reserved by volume statement or by mount request from
     another partition
24 - Invalid function code
28 - No GETVIS space available
2C - Device to be unassigned is not assigned
30 - Device owned by another partition
34 - Conflicting i/o assignment, e.g. tape cuu to be assigned is
     already assigned to syslst and/or syspch of calling partition
38 - Invalid lub or lub is not free (only for function asgtps)
3c - Defined mode not supported for defined device
40 - No device found which supports the specified mode
```

Return Codes from Phase $IJBCJC in Register 15

```
00 - Function successfully executed
04 - Wanted information not available
08 - Invalid parameter field
0C - No GETVIS space available
```

Return Codes from Phase $IJBPROC in Register 15

1. If called by macro PARMMAC

```
00 - Request was successful
08 - Invalid length in LENFLD
0C - Invalid pointer for a buffer parameter
10 - Parameter not defined in GETVAL-request
14 - SETPDF-request occurred twice
18 - SETPDF-request occurred after second GETREC
1C - So many symbolic parameters are used that more than 20K bytes
     of system or space GHETVIS area is needed.
20 - No system GETVIS space available
2C - No partition GETVIS space available
40 - Invalid function
```

2. If called by macro PROCMAC

```
00 - Request was successful
04 - Procedure not found
08 - EOPREQ was given on level 0
0C - GETREC was given on level 0
10 - ACCESS exceeds nesting level of 15
14 - Duplicate procedure name in nested stack
18 - Request outside member
1C - Invalid pointer to buffer
20 - No system GETVIS space available
24 - Librarian detects inconsistency
28 - Conflict in nested stack related to DATA=YES/NO option
2C - No partition GETVIS space available
30 - Error in LABEL request
34 - Partition FREEVIS failed
40 - Invalid function
```

Interface from and to Phase $IJBJCCN

Information received by $IJBJCCN when called :

```
R 3: Virtual partition start address
R 4: Physical end of virtual partition
R 5: Address of PARM information in job control work area
R 6: 0 if virtual, else real mode
R 9: Phase name (characters 0-3) if called by $JOBCTLE
R 9: Zero if called from $JOBCTLB
R10  Phase name (characters 4-7)
R11: Partition end address
R12: Load point of the program
```

Information passed when calling the program:

```
R 0: Load point of the program specified in EXEC
R 1: Pointer to the PARM information if specified,
     otherwise entry point of program
R13: Pointer to 18F save area
R14: Return point to this phase
R15: Entry point of the program specified in EXEC
```

Cross-References

Command-to-Phase Cross-Reference

Figure 26 (Page 1 of 2). Command-to-phase Cross-Reference

Command/Statement	Phase(s)
ALLOC	$JOBCTLJ
ALLOCR	$JOBCTLJ
ASSGN	$JOBCTLD
CANCEL	$JOBCTLG
CATALR	$JOBCTLJ
CLOSE	$JOBCTLD
DATE	$JOBCTLJ
DLBL	$JOBCTLK
DVCDN	$JOBCTLF
DVCUP	$JOBCTLF
END or ENTER	$JOBCTLA
EXEC	$JOBCTLE
EXTENT	$JOBCTLK
GOTO	$JOBCTLI
HOLD	$JOBCTLJ
ID	$JOBCTLG
IF	$JOBCTLI
IGNORE	$JOBCTLA
JCLEXIT	$JOBCTLO
JOB	$JOBCTLG
LIBDEF	$JOBCTLH
LIBDROP	$JOBCTLH
LIBLIST	$JOBCTLH
LIBSERV	$JOBCTLO $IJBSLIB
LISTIO	$JOBCTLF
LOG	$JOBCTLJ
MAP	$JOBCTLO $IJBMAP
MSECS	$JOBCTLE
MTC	$JOBCTLJ
NOLOG	$JOBCTLJ
NPGR	$JOBCTLJ
ON	$JOBCTLI
OPTION	$JOBCTLG
OVEND	$JOBCTLA
PAUSE	$JOBCTLA
PROC	$JOBCTLE
PRTY	$JOBCTLE $IJBPRTY
PWR	$JOBCTLE
QUERY	$JOBCTLO $IJBSDSP
RELSE	$JOBCTLJ
RESET	$JOBCTLF
ROD	$JOBCTLM
RSTRT	$JOBCTLB $JOBCTLK
SET	$JOBCTLJ
SETPARM	$JOBCTLE
SETPFIX	$JOBCTLO
SETPRT	$JOBCTLK
SIZE	$JOBCTLJ
START	$JOBCTLG
STDOPT	$JOBCTLJ

Figure 26 (Page 2 of 2). Command-to-phase Cross-Reference

Command/Statement	Phase(s)
STOP	$JOBCTLJ
SYSDEF	$JOBCTLO $IJBSDSP
TLBL	$JOBCTLK
UCS	$JOBCTLJ
UNA	$JOBCTLF
UNBATCH	$JOBCTLF
UPSI	$JOBCTLJ
VDISK	$JOBCTLO $IJBVDII
ZONE	$JOBCTLJ
*	$JOBCTLA
/.	$JOBCTLA
/+	$JOBCTLG
/&	$JOBCTLG
/*	$JOBCTLA
//	$JOBCTLA
:READ	$JOBCTLA

Notes:

1. The linkage editor commands ACTION, ENTRY, PHASE, MODE, INCLUDE are processed by subphase $JOBCTLJ, that is they are written to SYSLNK.
2. Most Attention Routine commands are allowed if specified in $0JCL.PROC. They are processed by subphase $JOBCTLO, that is they are passed to the attention routine by means of an SVC 30 interface. Symbolic parameters are resolved.
3. 'Modern' job control commands (such as LIBSERV, MAP, PRTY, QUERY, SYSDEF, VDISK) are handled by SVA-resident command-processors (such as $IJBSLIB, $IJBMAP, $IJBPRTY, $IJBSDSP, $IJBVDII). The interface between the SVA phase and root phase $JOBCTLA is provided by subphase $JOBCTLO.
4. Beside these 5 SVA-resident command-processors there are 5 other SVA-resident service routines related to job control.

   $IJBASGN This phase is called in the expansion of the ASSIGN macro. Refer to the *System Macros Reference* manual for a description of the ASSIGN macro.

   $IJBCJC This phase is called in the expansion of the CONDJC macro. Refer to the *IPL and Job Control Diagnosis Reference* manual for a description of the CONDJC macro.
   $IJBCJC stores and retrieves conditional job control information (such as ON conditions, last and maximal return code of a job step).

   $IJBJCCN This phase is executed during EXEC PGM processing, after $JOBCTLE has completed and before control is being passed to the user program. $IJBJCCN invalidates the partition, loads the user program into the partition (SVC 4) and passes control to the user program (SVC 133). When the user program returns (with BR R14 or EOJ macro), $IJBJCCN is invoked again to handle the return code passed by the user program (via R15 or via the RC operand of the EOJ macro).

   $IJBPROC This phase is called in the expansion of the GETSYMB, PARMMAC and PROCMAC macro. Refer to the *System Macros Reference* manual for a description of the GETSYMB macro. Refer to the *IPL and Job Control Diagnosis Reference* manual for a description of the PARMMAC macro.
   GETSYMB and PARMMAC provide services to store or retrieve symbolic parameters and their values (for example during SETPARM, PROC, EXEC PROC or EXEC REXX processing).

PROCMAC invokes LIBR services to retrieve records from a cataloged procedure (for example during EXEC PROC or EXEC REXX processing).

$IJBSLA This phase is called in the expansion of the LABEL macro. Refer to the *IPL and Job Control Diagnosis Reference* manual for a description of the LABEL macro.
$IJBSLA stores and retrieves label-information records. It writes to or reads from the system's label-information area.

$IJBSLA consists of two modules:

IJBSLA provides the label functions for a label-information area on disk (CKD or FBA).

IJBSLAD provides exactly the same label functions but with a label-information area that resides in data space (created by the command VDISK.....USAGE=DLA)

For these SVA-resident service routines the entry point is identical to the load point, which can be retrieved by means of the LIBR LISTDIR SDL command. Control is passed via BALR 14,15. Upon entry, Reg0 contains a function code, Reg1 points to a parameter list and R14 points back to the caller.
Example: When entering $IJBSLA, Reg1 contains the address of the LPL (label parameter list generated by the LPL macro). Reg0 contains the function code, such as 1 for GETLBL, 4 for ADDLBL or 6 for CLRGRPL etc.

Message-to-Phase Cross-Reference

Figure 27 (Page 1 of 5). Message-to-phase Cross-Reference

Message	Phase(s), prefix $JOB or $IJB omitted
0R00I	CTLB
0R01I	CTLB
0R03I	CTLB
0R04I	CTLB
0R05I	CTLB
0R06I	CTLB
0R07I	CTLB
0R08I	CTLB
0R09I	CTLB
0R10I	CTLB
0R11I	CTLB
0R12I	CTLB
0R13I	CTLB
0R15I	CTLB
0R16A	CTLB
0R17I	CTLB
0S35I	CTLE
1A0nD	CTLD
1A1nD	CTLD
1A2nt	CTLD CTLJ
1A4nD	CTLD CTLF CTLJ CTLK
1A5nt	CTLD CTLF CTLJ CTLK SLIB VDII VTAP
1A60t	CTLD
1A70D	CTLD CTLF CTLJ
1A80t	CTLD
1A81I	CTLA
1A82D	CTLD
1A83t	CTLD
1A84D	CTLF
1A85I	CTLF
1A86I	CTLF
1A87D	CTLD
1A88D	CTLD
1A89D	CTLD
1A9nt	CTLD CTLF
1AA0D	CTLF
1AA1t	CTLF
1B08I	CTLJ
1C00A	CTLA
1C10D	CTLA CTLJ
1C30t	CTLJ
1C60D	CTLG
1C70D	CTLA
1C80D	CTLA
1C90I	CTLA CTLG
1C91I	CTLE
1C92D	CTLD
1C93D	CTLD
1C94D	CTLD
1D01t	CTLA
1D02t	CTLH
1D03t	CTLH
1D04t	CTLH
1D06I	CTLH

Figure 27 (Page 2 of 5). Message-to-phase Cross-Reference

Message	Phase(s), prefix $JOB or $IJB omitted
1D07t	CTLH
1D09t	CTLH
1D10t	CTLH
1D12t	CTLH
1D13t	CTLH
1D14t	CTLH
1D3nt	CTLH
1D4nt	CTLH
1D5nt	CTLH
1D6nt	CTLH
1D7nt	CTLH
1D8nI	CTLH
1E1nt	CTLH
1E4nt	CTLH
1F00I	CTLG
1F01I	CTLG
1F02D	CTLE CTLI
1F03D	CTLE
1F04D	CTLE
1F05D	CTLE
1F06D	CTLI
1F07D	CTLE
1F08D	CTLE
1F09D	CTLE
1F1nD	CTLE CTLI
1F2nD	CTLA CTLE CTLI
1F3nD	CTLA CTLE CTLI
1F4nD	CTLE CTLI
1F5nD	CTLI
1F6nD	CTLI
1F7nD	CTLI
1I00D	CTLA CTLG
1I20I	CTLG
1I21I	CTLG
1I39t	CTLJ
1I50I	CTLA
1I70I	CTLA
1I81I	CTLM
1I82t	CTLM
1I83A	CTLM
1I84A	CTLM
1I86A	CTLM
1I90D	CTLM
1I92D	CTLM
1I93t	CTLM
1I94I	CTLC
1I95A	CTLC
1I96A	CTLC
1I97E	CTLC
1I98I	CTLJ
1I99A	CTLC
1L0nt	CTLK
1L1nD	CTLG CTLK
1L2nt	CTLK
1L30D	CTLK
1L40I	SLA

Figure 27 (Page 3 of 5). Message-to-phase Cross-Reference

Message	Phase(s), prefix $JOB or $IJB omitted
1L41A	SLA
1L5nD	CTLK
1L60D	CTLK
1L61I	CTLG
1L62D	CTLG
1L63I	CTLG
1L64D	CTLG
1L65t	CTLG
1L66D	CTLG
1L67D	CTLG
1L68D	CTLG
1L70D	CTLG
1L90I	LSERV
1L91I	LSERV
1L92I	LSERV
1M10A	CTLA
1M20D	CTLA
1M21D	CTLG
1M3nD	CTLA
1M4nD	CTLG
1M7nD	CTLE
1M80D	CTLE CTLI
1M81D	CTLE
1M82I	CTLG
1M9nt	CTLE
1N00I	CTLA
1N10D	CTLE
1N11t	CTLE
1N2nt	CTLE
1N7nD	CTLG
1N80I	CTLG
1N90I	CTLG
1N91I	CTLA
1N92D	CTLE
1P01D	CTLJ
1P02D	CTLJ
1P03I	CTLJ
1P04D	CTLE CTLJ
1P05D	CTLJ
1P1nD	CTLG
1P2nt	CTLE
1P76I	PRTY
1P77I	PRTY
1S0nt	CTLA CTLJ ATTN
1S1nt	CTLE CTLG CTLJ CTLK
1S2nI	CTLJ
1S3nt	CTLE
1S40t	CTLA CTLB CTLD CTLE CTLF CTLG CTLI CTLJ CTLK CTLO ATTN PRTY SDSP SLIB VDII VTAP
1S42A	CTLJ
1S43D	CTLE
1S44t	CTLE
1S45D	CTLJ
1S46I	CTLI
1S47I	CTLE
1S48D	CTLE
1S49I	CTLI

Figure 27 (Page 4 of 5). Message-to-phase Cross-Reference

Message	Phase(s), prefix $JOB or $IJB omitted
1S50D	CTLA
1S51D	CTLJ
1S52D	CTLJ
1S53D	CTLE
1S55I	CTLA
1S56D	CTLE
1S57D	CTLE
1S58D	CTLE
1S59D	CTLE
1S6nt	CTLE
1S70D	CTLK VTAP
1S71D	CTLJ
1S72D	CTLG
1S73t	CTLG
1S75I	PRTY
1S76I	CTLE
1S77D	CTLE
1S78I	CTLG
1S79D	CTLA
1S8nt	CTLG
1S9nD	CTLA
1SA0D	CTLA
1T10I	CTLJ
1T20I	CTLD
1T40D	CTLD
1T50A	CTLD
1T60A	CTLD
1T70A	CTLD
1T71D	CTLE
1T80I	CTLA
1U00t	CTLE CTLG CTLH
1U1nD	CTLJ
1U3nD	CTLJ
1U40t	CTLE
1U5nt	CTLE
1U6nt	CTLE
1U70A	CTLE
1U71I	CTLG
1U72I	CTLA
1U73D	CTLO
1U75D	CTLA CTLO VTAP
1U76I	CTLO
1U80t	CTLO
1U81I	CTLO
1U82I	CTLO
1UV1t	VDII
1UV2t	VDII
1UV3t	VDII
1UV4t	VDII VTAP
1UV5t	VDII
1UV6t	VDII
1UV7t	SDSP VDII
1UV8t	VDII
1UV9D	VDII
1Y01t	SDSP
1Y02I	MAP

Figure 27 (Page 5 of 5). Message-to-phase Cross-Reference

Message	Phase(s), prefix $JOB or $IJB omitted
1Y05t	SLIB VDII VTAP
1Y07t	VDII
1Y08t	VDII
1Y09t	SLIB
1Y0At	SLIB
1Y1nt	SDSP SLIB
1Y2nt	MAP PRTY SDSP SLIB VDII VTAP
1Y3nt	SDSP SLIB VDII VTAP
1Y4nt	CTLE PRTY SDSP VDII
1Y5nt	PRTY SDSP SLIB VDII VTAP
1Y6nt	PRTY SDSP SLIB VDII VTAP
1Y7nt	PRTY SDSP SLIB VDII VTAP
1Y8nt	MAP PRTY SDSP SLIB
1Y9nt	SLIB
1YAnt	SDSP SLIB
1YBnt	SLIB
1YCnt	SLIB VTAP
1YEnI	SLIB
1YFnI	SLIB
1YGnt	CTLE
1YH0I	SLIB
1YH1I	SLIB
1YH2I	SLIB
1YH3t	SLIB
1YH4I	SLIB
1YH5t	SDSP
1YH6I	SDSP
1YH7I	SDSP
1YH8t	CTLG SLIB
1YK0t	SLIB
1YK1t	PRTY SDSP
1YK2I	SDSP
1YK3t	SDSP
1YK4t	SDSP
1YK5t	SLIB
1YK6t	SDSP
1YK7I	SDSP
1YK8t	VTAP
1YK9t	VTAP
1YLnt	CTLJ
1YM1t	VTAP
1YM2t	VTAP
1YM3I	VTAP
1YM4I	VTAP
1YM5t	VTAP
1YM6I	VTAP
1YM7t	VTAP
1YM8t	VTAP
1YM9t	VTAP
1YN1t	VTAP
1YN2t	VTAP
1YN3t	VTAP
1YN4t	VTAP

Phase-to-Module Cross-Reference

Figure 28. Phase-to-Module Cross-Reference

Phase	Module
$IJBASGN	IJBASGN
$IJBATTN	IJBATTN
$IJBCJC	IJBCJC
$IJBJCCN	IJBJCLCN
$IJBMAP	IJBMAP
$IJBPROC	IJBPROC
$IJBPRTY	IJBPRTY
$IJBSDSP	IJBSDSP
$IJBSLA	IJBSLA
$IJBSLA	IJBSLAD
$IJBSLIB	IJBSLIB
$IJBSTRT	IJBSTRT
$IJBVDII	IJBVDII
$IJBVTAP	IJBVTAP
$JOBACCT	$JOBACCT
$JOBCTLA	IJBJC1
$JOBCTLB	IJBJCB
$JOBCTLC	IJBJCC
$JOBCTLD	IJBJC2
$JOBCTLE	IJBJC9
$JOBCTLF	IJBJC5
$JOBCTLG	IJBJC3
$JOBCTLH	IJBJCH
$JOBCTLI	IJBJCI
$JOBCTLJ	IJBJC4
$JOBCTLK	IJBJC6
$JOBCTLM	IJBJC7
$JOBCTLN	IJBJC8
$JOBCTLO	IJBJCO
$JOBEXIT	$JOBEXIT
$SYSOPEN	$SYSOPEN

Job Control Macros Quick Reference

Figure 29 (Page 1 of 2). Job Control Macros Quick Reference

Macro	Description
ASPL	Provides an ASsign Parameter List as required by the ASSIGN macro. Refer to the *System Macros Reference* manual for a description of the ASPL macro.
ASSIGN	Calls SVA-resident service routine $IJBASGN to dynamically assign or unassign I/O devices. Refer to the *System Macros Reference* manual for a description of the ASSIGN macro.
CONDJC	Calls SVA-resident service routine $IJBCJC to store or retrieve conditional job control information (such as ON conditions, last and maximal return code of a job step). Refer to “CONDJC Macro” on page 219 for a detailed description.
DFB	Generates CCB, CCWs, control bytes and control buffer space for the I/O on system files. Refer to “DFBs -- Data File Blocks in Phase $JOBCTLA” on page 248 for further information.
GETSYMB	Allows user programs and job control exit routine(s) to retrieve the value of symbolic parameters. Refer to the *System Macros Reference* manual for a description of the GETSYMB macro.
IJBJCLBR	Common interface mapping between JCL and LIBR (PL/X only).

Figure 29 (Page 2 of 2). Job Control Macros Quick Reference

Macro	Description
IJBLBRC	Provides a mapping for the DLBL/EXTENT information as described in Figure 21 on page 240 (PL/X only).
JCLARIF	Provides an interface area for $JOBCTLO or $IJBATTN on the one hand and the SVA-resident command-processors on the other. Refer to "JCARAREA - Interface Area Between JCL/AR and SVA Routines" on page 251 for further information.
JOBCOM	Calls SVA-resident service routine $IJBSLA to store or retrieve job-to-job-communication information (256 bytes). Also refer to the *System Macros Reference* manual for a description of the JOBCOM macro.
LABEL	Calls SVA-resident service routine $IJBSLA to store or retrieve label-information records. Refer to Figure 20 on page 239 or Figure 21 on page 240 for the format of label-information records. Refer to "LABEL Macro" on page 221 for a detailed description of the LABEL macro.
LPL	Provides a Label Parameter List as required by the LABEL macro. Refer to "LPL Macro" on page 229 for a detailed description of the LPL macro.
LPLDCT	Provides a mapping of the label parameter list generated by the LPL macro. Refer to "LPLDCT Macro" on page 230 for a detailed description of the LPLDCT macro.
MAPJCLA	Provides a mapping of the common area needed for the communication between the job control root phase $JOBCTLA and the job control subphases $JOBCTLx. Refer to "BASVCT -- Common Job Control Area" on page 247 for further information on this area.
MAPUE	Provides a mapping of the control blocks related to job control exit routines.
PARMMAC	Calls SVA-resident service routine $IJBPROC to store or retrieve symbolic parameters and their values. Refer to "PARMMAC Macro" on page 232 for a detailed description of the PARMMAC macro.
PROCMAC	Calls SVA-resident service routine $IJBPROC, which invokes LIBR services to retrieve records from a cataloged procedure. Refer to "PROCMAC Macro" on page 235 for a detailed description of the PROCMAC macro.
RELEASE	Calls $$BRELSE to release programmer logical units. Refer to the *System Macros Reference* manual for a description of the RELEASE macro.
SLADCT	Provides a mapping of the job control work area (where COMREG.IJBJCWA points to). Also provides a mapping of a work area (where SYSCOM.IJBSLACB points to) required for label processing.
STARTP	Calls SVA-resident service routine $IJBSTRT to start a partition. Refer to the *Supervisor Diagnosis Reference* manual for a description of the STARTP macro.

Job Control Dumps: Where to Look First?

The term "job control dump" denotes a partition dump taken at a time, when job control (and not a user program) was active in the partition. The most important storage area to look at is the save area located at the very start of the partition.

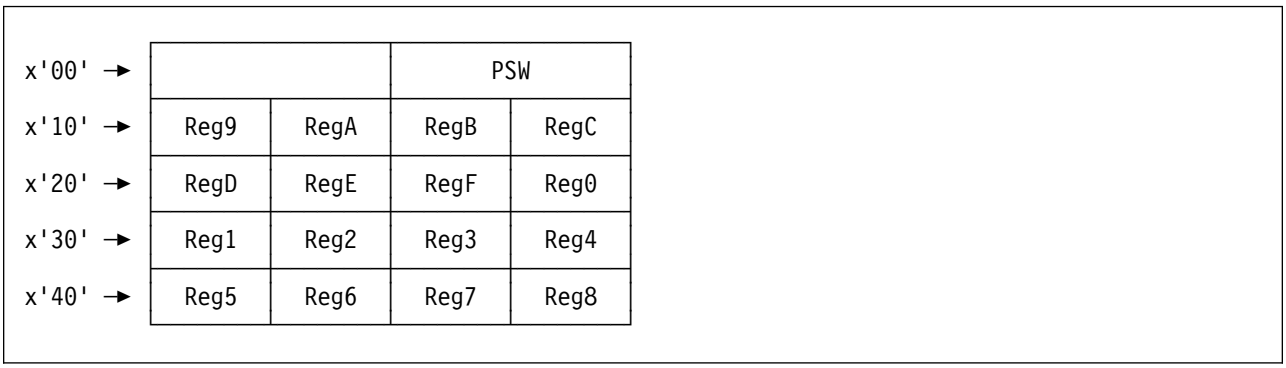


Figure 30. Partition Save Area Layout

For the examples discussed below we assume that the partition starts at X'600000'. When the root phase or one of the subphases is executing, the PSW contains instruction addresses in the range from X'600000' to X'613FFF'. In this case, the following register conventions are in effect:

RegA
Points to the partition's COMREG.

Reg7 Points to a branch vector table located at X'6000E0' in $JOBCTLA. This branch vector table allows the job control subphases to exploit common service routines provided by $JOBCTLA, such as operand scanners, message writers etc. Reg7 is also the first base register of root phase $JOBCTLA.

Reg8 Start of the overlay area (X'609000'), where subphases are loaded. Reg8 is also the first base register of each subphase $JOBCTLx.

Reg7, Reg6, Reg9, RegB
Base registers of root phase $JOBCTLA.

Reg8, Reg9 (, Reg6 (, RegC))
Base registers of subphase $JOBCTLx. It depends on the subphase, if 2, 3 or 4 base registers are needed.

If however one of the SVA-resident command-processors or service routines is executing, the PSW contains instruction addresses from the SVA-24 or SVA-31 area. In this case there are no general register conventions: it is up to the (PLS or PLX) compiler to choose registers.

The next storage area to look at is the eyecatcher at X'78' relative to the partition start address.

```
V00600070  40404040 40404040 5BD1D6C2 C3E3D3C1    *        $JOBCTLA*
V00600080  5CF1F5C3 5CC4E8F4 F4F2F4F1 5CF0F861    **15C*DY44241*08/*
V00600090  F2F261F9 F65CF5F6 F8F660F0 F6F6404D    *22/96*5686-066 (*
V006000A0  C35D40C3 D6D7E8D9 C9C7C8E3 40C9C2D4    *C) COPYRIGHT IBM*
V006000B0  40C3D6D9 D74B40F1 F9F7F76B 40F1F9F9    * CORP. 1977, 199*
```

Figure 31. Eyecatcher of Job Control Root Phase

This eyecatcher contains useful information. Separated by an asterisk we find:

$JOBCTLA
In this case this is (per chance) the name of the root phase itself. If a subphase (for example $JOBCTLE) is loaded, the corresponding suffix (in this example E) will overlay the A, thus resulting in $JOBCTLE and reflecting, which subphase is currently active (see Figure 16 on page 141). However the rest of the eyecatcher remains unchanged, that is it applies to $JOBCTLA.

15C This is the release code of VSE/ESA Version 2, Release 1 or 2.

DY44241 This is the service level of the root phase $JOBCTLA and therefore of particular interest for VSE Level 2.

08/22/96 This is the date of the last development activity for $JOBCTLA.

If job control is in command processing mode (not in initialization mode) the next storage area to look at is the eyecatcher near X'9000' relative to the partition start address. This eyecatcher is preceded by a branch table. Each branch table entry relates to a job control command or to an internal function provided by the subphase. Figure 32 shows 6 branch table entries, the first 5 of which are corresponding to the RSTRT, DLBL, EXTNT, TLBL, and SETPRT commands.

```
V00609000  47F090C0 47F087D0 47F089B2 47F08116 16 *.0...0g..0i..0a.*
V00609010  47F097A8 47F08F66 5BD1D6C2 C3E3D3D2    *.0py.0..$JOBCTLK*
V00609020  5CF1F5C3 5CC4E8F4 F2F8F2F7 5CF0F461    **15C*DY42827*04/*
V00609030  F1F761F9 F65CF5F6 F8F660F0 F6F6404D    *17/96*5686-066 (*
V00609040  C35D40C3 D6D7E8D9 C9C7C8E3 40C9C2D4    *C) COPYRIGHT IBM*
V00609050  40C3D6D9 D74B40F1 F9F7F76B 40F1F9F9    * CORP. 1977, 199*
```

Figure 32. Eyecatcher of Job Control Subphase

This eyecatcher contains useful information. Separated by an asterisk we find:

$JOBCTLK
This is the name of the subphase currently active (same as on location X'600078').

15C This is the release code of VSE/ESA Version 2, Release 1 or 2.

DY42827 This is the service level of the subphase $JOBCTLK and therefore of particular interest for VSE Level 2.

04/17/96 This is the date of the last development activity for $JOBCTLK.

Another storage location to look at is X'1BC' relative to the partition start address. There we find the command input buffer, showing, which command is currently being executed.

```
V006001B0  006051E4 40404040 40404040 616140C4 16 *.-.U        // D*
V006001C0  D3C2D340 C1C2C36B 7DC1C2C3 4BE3C5E2    *LBL ABC,'ABC.TES*
V006001D0  E34BC6C9 D3C57D6B F1F9F9F8 61F3F6F5    *T.FILE',1998/365*
V006001E0  40404040 40404040 40404040 40404040    *                *
```

Figure 33. Job Control Command Input Buffer

Index

Numerics

A

C

D

G

I

J

L

M

P

R

S

Communicating Your Comments to IBM

IBM VSE/Enterprise Systems Architecture
VSE Central Functions
Initial Program Load and Job Control
Diagnosis Reference
Version 6 Release 6

Publication No. SC33-6325-02

If you especially like or dislike anything about this book, please use one of the methods listed below to send your comments to IBM. Whichever method you choose, make sure you send your name, address, and telephone number if you would like a reply.

Feel free to comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of the book. However, the comments you send should pertain to only the information in this manual and the way in which the information is presented. To request additional publications, or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

If you are mailing a readers' comment form (RCF) from a country other than the United States, you can give the RCF to the local IBM branch office or IBM representative for postage-paid mailing.

- If you prefer to send comments by mail, use the RCF form and either send it postage-paid in the United States, or directly to:

  IBM Deutschland Entwicklung GmbH
  Department 3248
  Schoenaicher Strasse 220
  D-71032 Boeblingen
  Federal Republic of Germany

- If you prefer to send comments by FAX, use this number:
  - (Germany): 07031-16-3456
  - (Other countries): (+49)+7031-16-3456
- If you prefer to send comments electronically, use this network ID:

  INTERNET: s390id@de.ibm.com

Make sure to include the following in your note:

- Title and publication number of this book
- Page number or topic to which your comment applies.

Readers' Comments — We'd Like to Hear from You

IBM VSE/Enterprise Systems Architecture
VSE Central Functions
Initial Program Load and Job Control
Diagnosis Reference
Version 6 Release 6

Publication No. SC33-6325-02

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	□	□	□	□	□

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	□	□	□	□	□
Complete	□	□	□	□	□
Easy to find	□	□	□	□	□
Easy to understand	□	□	□	□	□
Well organized	□	□	□	□	□
Applicable to your tasks	□	□	□	□	□

Please tell us how we can improve this book:

Thank you for your responses. May we contact you? □ Yes □ No

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Name _____ Address _____

Company or Organization _____

Phone No. _____

Readers' Comments — We'd Like to Hear from You
SC33-6325-02

Cut or Fold
Along Line

Fold and Tape **Please do not staple** Fold and Tape

PLACE
POSTAGE
STAMP
HERE

IBM Deutschland Entwicklung GmbH
Department 3248
Schoenaicher Strasse 220
D-71032 Boeblingen
Federal Republic of Germany

Fold and Tape **Please do not staple** Fold and Tape

Cut or Fold
Along Line

File Number: S370/S390-37
Program Number: 5686-066

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

SC33-6325-02